ДАА семинар 13

Зад Minimum Coin Count Limited

Дадено е мултимножество от монети $\{c_1, c_2 \dots c_n\}_M$, с които разполагаме. Искаме да открем сума S с възможно най-малко монети.

Решение:

Ще разглеждаме подзадачите от всяка сума $x \leq S$ за „всяко" подмножество на $\{c_1 \dots c_n\}$.

// Ако разглеждаме наистина всички подмножества, то ще имаме таблица с 2^n реда (или колони)

За намиране на подмножествата на $\{a_1 \dots a_m\}$ имаме 2 възможности:
- a_1 + подмножества $\{a_2 \dots a_m\}$ // взимаме a_1
- подмножества $\{a_2 \dots a_m\}$ // или не

Нека $f(x, i)$ е минималният брой монети за сума x използвайки някои от първите i на брой монети.

$$\begin{cases} f(0, i) = 0 & \text{за } i = 1 \dots n \\ f(x, 0) = +\infty & \text{за } x \in [0; S] \\ f(0, 0) = 0 \end{cases}$$

$$f(x, i) = \min \begin{cases} 1 + f(x - c_i, i-1) & \text{, ако } x \geq c_i \quad \text{// взимаме } i\text{-тата монета} \\ f(x, i-1) & \text{// не я взимаме} \end{cases}$$

$\Rightarrow$ проверяме за следната сума с по-малкото монети

Зад. По колко начина може да покрием стена с размер $2 \times N$ с правоъгълни плочки 1×2?

Решение:

$f(x)$ - брят начини за покриване на дъска с дълж. x

$$\begin{cases} f(0) = 0 \\ f(1) = 1 \\ f(x) = f(x-1) + f(x-2) \end{cases}$$

Зад. Предложете ефикасен алгоритъм за намиране на брой на n-цифрените числа (в десетична бройна сис.), чиято сума на цифрите им е S.

Решение: n цифри, сума S $\longrightarrow$ $n-1$ ц., сума $S-1$

$\longrightarrow$ $n-1$ ц., сума $S-2$

...

$n-1$, $S-9$

$n-1$, S

За по-лесно да моментна гледаме n-цифрени стрингове.

$f(i, x)$ - брой i-цифрени стрингове със сума на цифрите x.

$f(i, 0) = 1$ за $i = 1..n$

$$f(1, k) = \begin{cases} 1, & \text{за } k = 0..9 \\ 0, & \text{за } k > 9 \end{cases}$$

$$f(i, x) = \sum_{k=0}^{\min(9;x)} f(i-1, x-k)$$

Псевдокод:

```
1  M[1..n][0..S] // (съдържа f(i,x), ако е попълнена
2. M[1][0..9] ← 1 ; M[1][10..S] ← 0
3. for i ← 2 to n
4.       M[i][0] ← 1
5.       for x ← 1 to S
6.           M[i][x] ← 0
7.               for k ← 0 to min(9;x)
8.                   M[i][x] += M[i-1, x-k]
9. return M[n][S] - M[n-1][S]
```

Отговорът на задачата не е просто M[n][S], защото не всички стойности са валидни числа: Тези, които започват с 0 са M[n-1][S] на брой.

$T(n,S) = S(n,S) = \Theta(n.S)$

// Може да направим $S(n,S) = \Theta(S)$ ако пазим само 2 реда - предишен и текущ, но губим възможност за възстановяване на решението.

Или $\Theta(n)$ с 10 стълба - аналогично

Зад Optimal Single Robot Path - В дадена матрица n x m е поставен робот в най-горния ляв ъгъл. Роботът се движи само надолу и надясно като целта е да стигне до долния десен ъгъл.

Предложете алгоритъм, който намира максималната сума на числата по пътя на робота.

Пр

1 → 7 12
3 11 5
12 8 → 10 → 37

Ин. f(i,j) - максимална печалба, започвайки от клетка (i,j)

$f(i,j) = A[i][j] + \max\{f(i+1,j);\ f(i,j+1)\}$

$f(i,m) = A[i][m] + f(i+1,m)$ // Последният ред

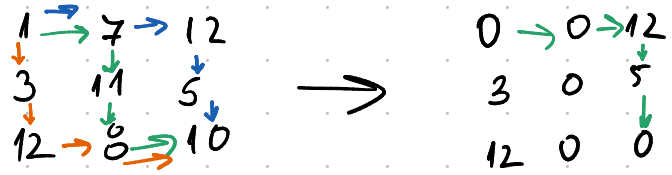
$f(n,j) = A[n][j] + f(n,j+1)$ // Последен стълб

II н. $f(i,j) = A[i][j] + \max\{f(i-1,j);\ f(i,j-1)\}$

максимална печалба по път до (i,j)

Зад Optimal Two Robot Path – Искаме максимална сума от двата робота заедно, която числото от дадена клетка може да се вземе само веднъж.

Пр



I робот – 37 II робот – 17 37+17 = 54

Алчното решение да пуснем единия робот след другия не е оптимално.

Може да съберем $58 = 1+3+12+8+10+7+12+5 = 58 > 54$

Нека (i_1,j_1) е позицията на I робот.
(i_2,j_2) – II робот.

$$f(i_1,j_1,i_2,j_2) = \begin{cases} A[i_1][j_1] + A[i_2][j_2] + \max\{*\}, & \text{ако } i_1 \neq i_2 \vee j_1 \neq j_2 \\ A[i_1][j_1] + \max\{*\}, & \text{иначе} \end{cases}$$

$\max\{*\}$ е най-добрата възможност за ход на роботите

$$\max\{*\} = \max\begin{cases} f(i_1+1, j_1, i_2+1, j_2) & // \text{Down и Down} \\ f(i_1+1, j_1, i_2, j_2+1) & // \text{Down и Right} \\ f(i_1, j_1+1, i_2+1, j_2) & // \text{RD} \\ f(i_1, j_1+1, i_2, j_2+1) & // \text{RR} \end{cases}$$

$T(n,m) = \Theta(n^2 \cdot m^2)$ - квадратично по входа

Можем да спестим един параметър: $i_1 + j_1 = i_2 + j_2$ - номерът на диагонала е винаги един и същ за роботите.

$\Rightarrow j_2 = i_1 + j_1 - i_2$ (например)

$\Rightarrow \Theta(n^2 m)$ или $\Theta(nm^2)$

Зад Minimum Deletions to Palindrome или
Longest Palindrome Sequence // ще използваме тази

Дадена е дума. Да се намери колко най-малко символа трябва да се премахнат, за да стане палиндром.

Пр a b x c b y z a w
— — — — — палиндром

Решение: Нека $f(i,j)$ е най-дългата палиндромична поредица в $w[i \dots j]$

Търсим $f(1,n) = ?$

$$f(i,j) = \max\begin{cases} f(i+1, j-1) + 2, \text{ако } w[i] = w[j] \\ // \text{Взимаме крайните и гледаме в по-малката дума} \end{cases}$$

$$\begin{cases} f(i+1,j) & // \text{Премахваме първата буква} \\ f(i,j-1) & // \text{Игнорираме последната буква} \\ f(i+1,j+1) & // \text{Игнорираме и двете} \end{cases}$$

$f(i,i) = 1$ // Имаме 2 дена по диагонала

$$f(i,i+1) = \begin{cases} 2, & \text{ако } w[i] = w[i+1] \\ 1, & \text{иначе} \end{cases}$$

Пр abba

i \ j	1	2	3	4
1	1	1	2	
2		1	2	
3			1	1
4				1

Псевдокод:

```
...
for d ← 3 to n
    for i ← 1 to n-d+1
        for j ← 1 to d-i+1
            dp[i][j] ← ...
return dp[1][n] // или n-dp[1][n] за брой изтривания
```

Зад Даден е регулярен израз $r[1..n] \in \{a, b, .., z, *, |\}^n$ и дума $w[1..m] \in \{a..z\}^m$. Да се провери дали думата се съдържа от регулярния израз.

Подсказките входни данни ще бъдат валидни.

Решение: Ще направим рекурсивна декомпозиция по дължината на думата и израза.

$$f(0,0) = T$$

$$f(i,j) = \begin{cases} f(i-1, j-1), & \text{ако } w[i] = regex[j] \vee regex[j] = '.' \quad 1) \\ f(i, j-2), & regex[j] = '*' \wedge f(i, j-2) \quad 2) \\ f(i-1, j), & regex[j] = '*' \wedge (w[i] = regex[j-1] \vee \\ & \qquad \vee regex[j-1]) \quad 3) \\ F, & \text{иначе} \quad 4) \end{cases}$$

1) Ако буквите в думата и израза съвпадат, гледаме дали досега сме разпознали.

Аналогично ако изр. е '.', т.е. всяка е буква.

2) Ако изразът е '*' (произволен брой пъти предния символ), то не ни интересуват този и предходния символ от израза.

Разглеждаме същата дума с по-кус израз и взимаме стойността само ако е истина.

3) Ако изр. е '*' и преходният символ съвпада с текущата буква от думата, то тя се разпознава, ако същия израз разпознава и думата без последния символ.

4) В останалите случаи, например ако се различават буквите, връщаме False.