Relatório 1º Projeto ASA 2022/2023

Grupo: AL011
Alunos: Raquel Braunschweig (102624), Madalena Cabrita (103595)

Descrição do Problema e da Solução

O problema apresentado consiste em conseguir determinar o número de combinações diferentes possíveis, utilizando quadrados de diversos tamanhos, para ladrilhar uma dada área delimitada por uma escada. A grelha em que a escada se encontra é definida por um retângulo com N linhas e M colunas.

Para resolver o problema decidiu-se criar uma função para calcular o número de combinações associadas a uma escada, que se chama a si própria recursivamente, retirando as partes mais à direita da escada original, transformando-a assim em escadas mais pequenas e calculando os valores associados a essas escadas, guardando-os num unordered map. Neste mapa, as keys correspondem a um valor numérico calculado por uma hash function associado a cada escada, enquanto os values são o número de combinações de cada escada. Quando a recursão termina, é então possível devolver o valor de combinações associado à escada original.

Análise Teórica

Seja N o número de linhas e M o número de colunas no retângulo, s[] o vetor representativo da escada e c(s,v) o unordered_map usado.

Leitura dos dados de entrada: simples leitura do input e inserção do input no vetor representativo da escada, de tamanho igual ao número de linhas, logo **Θ(N)**;

Transformação do vetor representativo de escadas num número inteiro(turnToInt(s[])): usa um ciclo linearmente dependente do tamanho do vetor para associar a este um número inteiro, ou seja, depende linearmente de N, logo **O(N)**;

Encontrar a posição mais à direita de uma escada:

```
findRightPosition(s[])
  biggest<-s[0];
  index<-0;
   for i=1 to N-1 do
     if s[i]>biggest then
       biggest<-s[i];
       index<-i;
     endif
     if biggest==M then
       return index;
   end for
 return index;
```

Usa um ciclo linearmente dependente de N, logo **O(N)**

Verificar se já é uma escada vazia:

```
checkIfAllZero(s[])
   for i=0 to N-1 do
     if s[i]!=0 then
         return false;
   end for
   return true;
```

Usa um ciclo linearmente dependente de N, logo **O(N)**

Retirar/adicionar um quadrado à escada: depende do tamanho do quadrado, que no máximo será (N - linha a ser analizada), ou seja, **O(N)**;

Aplicação do algoritmo de cálculo de número de combinações possíveis a uma escada:

```
findNumberOfConfig(s[])
    numberOfConfig<-0;
    hash<-turnToInt(s[]);
```

```
if hash ϵ escadas já calculadas then
      return v;
else
    line=findRightPosition(s[]);
    for i=1 to N do
        if quadrado de tamanho i pode ser retirado na linha line
            retirar “quadrado” da escada;    // (ou seja, de s[])
            numberOfConfig<- numberOfConfig + findNumberOfConfig(s[]);
            voltar a adicionar o quadrado retirado à escada;
        endif
        else
           break;
    endfor
endelse
inserir elemento (hash, v) no mapa
return numberOfConfig;
```

(v= value associado à key hash no mapa)

O(N)

Esta função chama-se recursivamente até serem calculados todos os números de configurações possíveis para todas as combinações de escada mais pequenas que a original. No pior caso, são determinadas todas as combinações de escadas para um retângulo de tamanho NxM.

Caso N=M, estamos perante $\binom{2n}{n}$ combinações de escada possíveis, ou seja, uma função com complexidade O$\left(\frac{2^{2n}}{\sqrt{n\pi}}\right)$;

Caso N≠M, estamos perante $\frac{(m+n)!}{m!n!}$ combinações de escada possíveis.
Desta forma, esta função tem complexidade exponencial.

Apresentação de dados: simples cout de um número, **O(1)**

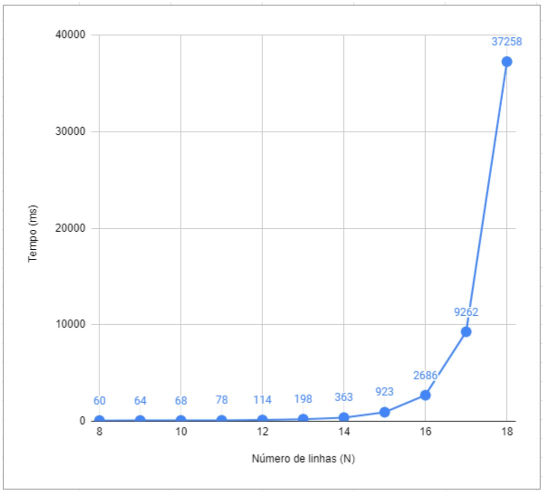
Complexidade global da solução=complexidade de cálculo de nº de combinações. Se for um quadrado, corresponde a O$\left(\frac{2^{2n}}{\sqrt{n\pi}}\right)$; se for um retângulo é igual a lim de $\frac{(m+n)!}{m!n!}$.

Avaliação Experimental dos Resultados

Testando o pior caso possível em quadrados de lado de dimensão N, entre N=8 e N=18, obtiveram-se os seguintes resultados:

8	60
9	64
10	68
11	78
12	114
13	198
14	363
15	923
16	2686
17	9262
18	37258

lado tempo(ms)



Observa-se claramente que o crescimento de tempo de execução da solução cresce exponencialmente em relação ao tamanho do quadrado, ou seja, a N, provando-se assim que a nossa implementação está de acordo com a análise teórica.