

ОАА Семинар 12

Заг Дадени са два масива с цели числа $A[1..n], B[1..m]$.
Можем да свържем прави линии м/у $A[i]$ и $B[j]$
(разположени един над друг) за $\forall i, j: A[i] = B[j]$

Да се състави ефективен алгоритъм, който изчислява
максималният брой линии, така че да няма пресичане

Пр
A: 1 4 2
B: 1 2 4
или 4-4,
или 2-2

//LeetCode 1035. Uncrossed Lines

Решение: Нека $f(i, j)$ е максималният брой
търсени линии за $A[1..i]$ и $B[1..j]$

Разглеждаме думите отзад напред. Имаме следните
възможности

- 1) Един от масивите е празен
- 2) Можем да свържем $A[i]$ с $B[j]$:
 - 2.1) Свързваме ги и разгл. за $i-1, j-1$
 - 2.2) Нещи свързваме и разгл. за $i-1, j$
 - 2.3) Нещи св. и разгл. за $i, j-1$
- 3) Не можем да ги свържем:
 - 3.1) Прескачаме и свата, т.е. разгл. за $i-1, j-1$
 - 3.2) Разгл. за $i-1, j$
 - 3.3) Разгл. за $i, j-1$

Ще обмислим тези случаи в дефиницията на
 $f(i, j)$:

$$f(i, j) = \begin{cases} 0, & \text{ако } i=0 \vee j=0 \\ \max \left\{ \begin{array}{l} 1 + f(i-1, j-1), \\ f(i-1, j-1) \\ f(i, j-1) \end{array} \right\}, & \text{ако } i \neq 0 \wedge j \neq 0 \wedge A[i] = B[j] \\ \max \left\{ \begin{array}{l} f(i-1, j-1) \\ f(i-1, j) \\ f(i, j-1) \end{array} \right\}, & \text{ако } i \neq 0 \wedge j \neq 0 \wedge A[i] \neq B[j] \end{cases}$$

Заг Изпит 06.07.2023

Зад. 5 Дадена е булева матрица M с m реда и n колони. За всяка нейна подматрица ще казваме, че е *единична*, ако съдържа само единици. Предложете алгоритъм със сложност $O(mn)$, който намира **квадратна** единична подматрица $M'_{k \times k}$, като това k е максимално. Вашият алгоритъм трябва да върне наредена тройка (i, j, k) , където $M[i, j]$ е клетката на M' , намираща се долу вдясно. Обосновете коректността и сложността по време на Вашия алгоритъм.

Решение:

Нека $opt(i, j)$ е размерът на страната на единична квадратна подматрица с долен десен ъгъл

$opt(1, j) = M[1, j]$ за $\forall j = 1 \dots n$, защото

Ако $M[1, j] = 0$, то няма единична матрица с долен десен ъгъл $M[1, j]$

Ако $M[1, j] = 1$, то има точно една матрица с долен десен ъгъл $M[1, j]$ със страна 1 $\rightarrow$ самата $M[1, j]$

Аналогично $opt(i, 1) = M[i, 1]$ за $i = 1 \dots m$

Нека $i > 1$ и $j > 1$. Възможни са следните случаи:

1) $M[i][j] = 0 \rightarrow$ единичната подматрица с долен диагонал във $M[i][j]$ е със страна 0 (т.е. няма такава)

$$\Rightarrow \text{opt}(i, j) = 0, \text{ ако } M[i][j] = 0$$

2) $M[i][j] = 1$

- Ако $M[i-1][j] = 0 \vee M[i][j-1] = 0 \vee M[i-1][j-1] = 0$,
то с $M[i][j]$ не „разширяваме“ подматрицата

$$\Rightarrow \text{opt}(i, j) = 1$$

- В противен случай, $M[i][j]$ „разширява“ единична подматрица, такава че:

Ако $\text{opt}(i-1, j) = \text{opt}(i, j-1)$, то $\text{opt}(i, j) = \text{opt}(i-1, j-1) + 1$
// страните на разширяваните подматрици са равни

Ако $\text{opt}(i-1, j) \neq \text{opt}(i, j-1)$, то $\text{opt}(i, j) = 1 + \min \{ \text{opt}(i-1, j), \text{opt}(i, j-1) \}$

// Можем да разширим най-много до по-малката страна + 1

Алгоритмично получаване:

$$\text{opt}(i, j) = \begin{cases} 0, & \text{ако } M[i][j] = 0 \\ 1, & \text{ако } M[i][j] = 1 \wedge (M[i-1][j] = 0 \vee M[i][j-1] = 0 \vee M[i-1][j-1] = 0) \\ 1 + \min \{ \text{opt}(i-1, j), \text{opt}(i, j-1), \text{opt}(i-1, j-1) \}, & \text{ако } M[i][j] = 1 \wedge M[i-1][j] = 1 \wedge M[i][j-1] = 1 \wedge M[i-1][j-1] = 1 \end{cases}$$

(2P) Псевдокод

$$T(n, m) = O(n, m)$$

Заг Независимо подмножество в граф $G = (V, E)$ ще наричаме всяко подмножество на V , такова че между върховете в него няма ребра.

Дадено е дърво T , като БОО T е кореново.

Да се предложи алгоритъм, който намира най-голямо независимо подмножество в T за $O(n)$ време.

Решение: Нека за $\forall v \in V$ $opt(v)$ е размерът на най-голямо подмножество за поддърво T' с корен v .

Когато разглеждаме връх $v \in T'$ имаме две възможности:

1) Не включваме v : Това независимо мнж. в T' ще е обединението от независими подмножества за наследниците на v .

2) Включваме v : В този случай всичките наследници на v трябва да бъдат изключени, а незав. подмн. за техните наследници да бъдат включени.

Получаваме:

$$opt(v) = \max \left\{ \sum_{u \in children(v)} opt(u), 1 + \sum_{u \in children(v)} \sum_{w \in children(u)} opt(w) \right\}$$

Можем да мемоизираме по различни начини:

- В масив $M[1..n]$, съотв. за всеки връх

- В самото горво T нато добавим нов атрибут
изв всеки връх.

Извършване одхождането в post-ордер (ляво-дясно-корен),
защото исаме стойностите за дедата да са изчислени.

Ⓟ LeetCode : 337. House Robber II

Долна граница

- Организация за **всички** алгоритми, които решават
дадена изчислителна задача

Техники:

- горво за взимане на решения
- аргументация с противник
- редукция

Заг Да се покаже, че $\Omega(\lg n)$ е асимптотична
долна граница за **Търсене**.

Решение: Нека входният масив е $A[1..n]$ и търсеният
елемент е key .

БОО Нека $A[1..n]$ има $2^{n/2}$ различни елемента.

За да работи коректно, алгоритъм, решаващ задачата
Търсене трябва да може да различава следните
състояния:

$A[1] = \text{key}$

$A[2] = \text{key}$

$\vdots$

$A[n] = \text{key}$

$\text{key} \notin A[1..n]$

} $n+1$ на дрой

(none)

Тоей в дървото за взимане на решения трябва
да има поне $\Theta(n)$ листа, което ни дава
долна граница $\Omega(n)$ за времето.

Доказва се*, че ако има $\Omega(n)$ листа, то
за височината на дървото има долна граница
 $\Omega(\lg(n))$.

Оттук, $\Omega(\lg n)$ е долна граница за сложността на
алгоритъма.

* Докажете, че всяко k -ично дърво с височина h
не повече от k^h листа.

БОО $k=2$.

Изпит - 29.06.2019

Задача Задачата за разпознаване СВЪРЗАНОСТ се дефинира така: общият пример е неориентиран
граф $G = (V, E)$, а въпросът е дали G е свързан.

Докажете, че всеки алгоритъм за СВЪРЗАНОСТ, който прави достъпи до графа само чрез задаване
на въпроси от вида "Има ли ребро между връх i и връх j ?", задава поне n^2 въпроса, ако броят на
върховете е $2n$.

Решение: Ще използваме аргументация с противник.

Нека n е число. $\forall$ е разбито на 2 подмножества A и B ,
т.е. $|A| = |B| = n$

На запитване за върховете i и j , противникът отговаря
с Да, ако са в един дял и Не иначе.

Одният двой подмножества $\{x, y\}$, т.е. x и y са от
различни дялове е n^2 .

Да допуснем, че за някое $\{x, y\}$ алгоритъмът не е попитан.
($x \in A, y \in B$)

- Ако алг. отговори, че графът е свързан, прот. ще констр.
граф, за който A и B са клики и няма нито едно
ребро м/у тях. Така ще опровергае алг, оставяйки
консистентен с отговорите си.
- Ако алг. отговори с Не, прот. конструира граф с A и B -
клики, както и на ребро $\{x, y\}$ м/у тях. Аналогично на
горното, опровергава алг.