Problema de Programação 2 - ARChitecture

Student ID: 2016228735 Name: Afonso Matoso Magalhães
Student ID: 2017251509 Name: Leandro Pais

Maio 2021

1 Algorithm Description

Inicialmente é feita uma passagem por todos os valores de $3 < i < n$ em que verificamos se existe pelo menos uma solução possível dados os valores de altura do bloco e da altura da sala (Ver 1) caso não exista pelo menos uma solução para nenhum dos valores podemos afirmar que não existe solução e o algoritmo retorna. No caso de haver alguma solução, é guardado o último valor de i com solução na variável last_n. Percorre-se a matriz de soluções (sols) desde i = last_n - 2 até i = 1.

Para cada valor de i é calculado o valor máximo da altura cujos blocos podem assumir e guardado na variável current_max_height, sendo que estes têm de chegar ao bloco inicial descendo h-1 de altura de cada vez, o que é garantido com a condição $current\_max\_height <= (h-1) * i$, e não podem ultrapassar a altura da sala, assegurado com $current\_max\_height <= H - h$. A variável next_max_height irá guardar a altura máxima possível que os blocos podem assumir na coluna asseguir à atual ($i + 1$, sendo que começa a 0, pois na última coluna da sala com solução só há uma solução no primeiro índice, e é atualizada para o valor de current_max_height em cada iteração do ciclo. Se $i = 1$, então temos de calcular as soluções ascendentes para dp[1][1], pois é o único índice de altura 1 que tem esse tipo de soluções, todos os outros têm apenas 1 solução descendente (incluindo dp[1][1]). No segundo ciclo for percorremos todos os indices com $j = 2$ até $j <= current\_max\_height$, de modo a calcular as soluções possíveis a partir de um bloco em cada posição da sala. Para otimizar o algoritmo, aproveitamos sempre as soluções do índice anterior para o atual, pois os índice visitados a partir desses são quase todos os mesmos.

No fim, vamos somar todas as soluções calculadas e que se encontram em dp[1], de modo a termos todas as soluções possíveis no índice inicial da matriz.

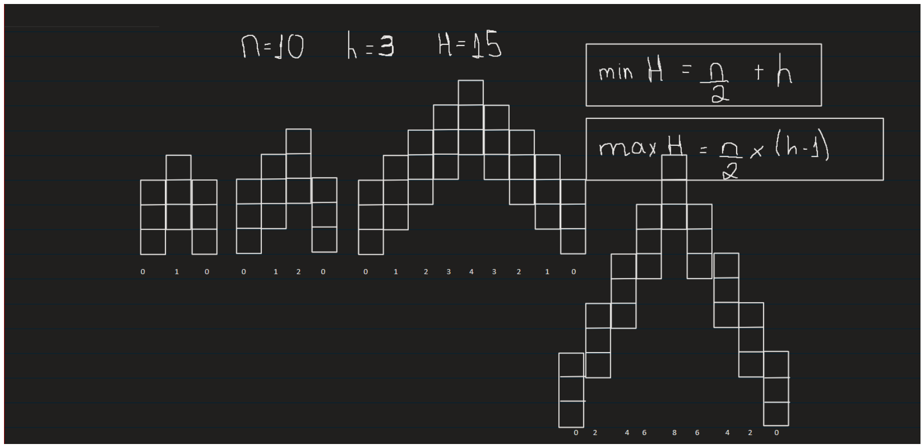

Figure 1: Visualização das condições

2 Data Structures

Relativamente às estruturas de dados utilizadas temos apenas uma matriz tridimensional estática alocada com os tamanhos de n e H que são respetivamente 500 e 60000 e com 2 índices na terceira dimensão, sendo o primeiro para guardar as soluções possíveis ascendentes e o segundo as descendentes, a partir de um bloco nessa posição.

3 Corretcness

No mooshak conseguimos obter a totalidade dos pontos com esta implementação.

4 Algorithm Analysis

Inicialmente, é feita uma única passagem por todos os valores $3 < k < n$, ou seja, aproximadamente $\mathcal{O}(n)$. De seguida, temos no core do algoritmo temos dois ciclos encadeados que apontaria para uma complexidade temporal quadrática, contudo, com os cortes que conseguimos efetuar, preservando cálculos de blocos anteriores, o algoritmo está mais próximo de $\mathcal{O}(n * log(n))$ o que faz com que o programa corra bastante rápido nas restrições dadas para n, H e h no mooshak. Em termos de complexidade espacial, e tirando partido destas mesma restrições temos uma matriz de tamanho fixo alocada na stack cujas dimensões são 500 (valor máximo de n) por 60000 (valor máximo de H) por 2 (os blocos ascendentes e descendentes). Como não é utilizada nenhuma outra estrutura de relevância apontamos para uma complexidade espacial final de $\mathcal{O}(500 * 60000 * 2)$.

References

Nenhuma referência usada.