Примерни решение на поправителния изпит по ДАА на 22 август 2024 г.

Зад. 1 Налага Ви се да пътувате от град А до град Б с кола. Има едно единствено шосе между град А и град Б. То има дължина n километра, където $n \in \mathbb{N}^+$. На всеки километър по шосето има точно един мотел.

Има важно ограничение: по някаква причина, може да шофирате най-много $k \in \mathbb{N}^+$ километра в един ден. Ако не може да достигнете до град Б при това ограничение, налага се да преспите в мотел, който е на не повече от k километра от мястото, от което сте тръгнали този ден, и да продължите на следващия ден, като това ограничение остава в сила.

За всеки мотел е дадена цената за нощувка в него. Вашата задача е конструирате колкото е възможно по-ефикасен алгоритъм, който при дадени n, k и цените на мотелите изчислява минималната сумарна цена за преспиване в мотели за пътуване от А до Б. Цените мотелите са дадени като числен масив $M[1\,..\,n-1]$, като $M[i]$ е цената за нощувка в мотела на i-ия километър. Обосновете коректността на Вашия алгоритъм и изследвайте сложността му по време. Ако алгоритъмът Ви е изграден по схемата **Динамично Програмиране**, трябва да напишете рекурсивната декомпозиция в явен вид и да я обосновете накратко – това е достатъчна обосновка за алгоритъма Ви. Ако ли не, трябва да обосновете коректността подробно и прецизно, с инвариант или по индукция.

Решение: Ще решим задачата с алгоритъм по схемата **Динамично Програмиране**. За удобство нека добавим елемент $M[0]$ със стойност 0 към M. Ако $k \geq n$, задачата се тривиализира, така че нека БОО $k < n$.

Нека за $i \in \{0, 1, \ldots, n\}$, $D(i)$ е оптималната (минималната) цена за достигане на километър i по шосето. Очевидно $D(0) = 0$, защото това е цената преди започването на пътуването, когато няма нощувки. За $i \in \{1, 2, \ldots, n\}$, $D(i)$ е минималната сума от следните:

- нощувка в мотела на километър j плюс
- оптималната цена за достигането на километър j,

по всички j, за които това има смисъл. Последното озвачава, че $j < i$, но $j \geq i - k$ заради ограничението "не повече от k километра на ден". Разбира се, разглеждат се само неотрицателните стойности на j. Всичко това може да запишем накратко така:

$$D(i) = \begin{cases} 0, & \text{ако } i = 0, \\ \min\{M[j] + D(j) \mid i - k \leq j \leq i - 1 \wedge j \geq 0\}, & \text{ако } i \geq 1 \end{cases}$$

Това е рекурсивната декомпозиция: има начално условие за $i = 0$, а за по-големите i, $D(i)$ се пресмята чрез по-малки стойности на индексната променлива. Пресмятането се извършва с масив $D[0\,..\,n]$, където $D[0] = 0$, а стойността, която ни интересува в края на изчислението, е $D[n]$. Ето псевдокод.

```
Пътуване(n, k, M)
1  създай масив D[0 .. n]
2  D[0] ← 0
3  for i ← 1 to n
4       D[i] ← ∞
5       for j ← max(0, i − k) to i − 1
6            D[i] ← min(D[i], M[j] + D[j])
7  return D[n]
```

Да изследваме сложността по време. Външният цикъл се изпълнява $\Theta(n)$ пъти, вътрешният цикъл се изпълнява $O(k)$ пъти и тялото на вътрешния се изпълнява в $\Theta(1)$, така че от общи съображения можем само да ограничим сложността отгоре с $O(nk)$.

Ще докажем, че сложността е $\Theta(nk)$. За целта ще преброим точно колко пъти се изпълнява тялото на вътрешния цикъл. Ако $i = 1$, то се изпълнява веднъж. Ако $i = 2$, то се изпълнява два пъти. И така нататък, ако $i = k - 1$, то се изпълнява $k - 1$ пъти. За $i \in \{k, k+1, \ldots, n\}$, то се изпълнява k пъти. Сумата е

$$1 + 2 + \cdots + k - 1 + \underbrace{k + k + \cdots + k}_{n-k+1\text{събираеми}} =$$

$$\frac{k(k-1)}{2} + k(n-k+1) = nk + k\left(\frac{k-1}{2} - k + 1\right) = nk + \frac{k}{2}(k - 1 - 2k + 2)$$

$$nk + \frac{k}{2}(-k+1) = \frac{1}{2}k(2n - k + 1)$$

Предвид допускането, че $k < n$, в сила е $n \leq 2n - k + 1$. Очевидно $2n - k + 1 \leq 2n$, така че $n \leq 2n - k + 1 \leq 2n$. От това следва, че $2n - k + 1 = \Theta(n)$. Тогава $\frac{1}{2}k(2n - k + 1) = \Theta(nk)$.

Зад. 2 Двама играчи, да кажем Иван и Мария, играят на следната игра. Известно е, че има масив $A[1\,..\,k]$ от уникални реални числа, като $k \geq 2$. Стойността на k е известна и на двамата играчи. Съдържанието на масива A е известно на Иван, но Мария не знае нищо за масива A. Иван казва на Мария една или повече подсказки от вида "$A[i] < A[j]$", като i и j са такива, че $1 \leq i < j \leq k$. Целта на Мария е да конструира масив $B[1\,..\,k]$ от реални числа, който е съвместим с всички подсказки в смисъл, че за всяка подсказка $A[i] < A[j]$, дадена от Иван, стойностите $B[i]$ и $B[j]$ са такива, че $B[i] < B[j]$.

Предложете колкото е възможно по-ефикасен алгоритъм за задачата на Мария. Входът на този алгоритъм е k и всички подсказки, а изходът е масив $B[1\,..\,k]$, съвместим с подсказките. Допуснете, че Иван не лъже с подсказките. Обосновете коректността и изследвайте сложността по време. За изследването на сложността по време допуснете, че k е кодирано унарно, тоест, числото k е представено от стринг от k на брой единици.

Бонус I, 5 точки: *Какъв е смисълът на това, k да бъде записано унарно?*

Бонус II, 10 точки: *Разгледайте същата задача, ако подсказките са нестроги неравенства от вида $A[i] \leq A[j]$. Решението Ви може да е формулирано само в общи линии. Важното е да е ясна идеята и обосновката на сложността по време.*

Решение: Нека броят на подсказките е p. По условие, $p \geq 1$. Очевидно е, че $p \leq \frac{k(k-1)}{2}$, защото броят на наредените двойки индекси (i, j), такива че $1 \leq i < j \leq k$, е точно $\frac{k(k-1)}{2}$; при положение, че Иван не лъже, ако той даде повече подсказки, то той се повтаря, което няма смисъл.

Нека $G = (V, E)$ е ориентиран граф, чиито върхове са елементите на A, а ребрата са подсказките; а именно, всяка подсказка $A[i] < A[j]$ е ребро от връх i към връх j, и други ребра няма. Щом Иван не лъже, G е даг. Решението на задачата е алгоритъм, който се състои в топо-сортиране на G, след което се генерира масив $B[1\,..\,k]$, в който $B[i]$ получава индекса на $A[i]$ в топологическото сортиране. Алгоритъмът връща получения масив B. Коректността е очевидна предвид свойствата на топологическото сортиране. Сложността по време е $\Theta(k+p)$. Това е в сила, понеже k във входа е кодирано унарно. Щом k е кодирано унарно, то размерът на входа е $\Theta(k+p)$. С други думи, графът има големина $\Theta(k+p)$. На лекции сме изучавали алгоритми за топо-сортиране, които са с линейна сложност по време. В случая, това означава сложност $\Theta(k+p)$. Ето псевдокод.

```
Създай съвместим масив(k, записано унарно; подсказки от вида A[i] < A[j])
1  създай граф G = ({1, 2, ..., k}, ∅), представен с празни списъци
2  за всяка подсказка A[i] < A[j], добави j в списъка на i
3  сортирай топологически G с алгоритъма на Tarjan
4  нека резултатът от топо-сортирането е масив C
5  създай празен масив B[1 .. k]
6  for i ← 1 to k
7      B[C[i]] ← i
8  върни B
```

За първия бонус: ако k има размер единица независимо от стойността си, каквото е стандартното допускане в курса, то сложността е недефинирана (ако предпочитате, сложността е безкрайна). Изискването k да е кодирано унарно гарантира, че размерът на k във входа е $\Theta(k)$. Ако k беше кодирано бинарно, размерът на k във входа би $\Theta(\lg k)$, откъдето алгоритъмът би бил експоненциален.

Колкото до втория бонус: отново моделираме задачата с ориентиран граф, но сега той не е непременно даг. Интересуват ни неговите силно свързани компоненти. Елементите на една и съща силно свързана компонента получават една и съща стойност, а алгоритъмът на Kosaraju ни дава силно свързаните компоненти, и то в линейно в размера на графа време, при това подредени в реда на топо-сортировка на фактор-графа. Трябва да се има предвид, че втората фаза на алгоритъма на Kosaraju, която намира силно свързаните компоненти като множества от върхове, работи върху транспонирания граф, така че стойностите за различни силно свързани компоненти трябва да се дават в намаляващ ред.

Зад. 3 Напишете алгоритъма, изучаван на лекции, намиращ всички срязващи върхове на граф, на псевдокод с горе-долу такава детайлност като псевдокода от лекциите. Обосновете коректността на алгоритъма. Не се иска подробно доказателство с инвариант или по индукция, а само основните леми, служещи за обосновка, с кратки доказателства, и формулировка на твърдението, от което следва коректността. Съвсем накратко изследвайте сложността по време.

Решение: Това е правено на лекции.