Отговори на задачите от изпита на 18.06.2012 г.

*Това **не са** пълни отговори. Пълни отговори за максимален брой точки би трябвало да са доста по-подробни.*

Зад. 1 Решете следните четири рекурентни отношения чрез мастър теоремата (Master Theorem).

а) $T(n) = 4T\left(\frac{n}{2}\right) + n^2$ б) $T(n) = 3T\left(\frac{n}{2}\right) + \frac{n^2}{(\lg n)^{11}}$

в) $T(n) = 5T\left(\frac{n}{2}\right) + n^2(\lg n)^{11}$ г) $T(n) = 4T\left(\frac{n}{2}\right) + n(\lg n) + \frac{n(\lg n)^9}{\lg\lg n} + n(\lg n)\left(\sqrt[4]{n^2 + n + (\lg n)^9}\right)$

Отговори: а) $T(n) = \Theta(n^2 \lg n)$; б) $T(n) = \Theta\left(\frac{n^2}{(\lg n)^{11}}\right)$; в) $T(n) = \Theta\left(n^{\lg_2 5}\right)$; г) $T(n) = \Theta(n^2)$.

Зад. 2 За всяко от следните четири рекурентни отношения, решете отношението чрез метода с характеристичното уравнение или обяснете защо не може да се реши чрез този метод.

а) $T(n) = T(n-1) + n^2$ б) $T(n) = T(n-1) + n^3 + (7n^2 + 5n - 33)\left(\frac{3}{2}\right)^n$

в) $T(n) = \left(n^2 + 2 - n^{\lg_2 4}\right) T(n-1) + 1$ г) $T(n) = T(n-1) + T(n-2) + \frac{n^3 + 3n^2}{n}$

Отговори: Всичките четири рекурентни отношения се решават с метода с характеристичното уравнение. Във в), $n^2 + 2 - n^{\lg_2 4} = n^2 + 2 - n^2 = 2$. И така,
а) $T(n) = \Theta(n^3)$; б) $T(n) = \Theta\left(n^2\left(\frac{3}{2}\right)^n\right)$; в) $T(n) = \Theta(2^n)$; д) $T(n) = \Theta\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$

Зад. 3 Намерете асимптотичната сложност на всеки от следните три фрагмента от програми.

```
int f(int n) {
 int i, s=2, m=n*n;
 for(i=0; i<m*n; i++)
  s += s;
 return s; }
```

```
int g(int n) {
 if(n < 10) return 1;
 int j=6, s=0;
 while(j > 8) {
  s += g(n-2);
  j --;}
 while(n-j > 1) {
  j = n;
  s += g(n-1) + g(n-2); }
 while(j >= n) {
  j = 2;
  s += g(n-j); } }
```

```
int h(int n) {
 int i, t=0;
 if(n < 2) return 2;
 t += h(n/3);
 for(i = 2; i < n; i <<=1)
  t ++;
 t *= h(n/3);
 return t; }
```

Отговори: Сложността на $f()$ е тривиално $\Theta(n^3)$. Сложността на $g()$ се намира чрез следните разсъждения: първият на `while` не се изпълнява изобщо, вторият `while` се изпълнява точно веднъж, оттам имаме по едно викане `g(n-1)` и `g(n-2)`, третият `while` се изпълнява точно веднъж и имаме още едно викане на `g(n-2)`; следователно, рекурентното отношение за сложността е $T(n) = T(n-1)+2T(n-2)+1$, което има решение $T(n) = \Theta(2^n)$. Сложността на $h()$ се намира чрез следните разсъждения: имаме точно две викания $h\left(\frac{n}{3}\right)$ и освен това логаритмична работа (цикълът `for`); следователно, рекурентното отношение за сложността е $T(n) = 2T\left(\frac{n}{3}\right) + \lg n$, което има решение $T(n) = \Theta\left(n^{\lg_3 2}\right)$.

Зад. 4 Даден е масив $A[1, 2, \ldots, n]$ от цели числа. Предложете алгоритъм, който размества елементите на $A[]$ така, че всички отрицателни числа да са вляво от всички неотрицателни числа. Не се иска полученият масив да бъде сортиран. Опишете решението си в псевдокод. Нека алгоритъмът да е колкото е възможно по-бърз и ползва колкото е възможно по-малко памет, в асимптотичния смисъл.

Отговор: Функцията PARTITION, известна от алгоритъма QUICKSORT ще свърши работа. Стойността на пивота трябва да е нула. Показали сме, че PARTITION работи в линейно време и ползва константна допълнителна памет, което е оптимално в асимптотичния смисъл.

Зад. 5 В тази задача графите са ориентирани, не са тегловни, не са мултиграфи и може би съдържат примки. За всеки граф $G = (V, E)$, *квадратът на* G наричаме графа $G^2 = (V, E^2)$, където

$$E^2 = \{(u, v) \in V \times V \,|\, \exists w \in V : (u, w) \in E \text{ и } (w, v) \in E\}$$

Предложете колкото може по-ефикасен алгоритми, които изчисляват G^2, ако

1. G е представен чрез списъци на съседства,

2. G е представен чрез матрица на съседства.

Накратко обосновете алгоритмите си и намерете асимптотичната им сложност по време. Не е необходимо да давате детайлен псевдокод, но отговорът Ви трябва да е абсолютно ясен и недвусмислен.

Отговор: Нека G е даден като списъци на съседства. Ще конструираме матрицата M на съседства на G^2 по следния начин: първо я инициализираме с нули, след това за всяко ребро (u, v) на G разглеждаме всички ребра $(v, x_1), \ldots, (v, x_k)$, инцидентни с v, и в матрицата M записваме ребрата $(u, x_1), \ldots, (u, x_k)$. Записването става така: ако $M[u, x_i]$ е 0, записваме 1, в противен случай не правим нищо. Самото записване става в константно време. Общо ребрата са m, за всяко ребро работата е $O(n)$, значи имаме $O(mn)$ работа общо по записването. Инициализирането на M отнема $\Theta(n^2)$ време; също толкова би отнело четенето на M впоследствие (примерно, ако искаме да съставим списъците на съседства на G^2). Общо имаме $O((m + n)n)$, което е $O(mn)$, ако G е свързан.

Сега да допуснем, че G е даден като матрица на съседства A. Конструираме матрицата M на G^2 тривиално във време $\Theta(n^3)$: инициализираме я с нули и после, за всяко възможно ребро (u, v) на G^2, записваме съществуването му с единица, проверявайки дали съществува връх x, такъв че $(u, x) \in E(G)$ и $(x, v) \in E(G)$. Проверката отнема $\Theta(n)$, тъй като има n върха за проверяване, а за всеки от тях проверката става в константно време. Тъй като всички възможни ребра на G^2 са n^2, имаме $\Theta(n^3)$ сложност в най-лошия случай.

Зад. 6 Предложете колкото е възможно по-бързи алгоритми, изчисляващи кликовото число и мощността на максимално независимо множество на **дърво**. Кликовото число на граф G е броят на върховете в коя да е максимална клика в G. Независимо множество върхове е всяко $U \subseteq V(G)$, такова че $\forall x, y \in U : (x, y) \notin E$. Под "максимално независимо множество" разбираме максимално по мощност (а не по включване).

Отговор: Кликовото число на дърво е 2, ако дървото има поне два върха, и 1, в противен случай. В $\Theta(1)$ можем да отговорим какво е кликовото число.

Мощността на максимално независимо множество се намира чрез лаком алгоритъм, разглеждан на лекции, или чрез алгоритъм с динамично програмиране, също разглеждан на лекции.

Зад. 7 Даден е двумерен масив $C[1, 2, \ldots, n][1, 2, \ldots, n]$ от естествени числа. *Специален път* в C наричаме всяка последователност $S_1, S_2, \ldots, S_n$ от клетки на масива, такава че

- S_1 е клетка от първия ред,
- S_j за $2 \leq j \leq n$ е елемент от ред номер j със следното ограничаващо условие. Нека k е номерът на колоната, в която се намира S_{j-1}. Тогава колоната, в която се намира S_j, е или k, или $k - 1$ (но само ако $k > 1$), или $k + 1$ (но само ако $k < n$).

Предложете колкото е възможно по-бърз алгоритъм, който намира специален път в C, такъв че сумата от стойностите на клетките му да е минимална. За пълен отговор е достатъчно да се намери само сумата, а не самият специален път.

Отговор: Задачата се решава чрез алгоритъм с динамично програмиране с двумерна таблица

$$D[1, \ldots, n][1, \ldots, n]$$

Най-горният ред на таблицата се инициализира с най-горния ред на $C[]$, и след това всеки следващ ред i надолу се изчислява чрез рекурсията $D[i,j] = \min\{D[i-1,j-1], D[i-1,j], D[i-1,j+1]\} + C[i,j]$, за $1 \leq j \leq n$. Тъй като $j-1$ и $j+1$ могат да станат съответно 0 и $n+1$, за тези стойности на втория индекс можем да дефинираме, че $D[i,0] = D[i,n+1] = \infty$. Отговорът очевидно е всеки минимален елемент на най-долния ред. Сложността по време е $\Theta(n^2)$, а сложността по памет е $\Theta(n^2)$ при директна имплементация с таблиця $n \times n$. Не е трудно да се види, че размерът на таблицата може да бъде подобрен до $2 \times n$.

Зад. 8 Обяснете колкото е възможно по-изчерпателно, какво е клас на сложност и какво е класът на сложност $\mathcal{NP}$-complete.

Нека е известно, че изчислителните задачи SAT, 3SAT и Хамилтонов Път са в $\mathcal{NP}$-complete. Докажете, че задачата Най-Дълъг Път в Граф е в $\mathcal{NP}$-complete.

Отговор: Нека задачата за най-дълъг път е в "тегловен вариант". И така:

> Най-Дълъг Път в Граф
> *общ екземпляр:* $\langle$неор. граф G, тегловна функция w, число $k\rangle$
> *въпрос:* Има ли път в G с претеглена дължина поне k?

Първо отбелязваме, че Най-Дълъг Път в Граф е в $\mathcal{NP}$, тъй като очевидно недетерминираната машина може да отгатне такъв път и после в полиномиално време да провери, че наистина това е прост път и дължината му е поне k.

Има тривиална редукция на Хамилтонов Път към Най-Дълъг Път в Граф. За всеки екземпляр на Хамилтонов Път, тоест неориентиран граф, създаваме екземпляр на Най-Дълъг Път в Граф: същият граф, тегла единици, число $n-1$, където n е броят на върховете. Очевидно хамилтонов път в оригиналния граф има тогава и само тогава, когато в другия граф има път с дължина $n-1$. Очевидно конструкцията може да се извърши в полиномиално време.