ИЗПИТ ПО “ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ”
(ПОПРАВИТЕЛНА СЕСИЯ — СУ, ФМИ, 4 СЕПТЕМВРИ 2017 Г.)

Задача №	1	2	3а	3б	4	5	6	Общо
получени точки								
максимален брой точки	20	20	20	20	20	20	20	140

Задача 1. С цел улеснение на паркирането общината на един град решила да направи всички улици еднопосочни. Има едно важно изискване към плана: от всяко място в пътната мрежа да можем да стигнем с кола до всяко друго място, като спазваме указаната посока на движение по улиците.

Съставете алгоритъм с линейна времева сложност, който проверява дали предложен план на посоките на улиците изпълнява изискването за достижимост.

За целта най-напред изберете подходящ математически модел на задачата. Направете подробно словесно описание на модела и на алгоритъма.

Задача 2. Следната задача за разпознаване ще наречем Problem2:

Вход: граф G и цяло положително число K.

Въпрос: G притежава ли покриващо дърво с не повече от K листа? (Листо се нарича всеки връх от първа степен.)

Докажете, че Problem2 е NP-трудна задача. Обосновете се подробно.

Задача 3. а) По шосето от град A до град B има n бензиностанции. Дадени са несортирани масиви $X[1...n]$ и $C[1...n]$, където $X[\,i\,]$ е разстоянието, на което i-тата бензиностанция се намира от A, а $C[\,i\,]$ е времето, за което можете да заредите колата си на i-тата бензиностанция (тоест времето за обслужване).

От град A потегляте с пълен резервоар, достатъчен за изминаване на 100 км без зареждане. Искате да стигнете до град B с минимални загуби на време за зареждане на бензин. Съставете алгоритъм с полиномиална времева сложност. Опишете го с думи и го демонстрирайте с поне един пример. Направете анализ на времевата сложност на алгоритъма.

б) Да се реши същата задача, ако е даден само масивът $X[1...n]$, а C липсва: времето за зареждане е едно и също за всички бензиностанции. Предложете алгоритъм с времева сложност $o(n^2)$ — словесно описание, пример и анализ.

Задача 4. За всяко цяло $n > 0$ постройте ориентиран граф с n върха, имащ:
а) минимален; б) максимален брой топологични сортировки.
Опишете конструкцията словесно и направете чертеж за $n = 5$.

Задача 5. Пресметнете времевата сложност $T(n)$ на алгоритъма Alg.

```
ALG(n: positive integer)
while n ≥ 6 do
    n ← ⌊√n⌋
print n
```

Задача 6. Да се намерят едновременно най-голямото и най-малкото число в даден масив $A[1...n]$ с не повече от $1{,}5n$ сравнения.

РЕШЕНИЯ

Задача 1. Предложеният план за посоките на улиците може да се представи чрез ориентиран нетегловен граф:
— Върховете на графа съответстват на кръстовищата.
— Ребрата съответстват на улиците.
— Посоката на всяко ребро съвпада с разрешената посока на движение по съответната улица.

Изискването за достижимост означава всеки връх на графа да е достижим от всеки друг връх. Тоест всеки два върха трябва да са достижими един от друг. С други думи, всеки два върха трябва да бъдат силно свързани. Следователно един план удовлетворява изискването за достижимост тогава и само тогава, когато съответният граф има ***само една компонента на силна свързаност.*** Намирането на компонентите на силна свързаност (и на техния брой) става чрез известния алгоритъм, използващ ***обхождане в дълбочина.***

Алгоритъмът има линейна времева сложност: $T(m,\ n) = \Theta(m+n)$, където n е броят на върховете, m е броят на ребрата на графа.

Задача 2. Че PROBLEM2 е NP-трудна задача, се доказва чрез полиномиална редукция: в частния случай $K = 2$ се получава известната NP-трудна задача ХАМИЛТОНОВПЪТ, защото дърво с не повече от две листа е (прост) път, а щом е покриващо дърво, то съдържа всички върхове на графа, тоест представлява хамилтонов път. Редукцията е полиномиална, тъй като присвояването $K = 2$ се извършва за константно (следователно полиномиално) време.

Задача 3. а) Построяваме ***граф*** с $n+2$ върха — A, B и бензиностанциите. Слагаме ребро от върха u към върха v тогава и само тогава, когато обектът v се намира след обекта u (по шосето, в посока от A към B) и разстоянието между u и v не надхвърля 100 км (тази информация получаваме от масива X). На всяко ребро приписваме тегло — времето за зареждане в бензиностанцията, в която реброто влиза (тези данни взимаме от масива C); ако реброто влиза в B, теглото му е нула.

Получава се ***ориентиран ацикличен граф:*** всички ребра сочат от A към B. "Дължината" на всеки път в този граф е общото време, изразходвано за зареждане на бензин. Търсим най-къс път в графа от върха A до върха B. Това става най-бързо с помощта на ***динамично програмиране.***

Анализ на алгоритъма: Построяването на графа изисква време $\Theta(n^2)$, защото всяка двойка обекти $\{u,\ v\}$ е потенциален кандидат за ребро и трябва да бъде проверена. За динамичното програмиране е нужно време $\Theta(m+n)$, където m е броят на ребрата на графа. Общо: $T(n) = \Theta(n^2) + \Theta(m+n) = \Theta(n^2)$, понеже $m = O(n^2)$. И така, $T(n) = \Theta(n^2)$, тоест алгоритъмът е полиномиален.

Задача 3. б) Най-напред ***сортираме*** масива $X[1...n]$, тоест подреждаме бензиностанциите в реда, в който се срещат при пътуване по шосето от A до B. По-нататък целта ни е да постигнем минимално общо време за зареждане. Понеже на всяка бензиностанция зареждаме за едно и също време, трябва да сведем до минимум броя на зарежданията. Това се постига с помощта на ***алчен алгоритъм:*** пътувайки от A до B, зареждаме максимално късно, т.е. отлагаме всяко зареждане дотогава, докогато е възможно.

Анализ на алгоритъма: Сортирането изисква време $\Theta(n \log n)$. Пътуването от A до B всъщност представлява обхождане на сортирания масив X с индекс, който расте от 1 до n, затова алчният алгоритъм има сложност по време $\Theta(n)$. Общото време на алгоритъма е $T(n) = \Theta(n \log n) + \Theta(n) = \Theta(n \log n)$.

Задача 4. а) Минималният брой топологични сортировки е 0 и се достига за кой да е цикличен ориентиран граф, например цикъл с n върха.

Ако условието се тълкува така, че се иска графът да има поне една топологична сортировка, то минималният брой е 1, например за граф, който представлява път с n върха и еднопосочни ребра: $a \to b \to c \to d \to e$.

б) Максималният брой топологични сортировки е $n!$ и се достига за единствения граф с n върха, който не съдържа нито едно ребро.

Задача 5. При $\mathtt{n} < 6$ алгоритъмът не отпечатва нищо. Това не е интересно, тъй като сложността се мери при $\mathtt{n} \to \infty$. Затова без ограничение нека $\mathtt{n} \geq 6$.

Трасираме алгоритъма. Той отпечатва следните числа:

— на първата итерация: $\left\lfloor \sqrt{\mathtt{n}} \right\rfloor$;

— на втората итерация: $\left\lfloor \sqrt{\left\lfloor \sqrt{\mathtt{n}} \right\rfloor} \right\rfloor = \left\lfloor \sqrt{\sqrt{\mathtt{n}}} \right\rfloor = \left\lfloor \sqrt[4]{\mathtt{n}} \right\rfloor$;

— на третата итерация: $\left\lfloor \sqrt{\left\lfloor \sqrt[4]{\mathtt{n}} \right\rfloor} \right\rfloor = \left\lfloor \sqrt{\sqrt[4]{\mathtt{n}}} \right\rfloor = \left\lfloor \sqrt[8]{\mathtt{n}} \right\rfloor$;

където с $\mathtt{n}$ е означена началната стойност на едноименната променлива от псевдокода, тоест стойността на входа на алгоритъма. Лесно е да се докаже, например чрез математическа индукция, че на $\mathtt{k}$-тата итерация алгоритъмът отпечатва числото $\left\lfloor \sqrt[2^{\mathtt{k}}]{\mathtt{n}} \right\rfloor$. Алгоритъмът излиза от цикъла при най-малкото цяло $\mathtt{k}$, за което отпечатаното число става по-малко от 6. Решаваме относно $\mathtt{k}$ неравенството $\left\lfloor \sqrt[2^{\mathtt{k}}]{\mathtt{n}} \right\rfloor < 6$. То е равносилно на $\sqrt[2^{\mathtt{k}}]{\mathtt{n}} < 6 \Leftrightarrow 6^{2^{\mathtt{k}}} > \mathtt{n} \Leftrightarrow$ $2^{\mathtt{k}} > \log_6 \mathtt{n} \Leftrightarrow \mathtt{k} > \log_2 \log_6 \mathtt{n}$. Най-малкото цяло $\mathtt{k}$, което удовлетворява това неравенство, е числото $\mathtt{k} = \left\lfloor \log_2 \log_6 \mathtt{n} \right\rfloor + 1 = \Theta(\log \log \mathtt{n})$.

Това $\mathtt{k}$ е броят на итерациите, тоест то е равно по порядък на времето за изпълнение на алгоритъма. Окончателно, $T(\mathtt{n}) = \Theta(\log \log \mathtt{n})$.

Задачата може да се реши и с помощта на рекурентното уравнение

$$T(\mathrm{n}) = T\left(\sqrt{\mathrm{n}}\right) + 1,$$

където единицата е порядъкът на времето за изпълнение на една итерация. Полагаме $\mathrm{n} = e^{\mathrm{m}}$, т.е. $\mathrm{m} = \log \mathrm{n}$.

$$T\left(e^{\mathrm{m}}\right) = T\left(\sqrt{e^{\mathrm{m}}}\right) + 1,$$

$$T\left(e^{\mathrm{m}}\right) = T\left(e^{\mathrm{m}/2}\right) + 1.$$

Полагаме $T\left(e^{\mathrm{m}}\right) = Q(\mathrm{m})$.

$$Q(\mathrm{m}) = Q\left(\frac{\mathrm{m}}{2}\right) + 1.$$

Това уравнение се решава с помощта на мастър-теоремата:

$$Q(\mathrm{m}) = \Theta(\log \mathrm{m}).$$

Заместваме $\mathrm{m} = \log \mathrm{n}$ и намираме решението:

$$T(\mathrm{n}) = \Theta(\log \log \mathrm{n}).$$

Задача 6. Отначало сравняваме първия и втория, третия и четвъртия елемент и така нататък. По-големия елемент от всяко сравнение пращаме в масив B, а по-малкия — в масив C. Дотук с $0{,}5\,n$ сравнения масивът $A[1...n]$ е разделен на два масива B и C с по $0{,}5\,n$ елемента. Търсим най-големия елемент на B и най-малкия елемент на C по стандартния начин; те са съответно най-големият и най-малкият елемент на A. Двете търсения изискват по $0{,}5\,n-1$ сравнения. Общият брой сравнения е равен на $0{,}5\,n + 2\,.\,(0{,}5\,n-1) = 1{,}5\,n - 2 \le 1{,}5\,n$.