КОНТРОЛНО № 2 ПО ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ (СУ, ФМИ, 26 НОЕМВРИ 2020 Г.)

Указания:

1) Всички решения да бъдат обосновани подробно.
2) Алгоритмите, изучени на лекции, могат да се използват наготово.
3) При всяко сортиране и търсене да се уточнява използваният алгоритъм.
4) Имената на алгоритмите и структурите от данни да са на български език.

Задача 1. Даден е масив $A[1\ldots n]$ от n положителни числа — дължините на n контролни работи, измерени в брой страници. Числото $A[k]$ е дължината на k-тата контролна работа и може да бъде дробно (например 6 ½ страници). Тъй като имате малко време, можете да проверите само S страници. Числото S също е дадено; то е положително и може да бъде цяло или дробно.

Искате да съобщите оценките на възможно най-много ученици. За целта трябва да проверите възможно най-много контролни за времето, което имате.

Съставете алгоритъм, който по дадени $A[1\ldots n]$ и S предлага за проверка възможно най-голямо множество от контролни. Алгоритъмът трябва да намира множеството от контролни, а не само техния брой.

Опишете алгоритъма словесно или на псевдокод, обосновете го накратко и анализирайте неговата времева сложност при най-лоши входни данни.

Ако времевата сложност е $O(n \log n)$ при най-лоши данни, се дава **1 точка.** Ако времевата сложност е $O(n)$ при най-лоши данни, решението носи **2 точки.** По-бавни алгоритми не носят точки, както и алгоритми с неясно описание, без обосновка (или с грешна обосновка) и алгоритми без анализ на сложността.

Задача 2. Какво връща алгоритъмът? Отговорът да се обоснове формално — с инвариант и полуинвариант.

(2 точки)

```
ALG(A[1...n]: реални числа)
1) k ← 1
2) while k ≤ n + 1 − k do
3)    if A[k] < 0
4)       return k
5)    if A[n + 1 − k] < 0
6)       return n + 1 − k
7)    k ← k + 1
8) return −1
```

Задача 3. Дадени са ви m указания за спешно копиране на n документа: "Документът X да се копира преди Y."

Изберете някой изучен алгоритъм, който за време $O(m + n)$ при всякакви входни данни намира един възможен ред за копиране на документите или открива, че няма такъв.

Опишете входните данни като граф. Какво означават върховете, ребрата и посоките (ако ребрата имат посоки)? Какво се търси (в термините на графи)?

С цел бързина, задачата да се реши с едно обхождане на графа. От кой вид е то — в ширина или в дълбочина?

Изпълнете алгоритъма постъпково върху граф с пет върха и седем ребра.

За пълно решение се дава **1 точка.** Непълни решения не носят точки.

Оценката = 1 + броя на точките.

РЕШЕНИЯ

Задача 1 може да се реши по различни начини. Един възможен алгоритъм: Сортираме масива $A[1\ldots n]$ от дължините на контролните работи с помощта на някоя бърза сортировка, например с пирамидалното сортиране. След това, тъй като разполагаме с ограничено време за проверка — само за S страници, започваме да проверяваме най-малките контролни. Тоест обхождаме масива по нарастващ ред на индексите — от малките към големите контролни работи. Събираме големините (бройките страници) на обходените контролни работи и спираме обхождането, когато сборът надхвърли S или изчерпим масива (което от двете събития настъпи първо). Проверяваме обходените контролни (без онази, след чието добавяне сборът от бройките страници надхвърля S).

При най-лоши входни данни този алгоритъм изразходва следното време: $\Theta(n \log n)$ — за сортирането; $\Theta(n)$ — за последващото обхождане и събиране; $\Theta(n \log n) + \Theta(n) = \Theta(n \log n)$ — общо за целия алгоритъм.

Втори начин: чрез ***алгоритъма PICK в съчетание с двоично търсене.*** Като избягваме сортирането, получаваме алгоритъм с времева сложност $\Theta(n)$ при всякакви входни данни:

1) Чрез алгоритъма PICK намираме медианата M на масива $A[1\ldots n]$.
2) Разделяме масива $A[1\ldots n]$ относно M, т.е. разделяме контролните работи на две групи — къси контролни и дълги контролни.
3) Пресмятаме сбора L от дължините на късите контролни.
4) Ако $L = S$, проверяваме само късите контролни (край на алгоритъма).
5) Ако $L > S$, то не можем да проверим дори само късите контролни. Затова отхвърляме всички дълги контролни и рекурсивно решаваме задачата върху подмасива от късите контролни, като използваме същото S.
6) Ако $L < S$, то проверяваме всички къси контролни. Тъй като разполагаме с време за проверка на още $S - L$ страници, то трябва да проверим колкото е възможно повече от дългите контролни, затова рекурсивно обработваме подмасива от дългите контролни с $S - L$ вместо S.

В описанието на алгоритъма не е изрично посочено дъното на рекурсията: при $n = 1$ проверяваме единствената контролна работа само ако тя съдържа не повече от S страници.

Анализ на времевата сложност на алгоритъма: При най-лоши входни данни не се изпълнява стъпка № 4 (т.е. $L \neq S$); в такъв случай рекурсията достига най-голяма дълбочина — масиви с дължина 1. Стъпките № 1, № 2 и № 3 изразходват време $\Theta(n)$. Изпълнява се само една от стъпките № 5 и № 6 върху половината масив. Която и да е от тях изразходва време $T\left(\frac{n}{2}\right)$, тоест $T(n) = T\left(\frac{n}{2}\right) + \Theta(n)$, където $T(n)$ е времевата сложност на целия алгоритъм. От мастър-теоремата намираме $T(n) = \Theta(n)$.

Задача 2. Алгоритъмът връща индекса на елемент от масива $A[1\ldots n]$ — индекса на онзи елемент с отрицателна стойност, който е възможно най-близо до някой от краищата на масива. Ако първото и последното отрицателно число са еднакво близо съответно до левия и десния край на масива, то алгоритъмът връща индекса на първото отрицателно число. Ако масивът $A[1\ldots n]$ не съдържа отрицателни числа, то алгоритъмът връща минус единица.

Полуинвариант на цикъла е например индексът k: той нараства с единица, а $n+1-k$ намалява с единица при всяко изпълнение на тялото на цикъла. Ако алгоритъмът не излезе чрез ред № 4 или ред № 6, то рано или късно k ще стане по-голямо от $n+1-k$ и цикълът ще завърши. В такъв случай алгоритъмът ще върне минус единица чрез ред № 8.

Инвариант: При всяка проверка за край на цикъла (тоест всеки път, когато алгоритъмът минава през ред № 2 от псевдокода) е вярно следното твърдение: подмасивите $A[1\ldots k-1]$ и $A[n+2-k\ldots n]$ не съдържат отрицателни числа.

Доказателство: с индукция по поредния номер на проверката на ред № 2.

База: При първото изпълнение на проверката k има стойност 1 (ред № 1). Тогава подмасивите $A[1\ldots k-1]$ и $A[n+2-k\ldots n]$ имат дължини $k-1=0$, тоест те са празни масиви, така че очевидно не съдържат отрицателни числа (нито каквито и да било числа).

Индуктивна стъпка: Нека инвариантът важи при някоя непоследна проверка на ред № 2. Ще докажем, че инвариантът ще важи и при следващата проверка. Действително, щом сегашната проверка не е последна, то тялото на цикъла се изпълнява още веднъж, като редовете № 4 и № 6 не се изпълняват. Ето защо $A[k] \geq 0$ и $A[n+1-k] \geq 0$. Присъединявайки индуктивното предположение, стигаме до извода, че подмасивите $A[1\ldots k]$ и $A[n+1-k\ldots n]$ не съдържат отрицателни числа. След като алгоритъмът изпълни ред № 7, стойността на k се увеличава с единица, така че същите два подмасива вече се записват като $A[1\ldots k-1]$ и $A[n+2-k\ldots n]$. После алгоритъмът преминава отново към проверката на ред № 2 за край на цикъла. Но ние току-що доказахме, че подмасивите $A[1\ldots k-1]$ и $A[n+2-k\ldots n]$ не съдържат отрицателни числа, а това съвпада с индуктивното заключение.

По-горе доказахме, че рано или късно цикълът завършва. Нека приложим инварианта към последната проверка за край на цикъла. Има две възможности.

Първи случай: $k \leq n+1-k$. Алгоритъмът пак изпълнява тялото на цикъла и тъй като разглеждаме последната проверка за край на цикъла, то алгоритъмът излиза от цикъла чрез ред № 4 или ред № 6, тоест $A[k] < 0$ или $A[n+1-k] < 0$. Подмасивите $A[1\ldots k-1]$ и $A[n+2-k\ldots n]$ не съдържат отрицателни числа съгласно с инварианта, затова върнатата стойност (k или $n+1-k$ съответно) е индексът на отрицателен елемент най-близо до някой от краищата на масива. Ако $A[k] < 0$ и $A[n+1-k] < 0$, то алгоритъмът връща k, защото след ред № 3 се изпълнява ред № 4.

Втори случай: $k > n + 1 - k$. Алгоритъмът не изпълнява тялото на цикъла, а отива на ред № 8 и връща минус единица като знак, че масивът $A[1\ldots n]$ не съдържа отрицателни числа. Действително, според доказания инвариант подмасивите $A[1\ldots k-1]$ и $A[n+2-k\ldots n]$ не съдържат отрицателни числа. Следователно тяхното обединение не съдържа отрицателни числа. Тъй като $k > n + 1 - k$ и числата k и n са цели, то важи неравенството $k \geq n + 2 - k$. Затова $A[n+2-k\ldots n] \supseteq A[k\ldots n]$, поради което

$$A[1\ldots k-1] \cup A[n+2-k\ldots n] \supseteq A[1\ldots k-1] \cup A[k\ldots n] = A[1\ldots n].$$

Щом $A[1\ldots k-1] \cup A[n+2-k\ldots n] \supseteq A[1\ldots n]$, то следва, че масивът $A[1\ldots n]$ не съдържа отрицателни числа, тоест стойността, върната от алгоритъма, съответства на нашия отговор.

Задача 3. Моделираме входните данни чрез ориентиран граф:
— Върховете на графа съответстват на документите.
— Ребрата на графа съответстват на указанията.
По-точно, указанието "документът X да се копира преди документа Y"
се представя като ребро от върха X към върха Y.

Да намерим един възможен ред за копиране на документите, означава да подредим върховете на графа водоравно така, че всички ребра да сочат отляво надясно. Това става с изучения алгоритъм за ***топологично сортиране,*** който използва ***обхождане в дълбочина*** и има времева сложност $O(m + n)$ при всякакви входни данни.

Указанията за копиране не могат да бъдат спазени тогава и само тогава, когато графът съдържа ориентиран цикъл. Търсенето на цикъл се извършва отново чрез ***обхождане в дълбочина*** и пак за време $O(m + n)$ при всякакви входни данни.

Няма нужда от две обхождания на графа. Достатъчно е едно обхождане: ако алгоритъмът открие цикъл, прекратява обхождането; в противен случай довършва обхождането и подрежда върховете на графа.

СХЕМА ЗА ТОЧКУВАНЕ

Задача 1 се оценява в зависимост от бързината на предложения алгоритъм: ако времевата сложност е $O(n)$ при най-лоши данни, решението носи 2 точки; иначе за времева сложност при най-лоши данни $O(n \log n)$ се дава 1 точка. По-бавни алгоритми не носят точки, както и алгоритми с неясно описание, без обосновка (или с грешна обосновка) и алгоритми без анализ на сложността.

Задача 2 се оценява с 2 точки, от които:
— за формулиране на верен отговор и верен и използваем инвариант: 1 точка;
— за полуинвариант и доказателство на инварианта и на отговора: 1 точка.

Задача 3 се оценява с 1 точка само ако е решена пълно.