ДАА семинар 13

Още за долни граници:

Зад. Даден е масив $A[1...n]$, чиито елементи са сортирани. Да се определи долна граница за всеки алгоритъм, който проверява дали A съдържа 2 по 2 различни елементи, използвайки функция Compare(i,j), която връща $<, =$ или $>$, в зависимост от резултата от сравнението на $A[i]$ с $A[j]$.

Решение: Ще покажем, че $n-1$ е долна граница за броя на сравненията, т.е. за броя извиквания на Compare.

// Часто се иска такава долна граница, защото, ако опитаме да си измислим максимално ефикасен алг., най-лесно е линейно да обходим масива и да сравним съседните елементи, които ни дава $n-1$ извиквания на Compare

Нека Alg X е алгоритъм, който решава задачата и нека той прави не повече от $n-2$ сравнения.

Достатъчно е да разгледаме само случая с точно $n-2$ сравнения.

Ще използваме аргументация с противник.

Броят на съседните двойки елементи в $A[1...n]$ е $n-1$. Щом са направени $n-2$ сравнения, то има поне една двойка елементи, които не са били сравнявани.

Стратегията на противника е следната:

Нека $A[i] = 2i$ за $i \in [1,n]$ и той отговаря с резултата

от функцията Compare.

Изл. Ако Alg X отговори с „Da", тогава елементите са 2 по 2 различни, то противникът го опровергава като за двойката A[i] и A[i+1], които не са били сравнени, задава стойност A[i]=A[i+1]=2i.

Така отговорите му са консистентни, но алг. е сгрешил.

II сл. Ако Alg X отговори с „Ne", то противникът го опровергава, показвайки оригиналния A[1..n].

Зад. Покажете, че $2n-1$ е долна граница за броя на сравненията при сливането на A[1..n] и B[1..n] в един сортиран масив C[1..2n].

Решение: Долна граница $2n-1$ ни подсказва, че всяка двойка съседи в C[1..2n] е била сравнена.

Отново ще използваме аргументация с противник, възползвайки се от този факт.

Нека Alg X е алгоритъм, който слива двата масива, правейки $2n-2$ сравнения.

Нека $A[i] = 2i-1$ за $i = 1..n$ и
$B[i] = 2i$ за $i = 1..n$.

//Зад.: Не е нужно да се посочват точно такива стойности. Така обаче е по-лесно, без да има загуба на общност.

Щом сравненията са $2n-2$, то има двойка съседни елементи, които не са били сравнени.

При този вход Alg X е върнал масива:

C = [A[1], B[1], A[2] ... A[n], B[n]]

Възможни са 2 случая за съседи, които не са сравн.

— A[i] не е дал сравнен с B[i]:

Противникът задава стойност A[i]=2i, B[i]=2i-1
Така передата в масивите A и B не се променя, т.е
той не може да бъде уличен в лъжа, но
Alg X връща сбъркан C, което е грешен резултат.

— A[i] не е дал сравнен с B[i-1]: Аналогично

на горния случай: A[i]=2(i-1) B[i-1]=2i-1

Зад. 3 Дадено е множество от наредени двойки $X = \{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$, такива че $a_i < b_i$ за $1 \leqslant i \leqslant n$. Известно е, че числата $a_1, \ldots, a_n, b_1, \ldots, b_n$ са две по две различни. За всеки две наредени двойки (a_i, b_i) и (a_j, b_j), където $i \neq j$, казваме, че са *независими*, ако е изпълнено $a_i < b_i < a_j < b_j$ или $a_j < b_j < a_i < b_i$. Ако е известно, че наредените двойки (a_{i_1}, b_{i_1}), (a_{i_2}, b_{i_2}), $\ldots$, (a_{i_k}, b_{i_k}) са две по две независими, казваме, че масивът от тях

$$[(a_{i_1}, b_{i_1}), (a_{i_2}, b_{i_2}), \ldots, (a_{i_k}, b_{i_k})]$$

е *растящ*, ако

$$a_{i_1} < b_{i_1} < a_{i_2} < b_{i_2} < \cdots < a_{i_k} < b_{i_k}$$

Професор Интервалски твърди, че разполага с алгоритъм, който във време $O(n \lg \lg n)$ намира множество $Y \subseteq X$, такова че наредените двойки в Y са две по две независими и $|Y| \geqslant |Z|$ за всяко $Z \subseteq X$, такова че наредените двойки в Z са независими. Нещо повече, професорът твърди, че неговият алгоритъм връща Y в растящ масив. Опровергайте го: докажете, че такъв алгоритъм няма.

Решение: Interval scheduling от лекции

Редукции (свеждане):

→ Изчислителна задача X свеждаме до изч. зад. Y, тоест използваме Y, за да решим X.

$X \leq_p Y$ // Полиномиална Карп редукция

→ „Ако X е лавна, то и Y е лавна", затова свеждаме познатата задача до тази, която разглеждаме сега

Зад. Дадени са два масива $A[1...n]$ и $B[1...n]$ от числа. Задачата е да се сортират елементите им един с друг, така че най-малкия* в A да е свързан с най-малкия в B, втория най-малък с втория и така нататък.

* Всеки елемент са 2 по 2 различни.

Решение: Нека означим тази задача с X.

Очевидно задачата X може да се реши със сортиране, но това не ни дава нищо:

Искаме да сведем позната задача до X, а не обратното, т.е. X до Сортиране.

Ако съставим редукция/покажем алгоритъм, който решава Сортиране чрез X, тоест да сведем Сортиране до X, ще получим долна граница за X.

Нека $A[1...n]$ са елементите, които искаме да сортираме, а $B[1...n]$ са валидни индекси от 1 до n.

Ако сортираме A с B, т.е. решим X, то сме решили и Сортиране (Очевидно?)

Знаем, че $\Omega(n \lg n)$ е долна гр. за Сортиране, след. това е и долна граница за X.

Зад. 6 Изчислителната задача ЗАД1 се дефинира така: Изпит 25.06.2016

Изчислителна Задача ЗАД1
Общ пример: Множество A, множество $\mathcal{B} \subseteq 2^A$, число $k \in \mathbb{N}$
Въпрос: Съществува ли $C \subseteq A$, такова че $|C| \leq k$ и $\forall X \in \mathcal{B} : X \cap C \neq \emptyset$?

Докажете, че ЗАД1 е **NP**-пълна.

// Нека Π е изчислителна задача. Казваме, че Π е NP-пълна тогава и само тогава, когато
1) $\Pi \in NP$
2) $(\forall \Pi' \in NP)[\Pi' \leq_p \Pi]$

Решение: 1) Ще покажем, че ЗАД1 $\in NP$:

Използваме НМТ-верификатор: Тази недетерминистична машина на Тюринг отгатва $C \subseteq A$ и проверява:
- $|C| \leq k$
- $(\forall x \in B)[x \cap C \neq \emptyset]$

Очевидно двете проверки могат да се извършат в полиномиално време.

2) Ще сведем позната NP-пълна задача до ЗАД1:

Ще покажем, че $VC \leq_p$ ЗАД1 (VC = Vertex Cover) // ЗАД1 = Hitting Set

Нека е даден екземпляр на VC: Неориентиран граф $G=(V,E)$ и естествено число k_0.

Ще построим екземпляр за ЗАД1 по следния начин:
$A = V$
$B = E$
$k = k_0$

Графът G има върхово покритие с мощност не по-голяма от k_0 тогава и само тогава, когато A има подмножество с не повече от k елемента, т.ге всяко (двуелементно) множество в B да има сечение с него.

Конструкцията може да се извърши в полиномално време.

Следователно, 3AD1 = HittingSet ∈ NP-c.

Някакви общи насоки за намиране на задача за редукция:

Нека ни е дадена зад. X и искаме да покажем, че $X \in NP\text{-c}$.
Ако X изисква:

- намиране на някакво подмножество от елементи, разделяне в 2 подмножества или пък „съпоставяне на двойки" или елементите → SAT или 2-PARTITION
- да се поставят „етикети" на някакви елементи или да се разпределят в малко на брой множества → kColour или 3Colour
- подреждане на елементи в някакъв ред → HC, HP, TSP
- намиране на „малко" подмножество на елементи по дадено условие → VC
- намиране на „голямо" подмножество по условие → IndependentSet / Clique / Max2SAT
- Ако има 3 някъде → 3SAT, 3Colour...

//last resort

↑ само ако нищо друго не става