Report for Programming Problem 2 - ARChitecture

Team: 2018297934_2018298731

Student ID: 2018297934 Name Luis Carlos Lopes Loureiro

Student ID: 2018298731 Name João Pedro Pacheco Silva

Descrição do Algoritmo:

Para solucionar o problema começámos por considerar apenas a construção de semi arcos em vez de arcos completos, tendo como subproblema a construção de semi arcos com uma dada altura e com um dado número de peças.

Para construir um semi arco de altura H e com n blocos basta adicionar 1 bloco aos semi arcos com n-1 blocos e de alturas inferiores a H. Tendo isso em conta consegue-se aproveitar os resultados dos subproblemas anteriores da seguinte forma:

$$f(H, n) = \begin{cases} 0, \text{casos impossiveis} \\ 1,\ n = \mathrm{min\_n}(\mathrm{H}, \mathrm{h}) \\ \sum_{i=1}^{h-1} \mathrm{f}(\mathrm{H}-\mathrm{i}, \mathrm{n}-1) \end{cases}$$

em que f(H, n) denota o número de combinações para semi arcos de altura H e n blocos, min_n(H, h) denota o número mínimo de blocos para chegar à altura H e h denota a altura dos blocos. Considera-se como caso impossível sempre o H é demasiado grade ou demasiado pequeno para conseguir construir o semi arco com n blocos.

Outra observação que se pode tirar é que:

$$\begin{aligned} \sum_{i=1}^{h-1} f(H-i, n-1) &= f(H-1, n-1) + \sum_{i=2}^{h-1} f(H-i, n-1) \\ &= f(H-1, n-1) + \sum_{i=1}^{h-2} f(H-(i+1), n-1) \\ &= f(H-1, n-1) + \left(\sum_{i=1}^{h-1} f(H-1-i, n-1) \right) - f(H-1-(h-1), n-1) \\ &= f(H-1, n-1) + f(H-1, n) - f(H-h, n-1) \end{aligned}$$

Ou seja, f(H, n) = f(H-1, n-1) + f(H-1, n) - f(H-h, n-1).

Esta simplificação tem bastante impacto na complexidade temporal do algoritmo dado que conseguimos transformar uma operação que crescia linearmente em função h em outra que apresenta complexidade constante. Trocando f por uma matriz conseguimos traduzir a expressão no seguinte pseudocódigo:

```
f[h+1][2] = 1                                              {caso base}
for n ← 3 to N do
    for H ← min_H(n, h) to max_H(n, h) do
        f[H][n] ← f[H-1] [n-1] + f[H-1][n] – f[H-h] [n-1]
    end for
    F[H] ← f[H] + f[H][n];
end for
```

Sendo N o numero máximo de blocos, min_H(n, h) a altura mínima para semi arcos de n blocos, max_H(n, h) a altura máxima para semi arcos de n blocos e F um array que guarda as cobinações de semi arcos de diferentes alturas (necessário para a proxima etapa do algoritmo).

Melhorias:

Em vez de considerar os casos impossíveis como 0, simplesmente saltar esses mesmos casos e não os incluir nos cálculos para os restantes semi arcos.

Para construir arcos completos de altura H é necessário combinar dois semi arcos de altura H. Portanto, o número de combinações para um arco resulta da multiplicação do número de combinações do seu lado esquerdo com o do seu lado direito, ou seja:

```
Arco[H] ← F[H] * F[H]
```

É importante ter em conta que só se pode combinar dois semi arcos caso o número total de blocos não exceda N, e portanto, a equação anterior só aplicável para alturas em que 2·max_n(H, h)-1 ≤ N. Para os restantes casos o cálculo deve ser feito da seguinte forma:

```
int Arco ← 0
for i ← min_n(H, h) to N-i+1 do
    if N-i+1 < max_n(H, h) then
        F[H] ← F[H] - f[H][N-i+1+1]
    end if
    F[H]  ← F[H]  - f[H][i]
    Arco ← Arco + f[H][i] · F[H] · 2          {incluir as combinações inversas}
    Arco ← Arco + f[H][i] · f[H][i]
end for
```

À medida que i aumenta deve retirar-se de F[H] as combinações para as quais deixa de haver blocos suficientes.

A abordagem usada é a correta dado que elimina as repetições de calculo computacional desnecessárias.

Estruturas usadas:

Usou-se uma matriz de inteiros para poder guardar as combinações dos semi arcos com diferentes alturas e número de blocos e usou-se um array de inteiros para guardar as combinações de semi arcos com diferentes alturas.

Complexidade:

A complexidade espacial do algoritmo é de $N \cdot (H-h) + (H-h)$.

Como $H \leq (h + ((N/2)-1)*(h-1))$ então no pior caso temos

$N \cdot ((N/2)-1) \cdot (h-1) + ((N/2)-1) \cdot (h-1) = ((N^2/2)-N) \cdot (h-1) + ((N/2)-1) \cdot (h-1)$

$= ((N/2)-1 + (N^2/2)-N) \cdot (h-1) \in O(N^2)$, sendo H a altura máxima dos arcos, N o número máximo de blocos de num arco e h a altura dos blocos.

A complexidade temporal do algoritmo é de $(H-h)(N-2)+(H-h)(N-2) = 2(H-h)(N-2)$.

Como $H \leq (h + ((N/2)-1)*(h-1))$ então no pior caso temos

$2((N/2)-1)*(h-1)(N-2) = (N-2)(h-1)(N-2) = (N-2)^2(h-1) \in O(N^2)$.

Referencias:

- Slides das aulas teoricas (“Week04 - T - Dynamic Programming” e “Week05 - T - Dynamic Programming (cont)”).