

Второ контролно за курса ДАА-избираем, 19.02.2015г.

Име: _____, ФН: _____, Спец./курс: _____

Задача	1	2	3	4	5	6	Общо
получени точки							
максимум точки	20	20	20	20	20	20	120

Забележка: За отлична оценка са достатъчни 100 точки!

Задача 1 Опитайте да разделите на две групи с равна сума редиците от числа, дадени по-долу. Ако това е невъзможно, предложете кратко доказателство.

(а - 10 точки) 12, 7, 31, 14, 17, 22

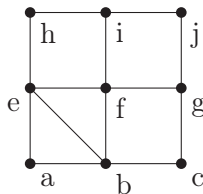
(б - 10 точки) 12, 9, 31, 15, 18, 27

Задача 2 Във всеки връх на свързан граф има плод с определен брой калории. Една маймунка яде плодовете в произволен ред. След изяждане на плод съответният връх се изтрива заедно с излизащите ребра. Ако графът се разпадне на няколко свързани компоненти, маймунката избира една от тях и продължава да яде от нея, а другите компоненти се изтриват (така графът остава свързан). Целта на маймунката е да изяде максимум калории.

Тя прилага алчен алгоритъм: всеки път изяжда най-калоричния плод и при разпадане на графа избира компонентата с най-калоричния плод.

Ще постигне ли целта си маймунката ?

Задача 3 Котка гони мишка в нарисувания по-долу граф:



Отначало котката е във връх a , а мишката — в j . Двете се придвижват последователно, като първа е котката. Всеки ход е придвижване по ребро в графа. Котката побеждава, ако достигне върха, в който е мишката.

Намерете печеливша стратегия за котката.

Задача 4 Даден е неориентиран граф $G(V, E)$ с теглова функция $w : E \rightarrow \{1, 2\}$ (теглото на всяко ребро е 1 или 2). Нека s и t са два върха в графа. Предложете линеен алгоритъм за намиране на най-краткия път от s до t .

Упътване: Алгоритъмът BFS намира най-кратък път за линейно време, но ако всички ребра имат тегло 1. Можете ли да модифицирате графа така, че BFS да реши задачата?

Задача 5 Превозвач има камион, побиращ K тона (K е цяло положително число), и трябва го натовари от склад с n контейнера, тежащи $A[1], A[2], \dots, A[n]$ тона (цели положителни числа), чийто сбор може да надхвърля K . Превозвачът иска да оползотвори възможно най-голяма част от капацитета на камиона, т. е. да натовари контейнери с най-голям сборен тонаж.

Пример: Ако $K = 20, n = 5, A = (2, 3, 3, 9, 10)$, то оптималното решение е $9 + 10 = 19 \leq 20$ тона.

а) (10 точки) Предложете бърз алгоритъм и го опишете на псевдокод като функция $OptimalTransport(A[1 \dots n], K)$, връщаща цяло неотрицателно число – максималния сбор от тонажите на натоварените контейнери. (В примера по-горе функцията трябва да върне стойност 19)

б) (3 точки) Оценете времевата сложност на предложения от Вас алгоритъм.

в) (7 точки) Демонстрирайте работата на алгоритъма върху дадения пример.

Задача 6 Разглеждаме следните две задачи за разпознаване:

$3PLine$: Дадено е множество от n точки в равнината. Има ли сред тях три точки, лежащи на една права?

$TArea$: Дадено е множество от n точки в равнината и цяло неотрицателно число S . Има ли сред дадените точки три, които образуват триъгълник с лице, не по-голямо от S ?

Намерете полиномиална сводимост $3PLine \propto TArea$ или докажете съществуването на такава сводимост.

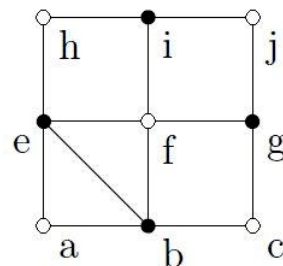
Забележка: Предполага се, че координатите на всички точки са цели числа.

РЕШЕНИЯ

Задача 1. Числата не могат да се разделят в две групи с равни суми, защото:

- а) сборът им е нечетен (има три нечетни числа); ако разделянето беше възможно, сборът щеше да бъде четен ($2.S$, където S е сумата на числата във всяка група);
- б) всички числа се делят на 3 с изключение на едно число (числото 31); както и да ги разпределяме в две групи, сборът на числата в едната група (която не съдържа числото 31) ще се дели на 3, а в другата група — не.

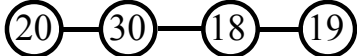
Задача 3. Оцветяваме върховете на графа шахматно — както е показано на картинката. Печелившата стратегия на котката е да отиде от върха **a** до върха **f**, като премине през върховете **b** и **e**, т.е. **a–b–e–f** или **a–e–b–f** (общо три хода).



На всеки ход се променя цветът на полето на мишката, затова след три хода мишката ще се намира на черно поле, т.е. на **b**, **e**, **i** или **g**. Затова котката (която в този момент е на полето **f**) може да хване мишката на четвъртия ход.


Да отбележим, че котката непременно трябва да използва реброто **be**: в противен случай цветът на нейното поле също ще се променя на всеки ход, поради което тя след всеки свой ход ще се намира на поле, различно по цвят от полето на мишката. Следователно, ако котката не използва реброто **be**, тя не може да залови мишката (освен ако мишката сама не отиде при котката).

Задача 2. Алчният алгоритъм, прилаган от маймунката, не е коректен.

Контрапример: графът 

Маймунката първо изяжда плода с 30 калории.

Графът се разпада на две свързани компоненти: 

Маймунката избира компонентата с най-калоричния плод, т.е. 20, следователно компонентата с 18 и 19 калории отпада: 

Сега маймунката изяжда плода с 20 калории и не остават повече плодове.

Така маймунката е изяла общо $30 + 20 = 50$ калории. Това не е максимумът: тя би могла да изяде всичките плодове, т.е. $20 + 30 + 18 + 19 = 87 > 50$ калории, като ги яде например отляво надясно.

Задача 5. Тази задача е разновидност на задачата за раницата: стойността на всяка поръчка е равна на нейното тегло, т.е. имаме един масив вместо два.

а) Задачата може да се реши чрез динамично програмиране:

```
OptimalTransport(A[1...n], K)
dyn[0...n, 0...K]: array of integers
for  $\tilde{n} \leftarrow 0$  to n
    dyn[ $\tilde{n}$ , 0]  $\leftarrow$  0
for  $\tilde{K} \leftarrow 1$  to K
    dyn[0,  $\tilde{K}$ ]  $\leftarrow$  0
for  $\tilde{n} \leftarrow 1$  to n
    for  $\tilde{K} \leftarrow 1$  to K
        a  $\leftarrow$  A[ $\tilde{n}$ ]
        if a >  $\tilde{K}$ 
            dyn[ $\tilde{n}$ ,  $\tilde{K}$ ]  $\leftarrow$  dyn[ $\tilde{n}-1$ ,  $\tilde{K}$ ]
        else
            dyn[ $\tilde{n}$ ,  $\tilde{K}$ ]  $\leftarrow$  max(dyn[ $\tilde{n}-1$ ,  $\tilde{K}$ ], dyn[ $\tilde{n}-1$ ,  $\tilde{K}-a$ ]+a)
return dyn[n, K]
```

б) Времето за изчисляване на всеки отделен елемент на таблицата d_{yn} е ограничено отгоре от константа, така че времевата сложност на алгоритъма е равна по порядък на броя на елементите на таблицата, т.е. $\Theta(n \cdot K)$.

в) В дадения пример таблицата d_{yn} има следния вид:

$\begin{matrix} \text{K} \\ \text{n} \end{matrix}$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	2	2	2	2	2
3	0	2	3	3	3	3
4	0	2	3	3	3	3
5	0	2	5	5	5	5
6	0	2	5	6	6	6
7	0	2	5	6	6	6
8	0	2	5	8	8	8
9	0	2	5	8	9	9
10	0	2	5	8	9	10
11	0	2	5	8	11	11
12	0	2	5	8	12	12
13	0	2	5	8	12	13
14	0	2	5	8	14	14
15	0	2	5	8	15	15
16	0	2	5	8	15	16
17	0	2	5	8	17	17
18	0	2	5	8	17	18
19	0	2	5	8	17	19
20	0	2	5	8	17	19

Задача 4. Върху всяко ребро с тегло 2 добавяме връх (напр. по средата). Новият връх разделя реброто на две части — нови ребра, на всяко от които приписваме тегло 1. С други думи, всяко ребро с тегло 2 се заменя с две последователно свързани ребра, всяко с тегло 1. Затова теглата на пътищата остават непроменени. Следователно най-краткият път от s до t остава същият (като някои от старите му ребра може да са заменени с по две нови). Обаче след извършване на описаната замяна графът съдържа само ребра с тегло 1, затова най-късият път от s до t може да бъде намерен чрез търсене в ширина (*BFS*).

Описаният алгоритъм е линеен (изпълнява се за време $\Theta(m + n)$, където m е броят на ребрата, а n е броят на върховете), защото се състои от три етапа, всеки от които е линеен (а сбор на линейни функции е линейна функция).

Първи етап: модификация на графа. Извършва се чрез еднократно обхождане на ребрата, поради което изисква линейно време.

Втори етап: търсене в ширина (*BFS*) в модифицирания граф. Също изисква линейно време (това е доказано в теорията).

Трети етап: Извършване на обратна модификация върху намерения път (ако има такъв): заменяне на всеки две последователни нови ребра със старото ребро, чиито половинки са те. Конкретните подробности по този етап зависят от използваните структури от данни. Ако например представяме пътя като списък от върхове на графа, то обратната модификация се свежда просто до еднократно обхождане на този списък и изтриване на онези елементи, които съответстват на новите върхове (средите на ребрата с тегло 2). Всяко изтриване на елемент от списък отнема константно време, поради което пълното време за обратна модификация на пътя е пропорционално на дължината му, т.е. $O(m + n)$ (защото дължината на пътя не надхвърля броя на върховете на модифицирания граф, а този брой е не по-голям от $m + n$: оригиналните n върха плюс няколко нови, чийто брой е равен на броя на ребрата с тегло 2 в оригиналния граф, поради което новите върхове са не повече от m).

Задача 6. Задачата *3PLine* се свежда до *TArea* при $S = 0$: лицето на триъгълник е нула точно тогава, когато върховете му са колинеарни. По-формално, ако `bool Triangle_Area(A[1...n]: array of points; S: number)` е функция, която връща `true` тогава и само тогава, когато някои три точки образуват триъгълник с лице, ненадвишаващо неотрицателното число S , то

```
bool Three_Points_On_Line(A[1...n]: array of points)
return Triangle_Area(A, 0)
```

е функция, която връща `true` само тогава, когато някои три точки са колинеарни. Редукцията е полиномиална, защото изисква константно време.