Programovací strategie: Maximalizujte svůj zisk

Tomáš Maršálek

6. října 2011

1 Zadání

Se soutěžemi se v televizi roztrhl pytel. Je na čase, abyste si vydělali také vy. Otázky jsou pro vás naprosto triviální a přichází čas vybrat si odměnu. Přivedou před vás n asistentek. Asistentky se drží za ruce a každá má na krku pověšenou ceduli s celočíselnou sumou, kterou získáte (kladné číslo) nebo naopak ztratíte (záporné číslo), když si tuto asistentku vyberete. Můžete si vybrat jakýkoliv počet asistentek, ale jejich spojené ruce smíte přerušit max. na dvou místech, tedy posloupnost odměn, kterou si vyberete, musí být spojitá, ale může být libovolně dlouhá. Asistentky nezískáváte, pouze sumy výher, takže jiná než finanční kritéria není třeba brát v úvahu.

2 Naivní řešení

Jednoduše vyzkoušíme všechny souvislé podposloupnosti a vybereme tu s nejvyšší sumou. Tedy pro všechny možné délky podposloupnosti a pro všechny jejich možné posuny zkoušíme, jestli je jejich suma lepší než dosavadní.

```
def naivniZisk(h):
        n = len(h)
        max_zisk, max_i, max_delka = 0, 0, 0
        pocatecni_zisk = 0
        for delka in range(1, n + 1):
                pocatecni_zisk += h[delka - 1]
                zisk = pocatecni_zisk

                for i in range(n - delka):
                        if zisk > max_zisk:
                                max_zisk, max_i, max_delka = zisk, i, delka
                        zisk += h[delka + i] - h[i]

                if zisk > max_zisk:
                        max_zisk, max_i, max_delka = zisk, n - delka, delka
        return max_zisk, max_i, max_delka
```

2.1 Složitost

Máme dvojitě vnořený cyklus, očekáváme kvadratickou složitost.

$$\begin{aligned} T(N) &= (\Theta(1) + (N-1)) + (\Theta(1) + (N-2)) + \ldots + (\Theta(1) + (N-N)) \\ &= \Theta(N) + N^2 - \frac{N}{2}(N+1) \\ &= \Theta(N^2) \end{aligned}$$

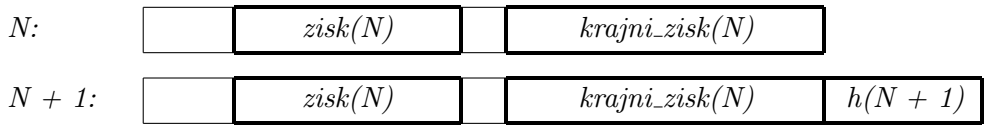
3 Dynamické programování

Definujme funkci zisk, kterou chceme maximalizovat pro posloupnost délky N

$$zisk(N) = nejvyšší\ suma\ v\ celé\ posloupnosti$$

Abychom našli algoritmus založený na dynamickém programování, hledáme rekurzivní vyjádření této funkce. Dodefinujeme ještě pomocné funkce

$$krajni\_zisk(N) = nejvyšší\ zisk\ z\ posloupnosti\ u\ pravého\ kraje$$
$$h(N) = hodnota\ N-tého\ prvku$$



Vyjádříme-li $zisk(N+1)$ pomocí $zisk(N)$, mohou nastat tři případy.

1. krajní zisk nám vůbec nepomohl a $zisk(N+1) = zisk(N)$

   N + 1: | | *zisk(N)* | |

2. krajní zisk s novou hodnotou je větší než $zisk(N)$, tedy
   $zisk(N+1) = krajni\_zisk(N) + h(N+1)$

   N + 1: | | *krajni_zisk(N)* | *h(N + 1)* |

3. $krajni\_zisk(N)$ je záporný, tedy je výhodnější vzít pouze poslední hodnotu.
   $zisk(N+1) = h(N+1)$

   N + 1: | | *h(N + 1)* |

Rekurzivním vyjádřením pak dostáváme základ pro algoritmus. Rekurzi pouze obrátíme na dynamické programování.

$$\begin{aligned} zisk(N+1) = \max(&zisk(N),\\ &krajni\_zisk(N) + h(N+1),\\ &h(N+1)) \end{aligned}$$

Stejně jako u předešlého algoritmu si zaznamenáváme hodnotu zisku, a polohu podposloupnosti (počáteční index a délku). Náročnost je očividně $\Theta(N)$.

```
def dpZisk(h):
        n = len(h)
        max_zisk, max_i, max_delka = 0, 0, 0
        krajni_zisk, krajni_i, krajni_delka = 0, 0, 0
        for i in range(n):
                if krajni_zisk > 0:         # Případ 2
                        krajni_zisk += h[i]
                        krajni_delka += 1
                else:                       # Případ 3
                        krajni_zisk ,krajni_i, krajni_delka = h[i], i, 1

                if krajni_zisk > max_zisk: # Případ 1 (obráceně)
                        max_zisk, max_i, max_delka = krajni_zisk, krajni_i, krajni_delka
        return max_zisk, max_i, max_delka
```

4 Implementace

Program `zisk.py` je v jazyce python 2.7.2. Vstupní data jsou kladná a záporná celá čísla. Výstupem je hodnota zisku, index první asistentky (od nuly), která začíná vybranou posloupnost a délka této posloupnosti. Jsou implementována řešení oběma metodami.

4.1 Příklad použití

```
# vygenerovani vstupnich dat:  300 cisel od -500 do 499
> test.data
for i in $(seq 1 300); do echo $(($RANDOM % 1000 - 500)) >> test.data; done

cat test.data | ./zisk.py
```