DYNAMICKÉ PROGRAMOVÁNÍ I

Karel Horák, Petr Ryšavý

27. dubna 2016

Katedra počítačů, FEL, ČVUT

Příklad 1

Naimplementujte výpočet n-tého prvku Fibonacciho posloupnosti. Použijte

1. rekurzi,
2. dynamické programování.

Co bude rychlejší? Proč? Porovnejte časy běhu.

Fibonacciho posloupnost je definovaná rekurentním vztahem

$$F(n) = F(n-1) + F(n-2),$$

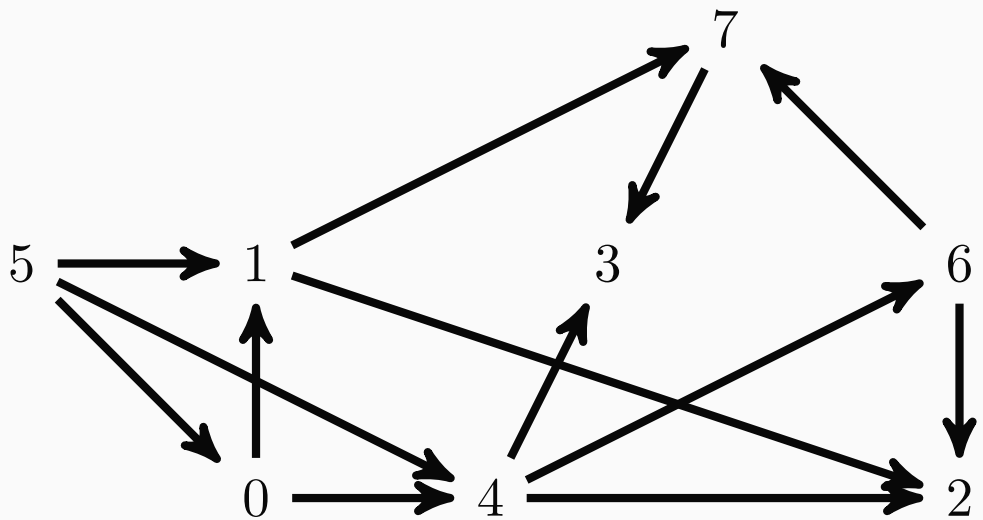
kde $F(0) = 0$ a $F(1) = 1$.

Řešení 1

Rychlejší bude dynamické programování, protože se oproti rekurzi vyhýbá opakovaným výpočtům.

Příklad 2

Určete topologické očíslování následujícího DAGu



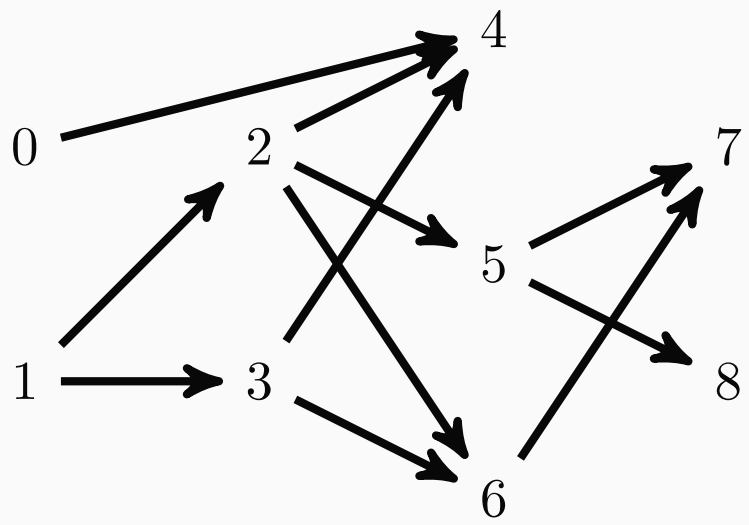
Řešení 2

Topologické očíslování daného DAGu je například

`5, 0, 1, 4, 6, 7, 3, 2`

Příklad 3

Navrhněte algoritmus pomocí něhož spočtete počet cest v DAGu. Poté ho aplikujte pro nalezení počtu cest v následujícím grafu:



Řešení 3

Pro každý vrchol je počet cest, které do něj vedou rovný součtu počtu cest do rodičů plus jedna za cestu o nulové délce. Platí tedy rekurzivní vztah

$$P(v) = 1 + \sum_{u:(u,v)\in E} P(u).$$

Vrcholy stačí procházet v pořadí topologického očíslování a aplikovat tento vztah. Pro získání výsledku pak stačí sečíst výsledek přes všechny vrcholy:

$$\#\text{paths in } G = \sum_{v\in V} P(v).$$

V DAGu na obrázku je počet cest 33.

Příklad 4

Navrhněte způsob jak lze pomocí dynamického programování spočíst kombinační číslo $\binom{n}{k}$ v čase $\mathcal{O}((n+k)^2)$.

Nápověda: zkuste použít některý z rekurzivních vztahů, které pro kombinační čísla platí.

Řešení 4

Vhodné je použít Pascalova trojúhelníku a vztahu

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}.$$

Příklad 5

Určujeme počet všech binárních vektorů délky N s vlastností, že v nich nikdy nestojí dvě (nebo více) jedničky těsně vedle sebe (vektor `0100100101` je přípustný, vektory `01100`, `1110011` přípustné nejsou).

Navrhněte algoritmus pro řešení tohoto problému a naimplementujte ho.

Řešení 5

Stačí nám pamatovat si pro každou délku n počet řetězců, které končí jednotkou $o(n)$ a nulou $z(n)$ a mají danou délku n. Platí rekurzivní vztahy:

$$z(n) = o(n-1) + z(n-1),$$
$$o(n) = z(n-1).$$

Za řetězce končící nulou lze přidat nulu nebo jednotku. Za řetězce končící jednotkou pouze nulu. Jako inicializaci lze použít $z(1) = o(1) = 1$.

Příklad 6

Každá hrana DAG G je obarvena buď modrou nebo zelenou barvou. Popište, jak metodou DP najdete co nejdelší cestu v G takovou, že se na ní pravidelně střídají barvy hran, to jest, barvy každých dvou bezprostředně navazujících hran na této cestě musí být různé.

Řešení 6

Stačí si pamatovat dvě nejdelší cesty, které vedou do každého vrcholu. Jednu co končí modrou hranou a jednu co zelenou.

Příklad 7

Popište, jak je nutno modifikovat standardní algoritmus hledání nejdelší cesty v DAG , aby našel nejdelší cestu v DAG za okolností:

1. Každý uzel DAG je ohodnocen kladným reálným číslem a hrany ohodnoceny nejsou.
2. Každý uzel DAG je ohodnocen libovolným reálným číslem a hrany ohodnoceny nejsou.
3. Každý uzel i každá hrana DAG jsou ohodnoceny libovolným reálným číslem.

Řešení 7

Algoritmus bude stejný, pouze upravíme funkci, která počítá cenu cesty, aby odpovídala zadaným podmínkám.