Домашно №2 по ДАА на Компютърни Науки

Камен Димитров Младенов, Група 6

7 Април 2023

Задача 1. Решение

а) Ляв фрагмент

Цикъла ще извърши точно $m.n = n.n.n = n^3$ итерации. Тоест за $c > 0, f(n) = n^3 + c \asymp n^3$ по дефиниция.

б) Централен фрагмент

Алгоритъма ще приключи директно ако $n < 10$. Иначе се изпълнява останалата част на алгоритъма.
Условието на цикъла на ред 4 никога няма да е вярно, понеже $6 \not> 8$.

Тялото на цикъла с условие на ред 7 ще се изпълни точно един път: при първо достигане на условието $n - j = n - 6 \geq 10 - 6 = 4 > 1$, на ред 8 j става n и на второто достигане на условието $n - j = n - n = 0 \not> 1$. От това единствено изпълнение имаме рекурентно извикване:

$$g(n-1) + g(n-2) \tag{1}$$

Тялото на цикъла с условие на ред 10 ще се изпълни точно един път: към момента $j = n$ и при първо достигане на условието, $n \geq n$. От ред 11 j става 2 и при второ достигане на условието, $2 = j \not\geq n$ понеже $n \geq 10$. От това единствено изпълнение имаме рекурентно извикване:

$$g(n-j) = g(n-2) \tag{2}$$

От (1) и (2): $g(n) = g(n-1) + 2g(n-2) + c$

Характеристично уравнение: $x^n = x^{n-1} + 2x^{n-2} \quad x^2 - x - 2 = 0 \quad$ с корени $\{-1, 2\}_M$

Нехомогенната част е константа и няма корени. $\quad g(n) = c_1(-1)^n + c_2 2^n \implies g(n) \asymp 2^n$

в) Десен фрагмент

Алгоритъма ще приключи директно ако $n < 2$. Иначе се изпълнява останалата част на алгоритъма.
От редове 4 и 7 имаме две еднакви рекурентни извиквания:

$$h\left(\frac{n}{3}\right) \tag{3}$$

Ще покажем, че тялото на цикъла с условие на ред 5 ще се изпълни точно

$$\lfloor \lg n \rfloor \text{ на брой пъти.} \tag{4}$$

Ако допуснем, че $n = 2^k$ за някое $k \in \mathbb{N}$, то тогава i последователно ще придобие стойностите $2^1, 2^2, ..., 2^k$, тоест цикъла ще се изпълни $k = \log_2 n \in \mathbb{N}$ на брой пъти.
Ако $n \neq 2^k$ за което и да е $k \in \mathbb{N}$, то тогава съществува $k' \in \mathbb{N}$, такова че $2^{k'} < n < 2^{k'+1}$ и тогава i придобива стойностите $2^1, 2^2, ..., 2^{k'}$. Понеже $k' < \lg n < k' + 1$, то цикъла ще се изпълни $k' = \lfloor \lg n \rfloor \in \mathbb{N}$ на брой пъти.

От (3) и (4): $h(n) = 2\,h\left(\frac{n}{3}\right) + \lfloor \lg n \rfloor$

Тогава $a = 2, b = 3, f(n) = c \quad$ За $\varepsilon = \frac{1}{2}, f(n) \preceq \frac{n^{\log_3 2}}{\sqrt{n}}, \log_3 2 - \frac{1}{2} > 0 \xrightarrow[\text{Мастър Теоремата}]{\text{1 случай на}} h(n) \asymp n^{\log_3 2}$

Задача 2. Решение

а) Инверсен вектор

Инверсния вектор е $(4, 1, 1, 0, 2, 3, 0, 0, 0)$

б) Алгоритъм

От условието, нека векторите (масивите) `invvec` и `result` да се индексират от 1 до n, където `invvec.size()` $= n$ и на ред 2, `result` се създава със същия размер като `invvec`:

```
1  vector<int> rebuild(vector<int> invvec) {
2      vector<int> result(invvec.size());
3      for (int i = 1; i <= result.size(); i++)
4          result[i] = i;
5      for (int i = invvec.size(); i >= 1; i--) {
6          for (int j = i; invvec[i] > 0; j++) {
7              swap(result[j], result[j+1]);
8              invvec[i]--;
9          }
10     }
11     return result;
12 }
```

От дефинициите, инверсният вектор съдържа броя инверсии за всяко j от 1 до n. Алгоритъмът `rebuild` създава сортирания вектор `result` на редове 2-4 и след това в цикъла на редове 5-10 в `result` се добавят инверсиите, определени от `invvec`. Инверсния вектор се итерира в обратен ред TODO

Цикъла на редове 3-4 се движи от `i = 1` до `i = n`, като в него се извършват единствено константни операции, тоест сложността му по време е $\Theta(n)$.

Цикъла на редове 5-10 се движи от `i = n` до `i = 1`, като за всяка негова итерация се изпълнява цикъла на редове 5-9. Цикъла на редове 5-9 се движи `invvec[i]` на брой пъти, като в него се извършват единствено константни операции. Максималният брой инверсии за дадено i е $n - i$ (колкото числа от 1 до n са по-големи от i).

Тоест, цикъла на редове 5-10 ще се изпълнява най-много:

$$\sum_{i=1}^{n}(n-i) = n^2 - \sum_{i=1}^{n} i = n^2 - \frac{n(n+1)}{2} = n^2 - \frac{1}{2}n^2 + \frac{1}{2}n = \frac{1}{2}n^2 + \frac{n}{2} \text{ на брой пъти}$$

Което означава, че максималната му сложност по време е $O(n^2)$. Следователно, `rebuild` $= O(n^2)$.

Задача 3. Решение

а) Алгоритъм

От условието, нека входния масив на алгоритъма `M` да е индексирана по редове от 1 до n и по колони от 1 до m, където `M.rows()` връща n и `M.cols()` връща m.

```
1  int isinteresting(array<int, int> M) {
2      for (int i = 2; i <= M.rows(); i++) {
3          for (int j = 2; j <= M.cols(); j++) {
4              if (M[i-1, j-1] + M[i, j] > M[i-1,j] + M[i,j-1])
5                  return 0;
6          }
7      }
8      return 1;
9  }
```

Нека $i \in \{2, ..., n\}, j \in \{2, ..., m\}$.

I. Ако $M[i-1,j-1]+M[i,j] \leq M[i-1,j]+M[i,j-1]$, и
$M[i-1,j-2]+M[i,j-1] \leq M[i-1,j-1]+M[i,j-2]$

$$\begin{aligned}\implies M[i-1,j-2]+M[i,j] &\leq (M[i-1,j-1]+M[i,j-2]-M[i,j-1]) \\ &\quad +(M[i-1,j]+M[i,j-1]-M[i-1,j-1]) \\ &= M[i,j-2]+M[i-1,j]\end{aligned}$$

II. Ако $M[i-1,j-1]+M[i,j] \leq M[i-1,j]+M[i,j-1]$, и
$M[i-2,j-1]+M[i-1,j] \leq M[i-2,j]+M[i-1,j-1]$

$$\begin{aligned}\implies M[i-2,j-1]+M[i,j] &\leq (M[i-2,j]+M[i-1,j-1]-M[i-1,j]) \\ &\quad +(M[i-1,j]+M[i,j-1]-M[i-1,j-1]) \\ &= M[i-2,j]+M[i,j-1]\end{aligned}$$

Твърдение: Ако $M[i-1,j-1]+M[i,j] \leq M[i-1,j]+M[i,j-1]$ е изпълнено, то за всяко $1 \leq p \leq i-1, 1 \leq q \leq j-1$, е изпълнено $M[p,q]+M[i,j] \leq M[p,j]+M[i,q]$.
Доказателство: Със силна индукция по i и j.

База: $i=2, j=2$ и $M[1,1]+M[2,2] \leq M[1,2]+M[2,1]$ е изпълнено. Разглеждаме $1 \leq p \leq 1, 1 \leq q \leq 1 \implies p=1, q=1$, тогава $M[p,q]+M[2,2]$ е винаги изпълнено.

Индукционно предположение: Допускаме, че за всички стойности до i и j включително твърдението е вярно.

Индукционна стъпка: Следните 3 случая са изчерпателни:

1) Разглеждаме i и $j+1$, за които е изпълнено $M[i-1,j]+M[i,j+1] \leq M[i-1,j+1]+M[i,j]$.
От I., следва че $M[i-1,j-1]+M[i,j+1] \leq M[i-1,j+1]+M[i-1,j-1]$.
От II., следва че $M[i-2,j]+M[i,j+1] \leq M[i-2,j+1]+M[i,j]$.
За $M[i-1,j-1]$ и $M[i-2,j]$ твърдението е съответно вярно при $1 \leq p \leq i-2, 1 \leq q \leq j-2$ и $1 \leq p \leq i-3, 1 \leq p \leq j-1$ по ИП.
Следователно, твърдението е изпълнено за $M[i,j+1]$ при $1 \leq p \leq i-2, 1 \leq q \leq j-1$.

2) Аналогично твърдението е изпълнено при $i+1$ и j

3) Аналогично твърдението е изпълнено при $i+1$ и $j+1$

Външният цикъл "минава" през всеки ред (без първия), а вътрешният през всяка колона (без първата). В тялото на вътрешния цикъл, за всеки елемент от масива правим константата проверка, която вече показахме че правилно зачита интересеността на целия масив. Следователно алгоритъмът работи коректно.

В най-лошия случай, условието на ред 4 никога няма да бъде истина, циклите никога няма да спрът действие преждевременно и ще минем през всеки един елемент от масива. Следователно има сложност по време $O(n.m)$.

б) Доказателство

Ще допуснем противното, тоест че за интересен масив M, $\phi(1) > \phi(2) > \cdots > \phi(n)$. Нека да разгледаме произволни $\phi(i), \phi(i+1)$, където $i > n$. То тогава:

$$M[i,\phi(i+1)]+M[i+1,\phi(i)] \leq M[i,\phi(i)]+M[i+1,\phi(i+1)]$$

Обаче, $M[i,\phi(i)]$ е най-левия най-малък елемент за ред i в масива M и $\phi(i) \neq \phi(i+1)$, тоест $M[i,\phi(i+1)] \not\leq M[i,\phi(i)]$. Аналогично $M[i+1,\phi(i)] \not\leq M[i+1,\phi(i+1)]$.

Следователно, M не интересен масив. $\lightning$

в) Обяснение

От б), знаем че $\phi(i-1) \leq \phi(i) \leq \phi(i+1), 1 < i < n, i \in n_o$. Тоест, за нечетните редове i, ще търсим $\phi(i)$ между стойностите на ϕ за съседните (четни) редове. Също, за ред 1, ще търсим между 1 и $\phi(2)$ (понеже $1 \leq \phi(1) \leq \phi(2)$) и ако n е нечетно, за ред n, ще търсим от $\phi(n-1)$ до m (понеже $\phi(n-1) \leq \phi(n) \leq m$).

Общо имаме $\lceil \frac{n}{2} \rceil$ на брой нечетни реда. За всеки нечетен ред търсенето се извършва върху участък с индекси $s, ..., k \quad 1 \leq s \leq k \leq m$, като за всяко търсене индексите ще са уникални, освен s и k, които може да се препокрият един път (с предходния и следващия). Следователно, общо работата за нечетните редове ще бъде най-много $n - 1 + m$ (препокривания за всички редове, освен последния който минава през целия ред). Тоест, сложността по време на алгоритъма ще бъде $O(n + m)$.