DAA семинар 14

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

Да се съставим ефикасен алгоритъм, който изчислява максималният брой линии, така че да няма пресичащи се

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) Не ги свързваме и разгл. за $i-1, j$
   - 2.3) Не ги св. и разгл. за $i, j-1$

3) Не можем да ги свържем:
   - 3.1) Прескачаме и двата, т.е. разгл. за $i-1, j-1$
   - 3.2) Разгл. за $i-1, j$
   - 3.3) Разгл. за $i, j-1$

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

$$f(i,j) = \begin{cases} 0, \text{ ако } i=0 \lor j=0 \\ \max\{1+f(i-1,j-1),\ f(i-1,j-1),\ f(i,j-1)\}, & \text{ако } i\neq 0 \land j\neq 0 \land A[i]=B[j] \\ \max\{f(i-1,j-1),\ f(i-1,j),\ f(i,j-1)\} & \text{ако } i\neq 0 \land j\neq 0 \land A[i]\neq B[j] \end{cases}$$

Зад 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 $\forall j = 1..n$, 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 $i = 1..m$

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

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

$\Rightarrow opt(i,j)=0$, 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" подматрица

$\Rightarrow opt(i,j)=1$

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

Ако $opt(i-1,j)=opt(i,j-1)$, то $opt(i,j)=opt(i-1,j-1)+1$

// страните на разширяваните подматрици са равни

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

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

Окончателно получаваме:

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

DP Псевдокод $T(n,m) = O(n,m)$

Зад. 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: Тогава независимо множ. в T' ще е обединението от независими подмножества за наследниците на v.

2) Включваме v: В този случай всички 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 дърво T като добавим нов атрибут към всеки връх.

Извършваме обхождането в post-order (ляво-дясно-корен), защото искаме стойностите за децата да са изчислени.

DP LeetCode: 337. House Robber III

Долни граница

- Ограничение за всички алгоритми, които решават дадена изчислителна задача

Техники:
- дърво за вземане на решения
- аргументация с противник
- редукция

Зад. 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 по 2 различни елемента

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

$$\left.\begin{array}{l} A[1] = key \\ A[2] = key \\ \vdots \\ A[n] = key \\ key \notin A[1..n] \end{array}\right\} n+1 \text{ на брой}$$

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

Доказва се*, че щом имаме $\Omega(n)$ листа, то за височината на дървото имаме 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 множ. V е разбито на 2 подмножества A и B, т.че $|A| = |B| = n$

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

Общият брой подмножества $\{x,y\}$, т.че. x и y са от различни дялове.

Да допуснем, че за някое $\{x,y\}$ алгоритъмът не е попитал.

- 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., оставайки консистентен с отговорите си.

- Ако алг. отговори с Не, прот. конструира граф с A и B - клики, като има ребро $\{x;y\}$ м/у тях. Аналогично на горното, опровергава алг.

• Редукция (свеждане): $X \propto Y$

Използване Y, за да решим X.

- Ако X е бавна, то и Y е бавна.

- Ако Y е бърза, то и X е бърза

IIр Дадени са два масива $A[1..n]$ и $B[1..n]$ от числа. Да се свържат елементите им, така че най-малкия в A да е св. с най-малък в B, втори с втори и т.н.

Нека тази изч. задача е X.

Очевидно задачата може да се реши със сортиране.

Можем да сведем X до Сортиране.

По този начин ще получим <u>горна</u> граница.

Ако покажем алгоритъм, който свежда Сортиране до X, то ще получим долна граница за X.

Нека $A[1..n]$ са числата, които искаме да сорт., а $B[1..n]$ са индекси от 1 до n.

Ако свържем A с B, т.е. решим X, ще сме решили и Сортиране

Щом $\Omega(n\lg n)$ е долна гр. за Сортиране, то това е долна гр. и за X.