Algorithms

Pseudocode

Usually, we will not represent algorithms as programs for a RAM but rather as programs in some high-level programming language.

We will not use any particular programming language.

Rather, we will write programs in a form of **pseudocode** whose syntax could be adjusted in arbitrary ways according to our needs (e.g., we will use things like arbitrary mathematical notation, descriptions in a natural language, and so on, freely).

Example:

Algorithm: An algorithm for finding the maximal element in an array

```
FIND-MAX(A, n):
    k := 0
    for i := 1 to n − 1 do
        if A[i] > A[k] then
            k := i
    return A[k]
```

Algorithms

Remark:

From the point of view of an analysis how a given algorithm works, it usually makes only a little difference if the algorithm:

- reads input data from some input device (e.g., from a file, from a keyboard, etc.)
- writes data to some output device (e.g., to a file, on a screen, etc.)

or

- reads input data from a memory (e.g., they are given to it as parameters)
- writes data somewhere to memory (e.g., it returns them as a return value)

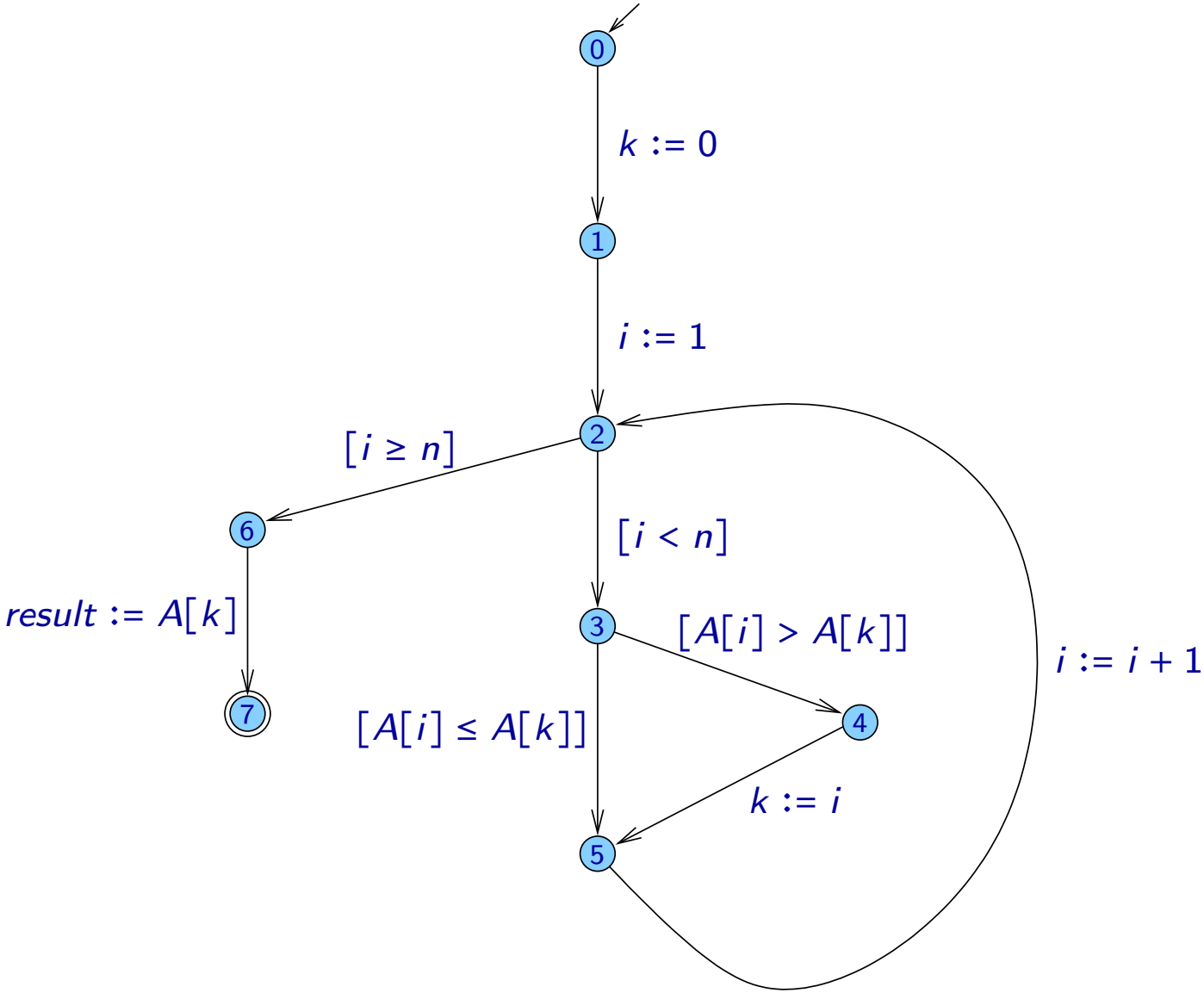
So in a pseudocode, input data will be often given as arguments of a function and an output will be represented as a return value of this function.

Control Flow

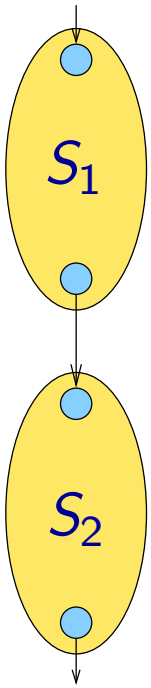
Instructions can be roughly devided into two groups:

- instructions working directly with data:
  - assignment
  - evaluation of values of expressions in conditions
  - reading input, writing output
  - ...
- instructions affecting the **control flow** — they determine, which instructions will be executed, in what order, etc.:
  - branching (if, switch, ...)
  - cycles (while, do .. while, for, ...)
  - organisation of instructions into blocks
  - returns from subprograms (return, ...)
  - ...

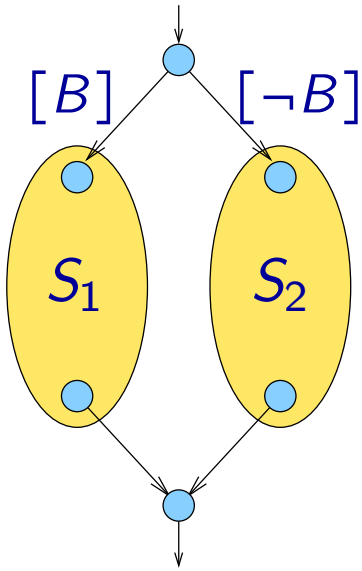
Control Flow Graph



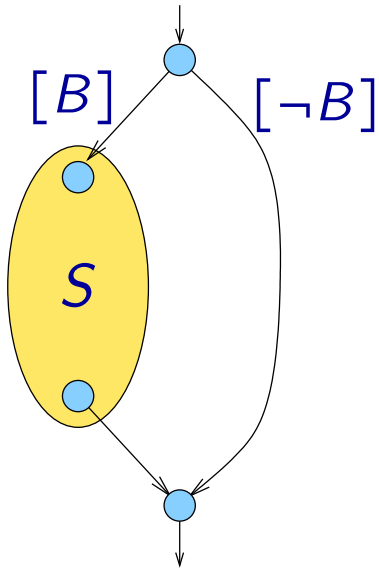
Some Basic Constructions of Structured Programming



$S_1; S_2$

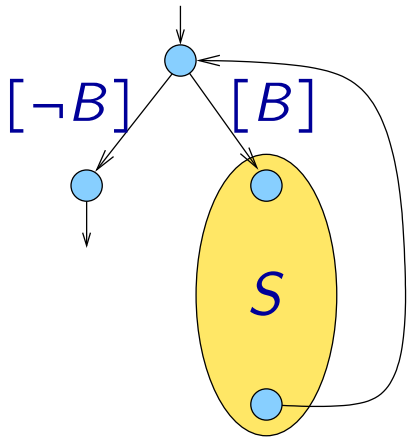


if B **then** S_1 **else** S_2

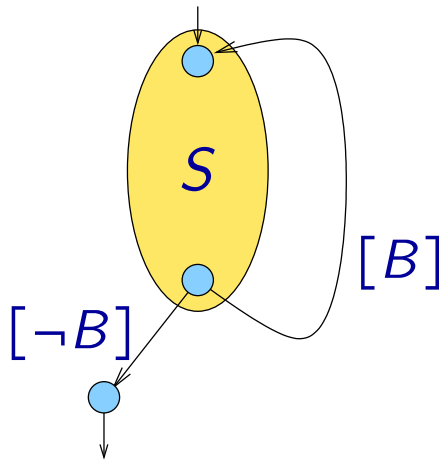


if B **then** S

Some Basic Constructions of Structured Programming

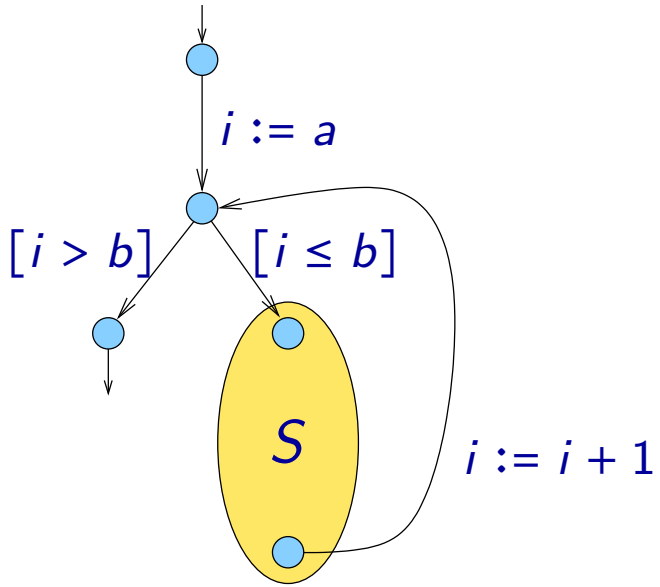


while B **do** S



do S **while** B

Some Basic Constructions of Structured Programming



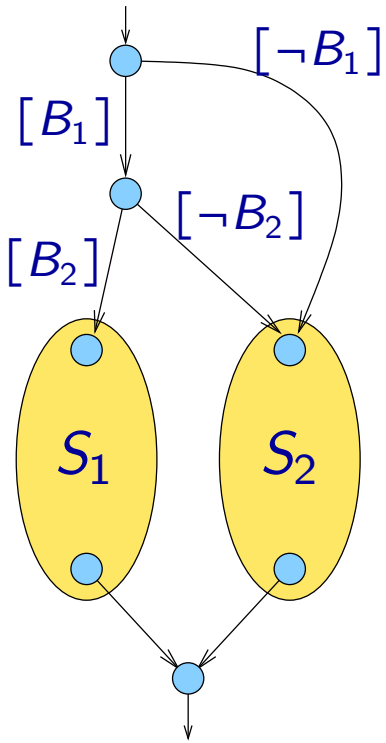
$i := a$
while $i \leq b$ **do**
  S
  $i := i + 1$

for $i := a$ **to** b **do** S

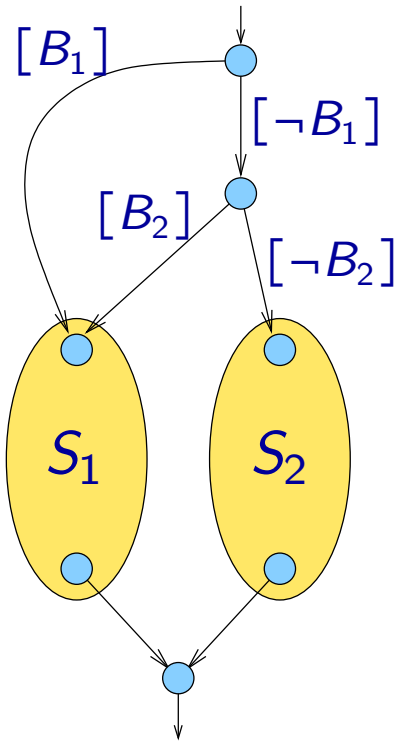
Some Basic Constructions of Structured Programming

Short-circuit evaluation of compound conditions, e.g.:

while $i < n$ **and** $A[i] > x$ **do** ...



if B_1 **and** B_2 **then** S_1 **else** S_2



if B_1 **or** B_2 **then** S_1 **else** S_2

Control-flow Realized by GOTO

- **goto** ℓ — **unconditional jump**
- **if** B **then goto** ℓ — **conditional jump**

Example:

0: $k := 0$
1: $i := 1$
2: **goto** *6*
3: **if** $A[i] \leq A[k]$ **then goto** *5*
4: $k := i$
5: $i := i + 1$
6: **if** $i < n$ **then goto** *3*
7: **return** $A[k]$

Control-flow Realized by GOTO

- **goto** ℓ — **unconditional jump**
- **if** B **then goto** ℓ — **conditional jump**

Example:

$$
\begin{array}{ll}
\mathit{start}: & k := 0 \\
& i := 1 \\
& \textbf{goto } \mathit{L3} \\
\mathit{L1}: & \textbf{if } A[i] \leq A[k] \textbf{ then goto } \mathit{L2} \\
& k := i \\
\mathit{L2}: & i := i + 1 \\
\mathit{L3}: & \textbf{if } i < n \textbf{ then goto } \mathit{L1} \\
& \textbf{return } A[k]
\end{array}
$$

Evaluation of Complicated Expressions

Evaluation of a complicated expression such as

$$A[i+s] := (B[3*j+1]+x)*y+8$$

can be replaced by a sequence of simpler instructions on the lower level, such as

$$\begin{aligned} t_1 &:= i+s \\ t_2 &:= 3*j \\ t_2 &:= t_2+1 \\ t_3 &:= B[t_2] \\ t_3 &:= t_3+x \\ t_3 &:= t_3*y \\ t_3 &:= t_3+8 \\ A[t_1] &:= t_3 \end{aligned}$$

Computation of an Algorithm

Configuration — the description of the global state of the machine in some particular step during a computation

Example: A configuration of the form

$$(q, mem)$$

where

- q — the current control state
- mem — the current content of memory of the machine — the values assigned currently to variables.

An example of a content of memory mem:

$$\langle A\colon [3,8,1,3,6],\quad n\colon 5,\quad i\colon 1,\quad k\colon 0,\quad result\colon ?\rangle$$

Computation of an Algorithm

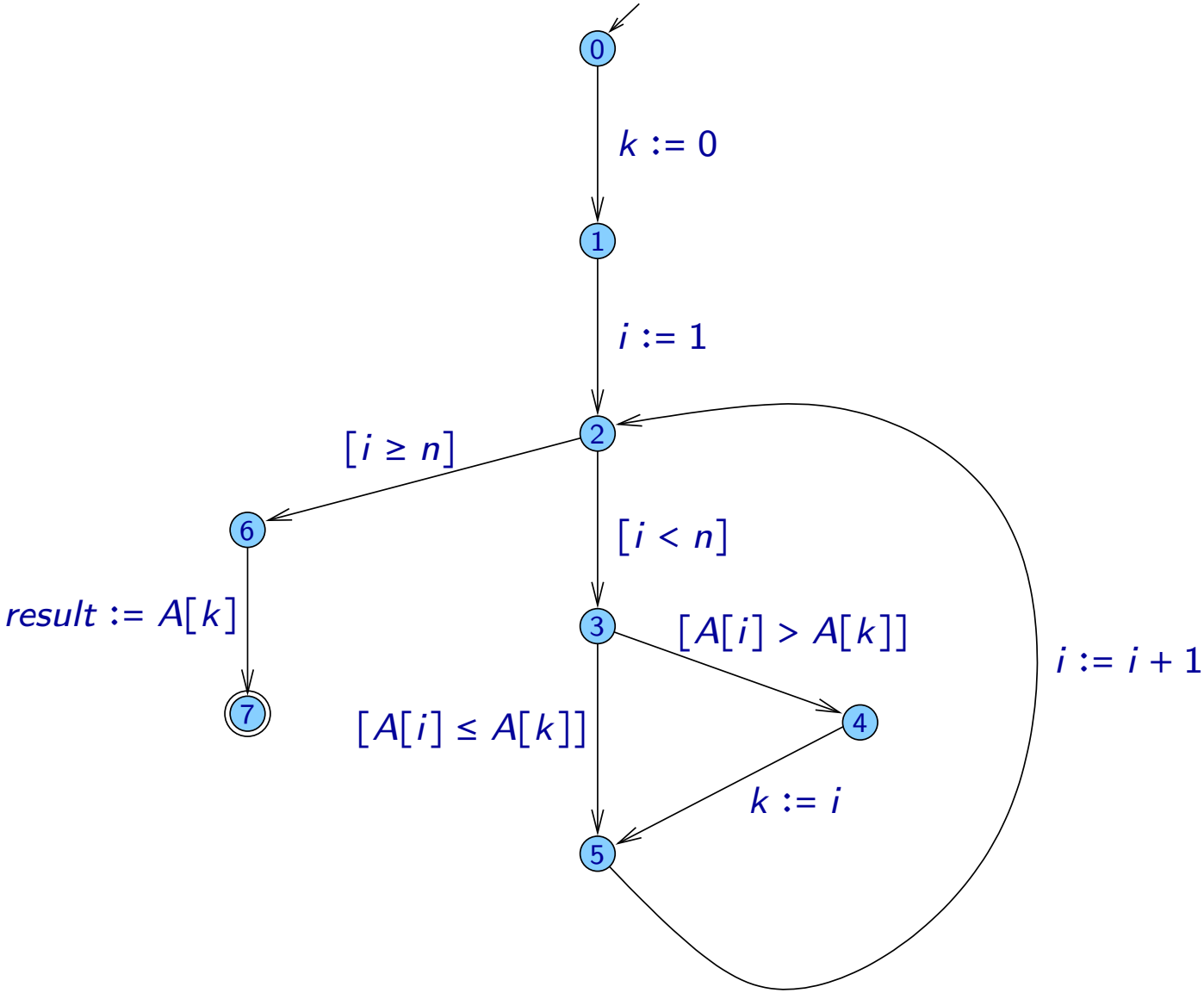
An example of a configuration:

$$(2,\ \langle A\colon [3,8,1,3,6],\ \ n\colon 5,\ \ i\colon 1,\ \ k\colon 0,\ \ \mathit{result}\colon ?\rangle)$$

A **computation** of a machine $\mathcal{M}$ executing an algorithm Alg, where it processes an input w, in a sequence of configurations.

- It starts in an **initial configuration**.
- In every step, the machine goes from one configuration to another.
- The computation ends in a **final configuration**.

Computation of an Algorithm



Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0,\ \langle A\text{: } [3,8,1,3,6],\ \ n\text{: } 5,\ \ i\text{: } ?,\ \ k\text{: } ?,\ \ \mathit{result}\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } ?, \quad \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } ?, \ k\text{: } ?, \ \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } ?, \ k\text{: } 0, \ \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ \mathit{result}\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } ?, \quad \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \quad n\text{: } 5, \quad i\text{: } 1, \quad k\text{: } 0, \quad \mathit{result}\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } ?, \ k\text{: } ?, \ \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } ?, \ k\text{: } 0, \ \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ \mathit{result}\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ \mathit{result}\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 1, \ \mathit{result}\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{10}: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_{10}: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_{11}: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_{10}: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_{11}: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_{12}: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{10}: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{11}: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{12}: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{13}: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ result\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_{10}: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_{11}: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_{12}: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_{13}: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$
α_{14}: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{10}: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{11}: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{12}: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{13}: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{14}: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{15}: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 5,\ k\text{: } 1,\ result\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{10}: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{11}: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{12}: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{13}: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{14}: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{15}: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 5,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{16}: $(6, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 5,\ k\text{: } 1,\ result\text{: } ?\rangle)$

Computation of an Algorithm

Example: A computation, where algorithm FIND-MAX processes an input where $A = [3, 8, 1, 3, 6]$ and $n = 5$.

α_0: $(0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_9: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{10}: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{11}: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{12}: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{13}: $(3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{14}: $(5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{15}: $(2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 5,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{16}: $(6, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 5,\ k\text{: } 1,\ result\text{: } ?\rangle)$
α_{17}: $(7, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 5,\ k\text{: } 1,\ result\text{: } 8\rangle)$

Computation of an Algorithm

By executing an instruction I, the machine goes from configuration α to configuration α':

$$\alpha \xrightarrow{I} \alpha'$$

A computation can be:

- **Finite**:

$$\alpha_0 \xrightarrow{I_0} \alpha_1 \xrightarrow{I_1} \alpha_2 \xrightarrow{I_2} \alpha_3 \xrightarrow{I_3} \alpha_4 \xrightarrow{I_4} \cdots \xrightarrow{I_{t-2}} \alpha_{t-1} \xrightarrow{I_{t-1}} \alpha_t$$

  where α_t is either a final configuration or a configuration where an error occurred and it is not possible to continue in the computation

- **Infinite**:

$$\alpha_0 \xrightarrow{I_0} \alpha_1 \xrightarrow{I_1} \alpha_2 \xrightarrow{I_2} \alpha_3 \xrightarrow{I_3} \alpha_4 \xrightarrow{I_4} \cdots$$

Computation of an Algorithm

A computation can be described in two different ways:

- as a sequence of configurations $\alpha_0, \alpha_1, \alpha_2, \ldots$
- as a sequence of executed instructions $I_0, I_1, I_2, \ldots$

Church-Turing Thesis

It should be clear from the previous discussion that:

- A program written in an arbitrary programming language could be translated to a program for a RAM.
- Behaviour of a RAM could be simulated by a Turing machine.

So the behaviour of a program written in an arbitrary programming language could be simulated by a Turing machine.

Church-Turing thesis

Every algorithm can be implemented as a Turing machine.

It is not a theorem that can be proved in a mathematical sense – it is not formally defined what an algorithm is.

The thesis was formulated in 1930s independently by Alan Turing and Alonzo Church.

Church-Turing Thesis

Examples of mathematical formalisms modelling the notion of an algorithm:

- Turing machines
- Random Access Machines
- Lambda calculus
- Recursive functions
- ...

We can also mention:

- An arbitrary (general purpose) programming language (for example C, Java, Python, Lisp, Haskell, Prolog, etc.).

All these models are equivalent with respect to algorithms that can be implemented by them.

Proving Correctness of Algorithms

Correctness of Algorithms

Algorithms are used for solving **problems**.

- **Problem** — a specification **what** should be computed by an algorithm:
  - Description of inputs
  - Description of outputs
  - How outputs are related to inputs
- **Algorithm** — a particular procedure that describes **how** to compute an output for each possible input

Algorithm is a correct solution of a given problem if it halts for all inputs and for all inputs it produces a correct output.

Example:

Problem: The problem of sorting
Algorithm: Quicksort

Correctness of Algorithms

Example:

The problem of finding a maximal element in an array:

Input: An array A indexed from zero and a number n representing the number of elements in array A. It is assumed that $n \geq 1$.

Output: A value *result* of a maximal element in the array A, i.e., the value *result* such that:

- $A[j] \leq result$ for all $j \in \mathbb{N}$, where $0 \leq j < n$, and
- there exists $j \in \mathbb{N}$ such that $0 \leq j < n$ and $A[j] = result$.

An **instance** of a problem — concreate input data, e.g.,

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$

The output for this instance is value 11.

Correctness of Algorithms

Algorithm: An algorithm for finding the maximal element in an array

```
FIND-MAX (A, n):
    k := 0
    for i := 1 to n − 1 do
        if A[i] > A[k] then
            k := i
    return A[k]
```

Correctness of Algorithms

Definition

An algorithm Alg **solves** a given problem P, if for **each** instance w of problem P, the following conditions are satisfied:

(a) The computation of algorithm Alg on input w halts after finite number of steps.

(b) Algorithm Alg generates a correct output for input w according to conditions in problem P.

An algorithm that solves problem P is a correct solution of this problem.

Correctness of Algorithms

Algorithm *Alg* is **not** a correct solution of problem *P* if there exists an input *w* such that in the computation on this input, one of the following incorrect behaviours occurs:

- some incorrect illegal operation is performed (an access to an element of an array with index out of bounds, division by zero, ...),
- the generated output does not satisfy the conditions specified in problem *P*,
- the computation never halts.

Testing — running the algorithm with different inputs and checking whether the algorithm behaves correctly on these inputs.

Testing can be used to show the presence of bugs but not to show that algorithm behaves correctly for **all** inputs.

Correctness of Algorithms

Typically, the set of possible instances of a given problem is infinite (or at least very big), so it is not possible to test the behaviour of the algorithm for all instances.

As a justification and a verificatoin of the fact that an algorithm is a correct solution of a given problem, we need to have a **proof** that takes into account all possible computations on all possible inputs.

Generally, it is reasonable to divide a proof of correctness of an algorithm into two parts:

- Showing that the algorithm never does anything "wrong" for any input:
  - no illegal operation is performed during a computation
  - if the program halts, the generated output will be "correct"
- Showing that for every input the algorithm halts after a finite number of steps.

Invariants

Consider an arbitrary system consisting of:

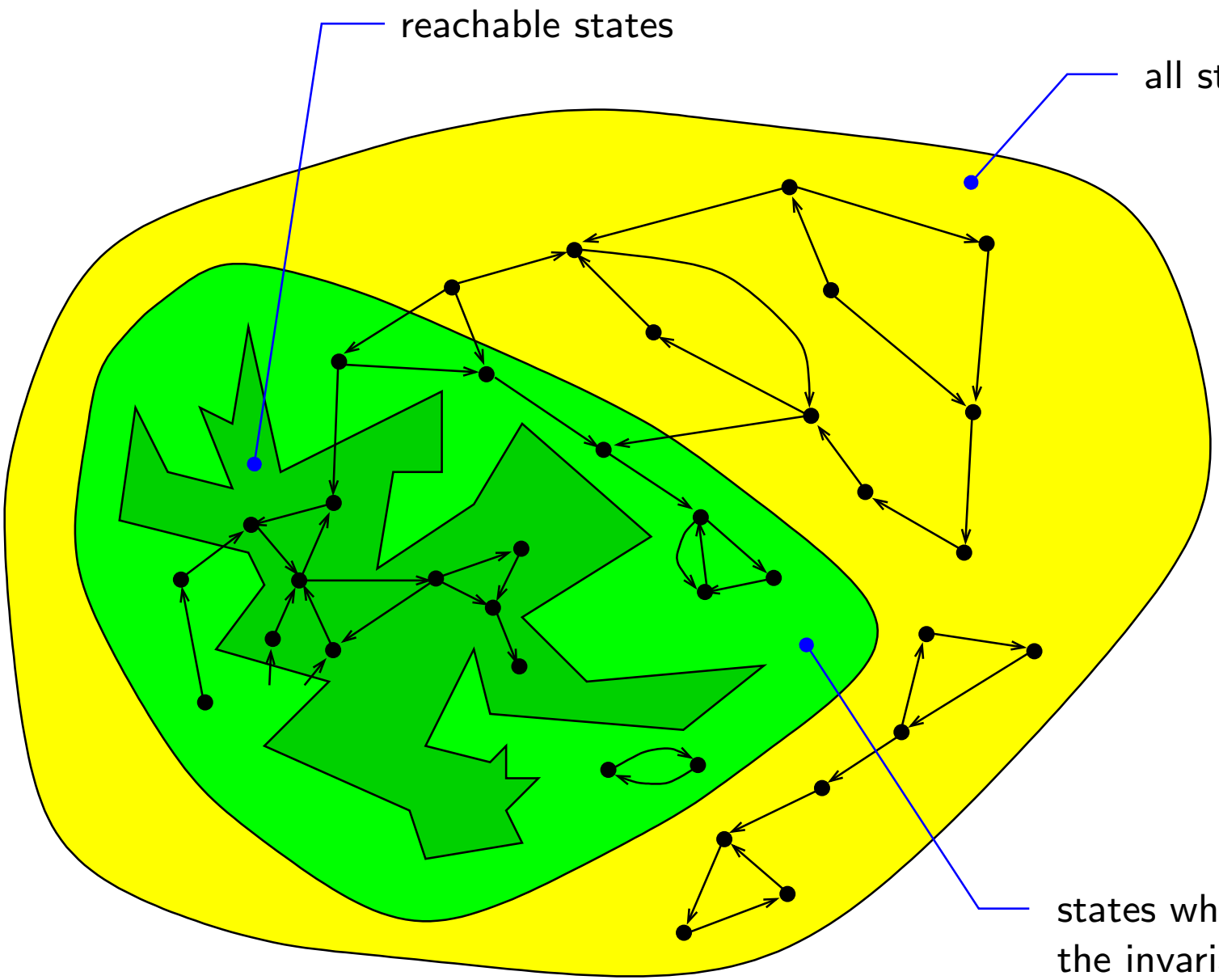
- a set of states (or configurations) — it can be infinite
- transitions between these states
- some states are specified as initial

A state is **reachable** if it is possible to reach it from some initial state using a sequence of transitions.

An **invariant** is a condition determining a subset of states such that all reachable states satisfy these condition:

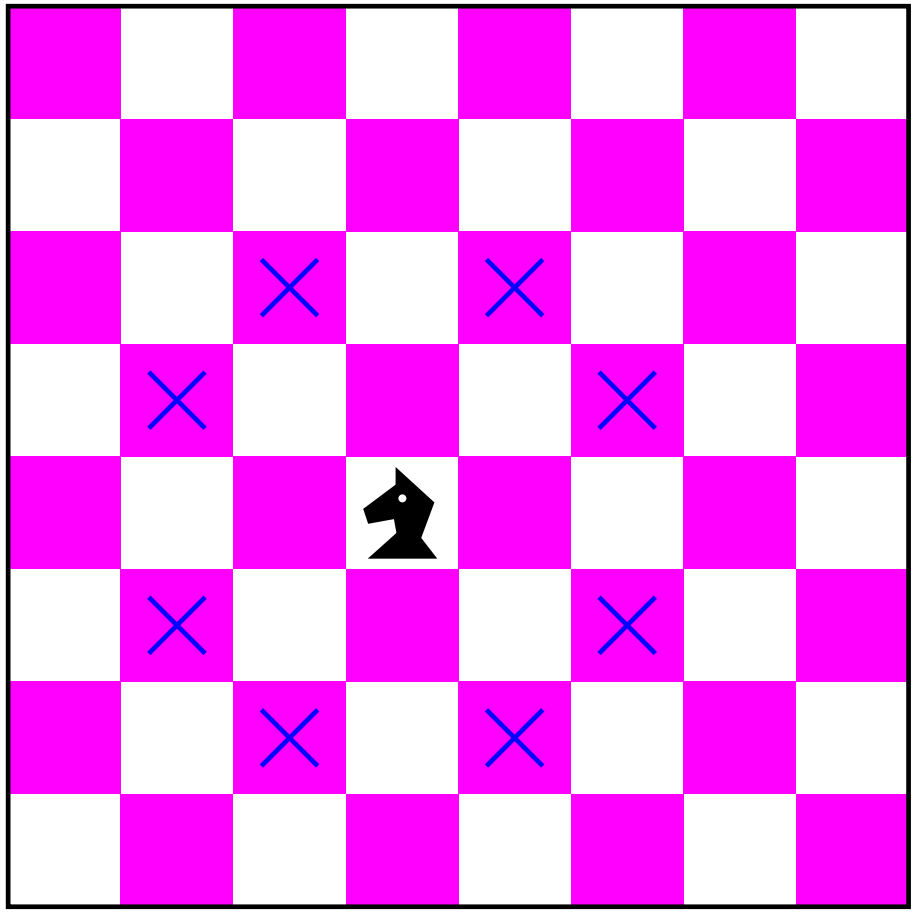
- it is satisfied in all initial states
- if it is satisfied in a state and there is a transition from this state, by which the system goes to another state in one step, then this condition will be satisfied also in this other state

Invariants


reachable states
all states
states where
the invariant holds

Invariants

Example: We will move with a knight (a chess piece) on a chessboard and at the same time we will count the number of the moves performed; the knight starts on some white square in the leftmost column:



Invariants

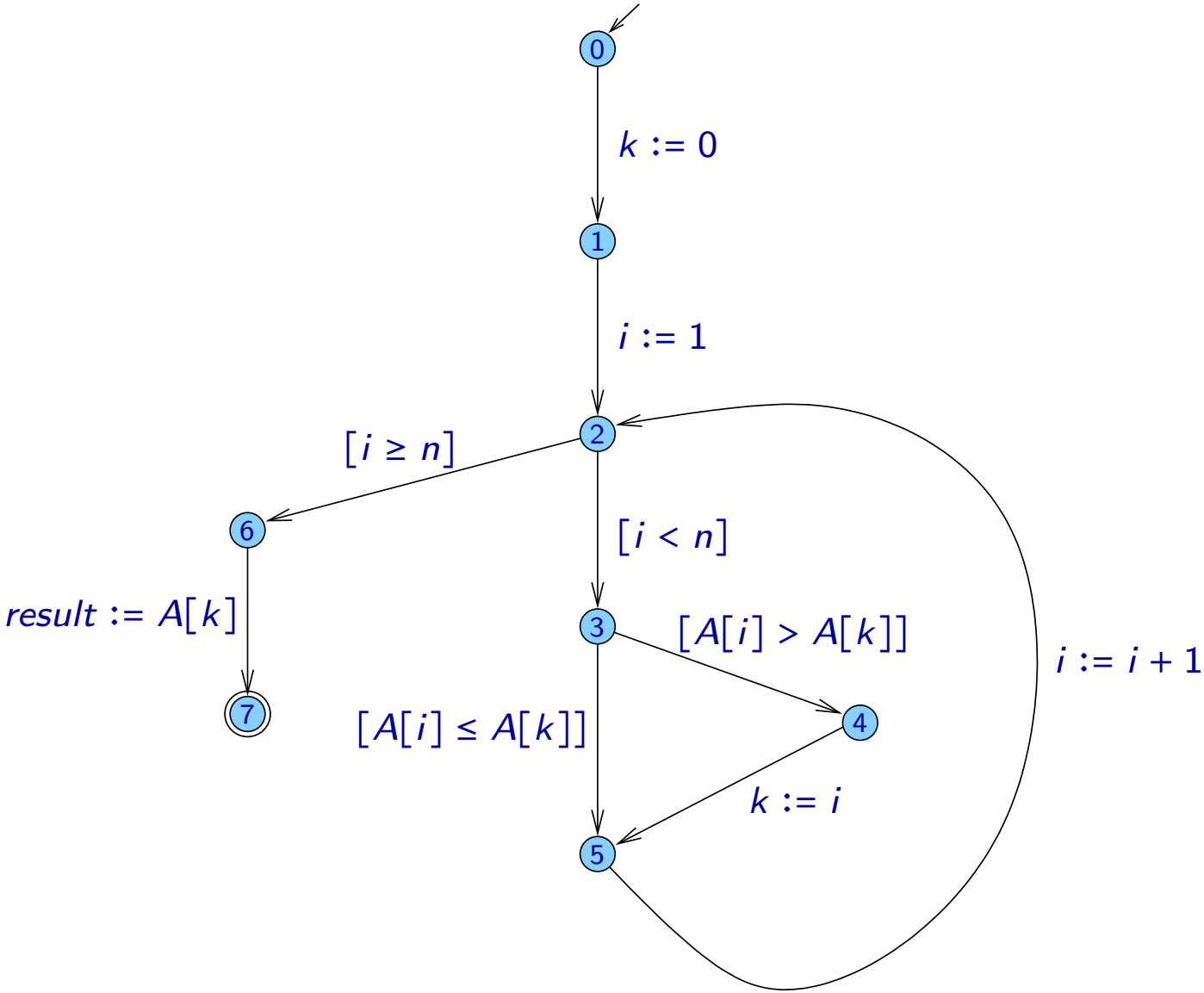
- **States** — pairs consisting of a current position of the knight on the chessboard and a value of the counter giving the number of moves performed so far
- **Transitions** — making one move with the knight (according to the rules of chess) and incrementing the counter by one
- **Initial states** — the knight is on a white square in the leftmost column and the value of the counter is 0

For example, the following invarint holds:

- if the value of the counter is even, the knight is on a white square
- if the value of the counter is odd, the knight is on a black square

Invariants

Example: Algorithm FIND-MAX represented as a control-flow graph



Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

α_0: $(0, \langle A\text{: } [3, 8, 1, 3, 6], \quad n\text{: } 5, \quad i\text{: } ?, \quad k\text{: } ?, \quad \mathit{result}\text{: } ?\rangle)$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

α_0: $(0, \langle A\text{: } [3, 8, 1, 3, 6], \ n\text{: } 5, \ i\text{: } ?, \ k\text{: } ?, \ \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3, 8, 1, 3, 6], \ n\text{: } 5, \ i\text{: } ?, \ k\text{: } 0, \ \mathit{result}\text{: } ?\rangle)$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } ?, \; \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } ?, \; \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } ?, \; \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } ?, \; \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle)$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } ?, \; \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 2, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle)$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } ?, \; \mathit{result}\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 2, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 2, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle)$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

α_0: $(0, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } ?, \; result\text{: } ?\rangle)$
α_1: $(1, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } 0, \; result\text{: } ?\rangle)$
α_2: $(2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; result\text{: } ?\rangle)$
α_3: $(3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; result\text{: } ?\rangle)$
α_4: $(4, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; result\text{: } ?\rangle)$
α_5: $(5, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 1, \; result\text{: } ?\rangle)$
α_6: $(2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 2, \; k\text{: } 1, \; result\text{: } ?\rangle)$
α_7: $(3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 2, \; k\text{: } 1, \; result\text{: } ?\rangle)$
α_8: $(5, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 2, \; k\text{: } 1, \; result\text{: } ?\rangle)$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

α_0: $(0, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon ?,\ result\colon ?\rangle)$
α_1: $(1, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon 0,\ result\colon ?\rangle)$
α_2: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_3: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_4: $(4, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ result\colon ?\rangle)$
α_5: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 1,\ result\colon ?\rangle)$
α_6: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_7: $(3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_8: $(5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ result\colon ?\rangle)$
α_9: $(2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ result\colon ?\rangle)$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

$$
\begin{array}{rl}
\alpha_0: & (0, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon ?, \; k\colon ?, \; \mathit{result}\colon ?\rangle) \\
\alpha_1: & (1, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon ?, \; k\colon 0, \; \mathit{result}\colon ?\rangle) \\
\alpha_2: & (2, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 1, \; k\colon 0, \; \mathit{result}\colon ?\rangle) \\
\alpha_3: & (3, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 1, \; k\colon 0, \; \mathit{result}\colon ?\rangle) \\
\alpha_4: & (4, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 1, \; k\colon 0, \; \mathit{result}\colon ?\rangle) \\
\alpha_5: & (5, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 1, \; k\colon 1, \; \mathit{result}\colon ?\rangle) \\
\alpha_6: & (2, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 2, \; k\colon 1, \; \mathit{result}\colon ?\rangle) \\
\alpha_7: & (3, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 2, \; k\colon 1, \; \mathit{result}\colon ?\rangle) \\
\alpha_8: & (5, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 2, \; k\colon 1, \; \mathit{result}\colon ?\rangle) \\
\alpha_9: & (2, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 3, \; k\colon 1, \; \mathit{result}\colon ?\rangle) \\
\alpha_{10}: & (3, \langle A\colon [3,8,1,3,6], \; n\colon 5, \; i\colon 3, \; k\colon 1, \; \mathit{result}\colon ?\rangle)
\end{array}
$$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

$$
\begin{array}{rl}
\alpha_0: & (0, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } ?, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_1: & (1, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_2: & (2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_3: & (3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_4: & (4, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_5: & (5, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_6: & (2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 2, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_7: & (3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 2, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_8: & (5, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 2, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_9: & (2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 3, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_{10}: & (3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 3, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_{11}: & (5, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 3, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle)
\end{array}
$$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

$$
\begin{array}{rl}
\alpha_0: & (0, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } ?, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_1: & (1, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } ?, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_2: & (2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_3: & (3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_4: & (4, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 0, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_5: & (5, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 1, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_6: & (2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 2, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_7: & (3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 2, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_8: & (5, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 2, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_9: & (2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 3, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_{10}: & (3, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 3, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_{11}: & (5, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 3, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle) \\
\alpha_{12}: & (2, \langle A\text{: } [3,8,1,3,6], \; n\text{: } 5, \; i\text{: } 4, \; k\text{: } 1, \; \mathit{result}\text{: } ?\rangle)
\end{array}
$$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

$$
\begin{array}{rl}
\alpha_0: & (0, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon ?,\ \mathit{result}\colon ?\rangle) \\
\alpha_1: & (1, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon 0,\ \mathit{result}\colon ?\rangle) \\
\alpha_2: & (2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ \mathit{result}\colon ?\rangle) \\
\alpha_3: & (3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ \mathit{result}\colon ?\rangle) \\
\alpha_4: & (4, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ \mathit{result}\colon ?\rangle) \\
\alpha_5: & (5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_6: & (2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_7: & (3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_8: & (5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_9: & (2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{10}: & (3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{11}: & (5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{12}: & (2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{13}: & (3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ \mathit{result}\colon ?\rangle)
\end{array}
$$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

$$
\begin{array}{rl}
\alpha_0: & (0, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } ?, \ k\text{: } ?, \ \mathit{result}\text{: } ?\rangle) \\
\alpha_1: & (1, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } ?, \ k\text{: } 0, \ \mathit{result}\text{: } ?\rangle) \\
\alpha_2: & (2, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ \mathit{result}\text{: } ?\rangle) \\
\alpha_3: & (3, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ \mathit{result}\text{: } ?\rangle) \\
\alpha_4: & (4, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 0, \ \mathit{result}\text{: } ?\rangle) \\
\alpha_5: & (5, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 1, \ k\text{: } 1, \ \mathit{result}\text{: } ?\rangle) \\
\alpha_6: & (2, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 2, \ k\text{: } 1, \ \mathit{result}\text{: } ?\rangle) \\
\alpha_7: & (3, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 2, \ k\text{: } 1, \ \mathit{result}\text{: } ?\rangle) \\
\alpha_8: & (5, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 2, \ k\text{: } 1, \ \mathit{result}\text{: } ?\rangle) \\
\alpha_9: & (2, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 3, \ k\text{: } 1, \ \mathit{result}\text{: } ?\rangle) \\
\alpha_{10}: & (3, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 3, \ k\text{: } 1, \ \mathit{result}\text{: } ?\rangle) \\
\alpha_{11}: & (5, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 3, \ k\text{: } 1, \ \mathit{result}\text{: } ?\rangle) \\
\alpha_{12}: & (2, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 4, \ k\text{: } 1, \ \mathit{result}\text{: } ?\rangle) \\
\alpha_{13}: & (3, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 4, \ k\text{: } 1, \ \mathit{result}\text{: } ?\rangle) \\
\alpha_{14}: & (5, \langle A\text{: } [3,8,1,3,6], \ n\text{: } 5, \ i\text{: } 4, \ k\text{: } 1, \ \mathit{result}\text{: } ?\rangle)
\end{array}
$$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

$$
\begin{array}{rl}
\alpha_0: & (0, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } ?,\ \mathit{result}\text{: } ?\rangle) \\
\alpha_1: & (1, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } ?,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle) \\
\alpha_2: & (2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle) \\
\alpha_3: & (3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle) \\
\alpha_4: & (4, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 0,\ \mathit{result}\text{: } ?\rangle) \\
\alpha_5: & (5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 1,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle) \\
\alpha_6: & (2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle) \\
\alpha_7: & (3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle) \\
\alpha_8: & (5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 2,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle) \\
\alpha_9: & (2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle) \\
\alpha_{10}: & (3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle) \\
\alpha_{11}: & (5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 3,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle) \\
\alpha_{12}: & (2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle) \\
\alpha_{13}: & (3, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle) \\
\alpha_{14}: & (5, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 4,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle) \\
\alpha_{15}: & (2, \langle A\text{: } [3,8,1,3,6],\ n\text{: } 5,\ i\text{: } 5,\ k\text{: } 1,\ \mathit{result}\text{: } ?\rangle)
\end{array}
$$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

$$
\begin{array}{rlllllll}
\alpha_0: & (0, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon ?, & k\colon ?, & \mathit{result}\colon ?\rangle) \\
\alpha_1: & (1, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon ?, & k\colon 0, & \mathit{result}\colon ?\rangle) \\
\alpha_2: & (2, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon 1, & k\colon 0, & \mathit{result}\colon ?\rangle) \\
\alpha_3: & (3, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon 1, & k\colon 0, & \mathit{result}\colon ?\rangle) \\
\alpha_4: & (4, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon 1, & k\colon 0, & \mathit{result}\colon ?\rangle) \\
\alpha_5: & (5, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon 1, & k\colon 1, & \mathit{result}\colon ?\rangle) \\
\alpha_6: & (2, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon 2, & k\colon 1, & \mathit{result}\colon ?\rangle) \\
\alpha_7: & (3, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon 2, & k\colon 1, & \mathit{result}\colon ?\rangle) \\
\alpha_8: & (5, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon 2, & k\colon 1, & \mathit{result}\colon ?\rangle) \\
\alpha_9: & (2, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon 3, & k\colon 1, & \mathit{result}\colon ?\rangle) \\
\alpha_{10}: & (3, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon 3, & k\colon 1, & \mathit{result}\colon ?\rangle) \\
\alpha_{11}: & (5, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon 3, & k\colon 1, & \mathit{result}\colon ?\rangle) \\
\alpha_{12}: & (2, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon 4, & k\colon 1, & \mathit{result}\colon ?\rangle) \\
\alpha_{13}: & (3, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon 4, & k\colon 1, & \mathit{result}\colon ?\rangle) \\
\alpha_{14}: & (5, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon 4, & k\colon 1, & \mathit{result}\colon ?\rangle) \\
\alpha_{15}: & (2, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon 5, & k\colon 1, & \mathit{result}\colon ?\rangle) \\
\alpha_{16}: & (6, & \langle A\colon [3,8,1,3,6], & n\colon 5, & i\colon 5, & k\colon 1, & \mathit{result}\colon ?\rangle)
\end{array}
$$

Invariants

A computation for input $A = [3, 8, 1, 3, 6]$ and $n = 5$ as a sequence of configurations:

$$
\begin{array}{rl}
\alpha_0: & (0, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon ?,\ \mathit{result}\colon ?\rangle) \\
\alpha_1: & (1, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon ?,\ k\colon 0,\ \mathit{result}\colon ?\rangle) \\
\alpha_2: & (2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ \mathit{result}\colon ?\rangle) \\
\alpha_3: & (3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ \mathit{result}\colon ?\rangle) \\
\alpha_4: & (4, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 0,\ \mathit{result}\colon ?\rangle) \\
\alpha_5: & (5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 1,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_6: & (2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_7: & (3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_8: & (5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 2,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_9: & (2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{10}: & (3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{11}: & (5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 3,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{12}: & (2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{13}: & (3, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{14}: & (5, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 4,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{15}: & (2, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 5,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{16}: & (6, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 5,\ k\colon 1,\ \mathit{result}\colon ?\rangle) \\
\alpha_{17}: & (7, \langle A\colon [3,8,1,3,6],\ n\colon 5,\ i\colon 5,\ k\colon 1,\ \mathit{result}\colon 8\rangle)
\end{array}
$$

Invariants

- **States** — configurations consisting of a state of the control unit and a content of the memory represented by values of all variables.
- **Transitions** — they are determined by instructions on the edges of the control-flow graph, they change both the control state and the content of the memory by assigning values to variables
- **Initial states** — all possible initial configurations for all possible input instances that are allowed according to a specification of the problem

Invariants will be propositions referring to configurations, i.e., they talk about states of the control unit and values of the variables

- If the control state is 2, then, in the given configuration, it holds that $1 \leq i \leq n$, $0 \leq k < i$, and $A[k]$ is the greatest of the elements $A[0], A[1], \ldots, A[i-1]$.

Invariants

For those systems, where configurations contain a control state, it can be convenient to state invariants in the form:

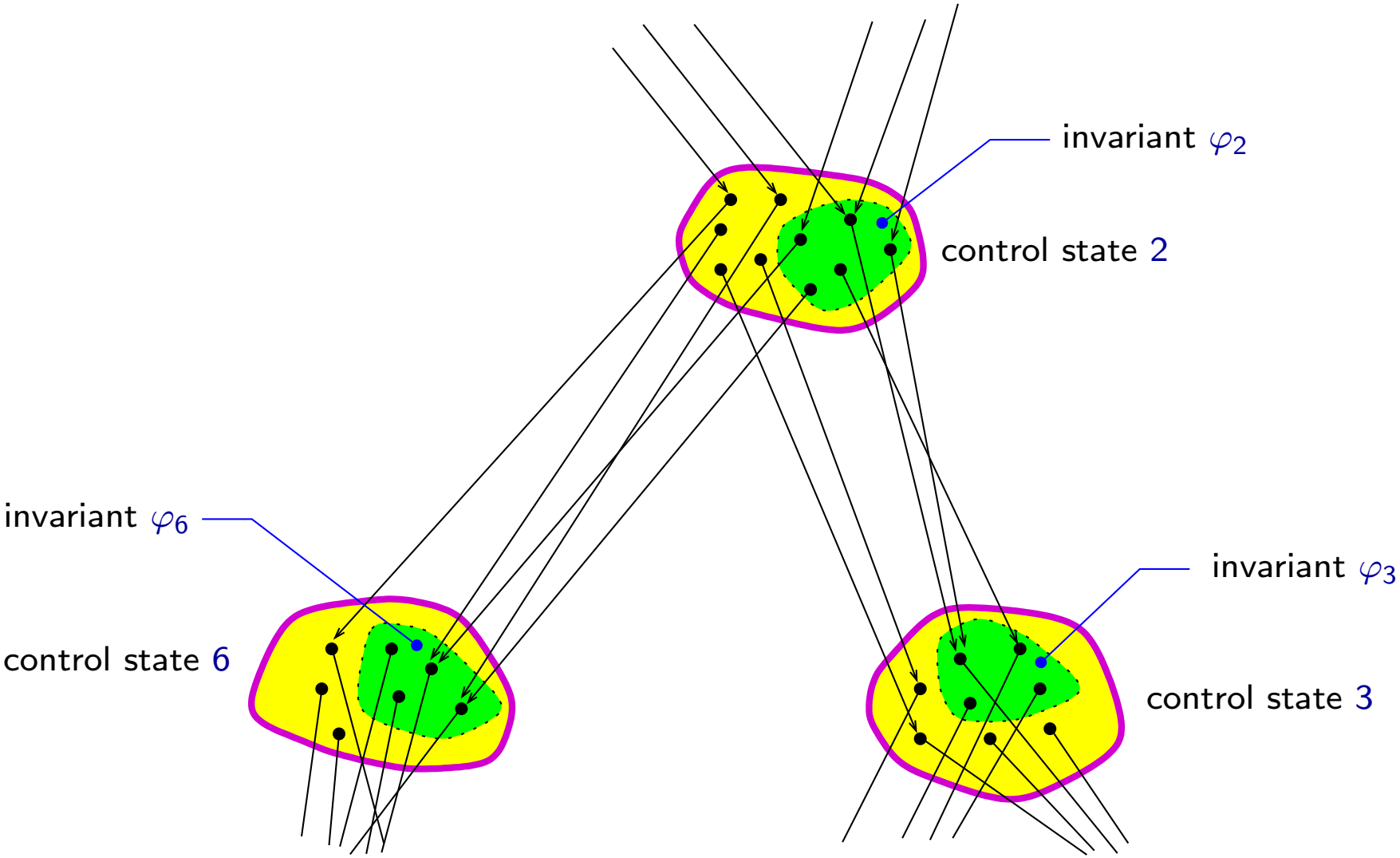
- if the control state is 0, then φ_0 holds
- if the control state is 1, then φ_1 holds

  $\vdots$
- if the control state is r, then φ_r holds

where the propositions $\varphi_0, \varphi_1, \ldots, \varphi_r$ refer only to the content of the memory, not to the control state.

Configurations can divided into (finitely many) groups according to the states of the control unit.

Invariants


invariant φ_2
control state 2
invariant φ_6
control state 6
invariant φ_3
control state 3

Invariants

Invariant — a condition that must be always satisfied in a given position in a code of the algorithm (i.e., in all possible computations for all allowed inputs) whenever the algorithm goes through this position.

Invariants can be written as formulas of predicate logic:

- **free** variables correspond to variables of the program
- a **valuation** is determined by values of program variables in a given configuration

Example: Formula

$$(1 \leq i) \wedge (i \leq n)$$

holds for example in a configuration where variable i has value 5 and variable n has value 14.

Invariants

Established invariants can be useful for many different purposes:

- They can help in better understanding the behaviour of the algorithm.
- They can be used to verify that ceirtain types of errors do not occur — e.g., an out of bounds array access, division by zero, ...

  We can verify that in those places in the code where such errors could potentially occur the invariants hold that ensure that the variables will always have values, for which the given error can not occur.

  Example: When element $A[i]$ will be accessed, it will always hold that $0 \leq i < n$, where n is the length of the array.
- An invariant that will hold in the final configurations will ensure that the output of the algorithm is correct with respect to the specification of the problem.
- In an analysis of the computational complexity, they could be useful in the examination how many times some instructions will be performed or how much memory is needed during the computation.

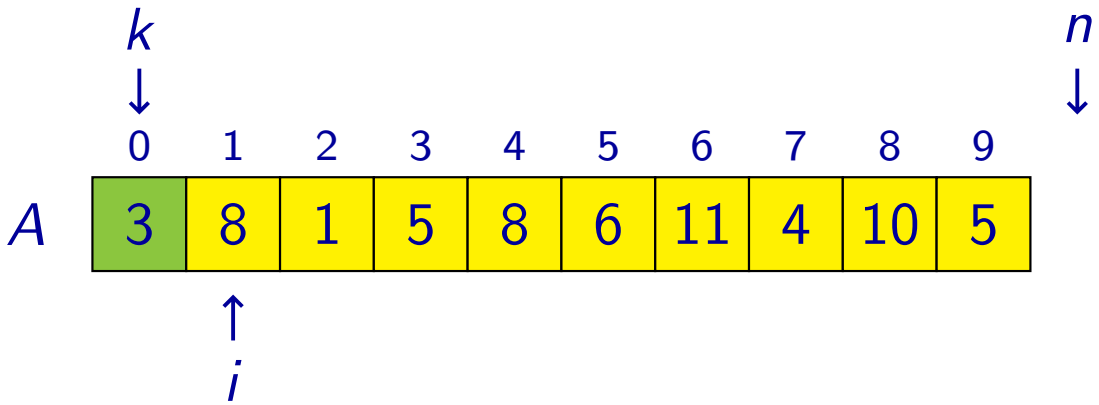
Invariants

Determining invariants is not a completely mechanical process. It requires a certain understanding of the behaviour of the algorithm.

Before formulating hypothesis what invariants hold in different control states, it can be useful to look at the behaviour of the algorithm on some particular concrete inputs.

Example: A computation of algorithm FIND-MAX for input

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



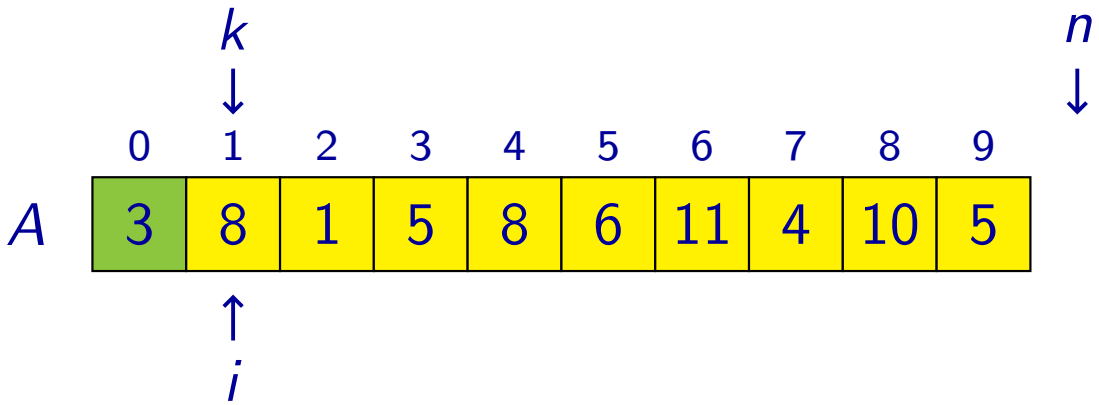
Invariants

Determining invariants is not a completely mechanical process. It requires a certain understanding of the behaviour of the algorithm.

Before formulating hypothesis what invariants hold in different control states, it can be useful to look at the behaviour of the algorithm on some particular concrete inputs.

Example: A computation of algorithm FIND-MAX for input

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



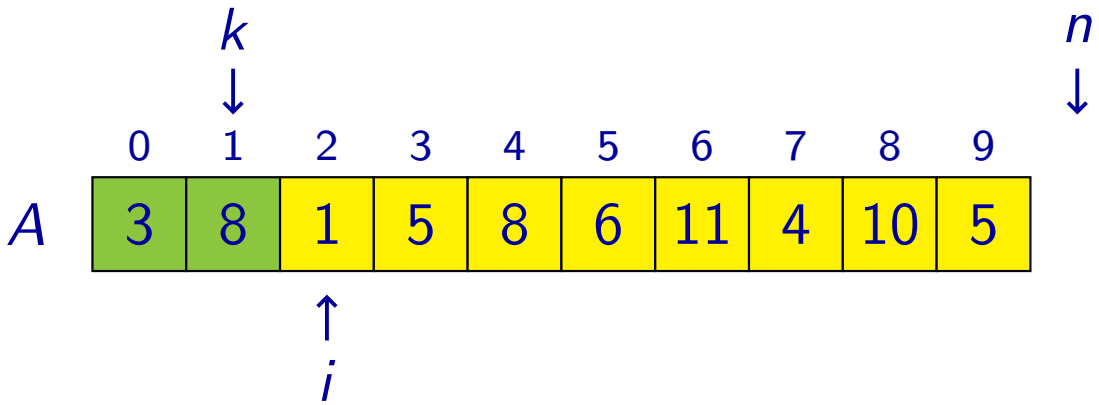
Invariants

Determining invariants is not a completely mechanical process. It requires a certain understanding of the behaviour of the algorithm.

Before formulating hypothesis what invariants hold in different control states, it can be useful to look at the behaviour of the algorithm on some particular concrete inputs.

Example: A computation of algorithm FIND-MAX for input

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



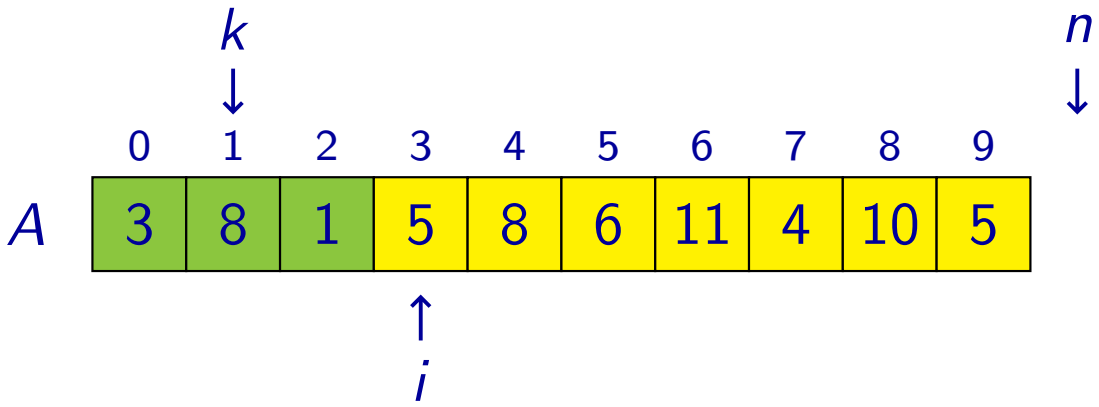
Invariants

Determining invariants is not a completely mechanical process. It requires a certain understanding of the behaviour of the algorithm.

Before formulating hypothesis what invariants hold in different control states, it can be useful to look at the behaviour of the algorithm on some particular concrete inputs.

Example: A computation of algorithm FIND-MAX for input

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



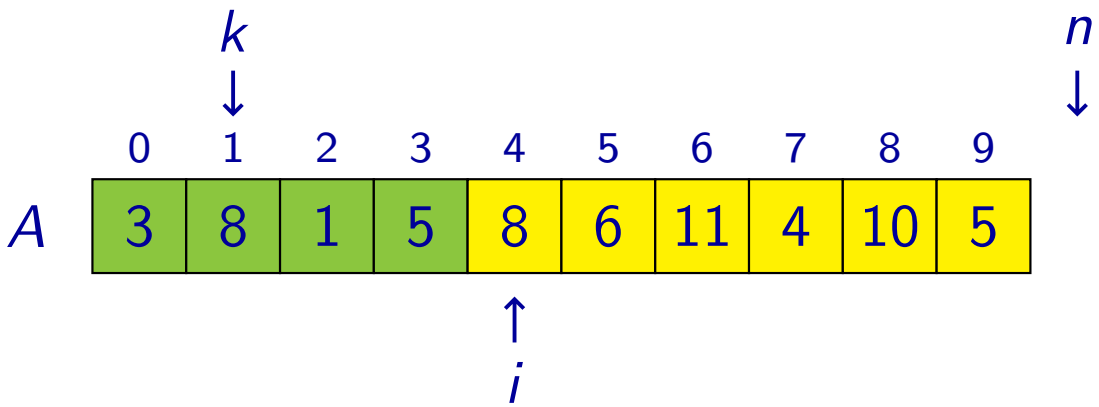
Invariants

Determining invariants is not a completely mechanical process. It requires a certain understanding of the behaviour of the algorithm.

Before formulating hypothesis what invariants hold in different control states, it can be useful to look at the behaviour of the algorithm on some particular concrete inputs.

Example: A computation of algorithm FIND-MAX for input

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



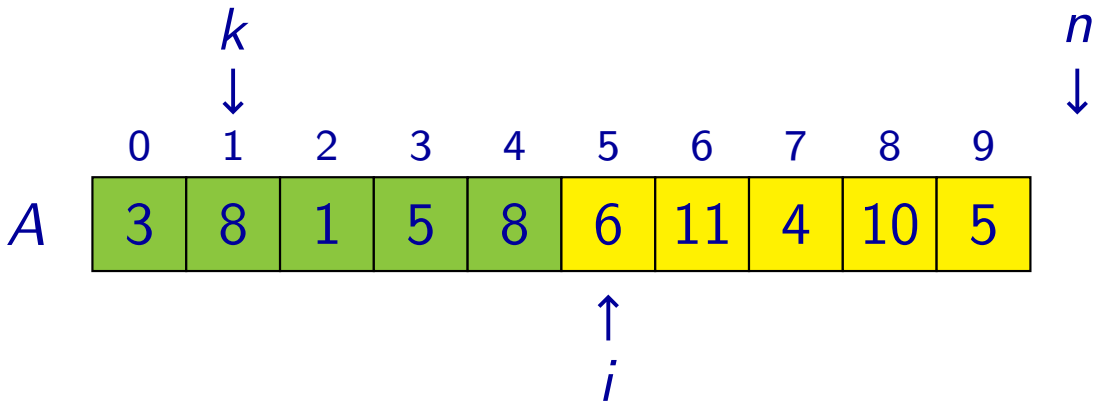
Invariants

Determining invariants is not a completely mechanical process. It requires a certain understanding of the behaviour of the algorithm.

Before formulating hypothesis what invariants hold in different control states, it can be useful to look at the behaviour of the algorithm on some particular concrete inputs.

Example: A computation of algorithm FIND-MAX for input

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



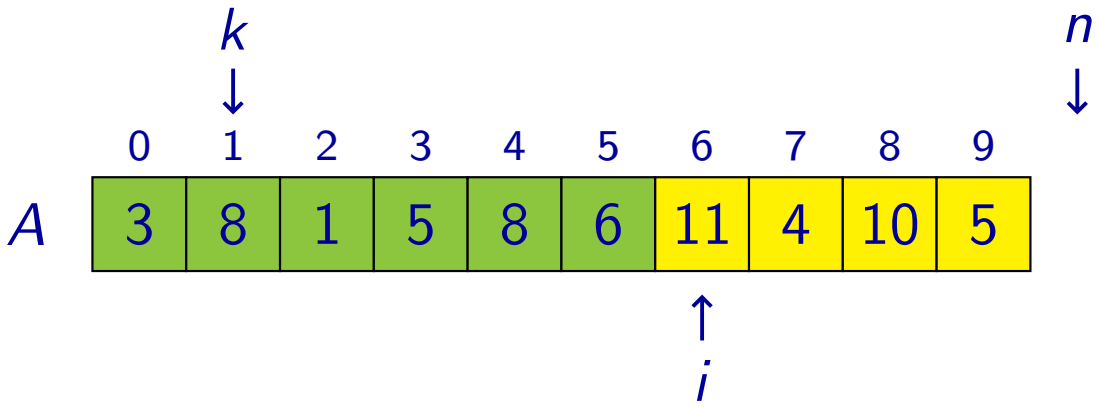
Invariants

Determining invariants is not a completely mechanical process. It requires a certain understanding of the behaviour of the algorithm.

Before formulating hypothesis what invariants hold in different control states, it can be useful to look at the behaviour of the algorithm on some particular concrete inputs.

Example: A computation of algorithm FIND-MAX for input

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



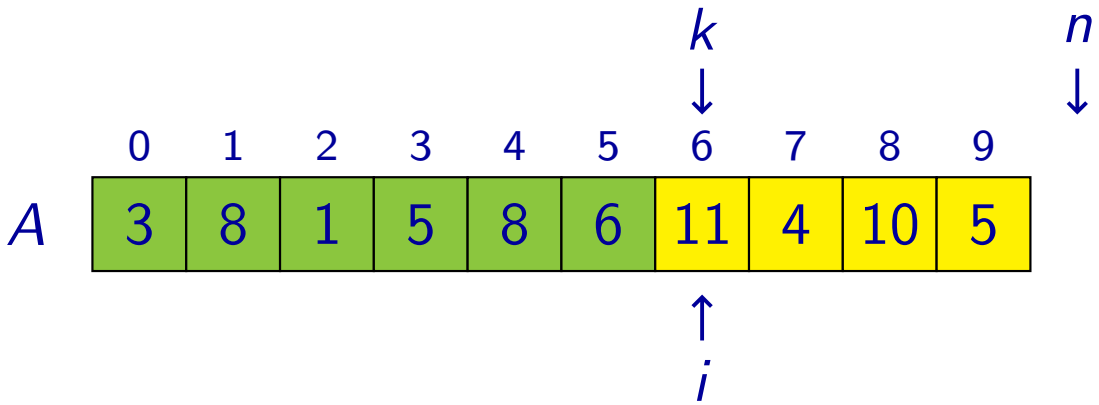
Invariants

Determining invariants is not a completely mechanical process. It requires a certain understanding of the behaviour of the algorithm.

Before formulating hypothesis what invariants hold in different control states, it can be useful to look at the behaviour of the algorithm on some particular concrete inputs.

Example: A computation of algorithm FIND-MAX for input

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



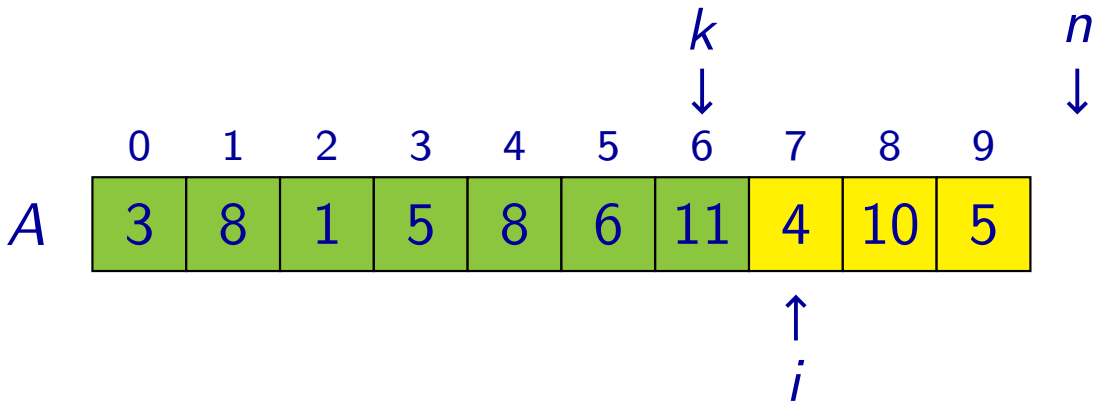
Invariants

Determining invariants is not a completely mechanical process. It requires a certain understanding of the behaviour of the algorithm.

Before formulating hypothesis what invariants hold in different control states, it can be useful to look at the behaviour of the algorithm on some particular concrete inputs.

Example: A computation of algorithm FIND-MAX for input

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



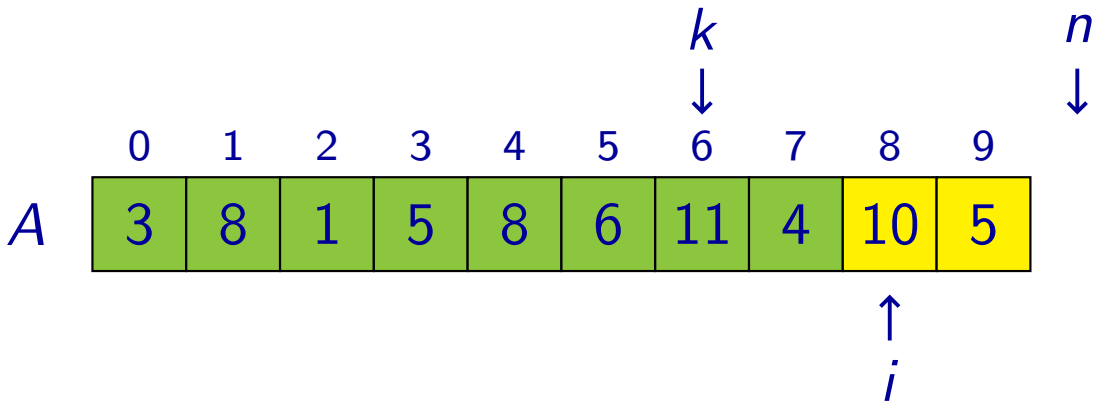
Invariants

Determining invariants is not a completely mechanical process. It requires a certain understanding of the behaviour of the algorithm.

Before formulating hypothesis what invariants hold in different control states, it can be useful to look at the behaviour of the algorithm on some particular concrete inputs.

Example: A computation of algorithm FIND-MAX for input

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



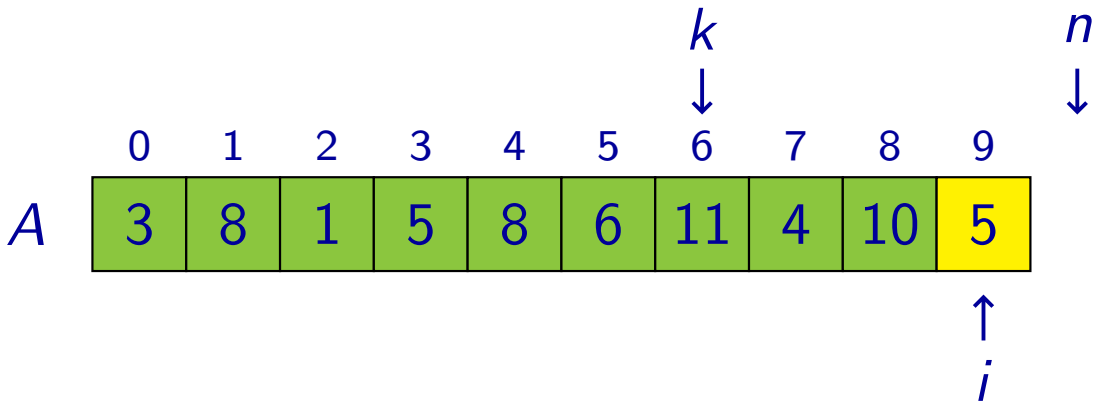
Invariants

Determining invariants is not a completely mechanical process. It requires a certain understanding of the behaviour of the algorithm.

Before formulating hypothesis what invariants hold in different control states, it can be useful to look at the behaviour of the algorithm on some particular concrete inputs.

Example: A computation of algorithm FIND-MAX for input

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



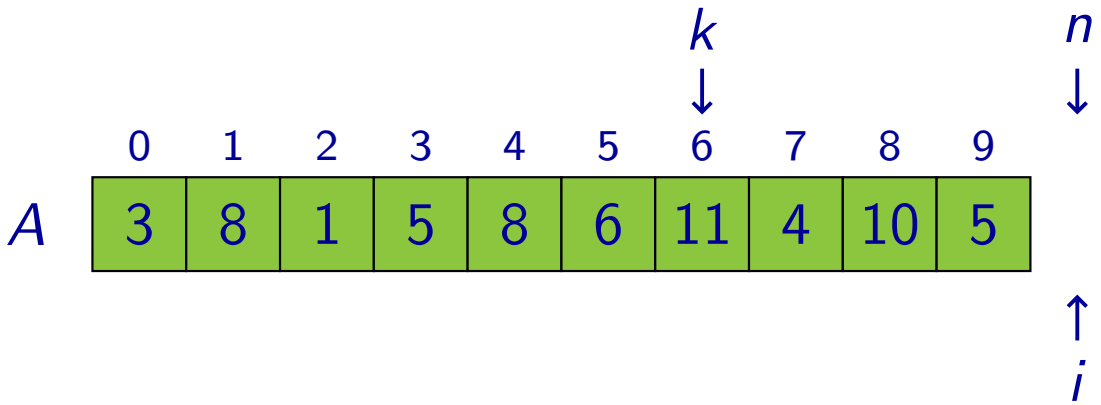
Invariants

Determining invariants is not a completely mechanical process. It requires a certain understanding of the behaviour of the algorithm.

Before formulating hypothesis what invariants hold in different control states, it can be useful to look at the behaviour of the algorithm on some particular concrete inputs.

Example: A computation of algorithm FIND-MAX for input

$$A = [\,3, 8, 1, 5, 8, 6, 11, 4, 10, 5\,],\ n = 10.$$



Invariants

Examples of invariants:

- an invariant in a control state q is represented by a formula φ_q

Invariants for individual control states (so far only hypotheses):

- φ_0: $(n \geq 1)$
- φ_1: $(n \geq 1) \wedge (k = 0)$
- φ_2: $(n \geq 1) \wedge (1 \leq i \leq n) \wedge (0 \leq k < i)$
- φ_3: $(n \geq 1) \wedge (1 \leq i < n) \wedge (0 \leq k < i)$
- φ_4: $(n \geq 1) \wedge (1 \leq i < n) \wedge (0 \leq k < i)$
- φ_5: $(n \geq 1) \wedge (1 \leq i < n) \wedge (0 \leq k \leq i)$
- φ_6: $(n \geq 1) \wedge (i = n) \wedge (0 \leq k < n)$
- φ_7: $(n \geq 1) \wedge (i = n) \wedge (0 \leq k < n)$

Invariants

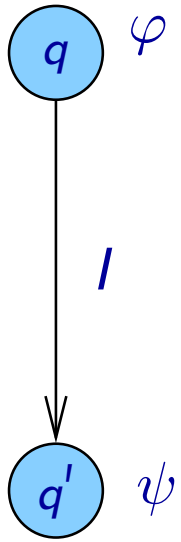
Checking that the given invariants really hold:

- We must check that the given invariants hold in the initial configurations — this is usually simple.
- It is necessary to check for each instruction of the algorithm that under the assumption that a specified invariant holds before an execution of the instruction, the other specified invariant holds after the execution of the instruction.

Let us assume the algorithm is represented as a control-flow graph:

- edges correspond to instructions
- consider an edge from state q to state q' labelled with instruction I
- let us say that (so far non-verified) invariants for states q and q' are expressed by formulas φ and ψ

Invariants



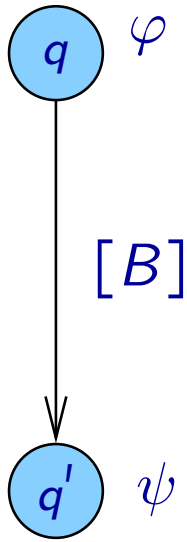
- for this edge we must check that for every configurations $\alpha = (q, mem)$ and $\alpha' = (q', mem')$ such that $\alpha \xrightarrow{l} \alpha'$, it holds that if
  - φ holds is configuration α,

  then
  - ψ holds in configuration α'

Invariants

Checking instructions, which are conditional tests:

- an edge labelled with a conditional test $[B]$

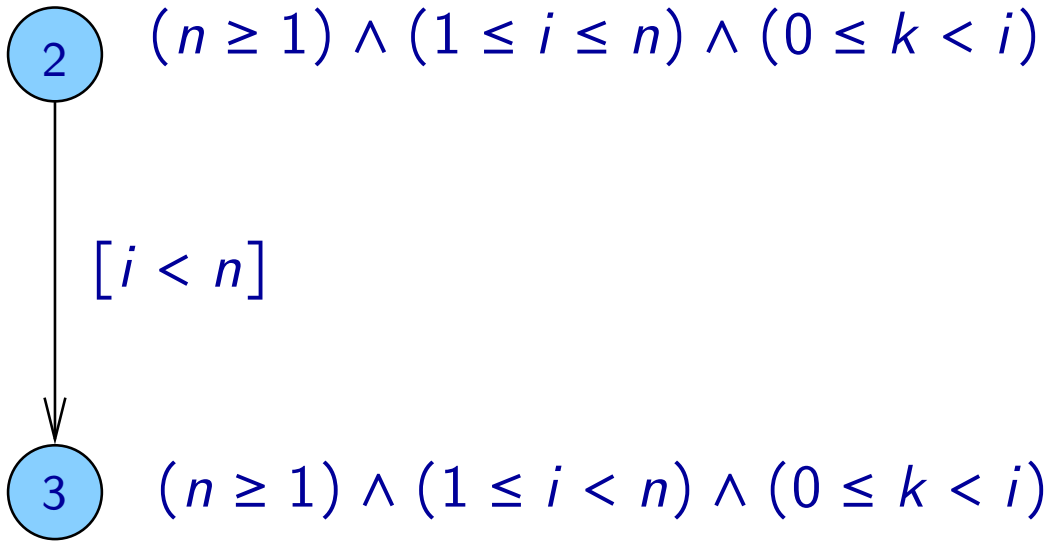


A content of memory is not modified, so it is sufficient to check that the following implication holds

$$(\varphi \wedge B) \Rightarrow \psi$$

Invariants

Example:

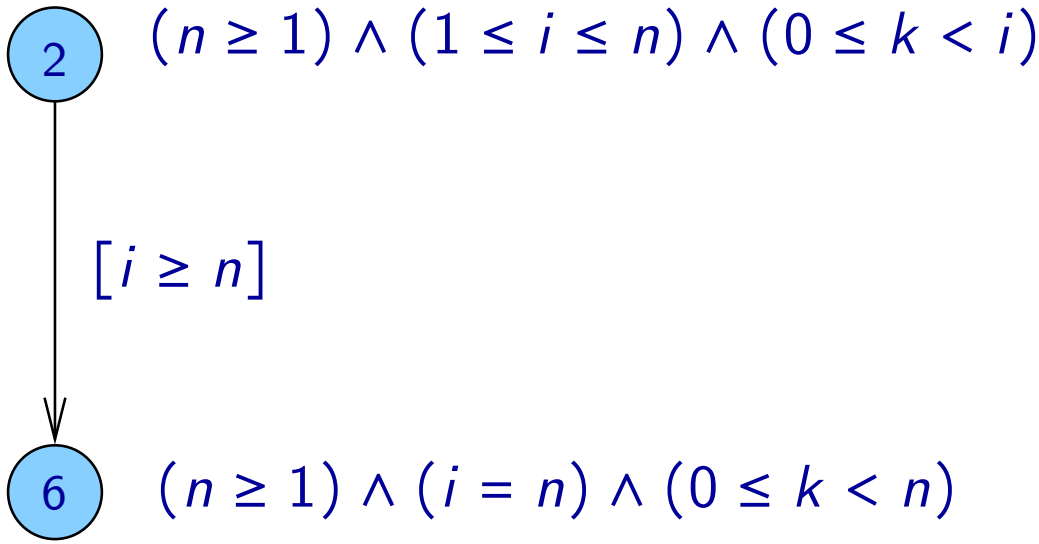


It is sufficient to check that the followng implication holds:

- If $(n \geq 1) \wedge (1 \leq i \leq n) \wedge (0 \leq k < i) \wedge (i < n)$,
  then $(n \geq 1) \wedge (1 \leq i < n) \wedge (0 \leq k < i)$.

Invariants

Example:



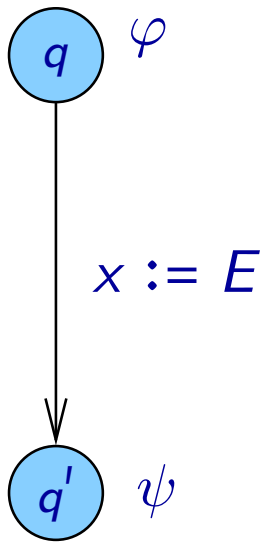
It is sufficient to check that the followng implication holds:

- If $(n \geq 1) \wedge (1 \leq i \leq n) \wedge (0 \leq k < i) \wedge (i \geq n)$,
  then $(n \geq 1) \wedge (i = n) \wedge (0 \leq k < n)$.

Invariants

Checking those instructions that assign values to variables (they modify a content of memory):

- an edge labelled with assignment $x := E$



We must distinguish between the values of variable x before this assignment and after this assignment.

Invariants

We will need the following operation of **substitution** on formulas:

$$\varphi[E/x]$$

denotes a formula obtained from variable φ when we substitute an expressiion E for all free occurrences of variable x in formula φ.

Example: Let us say that φ is formula $(1 \leq i) \wedge (i \leq n)$.

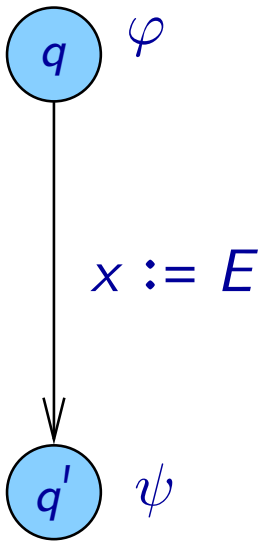
Notation $\varphi[i'/i]$ then denotes formula

$$(1 \leq i') \wedge (i' \leq n)$$

and notation $\varphi[(i+1)/i]$ denotes formula

$$(1 \leq i+1) \wedge (i+1 \leq n)$$

Invariants



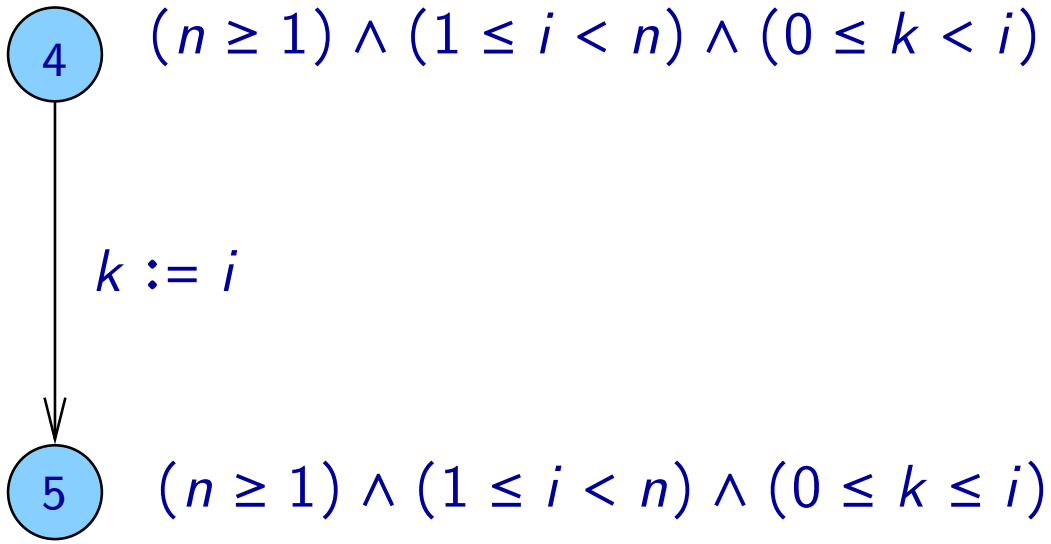
We will introduce a new variable x' representing the value of variable x after executing this assignment.

We need to check that the following implication holds:

$$(\varphi \wedge (x' = E)) \Rightarrow \psi[x'/x]$$

Invariants

Example:

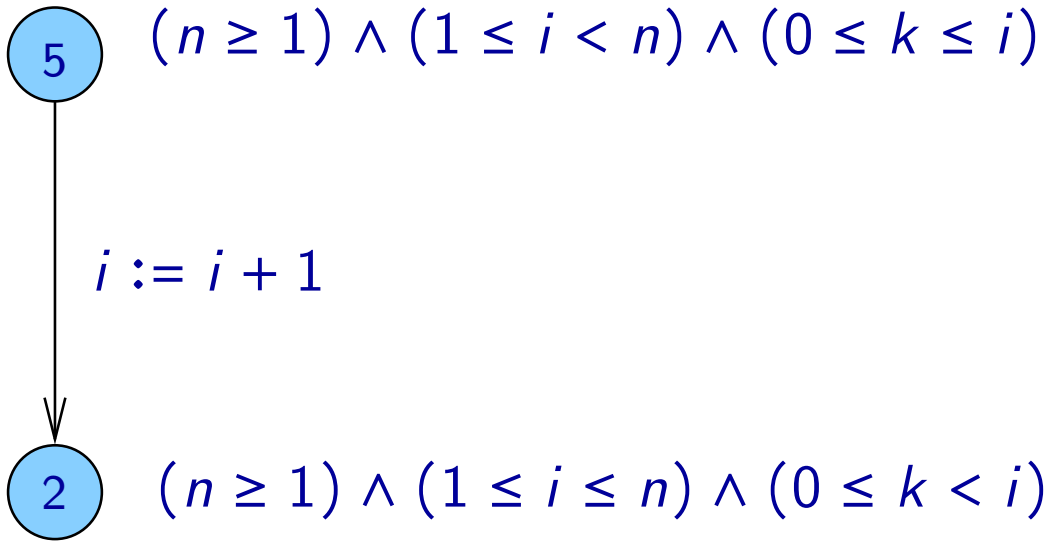


It is sufficient to check that the following implication holds:

- If $(n \geq 1) \wedge (1 \leq i < n) \wedge (0 \leq k < i) \wedge (k' = i)$, then $(n \geq 1) \wedge (1 \leq i < n) \wedge (0 \leq k' \leq i)$.

Invariants

Example:



It is sufficient to check that the following implication holds:

- If $(n \geq 1) \wedge (1 \leq i < n) \wedge (0 \leq k \leq i) \wedge (i' = i + 1)$, then $(n \geq 1) \wedge (1 \leq i' \leq n) \wedge (0 \leq k < i')$.

Invariants

Finishing the checking that the algorithm FIND-MAX returns a correct result (under assumption that it halts):

- ψ_0: φ_0
- ψ_1: $\varphi_1 \wedge (\forall j \in \mathbb{N})(0 \leq j < 1 \rightarrow A[j] \leq A[k])$
- ψ_2: $\varphi_2 \wedge (\forall j \in \mathbb{N})(0 \leq j < i \rightarrow A[j] \leq A[k])$
- ψ_3: $\varphi_3 \wedge (\forall j \in \mathbb{N})(0 \leq j < i \rightarrow A[j] \leq A[k])$
- ψ_4: $\varphi_4 \wedge (\forall j \in \mathbb{N})(0 \leq j < i \rightarrow A[j] \leq A[k]) \wedge (A[i] > A[k])$
- ψ_5: $\varphi_5 \wedge (\forall j \in \mathbb{N})(0 \leq j \leq i \rightarrow A[j] \leq A[k])$
- ψ_6: $\varphi_6 \wedge (\forall j \in \mathbb{N})(0 \leq j < n \rightarrow A[j] \leq A[k])$
- ψ_7: $\varphi_7 \wedge (\mathit{result} = A[k]) \wedge (\forall j \in \mathbb{N})(0 \leq j < n \rightarrow A[j] \leq \mathit{result}) \wedge (\exists j \in \mathbb{N})(0 \leq j < n \wedge A[j] = \mathit{result})$

Invariants

Usually it is not necessary to specify invariants in all control states but only in some "important" states — in particular, in states where the algorithm enters or leaves loops:

It is necessary to verify:

- That the invariant holds before entering the loop.
- That if the invariant holds before an iteration of the loop then it holds also after the iteration.
- That the invariant holds when the loop is left.

Invariants

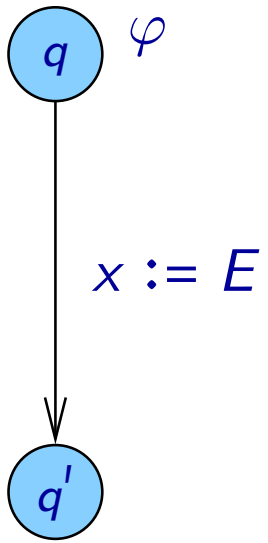
Example: In algorithm FIND-MAX, state 2 is such "important" state.

In state 2, the following holds:

- $n \geq 1$
- $1 \leq i \leq n$
- $0 \leq k < i$
- For each j such that $0 \leq j < i$ it holds that $A[j] \leq A[k]$.

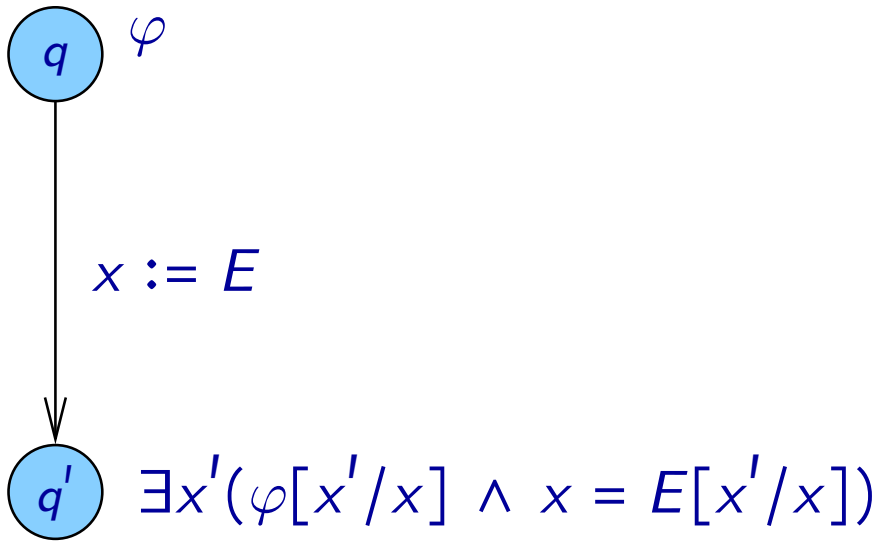
Invariants

Examples that show how invariants for some other states could be determined, if we already have determinants for some states:



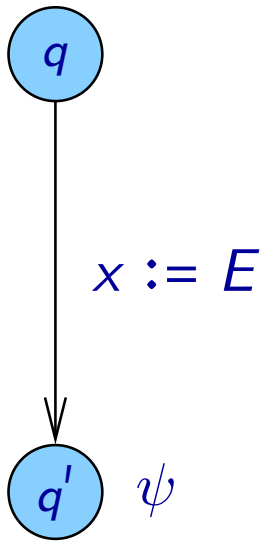
Invariants

Examples that show how invariants for some other states could be determined, if we already have determinants for some states:



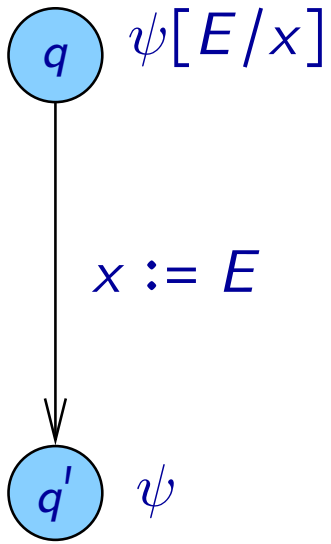
Invariants

Examples that show how invariants for some other states could be determined, if we already have determinants for some states:



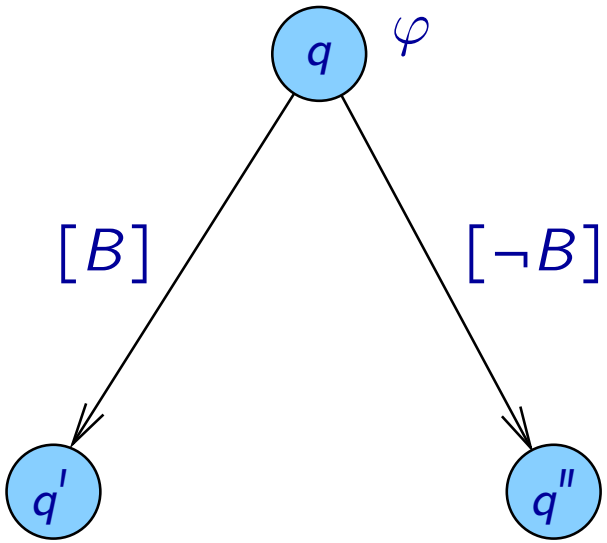
Invariants

Examples that show how invariants for some other states could be determined, if we already have determinants for some states:



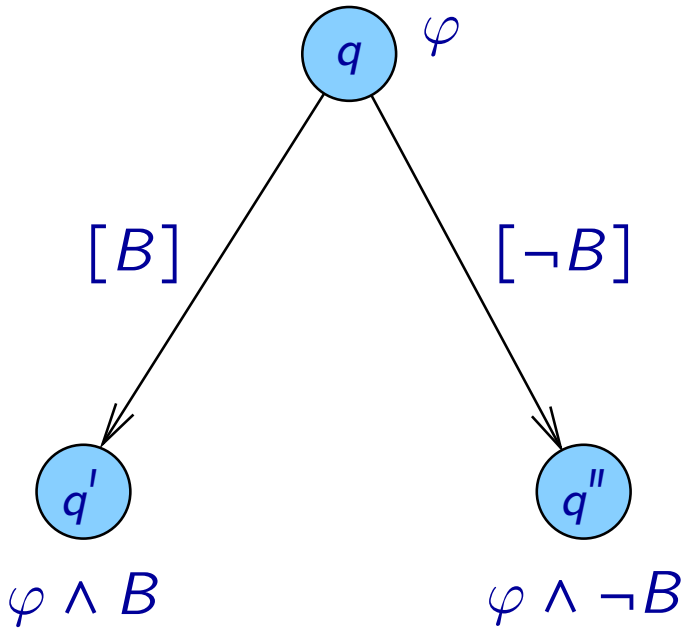
Invariants

Examples that show how invariants for some other states could be determined, if we already have determinants for some states:



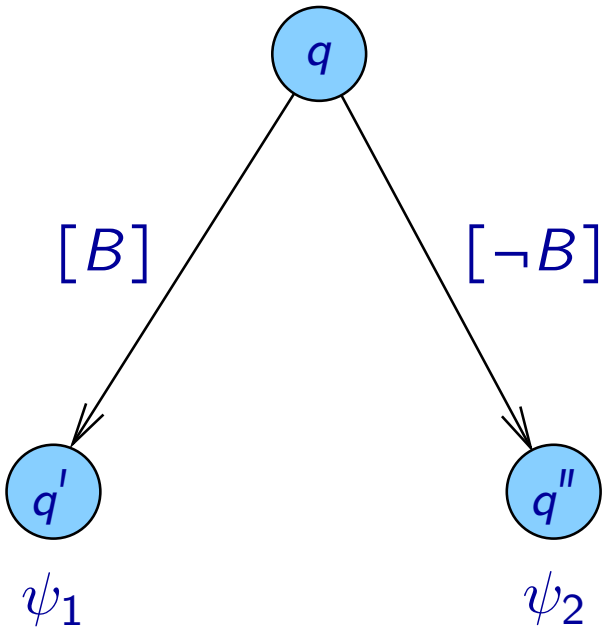
Invariants

Examples that show how invariants for some other states could be determined, if we already have determinants for some states:



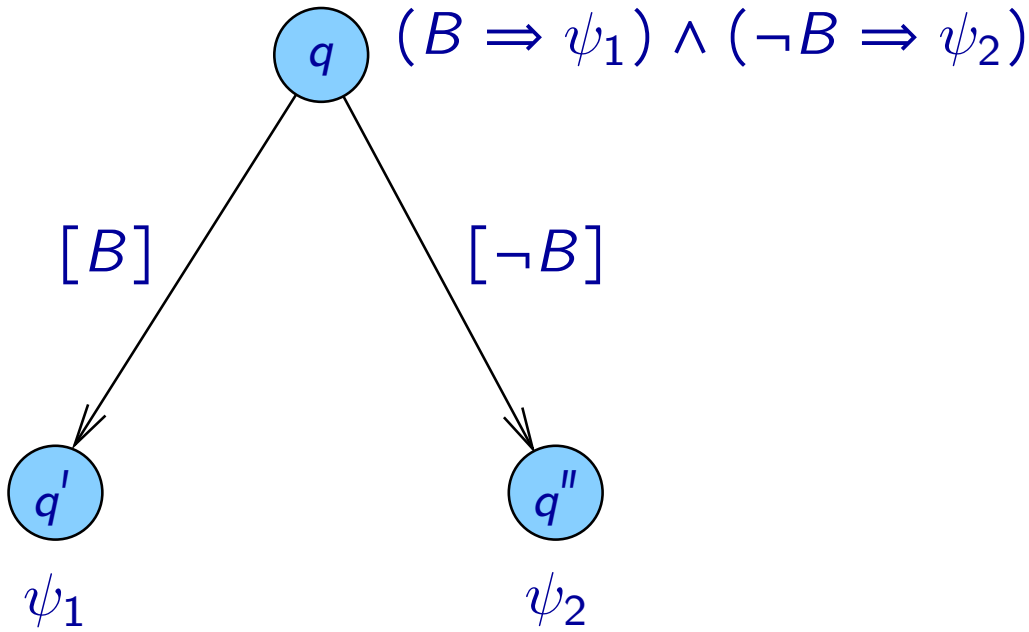
Invariants

Examples that show how invariants for some other states could be determined, if we already have determinants for some states:



Invariants

Examples that show how invariants for some other states could be determined, if we already have determinants for some states:



Invariants

Example:

Algorithm: Insertion sort

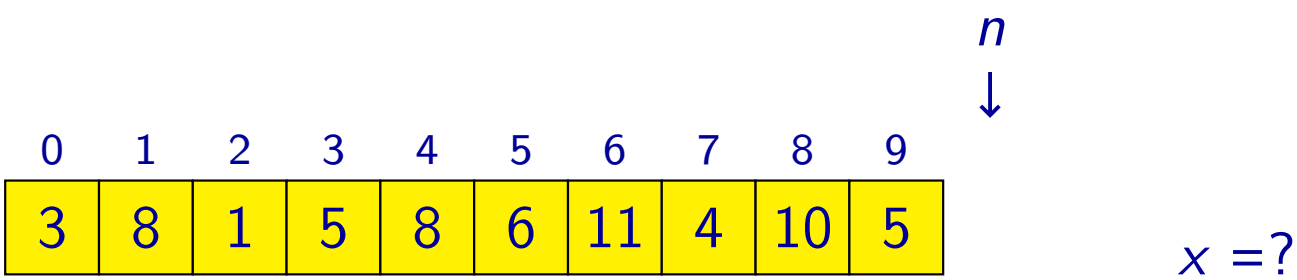
INSERTION-SORT (A, n):

```
for j := 1 to n − 1 do
    x := A[j]
    i := j − 1
    while i ≥ 0 and A[i] > x do
        A[i + 1] := A[i]
        i := i − 1
    A[i + 1] := x
```

Invariants

Example: A computation of algorithm INSERTION-SORT for input

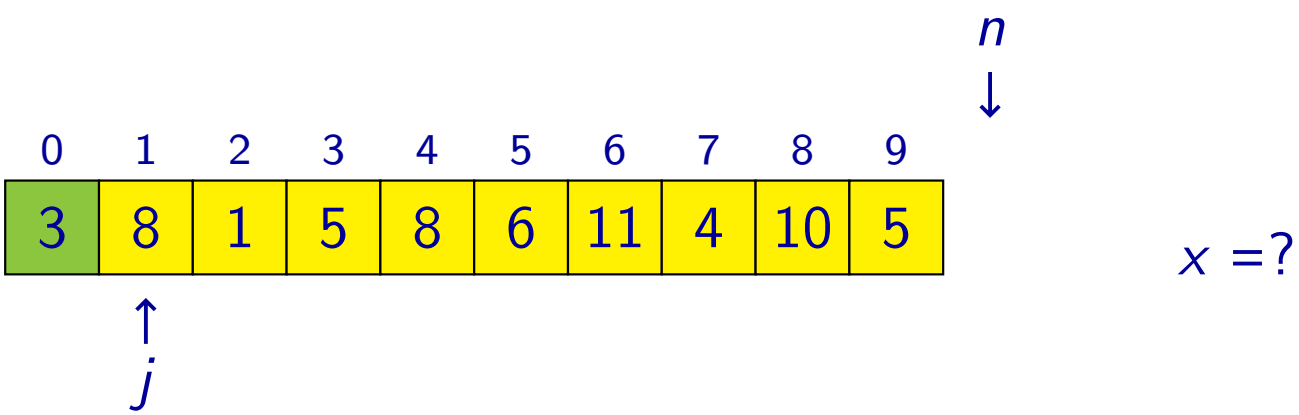
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

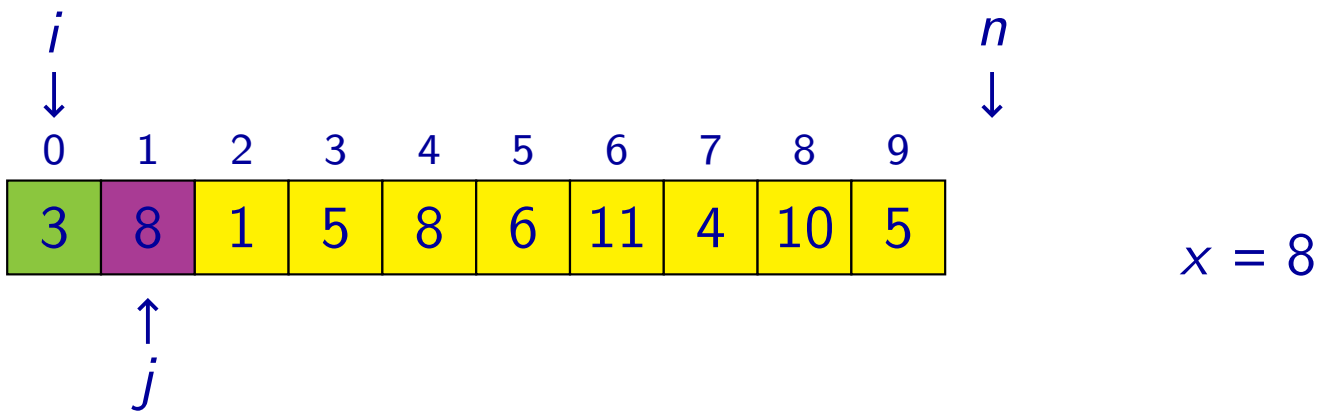
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

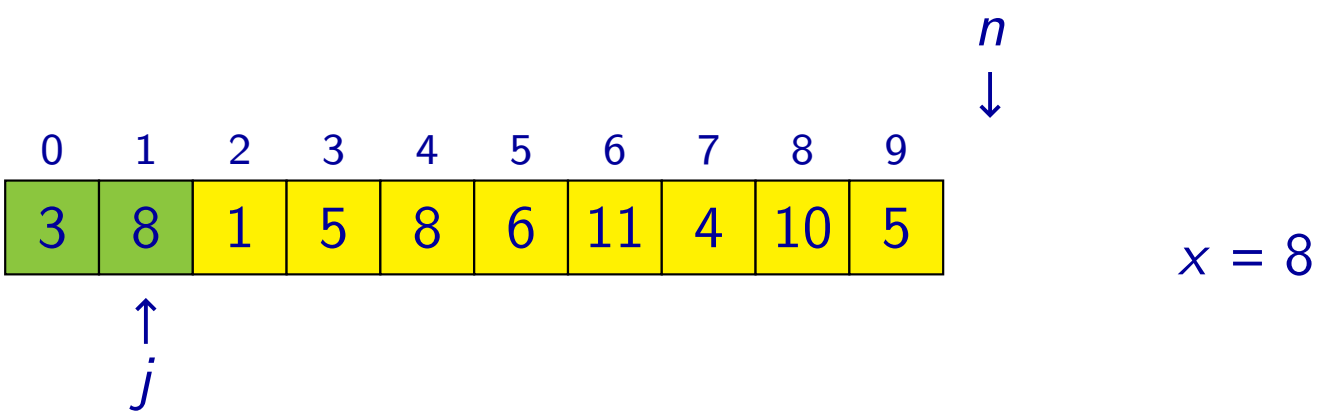
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

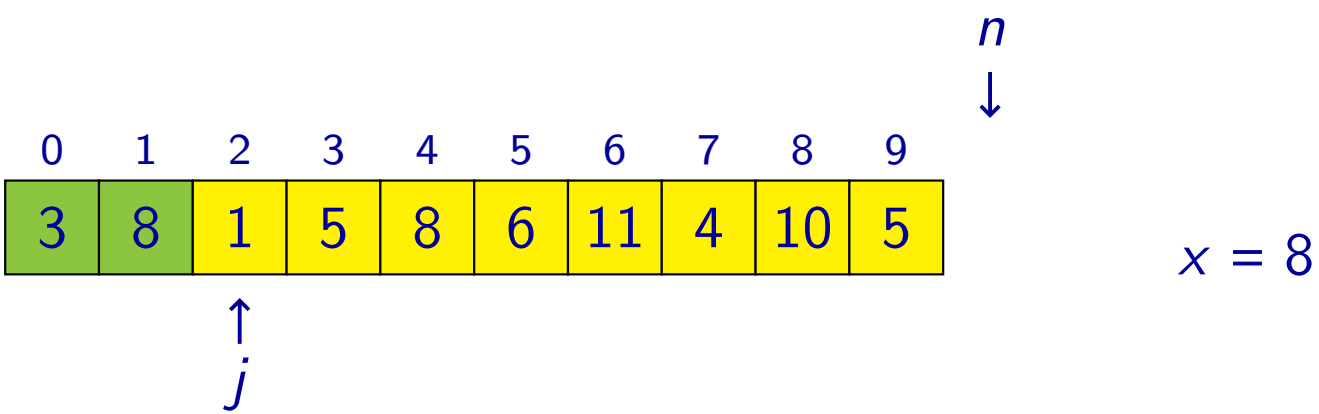
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

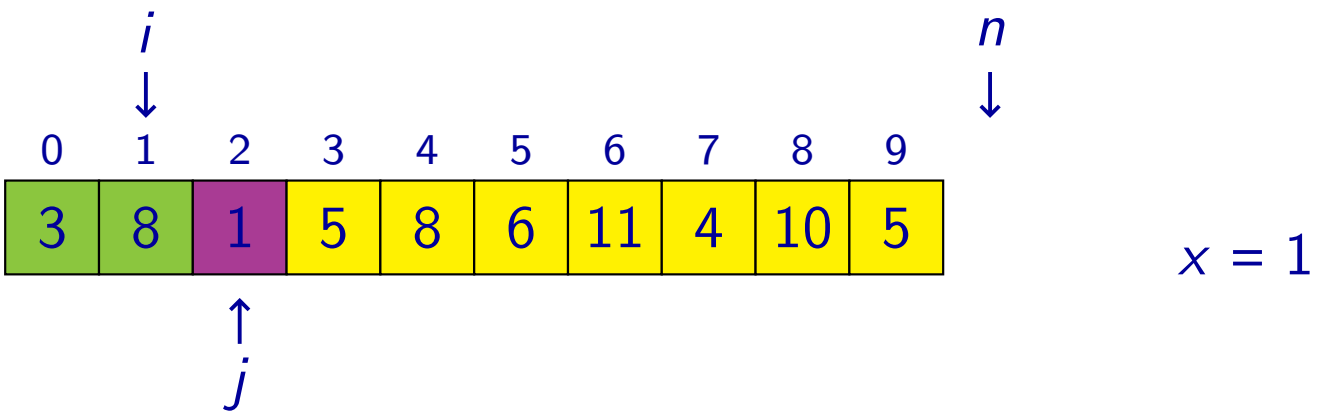
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

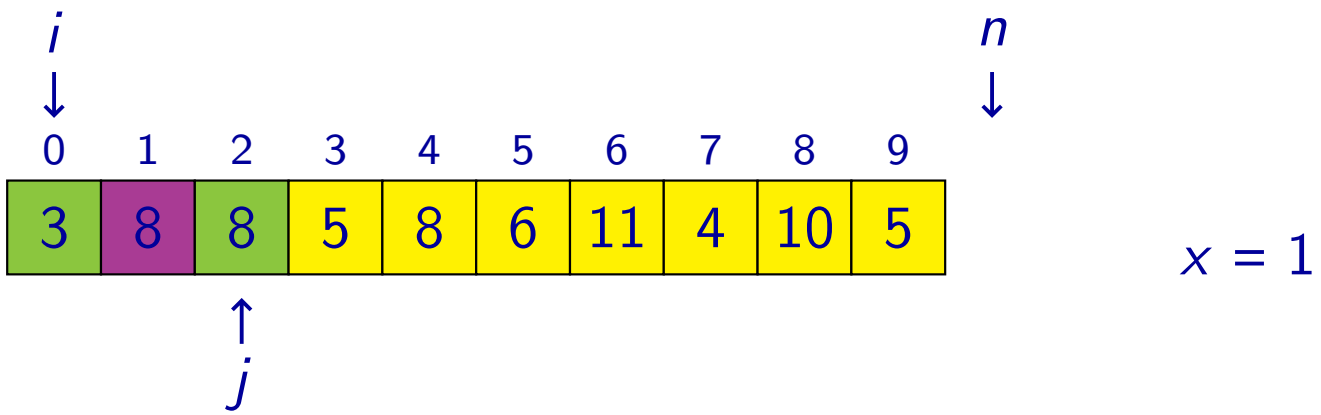
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

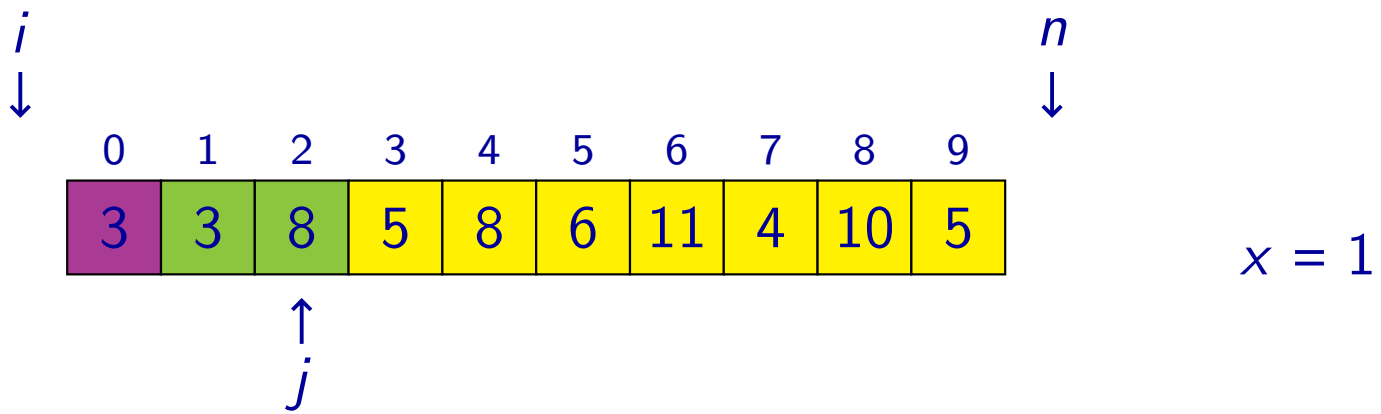
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

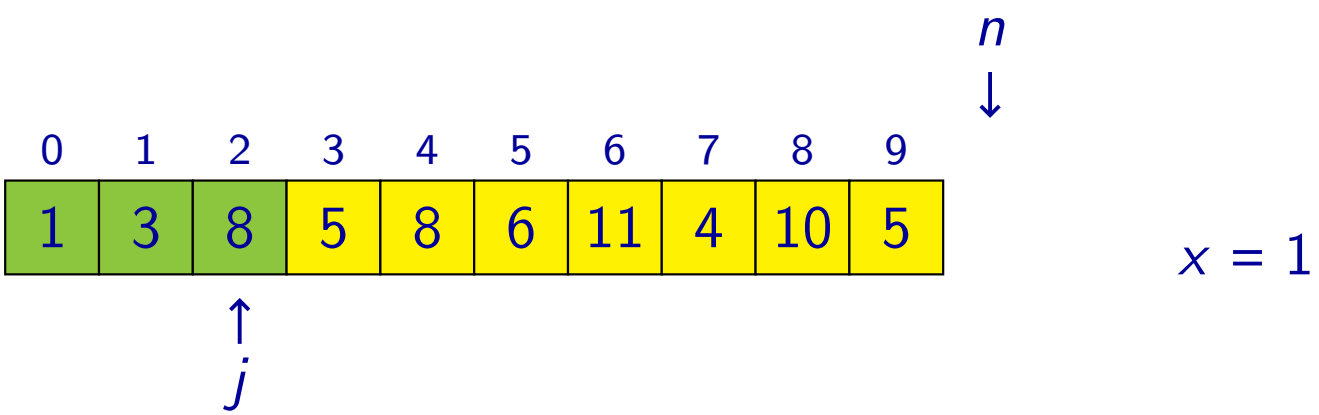
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

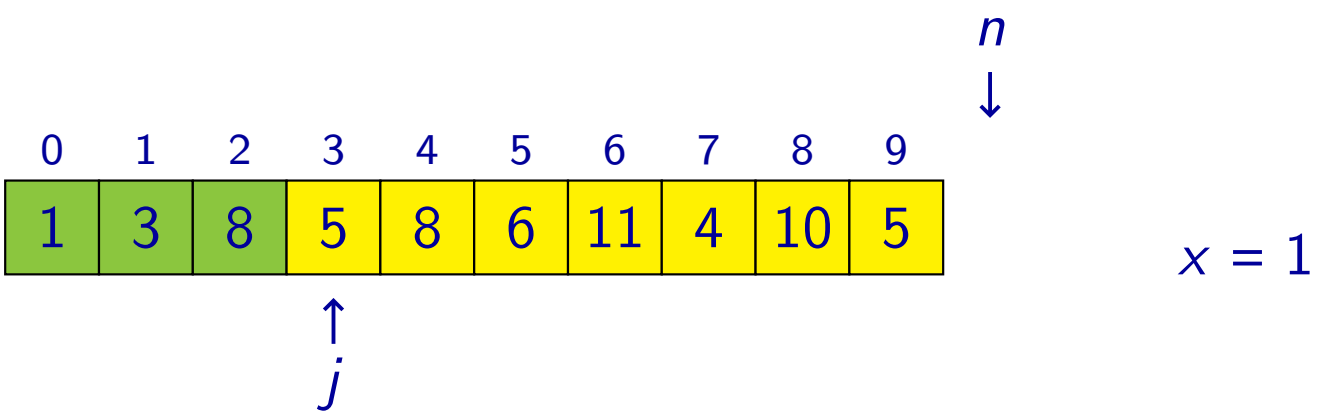
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

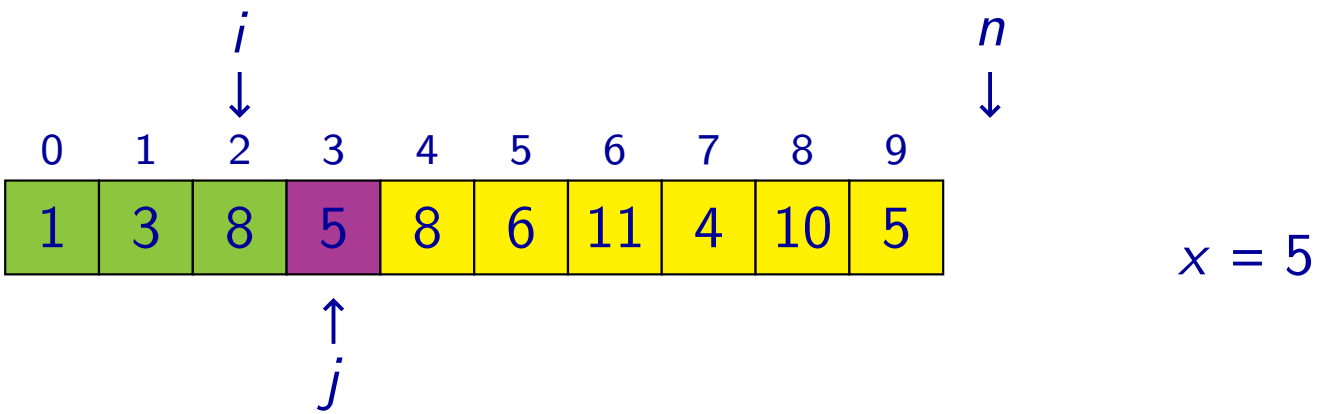
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

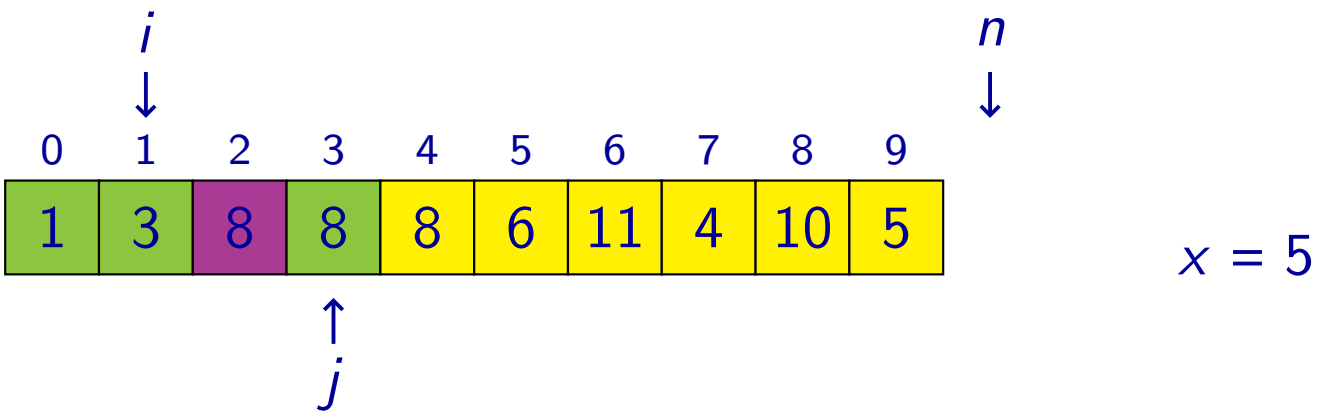
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

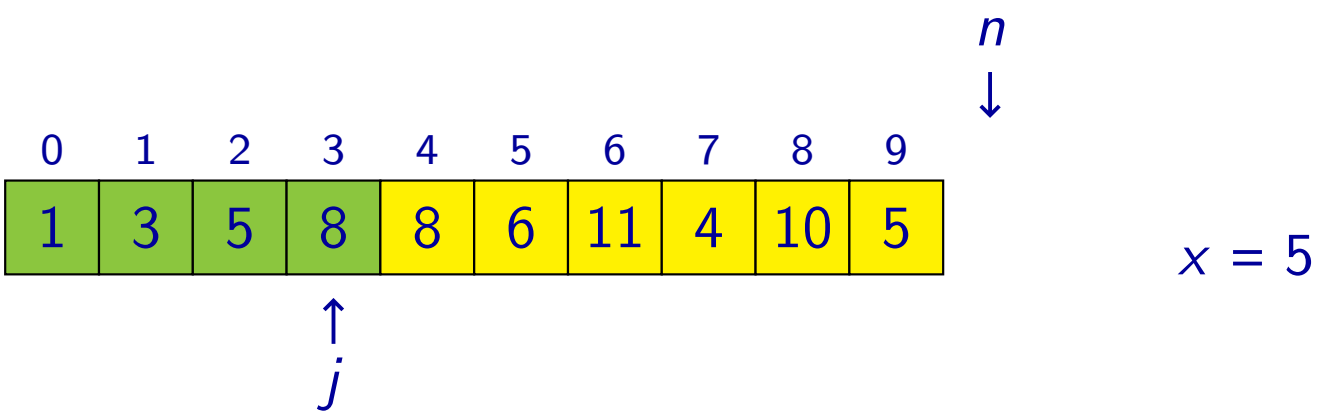
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

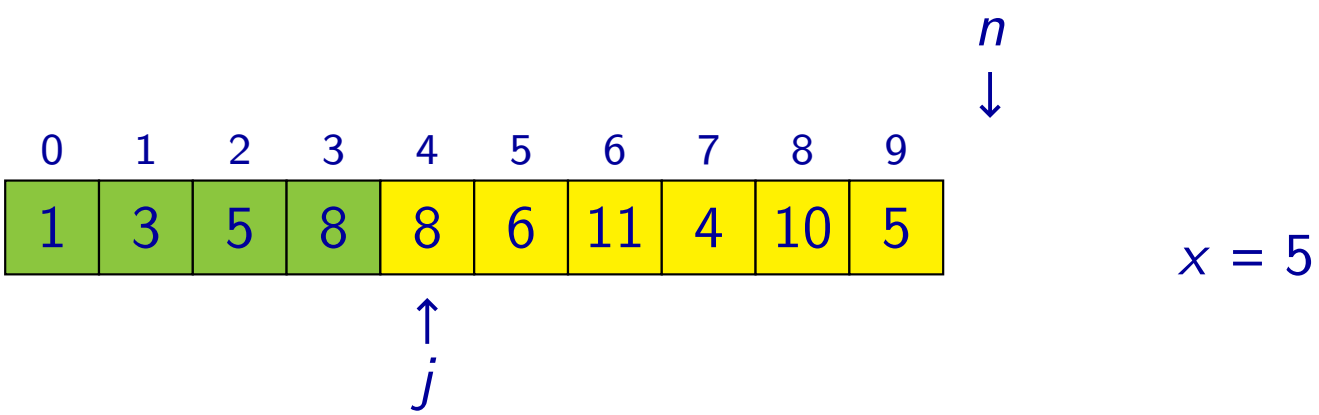
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

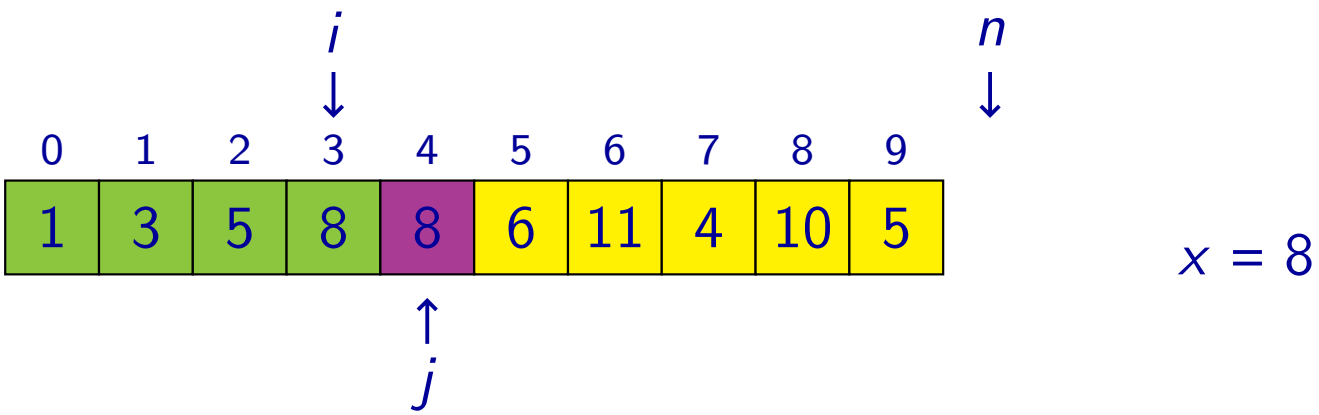
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

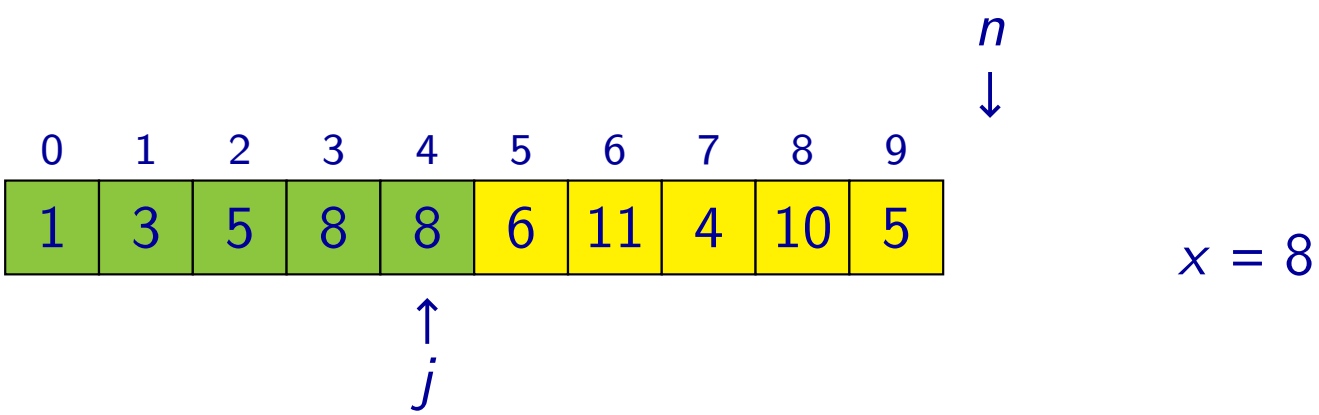
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

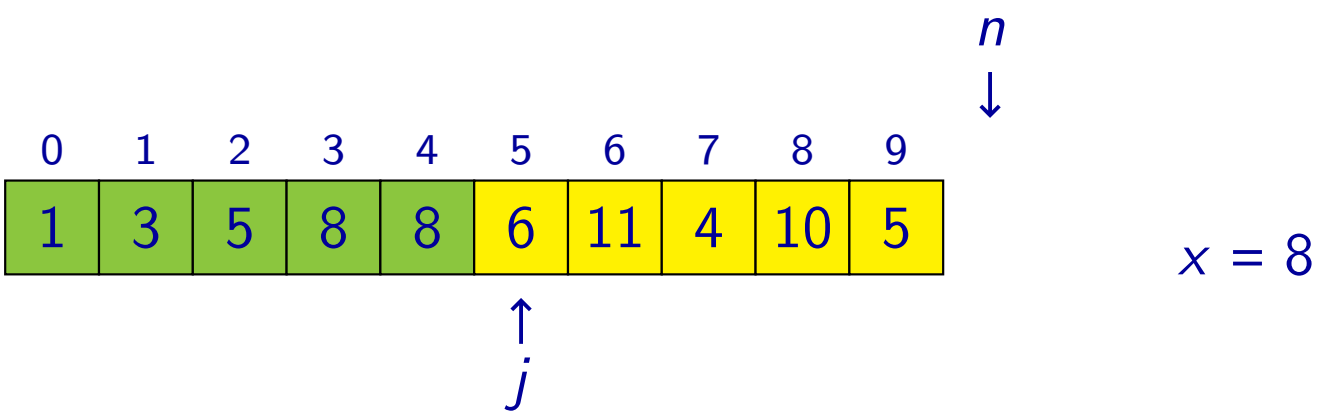
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

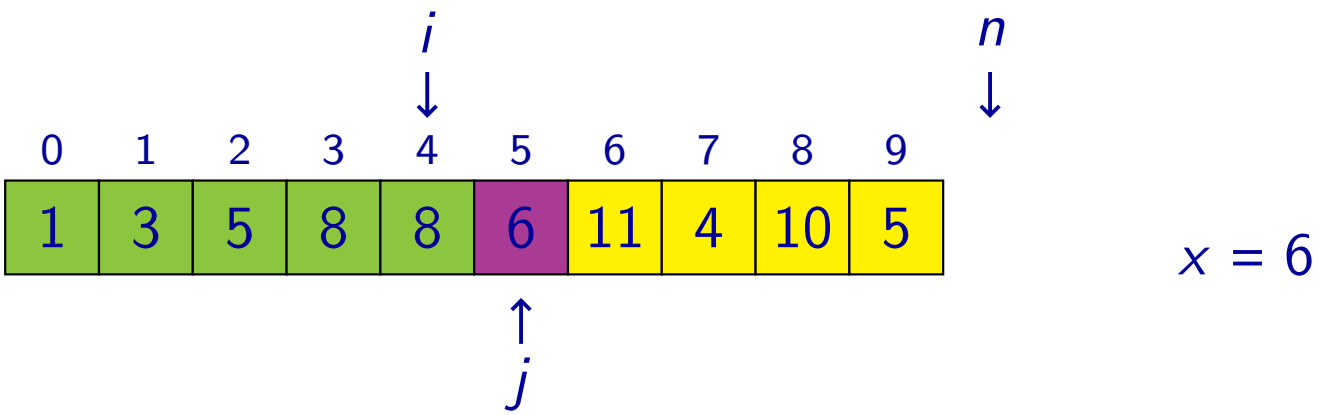
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

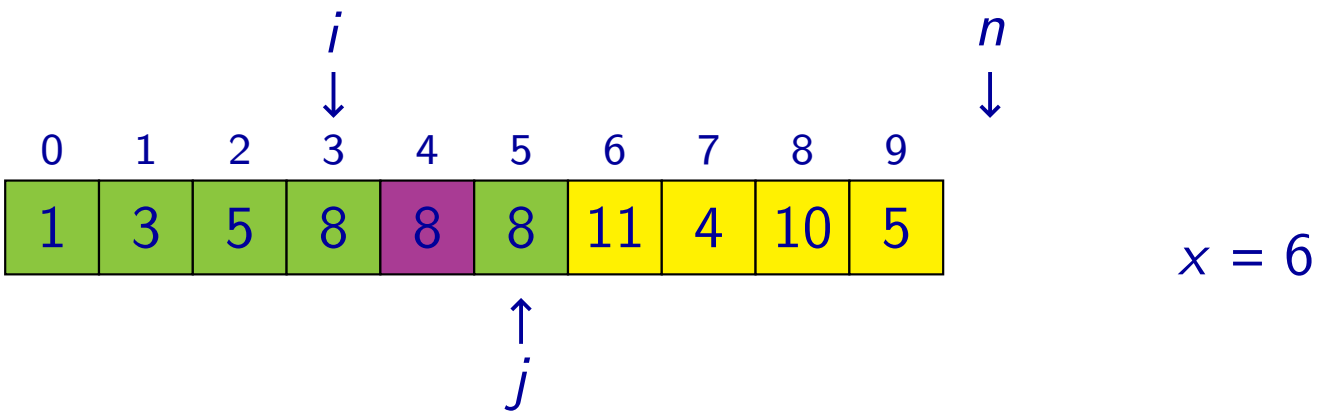
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

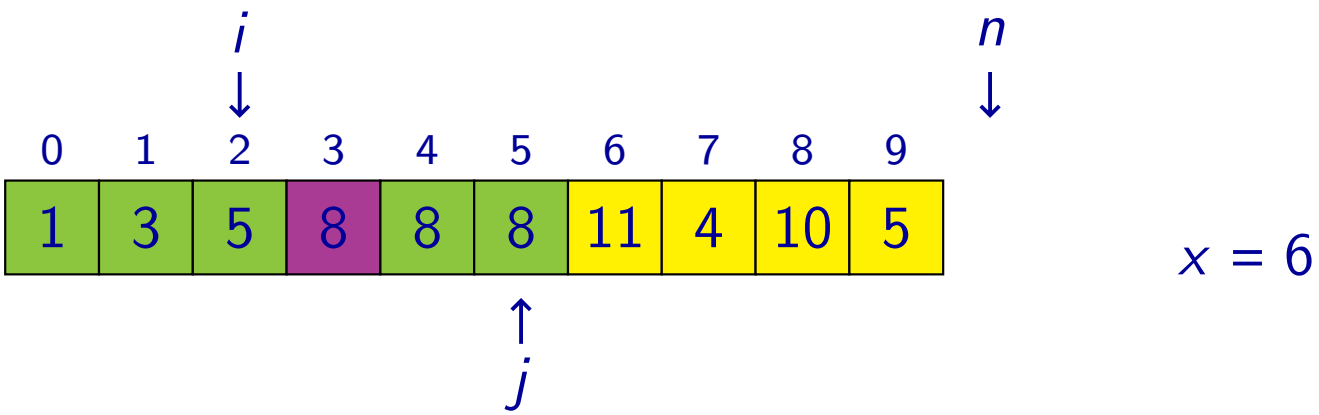
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

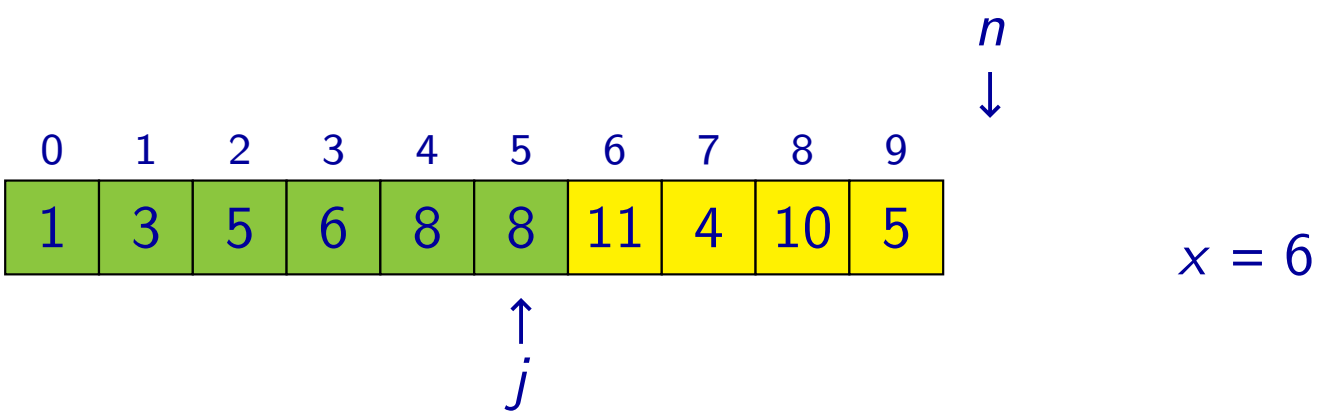
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

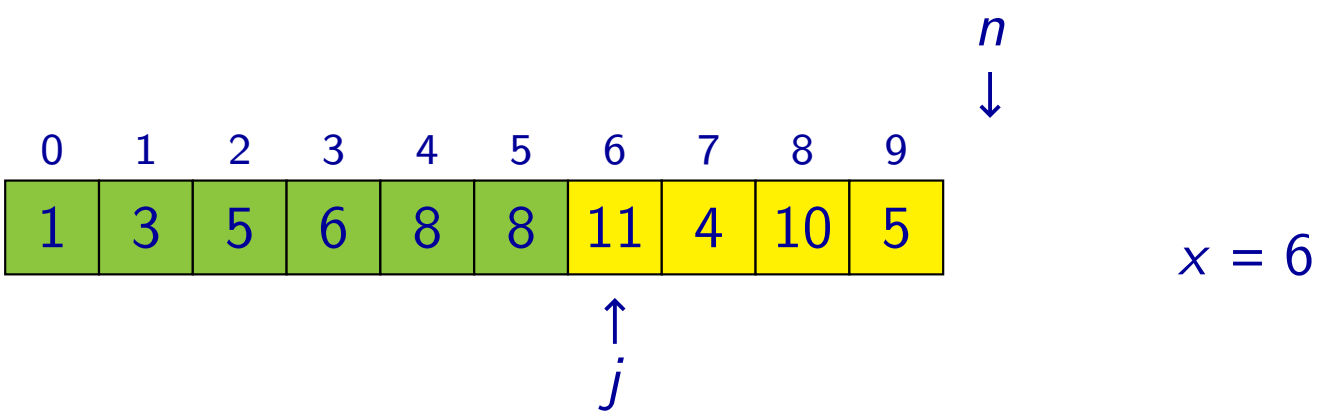
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

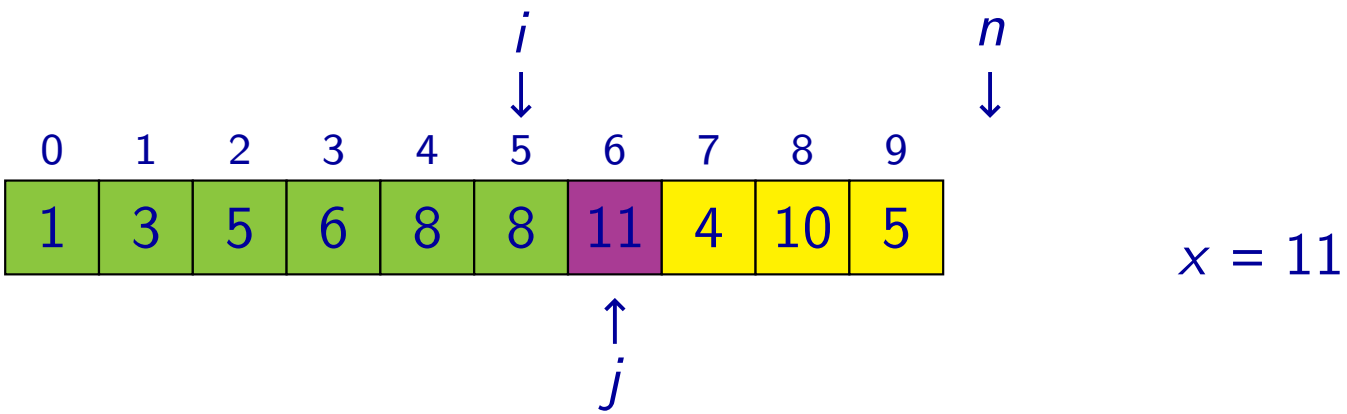
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

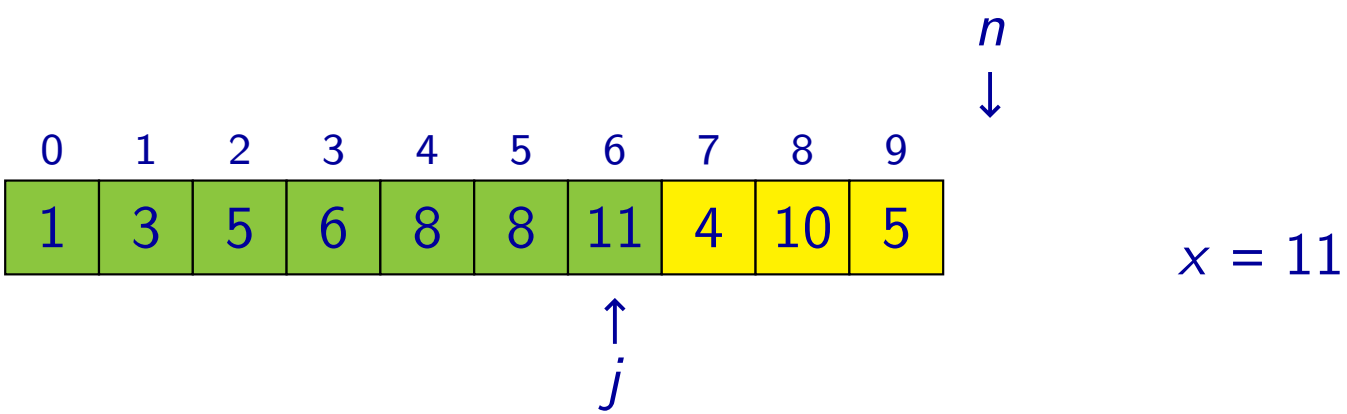
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

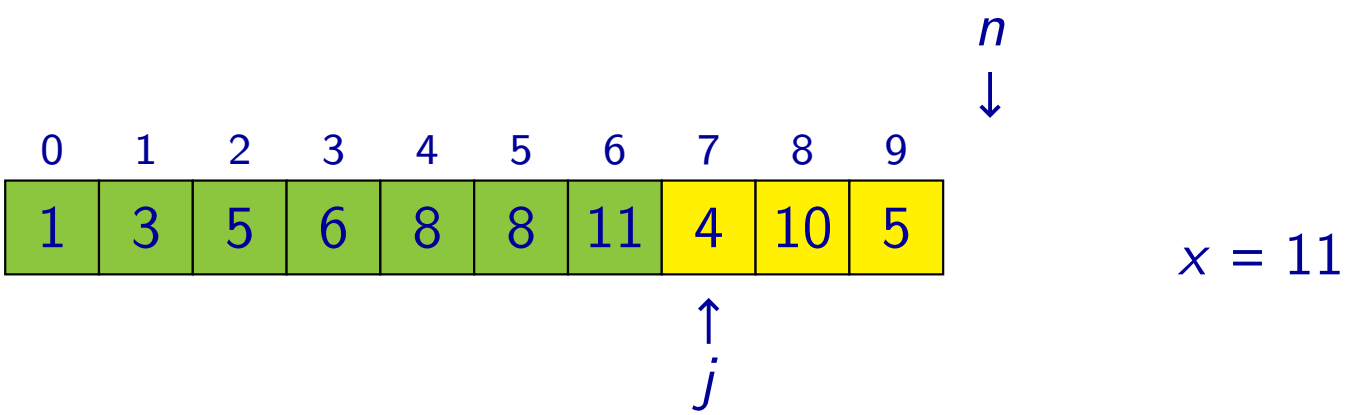
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

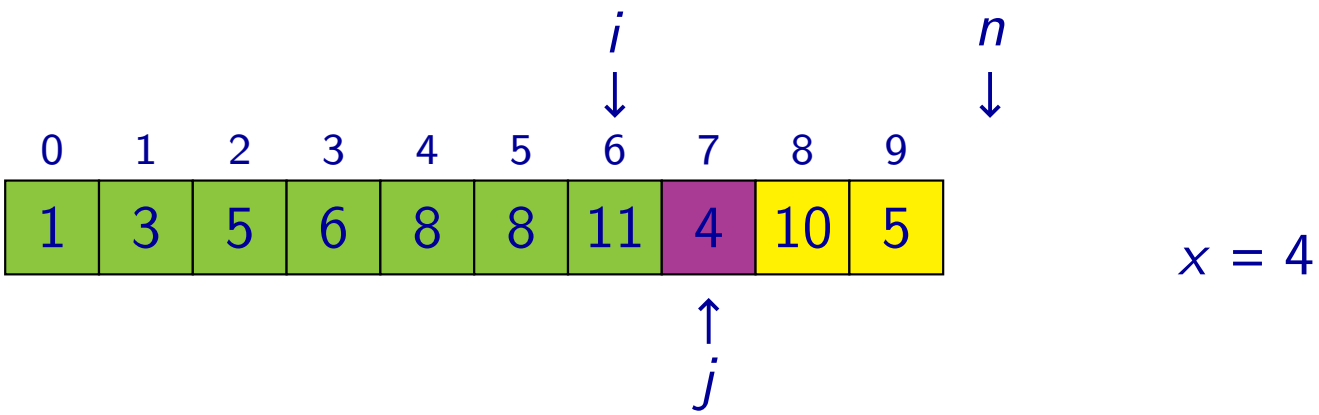
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

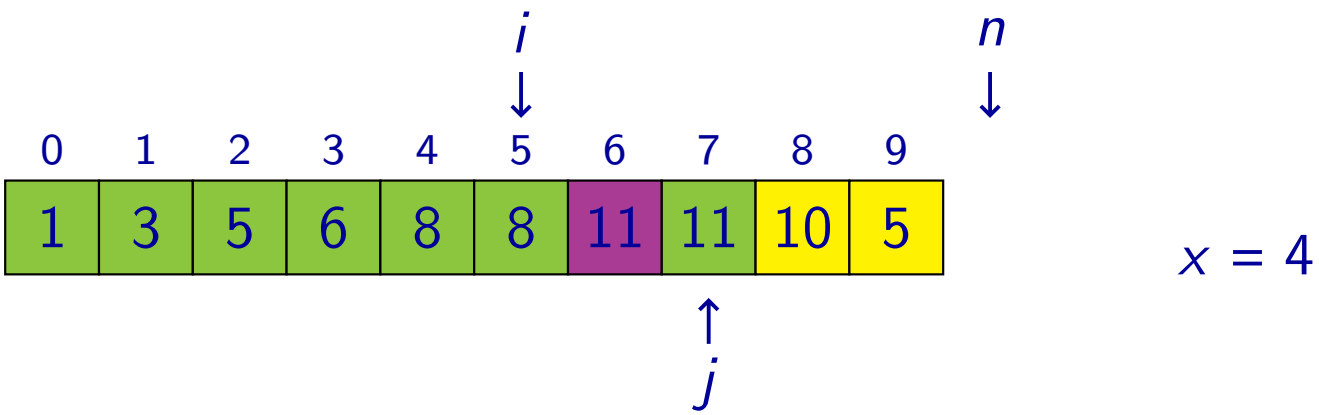
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

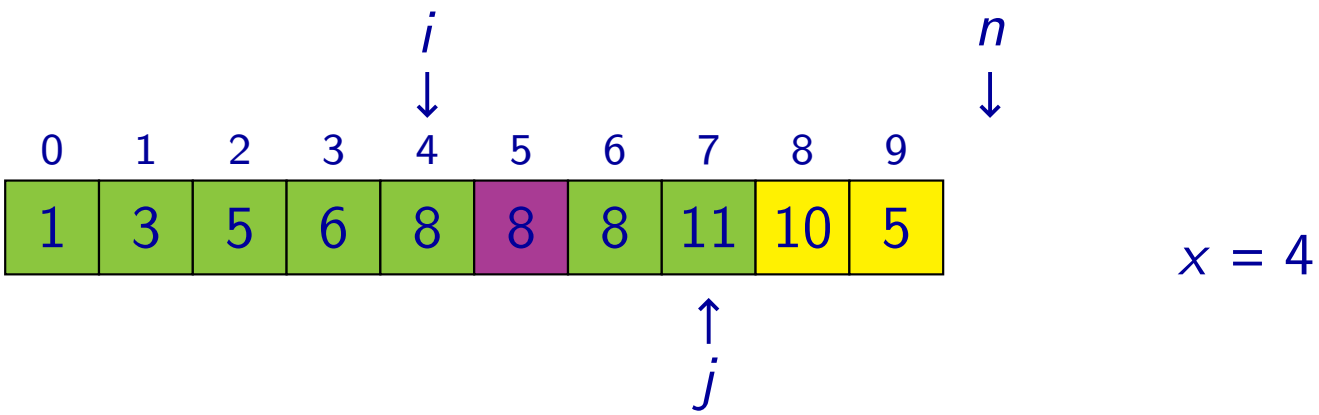
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

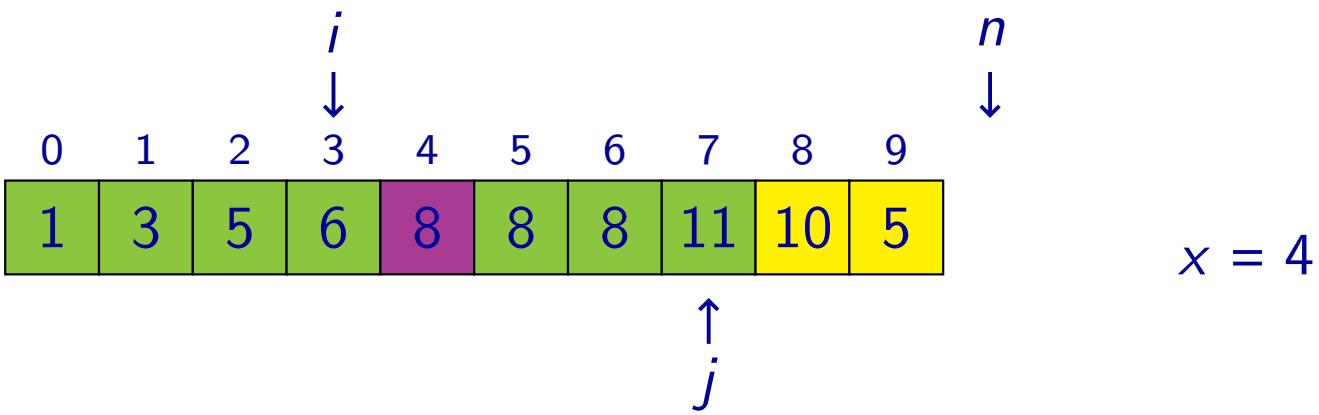
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

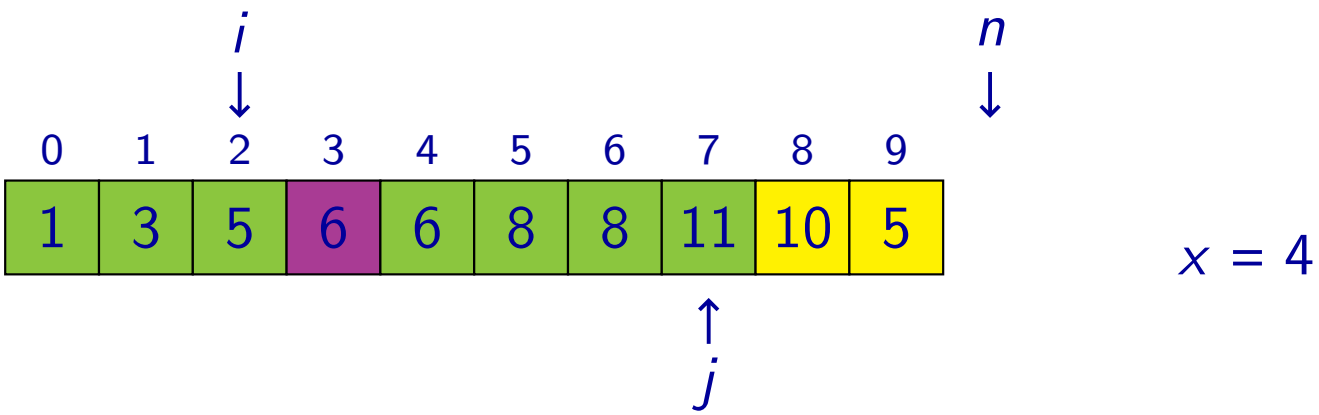
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

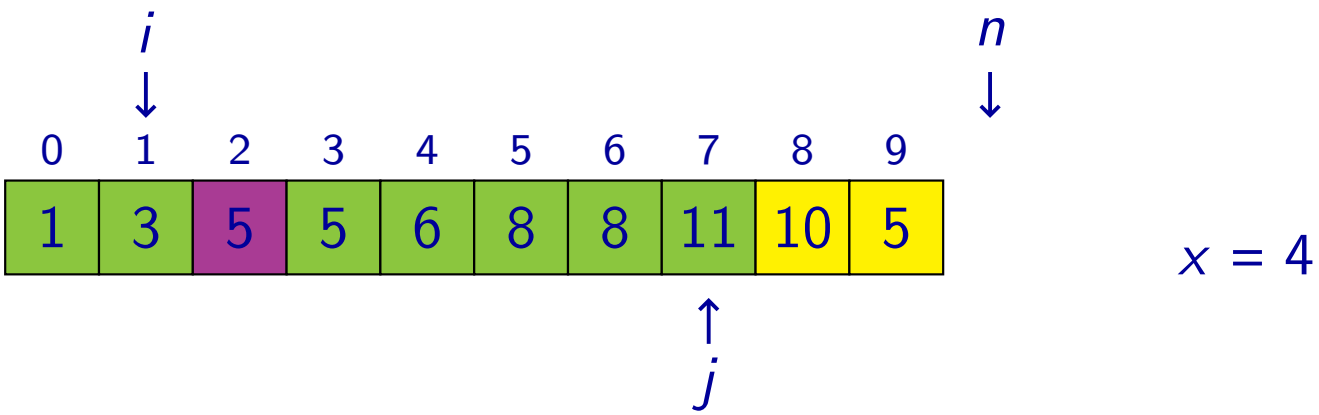
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

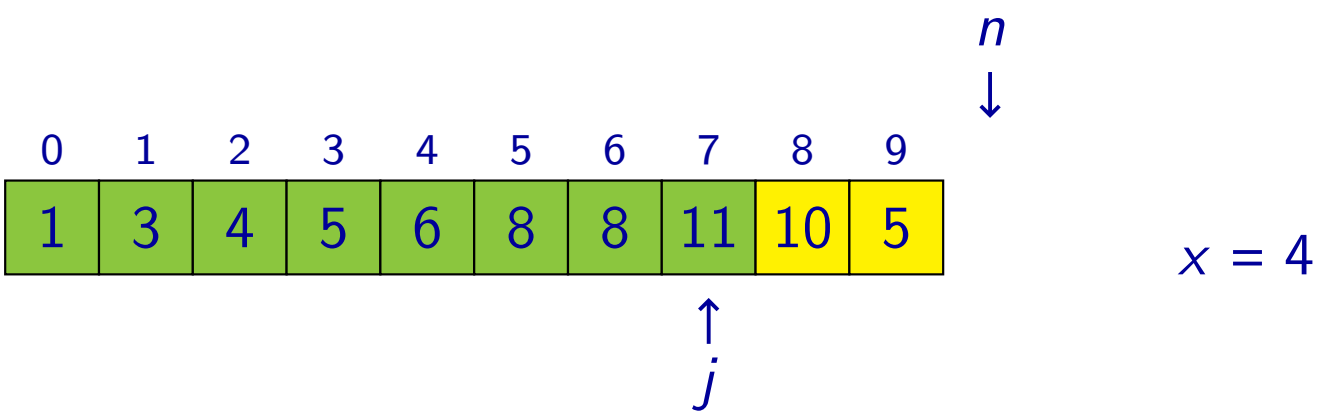
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

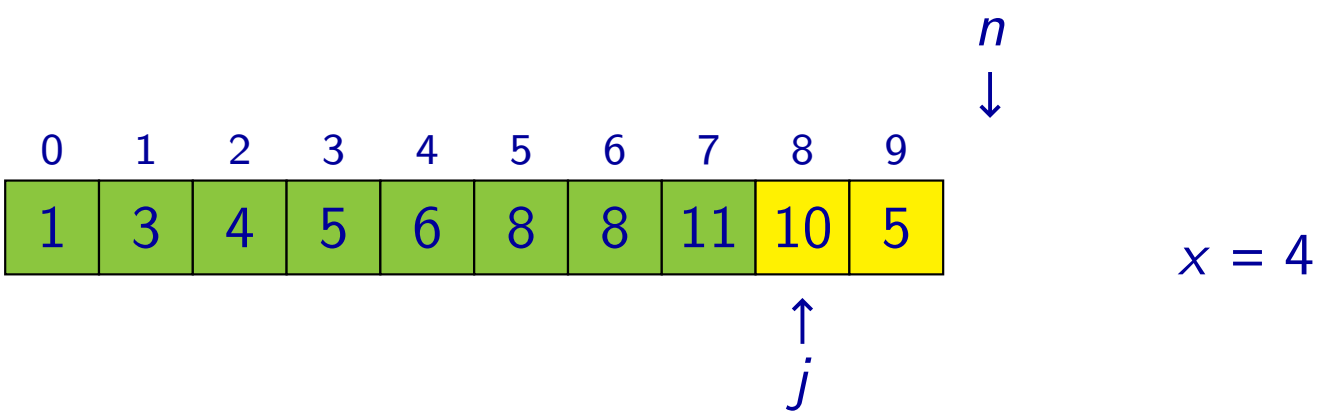
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

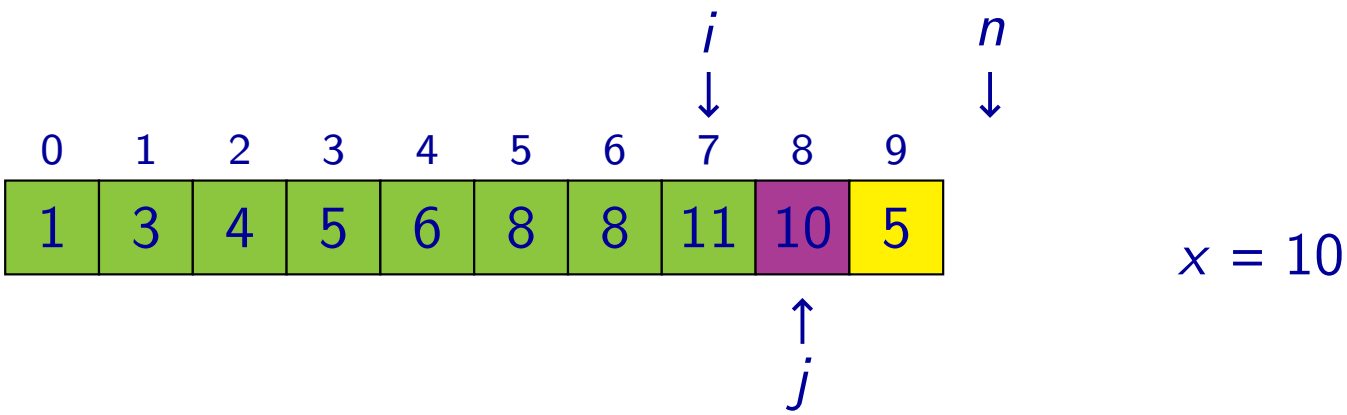
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

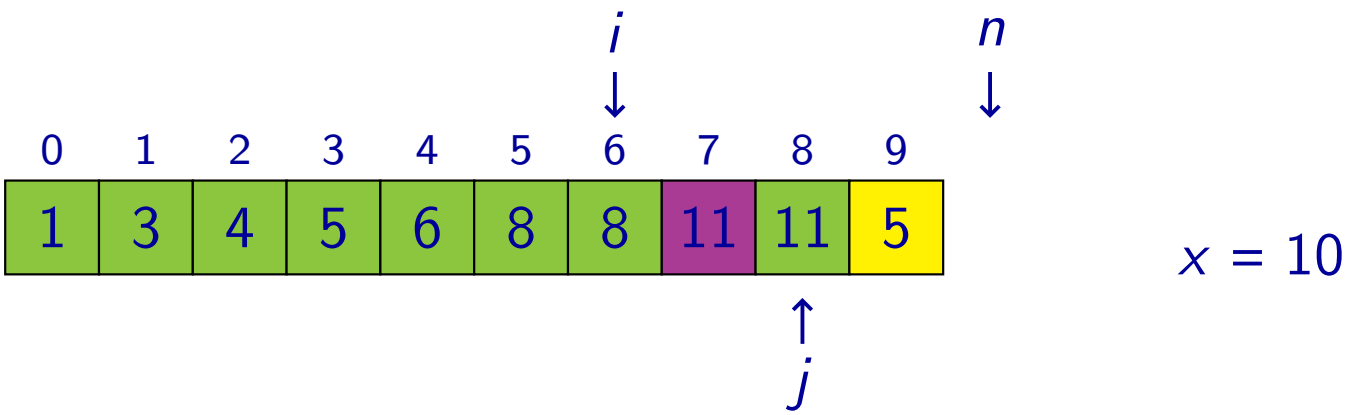
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

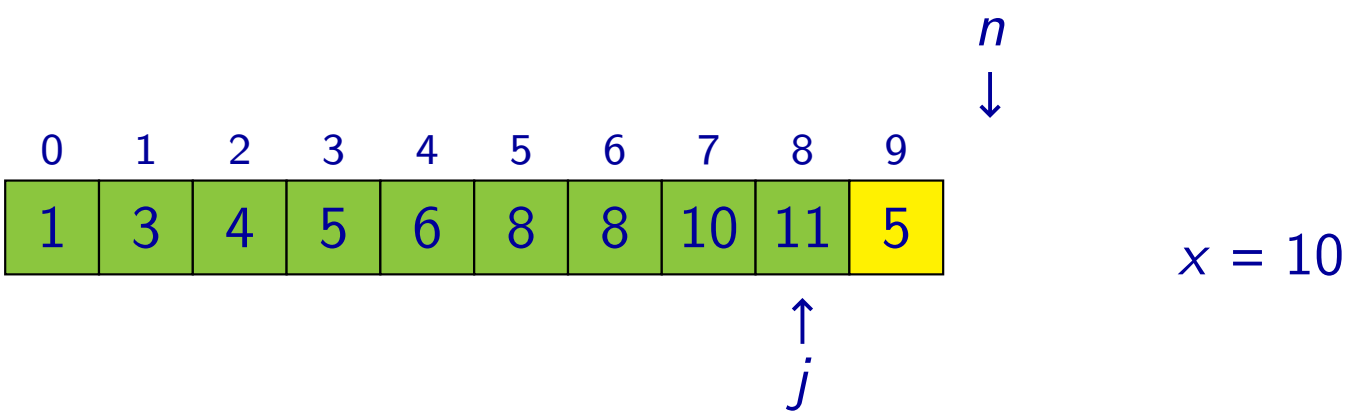
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

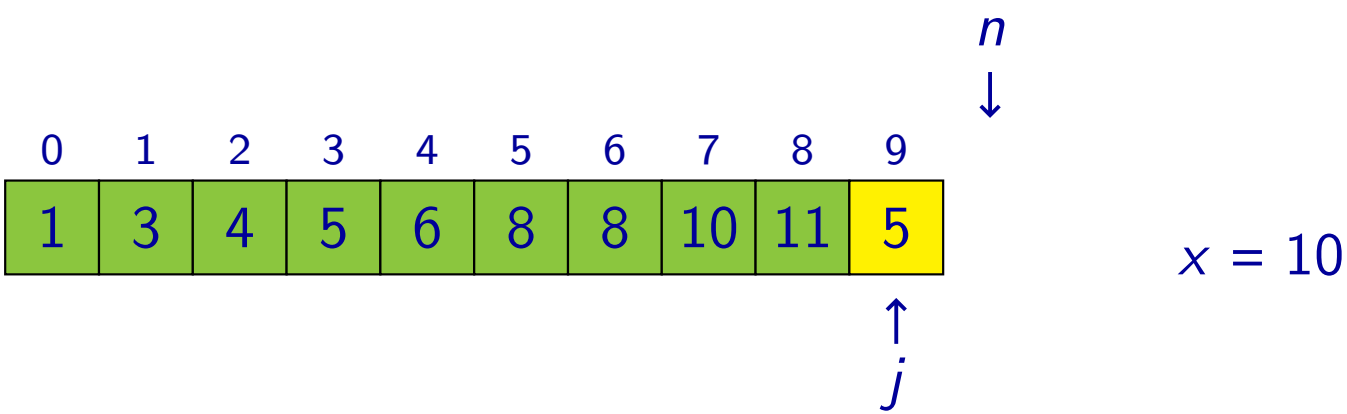
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

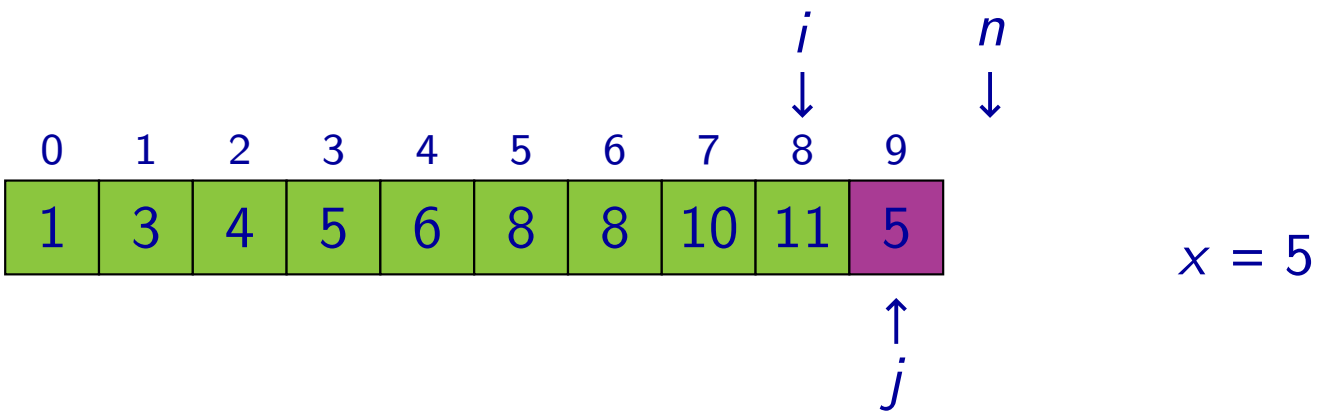
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

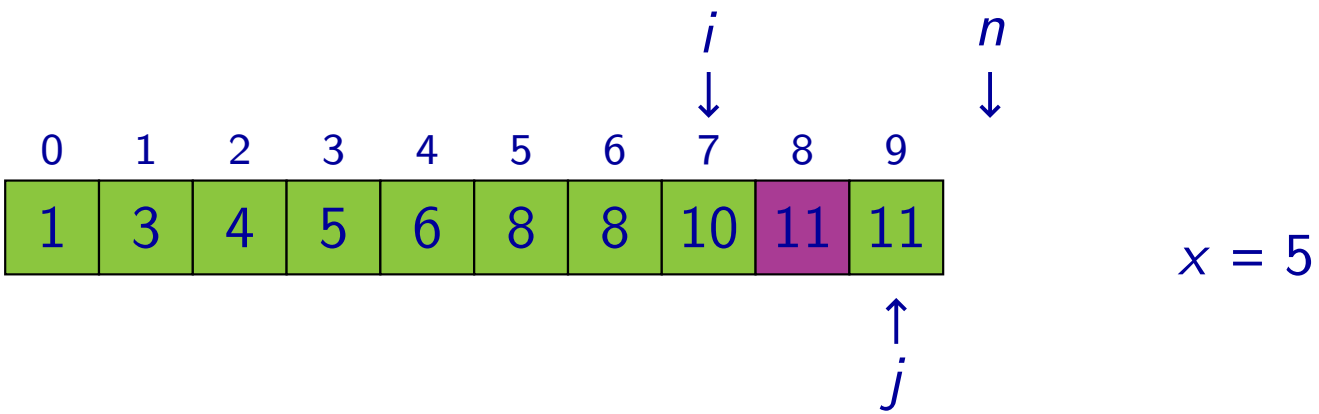
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

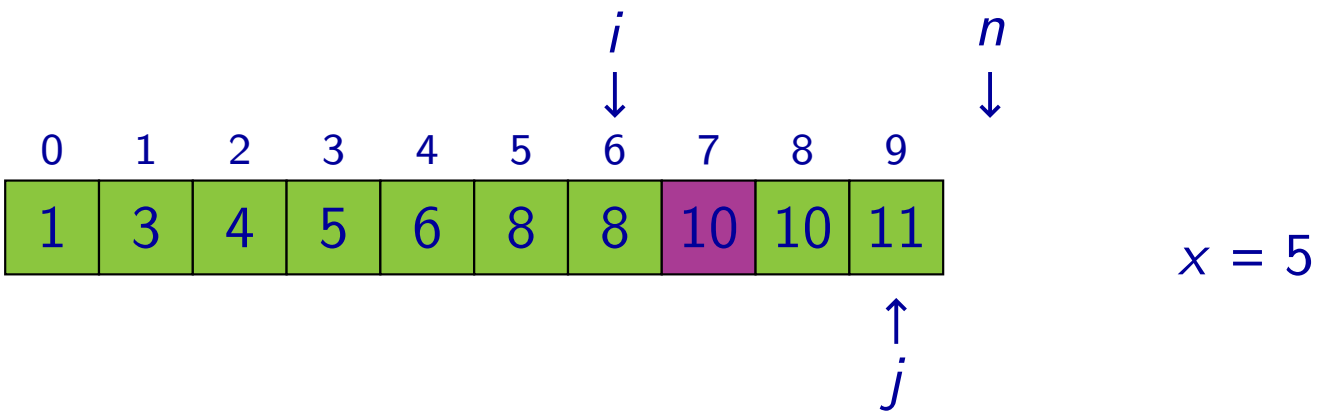
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

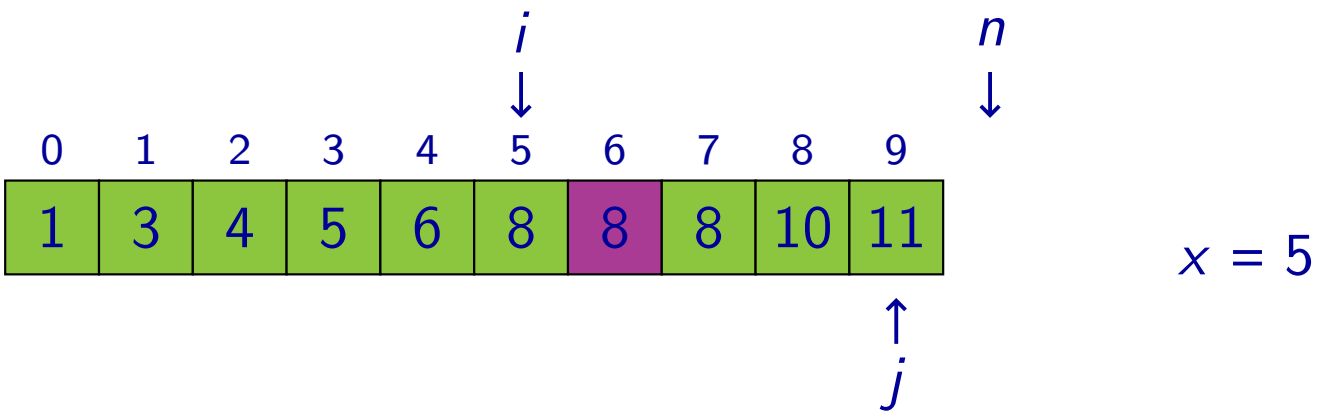
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

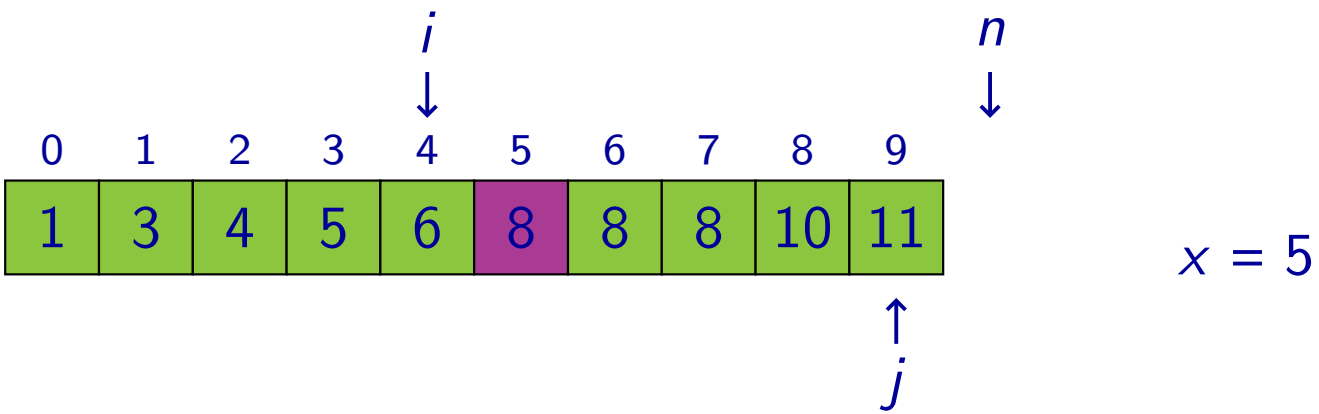
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

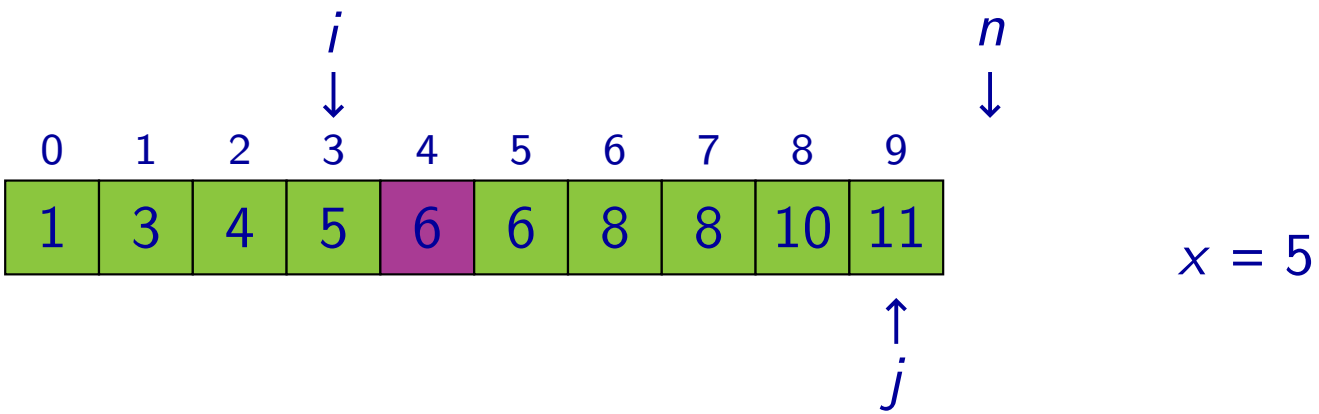
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

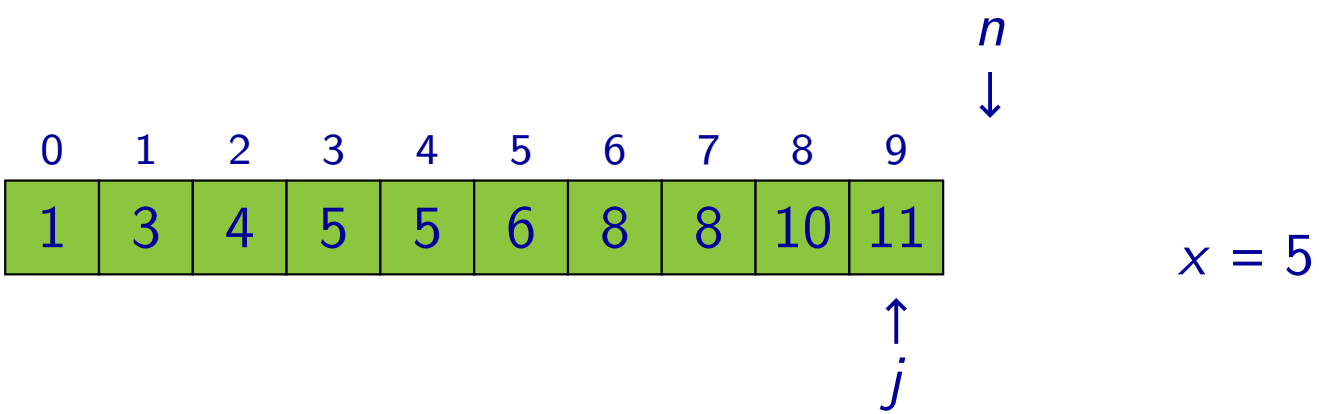
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

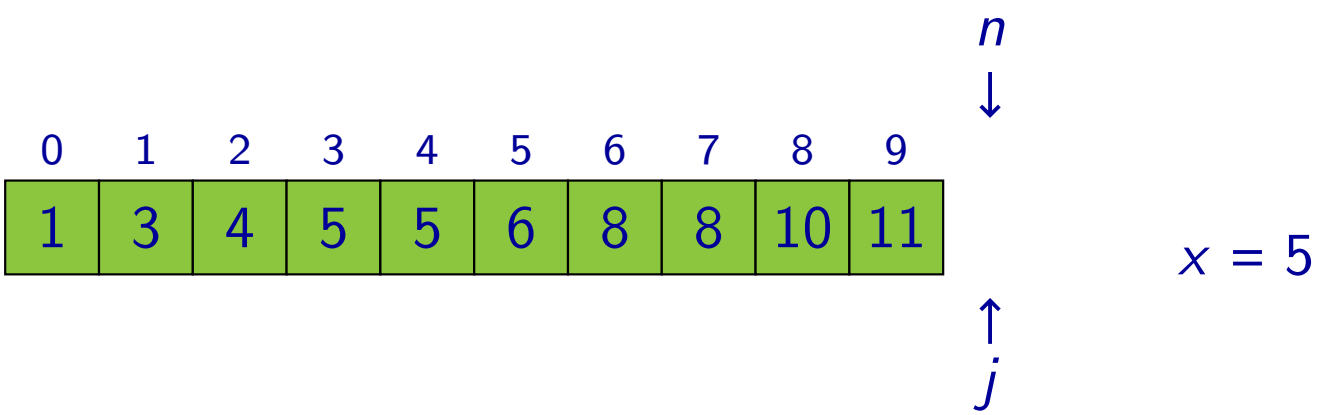
$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Example: A computation of algorithm INSERTION-SORT for input

$$A = [\, 3, 8, 1, 5, 8, 6, 11, 4, 10, 5 \,],\ n = 10.$$



Invariants

Let us assume that the input is an array $A = [a_0, a_1, \ldots, a_{n-1}]$ and number n (where $n \geq 1$) specifying the length of this array, i.e., at the beginning, it holds for each i, where $0 \leq i < n$, that $A[i] = a_i$.

- At the beginning of the **for** cykle (i.e., always before executing test $j < n$, resp. $j \leq n - 1$), the following invariants hold:
  - $1 \leq j \leq n$
  - Elements of the array $A[0], A[1], \ldots, A[j-1]$ contain values $a_0, a_1, \ldots, a_{j-1}$ sorted from the smallest to the biggest, i.e.,
    $$A[0] \leq A[1] \leq \cdots \leq A[j-1]$$
  - Elements of the array $A[j], A[j+1], \ldots, A[n-1]$ contain values $a_j, a_{j+1}, \ldots, a_{n-1}$, i.e.,
    $$A[j] = a_j,\ A[j+1] = a_{j+1},\ \ldots,\ A[n-1] = a_{n-1}$$

Invariants

- At the beginning **while** cycle (i.e., always before executing test $i \geq 0$), the following invariants hold:
  - $1 \leq j < n$
  - $-1 \leq i < j$
  - Variable x contains value a_j, i.e., $x = a_j$.
  - Elements of the array $A[0], A[1], \ldots, A[i]$ and $A[i+2], A[i+3], \ldots, A[j]$ contain values $a_0, a_1, \ldots, a_{j-1}$ ordered from the smallest to the biggest, i.e.,
    $$A[0] \leq A[1] \leq \cdots \leq A[i] \leq A[i+2] \leq A[i+3] \leq \cdots \leq A[j]$$
  - All elements $A[i+2], A[i+3], \ldots, A[j]$ are strictly greater than x.
  - Elements of the array $A[j+1], A[j+2], \ldots, A[n-1]$ contain values $a_{j+1}, a_{j+2}, \ldots, a_{n-1}$, i.e.,
    $$A[j+1] = a_{j+1},\ A[j+2] = a_{j+2},\ \ldots,\ A[n-1] = a_{n-1}$$

Finiteness of a Computation

Two possibilities how an infinite computation can look:

- some configuration is repeated — then all following configurations are also repeated
- all configurations in a computation are different but a final configuration is never reached

Finiteness of a Computation

One of standard ways of proving that an algorithm halts for every input after a finite number of steps:

- to assign a value from a set W to every (reachable) configuration
- to define an order $\leq$ on set W such that there are no infinite (strictly) decreasing sequences of elements of W
- to show that the values assigned to configuration decrease with every execution of each instruction, i.e., if $\alpha \xrightarrow{I} \alpha'$ then

$$f(\alpha) > f(\alpha')$$

($f(\alpha)$, $f(\alpha')$ are values from set W assigned to configurations α and α')

Finiteness of a Computation

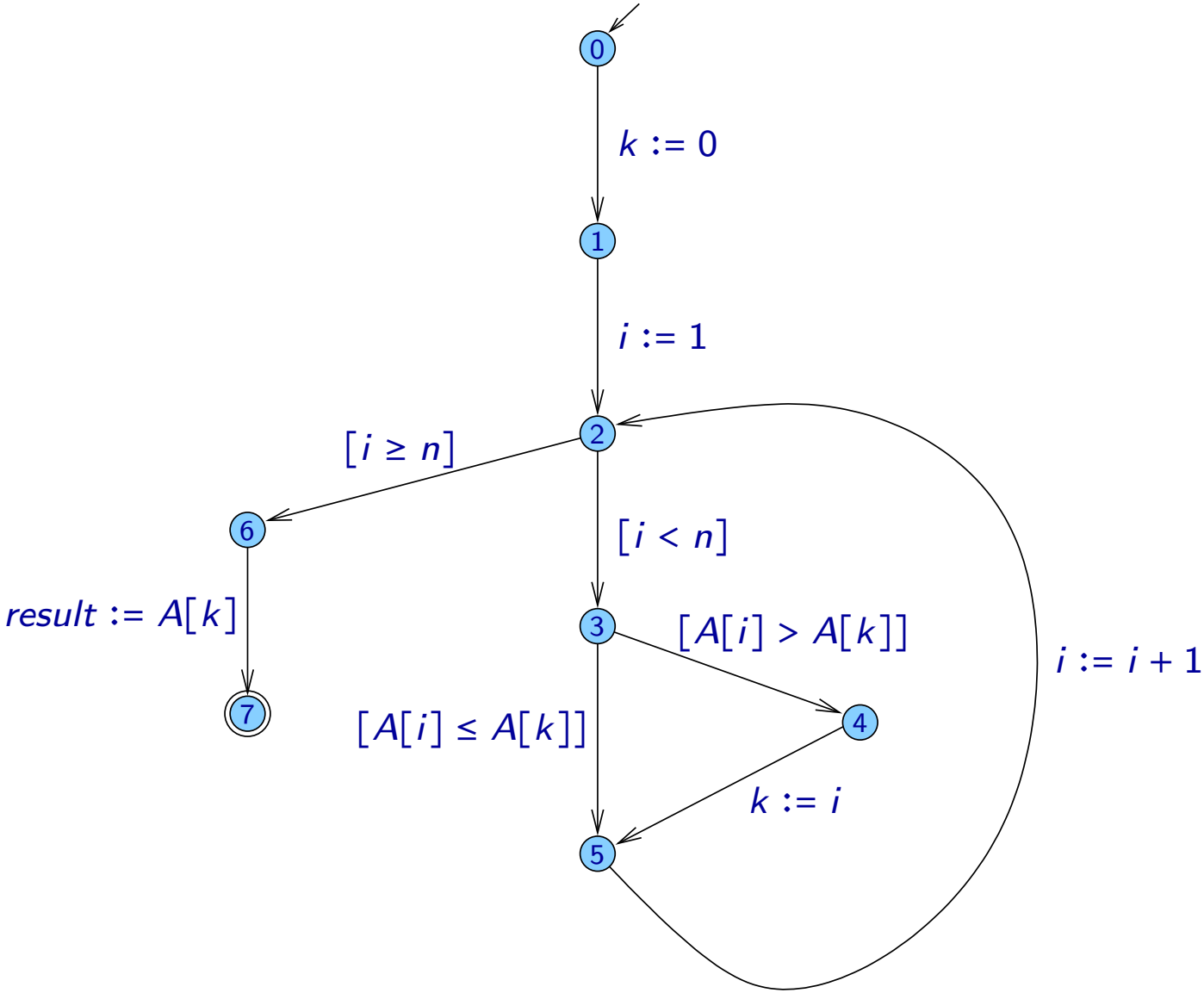
As a set W, we can use for example:

- The set of natural numbers $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ with ordering $\leq$.
- The set of vectors of natural numbers with lexicographic ordering, i.e., the ordering where vector $(a_1, a_2, \dots, a_m)$ is smaller than $(b_1, b_2, \dots, b_n)$, if
  - there exists i such that $1 \leq i \leq m$ and $i \leq n$, where $a_i < b_i$ and for all j such that $1 \leq j < i$ it holds that $a_j = b_j$, or
  - $m < n$ and for all j such that $1 \leq j \leq m$ is $a_j = b_j$.

  For example, $(5, 1, 3, 6, 4) < (5, 1, 4, 1)$ and $(4, 1, 1) < (4, 1, 1, 3)$.

  Remark: The number of elemets in vectors must be bounded by some constant.

Finiteness of a Computation

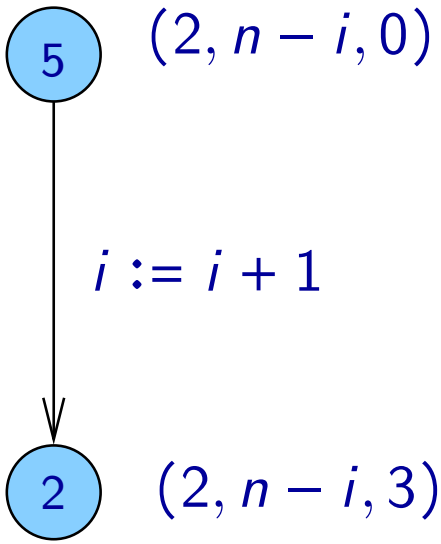


Finiteness of a Computation

Example: Vectors assigned to individual configurations:

- State 0: $f(\alpha) = (4)$
- State 1: $f(\alpha) = (3)$
- State 2: $f(\alpha) = (2, n - i, 3)$
- State 3: $f(\alpha) = (2, n - i, 2)$
- State 4: $f(\alpha) = (2, n - i, 1)$
- State 5: $f(\alpha) = (2, n - i, 0)$
- State 6: $f(\alpha) = (1)$
- State 7: $f(\alpha) = (0)$

Finiteness of a Computation



We must take into account that the value of variable i is modified by this instruction.

A transition from a configuration with assigned vector $(2, n-i, 0)$ to a configuration with assigned vector $(2, n-i', 3)$, where $i' = i+1$.

It is obvious that $n-i' < n-i$, since $n-(i+1) < n-i$.