Algoritmy

Algoritmy

Příklad: Popis algoritmu pomocí **pseudokódu**:

Algoritmus 1: Algoritmus pro nalezení největšího prvku v poli

```
1  FIND-MAX (A, n):
2  begin
3      k := 0
4      for i := 1 to n − 1 do
5          if A[i] > A[k] then
6              k := i
7          end
8      end
9      return A[k]
10 end
```

Algoritmy

Algoritmus

- zpracovává **vstup**
- generuje **výstup**

Z hlediska analýzy toho, jak daný algoritmus funguje, většinou není příliš podstatný rozdíl v tom, jestli algoritmus:

- čte vstupní data z nějakého vstupního zařízení (např. ze souboru na disku, z klávesnice, apod.)
- zapisuje data na nějaké výstupní zařízení (např. do souboru, na obrazovku, apod.)

nebo

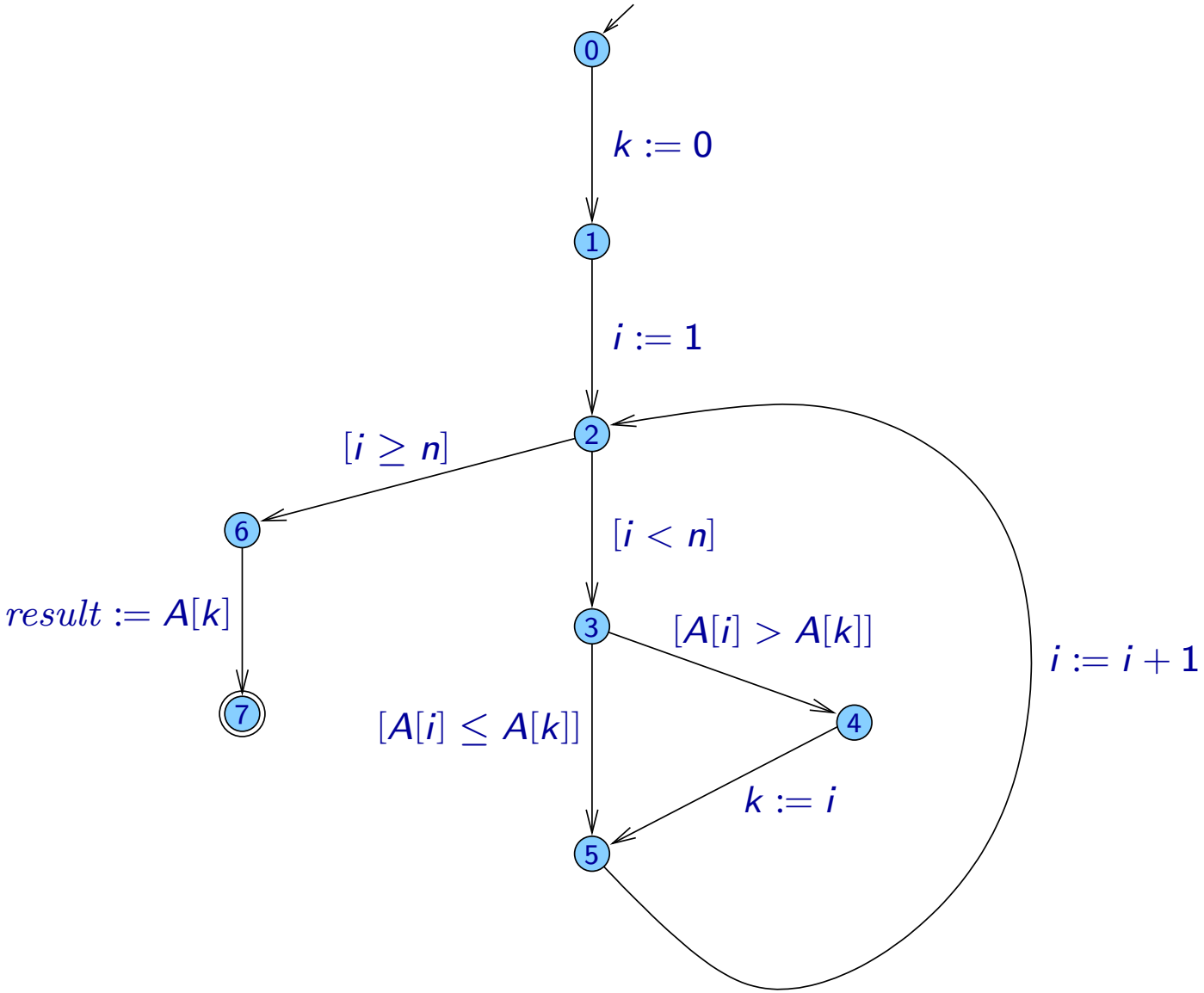
- čte vstupní data z paměti (např. jsou mu předány jako parametry)
- zapisuje data na do paměti (např. je vrátí jako návratovou hodnotu)

Řídící tok

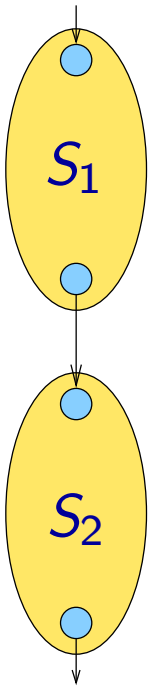
Instrukce lze zhruba rozdělit na dvě skupiny:

- intrukce přímo pracující s daty:
  - přiřazení
  - vyhodnocení hodnot výrazů v podmínkách
  - čtení vstupu, zápis na výstup
  - ...
- instrukce ovlivňující **řídící tok** — určují, které instrukce se budou provádět, v jakém pořadí, apod.:
  - větvení (if, switch, ...)
  - cykly (while, do .. while, for, ...)
  - uspořádání instrukcí do bloků
  - návraty z podpogramů (return, ...)
  - ...

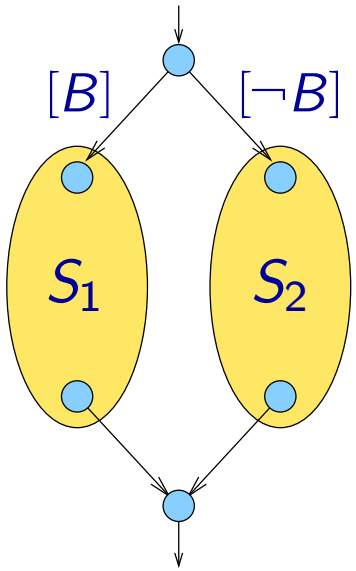
Graf řídícího toku



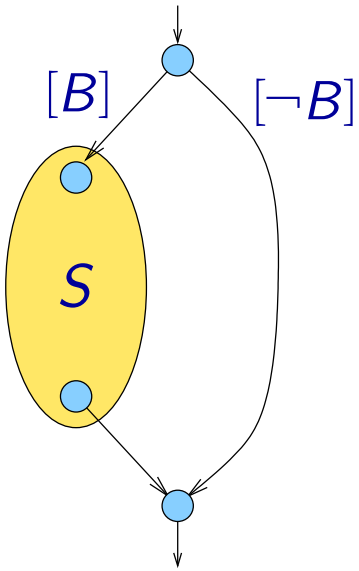
Některé základní konstrukce strukturovaného programání



$S_1; S_2$

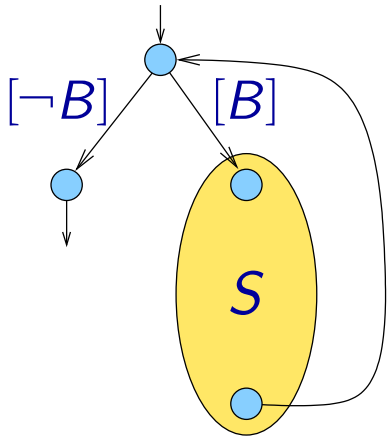


if B **then** S_1 **else** S_2

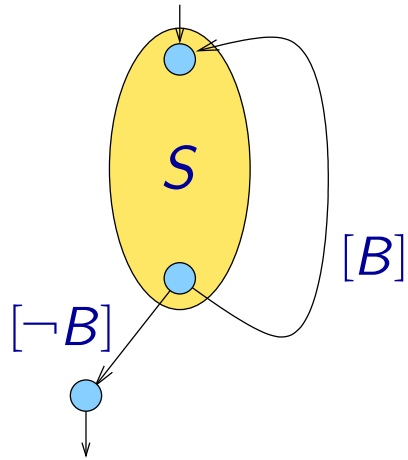


if B **then** S

Některé základní konstrukce strukturovaného programání

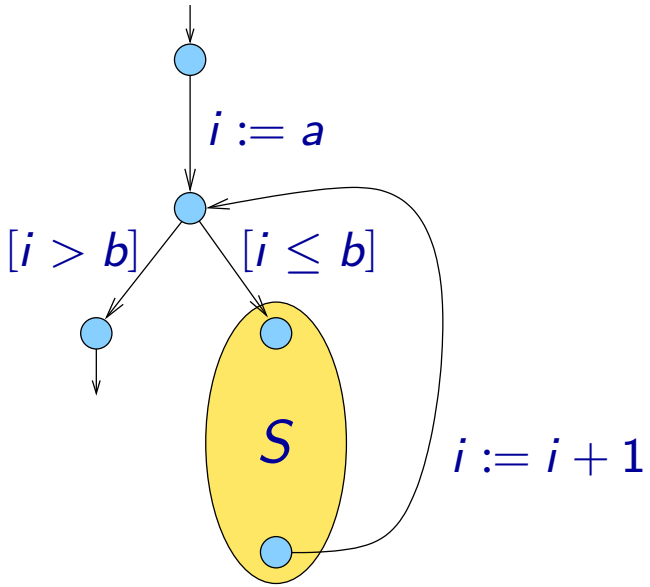


while B **do** S



do S **while** B

Některé základní konstrukce strukturovaného programání



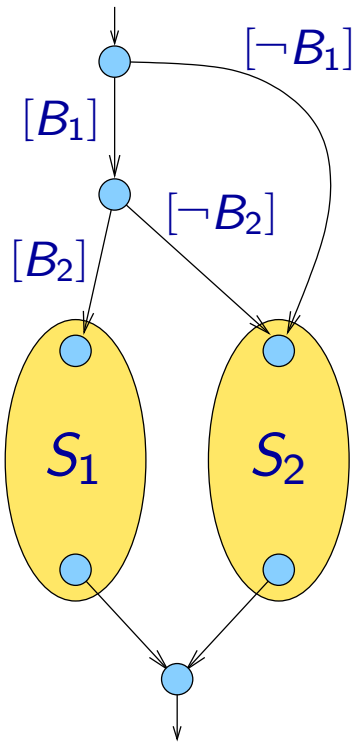
for $i := a$ **to** b **do** S

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

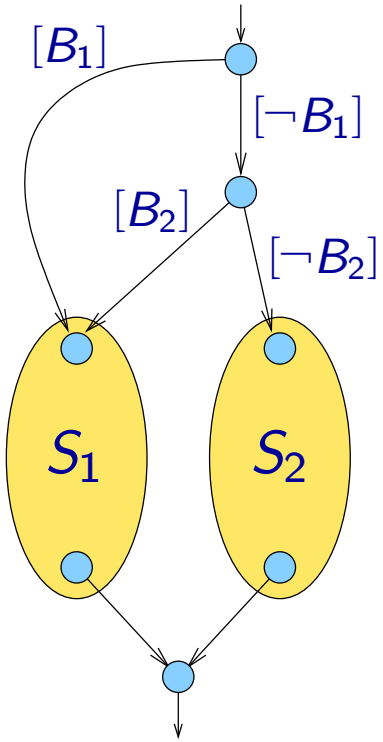
Některé základní konstrukce strukturovaného programání

Zkrácené vyhodnocování složených podmínek, např.:

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

Řídící tok realizovaný pomocí goto

- **goto** ℓ — **nepodmíněný skok**
- **if** B **then goto** ℓ — **podmíněný skok**

Příklad:

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** $A[k]$

Řídící tok realizovaný pomocí goto

- **goto** ℓ — **nepodmíněný skok**
- **if** B **then goto** ℓ — **podmíněný skok**

Příklad:

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

Vyhodnocení složitých výrazů

Vyhodnocení složitého výrazu, jako třeba

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

může být na nižší úrovni nahrazeno posloupností jednodušších příkazů, jako třeba

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

Výpočet algoritmu

Algoritmus je vykonáván strojem — může to být například:

- skutečný počítač — vykonává instrukce strojového kódu
- virtuální stroj — vykonává instrukce bytekódu
- nějaký idealizovaný matematický model počítače
- ...

Stroj může být:

- jednoúčelový — vykonává jen jeden algoritmus
- obecnější — algoritmus dostává ve formě **programu**

Stroj pracuje po **krocích**.

Algoritmus během výpočtu zpracovává konkrétní **vstup**.

Výpočet algoritmu

Během výpočtu si stroj musí pamatovat:

- která intrukce se právě provádí
- obsah pracovní paměti

Podle typu stroje je určeno:

- s jakým typem dat stroj pracuje
- jak jsou tato data v paměti organizována

Podle typu algoritmu a typu analýzy, kterou chceme provádět, se můžeme rozhodnout, zda má smysl mezi obsah paměti zahranout i místa

- odkud se čtou vstupní data
- kam se zapisují výstupní data

Výpočet algoritmu

Konfigurace — popis celkového stavu stroje v nějakém okamžiku během výpočtu

Příklad: Konfigurace tvaru

$$(q, mem)$$

kde

- q — aktuální řídící stav
- mem — představuje aktuální obsah paměti stroje — jaké hodnoty jsou momentálně přiřazeny jednotlivým proměnným.

Příklad obsahu paměti mem:

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

Výpočet algoritmu

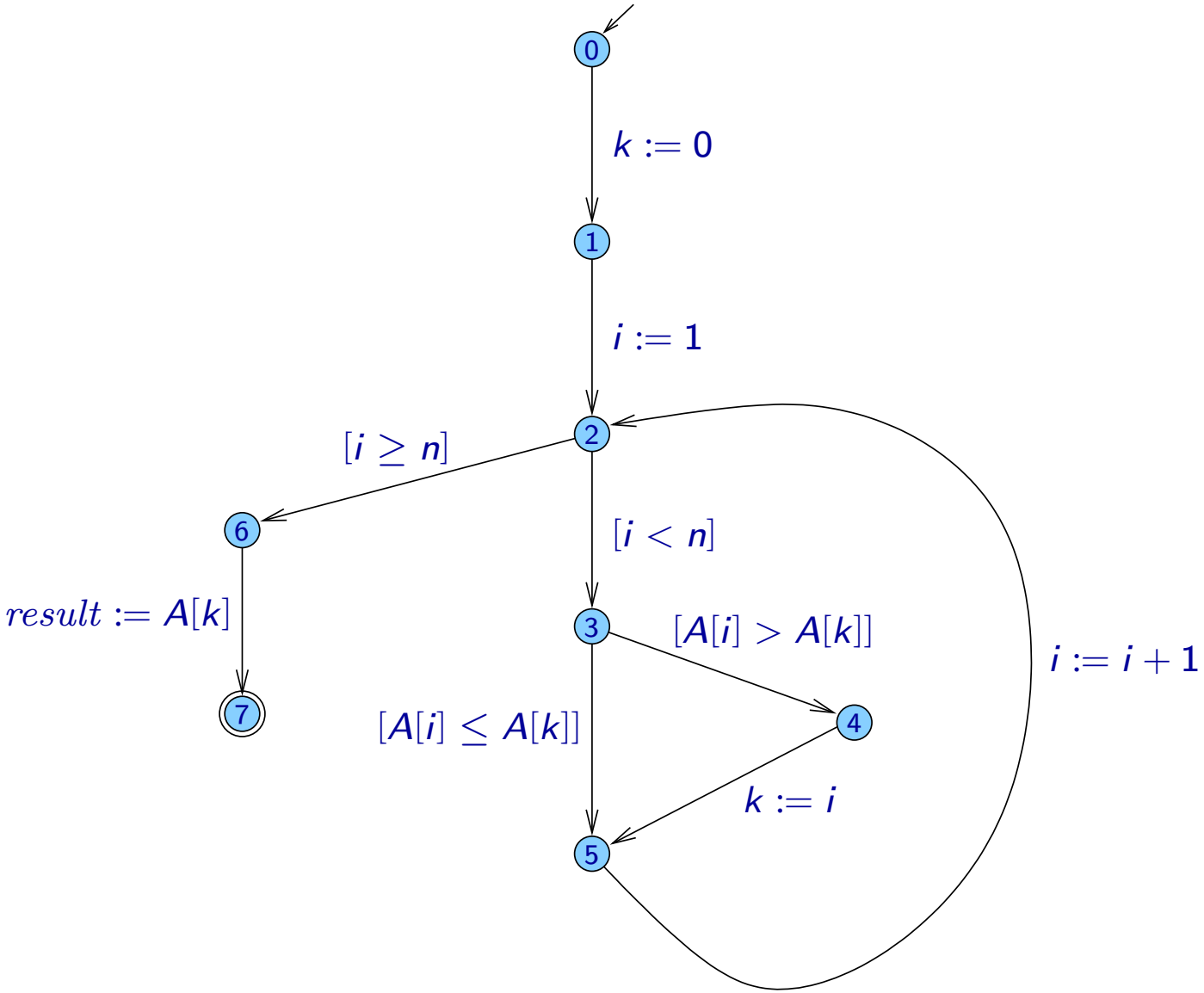
Příklad konfigurace:

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

Výpočet stroje $\mathcal{M}$ provádějícího algoritmus Alg, kde zpracovává vstup w, je posloupnost konfigurací.

- Začíná se v **počáteční konfiguraci**.
- Každým krokem se přejde z jedné konfigurace do druhé.
- Výpočet končí v **koncové konfiguraci**.

Výpočet algoritmu



Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Příklad: Výpočet, kde algoritmus FIND-MAX zpracovává vstup, kde $A = [3, 8, 1, 3, 6]$ a $n = 5$.

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

Výpočet algoritmu

Provedením instrukce I se přejde z konfigurace α do konfigurace α':

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

Výpočet může být:

- **Konečný**:

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

  kde α_t je koncová konfigurace

- **Nekonečný**:

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

Výpočet algoritmu

Výpočet je možné popsat dvěma různými způsoby:

- jako posloupnost konfigurací $\alpha_0, \alpha_1, \alpha_2, \ldots$
- jako posloupnost provedených instrukcí $l_0, l_1, l_2, \ldots$

Korektnost algoritmu

Algoritmy slouží k řešení **problémů**.

- **Problém** — specifikace toho, **co** má algoritmus dělat:
  - Popis vstupu
  - Popis výstupu
  - Vztah mezi vstupy a výstupy
- **Algoritmus** — konkrétní postup, **jak** při výpočtu postupovat

Korektnost algoritmu

Příklad: Problém nalezení maximálního prvku v poli:

Vstup: Pole A indexované od nuly a číslo n udávající počet prvků v tomto poli, přičemž se předpokládá, že $n \geq 1$.

Výstup: Hodnota $result$, která je hodnotou maximálního prvku v poli A, tj. hodnota $result$, pro kterou platí:

- $A[j] \leq result$ pro všechna $j \in \mathbb{N}$, kde $0 \leq j < n$, a
- existuje $j \in \mathbb{N}$ takové, že $0 \leq j < n$ a $A[j] = result$.

Instance problému — konkrétní vstup, např.

$$A = [3, 8, 1, 3, 6], \; n = 5.$$

Pro tuto instanci je výstupem hodnota 8.

Korektnost algoritmu

Definice

Algoritmus Alg **řeší** problém P, jestliže pro **každou** instanci w problému P jsou splněny následující dvě podmínky:

(a) Výpočet algoritmu Alg nad vstupem w se po konečném počtu kroků (korektně) zastaví.

(b) Algoritmus Alg vygeneruje pro vstup w výstup, který odpovídá podmínkám kladeným na výstup ve specifikaci problému P.

Algoritmus, který řeší problém P, je korektním řešením tohoto problému.

Korektnost algoritmu

Algoritmus Alg **není** korektním řešení problému P, jestliže existuje vstup w takový, že při výpočtu nad tímto vstupem nastane některá z následujících chyb:

- provedení nějaké chybné nepovolené operace (přístup k prvku pole mimo povolený rozsah indexů, dělení nulou, ...),
- vygenerovaný výstup neodpovídá podmínkám specifikovaným v zadání problému P,
- výpočet se nikdy nezastaví.

Testování — spustění algoritmu nad různými vstupy a zkontrolování, zda se algoritmus pro tyto vstupy chová „správně“.

Testování může prokázat přítomnost chyb, ale ne to, že se algoritmus chová korektně pro **všechny** vstupy.

Korektnost algoritmu

Důkaz korektnosti algoritmu je obecně vhodné rozdělit na dvě části:

- Zdůvodnění toho, že algoritmus pro žádný vstup nikdy neudělá nic „špatně“:
  - během výpočtu nedojde k žádné chybné operaci
  - pokud program skončí, výstup bude „správně“
- Zdůvodnění toho, že se algoritmus pro každý vstup po konečném počtu kroků zastaví.

Korektnost algoritmu

Invariant — podmínka, která musí být v určitém místě kódu algoritmu vždy (tj. ve všech možných výpočtech pro všechny možné vstupy) splněna v okamžiku, kdy algoritmus tímto místem prochází.

Řekneme, že konfigurace α je **dosažitelná**, jestliže existuje vstup w takový, že je α jednou z konfigurací, kterými algoritmus Alg projde při výpočtu nad vstupem w.

Pokud je algoritmus reprezentován ve formě grafu řídícího toku, můžeme pro **řídící stav** q (tj. vrchol grafu) specifikovat invarianty, které platí v každé dosažitelné konfiguraci, kde je řídícím stavem q.

Invarianty

Invarianty můžeme zapisovat formulemi predikátové logiky:

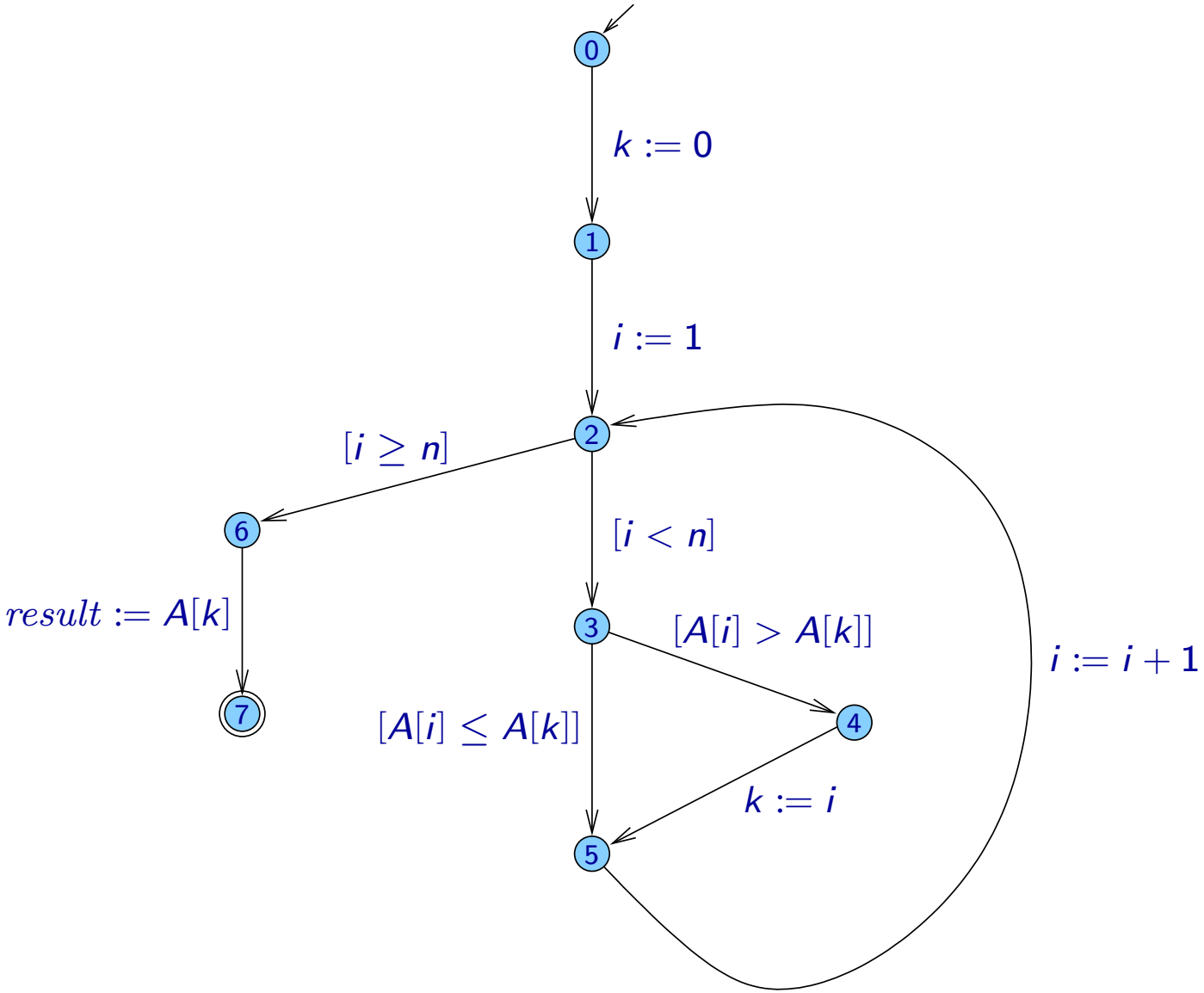
- **volné** proměnné odpovídají proměnným programu
- **valuace** je dána hodnotami proměnných programu v dané konfiguraci

Příklad: Formule

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

bude platit například v konfiguraci, kde proměnná i má hodnotu 5 a proměnná n má hodnotu 14.

Invarianty



Invarianty

Příklady invariantů:

- invariant v řídícím stavu q zapíšeme formulí φ_q

Invarianty v jednotlivých řídících stavech (zatím jen hypotézy):

- 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: $(n \geq 1) \wedge (i = n) \wedge (0 \leq k < n)$

Invarianty

Příklady invariantů:

- invariant v řídícím stavu q zapíšeme formulí φ_q

Invarianty v jednotlivých řídících stavech (zatím jen hypotézy):

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

Invarianty

Zkontrolování toho, že invarianty opravdu platí:

- Pro každou instrukci algoritmu je třeba zkontrolovat, zda za předpokladu, že bude platit příslušný invariant před provedením této instrukce, bude platit i příslušný invariant po provedení této instrukce.

Předpokládejme algoritmus ve formě grafu řídícího toku:

- hrany odpovídají instrukcím
- vezměme si hranu ze stavu q do stavu q' označenou instrukcí I
- řekněme, že (zatím neověřené) invarianty pro stavy q a q' jsou vyjádřeny formulemi φ a φ'
- pro tuto hranu musíme zkontrolovat, že pro všechny konfigurace $\alpha = (q, mem)$ a $\alpha' = (q', mem')$ takové, že $\alpha \xrightarrow{I} \alpha'$, platí, že pokud
  - v konfiguraci α platí φ,

  pak
  - v konfiguraci α' platí φ'

Invarianty

Zkotrolování instrukcí, které jsou testy podmínek:

- hrana označená testem podmínky $[B]$

Obsah paměti se nemění.
Stačí ověřit, že platí implikace

$$(\varphi \wedge B) \rightarrow \varphi'$$

Poznámka: Příslušná implikace musí platit pro všechny možné hodnoty proměnných.

Příklad: Předpokládáme, že se formulích objevují jen proměnné n, i, k, a že hodnotami těchto proměnných mohou být jen celá čísla:

$$(\forall n \in \mathbb{Z})(\forall i \in \mathbb{Z})(\forall k \in \mathbb{Z})\,(\varphi \wedge B \rightarrow \varphi')$$

Invarianty

Zkotrolování instrukcí, které přiřazují hodnoty proměnným (mění obsah paměti):

- hrana označená přiřazením $x := E$

φ'' — formule, kterou dostaneme z formule φ' přejmenováním všech volných výskytů proměnné x na proměnnou x'

Je třeba ověřit platnost implikace

$$(\varphi \wedge (x' = E)) \rightarrow \varphi''$$

Příklad: Přiřazení $k := 3 * k + i + 1$:

$$(\forall n \in \mathbb{Z})(\forall i \in \mathbb{Z})(\forall k \in \mathbb{Z})(\forall k' \in \mathbb{Z})\,(\varphi \wedge (k' = 3 * k + i + 1) \rightarrow \varphi'')$$

Invarianty

Dokončení ověření toho, že algoritmus pro nalezení maximálního prvku v poli vrací správný výsledek (za předpokladu, že skončí):

- 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: $\varphi_7 \wedge (result = A[k]) \wedge (\forall j \in \mathbb{N})(0 \leq j < n \rightarrow A[j] \leq result) \wedge (\exists j \in \mathbb{N})(0 \leq j < n \wedge A[j] = result)$

Invarianty

Často není třeba specifikovat invarianty ve všech řídících stavech, ale je v některých „důležitých“ — zejména stavy, kde se vstupuje do nebo vystupuje z cyklů:

Je pak třeba ověřit:

- Že invariant platí před vstupem do cyklu.
- Že pokud invariant platí před provedením cyklu, tak bude platit i po jeho provedení.
- Že invariant platí při opuštění cyklu.

Invarianty

Příklad: V algoritmu FIND-MAX je takovým „důležitým“ stavem stav 2.

Ve stavu 2 platí:

- $n \geq 1$
- $1 \leq i \leq n$
- $0 \leq k < i$
- Pro všechna j taková, že $0 \leq j < i$, platí $A[j] \leq A[k]$.

Konečnost výpočtu

Dva možné případy, jak může vypadat nekonečný výpočet:

- nějaká konfigurace se zopakuje — následující konfigurace se opakují stále dokola
- objevují se stále nové a nové konfigurace

Konečnost výpočtu

Jeden z běžných způsobů dokazování toho, že se algoritmus zaručeně pro každý vstup po konečném počtu kroků zastaví:

- každé (dosažitelné) konfiguraci přiřadit hodnotu z nějaké vhodně zvolené množiny W
- na množině W definovat uspořádání $\leq$ takové, že ve W neexistují nekonečné (ostře) klesající posloupnosti
- ukázat, že s provedením každé instrukce se hodnota přiřazená konfiguraci zmenšuje, tj. pro $\alpha \xrightarrow{I} \alpha'$ je

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

($f(\alpha)$, $f(\alpha')$ jsou hodnoty z množiny W přiřazené konfiguracím α a α')

Konečnost výpočtu

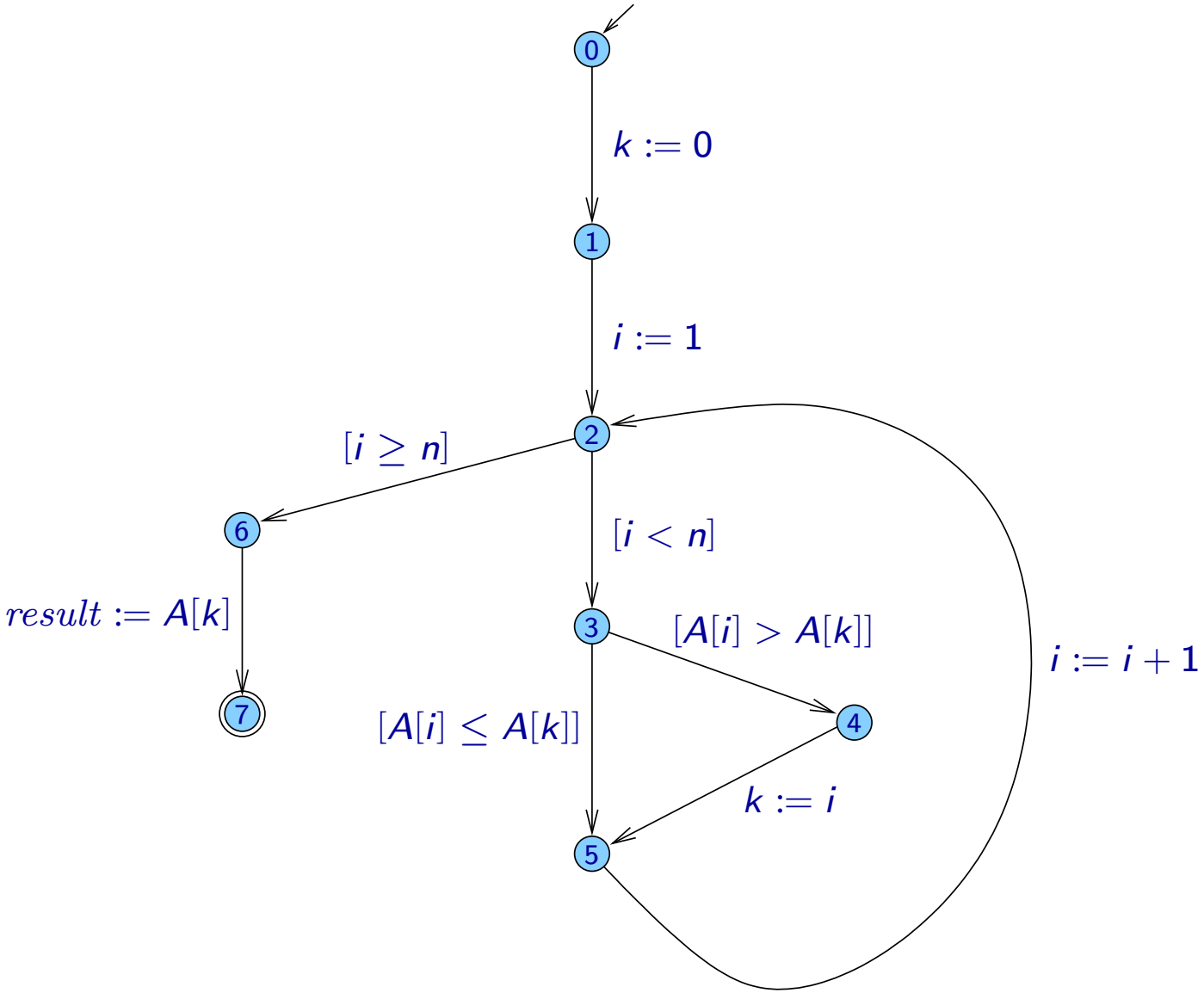
Jako množinu W je možno použít například:

- Množinu přirozených čísel $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ s uspořádáním $\leq$.
- Množinu vektorů přirozených číslel s lexikografickým uspořádáním, tj. s uspořádáním, kde vektor $(a_1, a_2, \ldots, a_m)$ je menší než vektor $(b_1, b_2, \ldots, b_n)$, jestliže
  - existuje i takové, že $1 \leq i \leq m$ a $i \leq n$, kde $a_i < b_i$ a pro všechna j taková, že $1 \leq j < i$, platí $a_j = b_j$, nebo
  - $m < n$ a pro všechna j taková, že $1 \leq j \leq m$, je $a_j = b_j$.

  Například $(5, 1, 3, 6, 4) < (5, 1, 4, 1)$ a $(4, 1, 1) < (4, 1, 1, 3)$.

  Poznámka: Počet prvků vektorů musí být omezen nějakou konstantou.

Konečnost výpočtu



Konečnost výpočtu

Příklad: Vektory přiřazené jednotlivým konfiguracím:

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