Jméno: Petr Valenta **UČO:** 487561

0007 *list* 1 *učo* 487561 *body*

Oblast strojově snímaných informací. Své učo a číslo listu vyplňte zleva dle vzoru číslic. Jinak do této oblasti nezasahujte. 0 1 2 3 4 5 6 7 8 9

3. [20 bodů] Pan Dřevorubec má za úkol rozřezat dlouhé poleno na předem určených pozicích. Cena každého řezu je určena délkou polena, které se řeže. Pomozte panu Dřevorubcovi určit, jak postupovat při řezání tak, aby splnil úkol za co nejmenší cenu.
Vstupem problému Dřevorubce je posloupnost přirozených čísel $p_0, p_1, \ldots, p_{n+1}$ taková, že $p_0 = 0$, $p_{n+1} = L$, a $p_0 < p_1 < \ldots < p_{n+1}$. Čísla $p_1, \ldots, p_n$ určují pozice, na kterých má být poleno délky L rozřezáno. Použitím principů dynamického programování navrhněte algoritmus polynomiální složitosti, který určí optimální pořadí řezání.

Příklad. Poleno délky 10 má být rozřezáno na pozicích 1, 3 a 7. Jestliže pan D. udělá řez nejdříve na pozici 1, pak 3 a nakonec 7, tak celková cena bude 10 + 9 + 7. Jestliže zvolí pořadí 3, 1 a 7, tak cena bude 10 + 3 + 7.

Rozdělení na podproblémy

Podproblém je posloupnost přirozených čísel, která je částí vstupní posloupnosti. $p_x, p_{x+1}, \ldots, p_y$ taková, že $0 \leq x \leq y$, $x \leq y \leq n+1$, $p_{n+1} = L$. Čísla $p_{x+1}, \ldots, p_{k-1}$ určují pozice, na kterých má být poleno délky $p_y - p_x$ rozřezáno. Posloupnost $p_x, \ldots, p_y$ budu zapisovat $p_{x,y}$. Množinu všech možných posloupností o délce $k = y - x$, které je možné vytvořit ze vstupní posloupnosti, budu zapisovat jako P_k.

Rekurentní rovnice

$$OPT(p_{x,y}) = \begin{cases} 0, & y - x = 1 \\ p_y - p_x + \min\limits_{x<c<y} (OPT(p_{x,c}) + OPT(p_{c,y})), & y - x > 1 \end{cases}$$

Pořadí podproblémů

Podproblémy $p_{x,y}$ budeme řešit vzestupně, dle délky posloupnosti $k = y - x$. Na pořadí podproblémů stejné velikosti nezáleží.

Algoritmus

function `cut`($p_{0,n+1}$)
  for *all* $p_{x,y}$ *in* P_2 **do**
    values[$p_{x,y}$], *cuts*[$p_{x,y}$] $\leftarrow 0$
  for k *in* $[3, n+1]$ **do**
    for *all* $p_{x,y}$ *in* P_k **do**
      $minValue \leftarrow \infty$
      $minCut \leftarrow 0$
      for *all* c , $x < c < y$ **do**
        $value \leftarrow values[p_{x,c}] + values[p_{c,y}]$
        if $value < minValue$ **then**
          $minValue \leftarrow value$
          $minCut \leftarrow c$
      $values[p_{x,y}] \leftarrow minValue + p_y - p_x$
      $cuts[p_{x,y}] \leftarrow minCut$
  `solution`(*values, cuts,* $p_{0,n+1}$)

Sestavení řešení

U každého vyřešeného podproblému $p_{x,y}$ si kromě hodnoty jeho řešení uložím navíc i c — index řezu p_c, který byl k dosažení řešení využit. K získání řešení procházím po problémech shora dolů. -

začnu problémem $p_{0,n+1}$, poznamenám si zvolený řez, rozdělím problém na podproblémy $p_{a,c}$ a $p_{c,b}$ a rekurzivně opakuji.

```
function solution(values, cuts, p_{a,b})
    if b − a ≥ 3 then
        c ← cuts[p_{a,b}]
        output ← p_c
        solution(values, cuts, p_{a,c})
        solution(values, cuts, p_{c,b})
```

Zdůvodnění korektnosti

Optimální strukturou řešení je binární strom. Kořen reprezentuje původní poleno, uzly jsou části, na které bylo rozřezáno. Poleno, které je listem už není možné dále řezat. Počet listů je roven počtu řezů +1. Počet řezů je roven dvojnásobku počtu hran. Listy jsou báze. Indukčním krokem je vytvoření stromu ze dvou podstromů.

Jestliže hledám takové řezy, jejichž součet bude co nejmenší, pak skutečně optimální řešení $SOPT$ ho najde.

Předpokládejme, že do jisté volby se $SOPT$ a OPT rozhodovaly stejně. V této volbě se poprvé neshodly. OPT vybere nejmenší hodnotu z optimálních řešení všech podproblémů, což znamená, že $SOPT$ si zvolilo stejně drahý, ale jiný podproblém, nebo dražší podproblém. V případě, že si $SOPT$ zvolil dražší podproblém, pak je striktně horší. V případě, že si zvolil jiný, stejně ohodnocený podproblém, na základě exchange argumentu tvrdím, že si mohl zvolit stejný podproblém jako OPT a výsledek by to nezměnilo.

Analýza složitosti

Algoritmus se sestává z množin podproblémů P_1 , ... , P_n. Velikost množiny P_k je $n - k + 1$. V každém podproblému z množiny P_k je možné vykonat $k - 1$ řezů, každý řez vytvoří dvě polena, tj. dva dotazy. Celkový počet dotazů je tedy:

$$\sum_{k=1}^{n} (n - k + 1)(2(k - 1)) = \frac{n^3}{3} - \frac{n}{3}$$

Složitost sestavení řešení je zanedbatelná. Celková složitost je $\mathcal{O}(n^3)$ — kubická.

Algoritmus si ukládá pole, které pro každý podproblém obsahuje hodnotu jeho optimálního řešení a řez, pomocí které bylo tohoto řešení dosaženo. Velikost těchto informací nezávisí na velikosti vstupu, pojmenujeme si je tedy C. Velikost pole je tedy:

$$\sum_{k=1}^{n} (n - k + 1) \times C = (\frac{n^2}{2} - \frac{n}{2}) \times C$$

Prostorová náročnost je $\mathcal{O}(n^2)$ — kvadratická.