Relatório do Trabalho Prático 2
Algoritmos I

Nome do Aluno: Isadora Cunha Pimentel
Curso: Sistemas de Informação - DCC/UFMG
Matrícula: 2021039506

Introdução

Este trabalho prático tem como objetivo resolver dois problemas distintos de otimização propostos pela prefeitura de Triangulândia, aplicando paradigmas fundamentais de projeto de algoritmos: Algoritmos Gulosos e Divisão e Conquista.

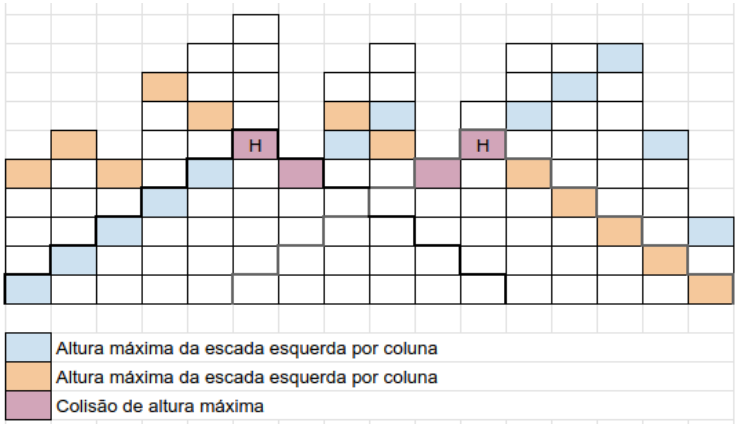
O primeiro problema consiste em determinar a altura máxima de um muro em formato de triângulo isósceles que pode ser construído a partir de pilhas de blocos existentes, sem adicionar novos materiais. O segundo problema envolve a seleção de três árvores em um plano cartesiano para formar uma área de preservação ambiental triangular com o menor perímetro possível.

Modelagem

Parte 1: O Muro Simétrico (Algoritmo Guloso)

O problema do muro foi modelado como um vetor de inteiros *muro* de tamanho *nColunas*, onde *M[i]* representa a altura da pilha na posição [i]. O objetivo é encontrar a altura máxima, tal que seja possível formar uma estrutura triangular onde a altura dos blocos cresce linearmente até um pico e depois decresce.

Para resolver este problema, utilizou-se uma abordagem gulosa baseada em programação dinâmica simplificada. A "escolha gulosa" local consiste em garantir que, para cada posição, a altura da "escada" que sobe ou desce respeite tanto a quantidade de blocos disponíveis naquela posição quanto a altura da escada adjacente. Ao cruzar esse limite entre as escadas crescente e decrescente é possível determinar os mínimos em cada coluna e, assim, encontrar a maior altura para fazer um muro triangular simétrico.



Parte 2: Área Ambiental (Divisão e Conquista)
O problema da área ambiental foi modelado como um conjunto de Árvores no plano. O objetivo é encontrar a tripla de pontos (a1, a2, a3) que minimiza a soma das distâncias euclidianas entre eles (perímetro), com o critério de desempate sendo a ordem lexicográfica dos índices originais das árvores.

A solução utiliza o paradigma de Divisão e Conquista, similar ao clássico algoritmo para encontrar o par de pontos mais próximo (Closest Pair of Points). O problema é dividido recursivamente em metades menores baseadas na coordenada X, resolvido para cada metade, e então combinado verificando se existe um triângulo de menor perímetro que cruza a linha divisória.

Solução

Parte 1: Solução Gulosa
A solução baseia-se no cálculo de dois vetores auxiliares:

- Escada Esquerda (L): *L[i]* armazena a altura máxima de uma rampa crescente que termina em *i*. A recorrência é *L[i] = min(M[i], L[i-1] + 1)*.
- Escada Direita (R): *R[i]* armazena a altura máxima de uma rampa decrescente que começa em *i*. A recorrência é R[i] = min(M[i], R[i+1] + 1).

A altura máxima possível em qualquer posição *i* é limitada pelo menor valor entre as duas rampas que convergem para aquele ponto: *min(L[i], R[i])*. A resposta final é o máximo valor encontrado para todas as posições.

Algorithm 1 Altura Máxima do Triângulo

```
Calcular L[i] da esquerda para a direita
Calcular R[i] da direita para a esquerda
max_altura ← 0
for i ← 0 até N − 1 do
    altura_local ← min(L[i], R[i])
    max_altura ← max(max_altura, altura_local)
end for
return max_altura
```

Parte 2: Solução por Divisão e Conquista

O algoritmo segue os passos:

1) **Pré-processamento:** Ordenar os pontos pela coordenada X com MergeSort.

2) **Caso Base:** Se o número de pontos for pequeno é usada força bruta para encontrar o menor perímetro. Foi considerado o limite de 6 pontos para o caso base, pois não chegava a afetar o tempo de processamento (força bruta é O(N^3) ao mesmo tempo que reduzia o número de chamadas recursivas.
3) **Divisão:** Dividir o conjunto de pontos em duas metades, esquerda e direita, divididas por uma linha vertical média.
4) **Conquista:** Calcular recursivamente o menor perímetro em dL e dR. Seja d = min(dL, dR).
5) **Combinação (Faixa):** Criar uma faixa contendo apenas os pontos cuja distância horizontal para a linha média é menor que d. Ordenar esses pontos por Y. Verificar trios de pontos dentro dessa faixa. Devido à geometria, para cada ponto, basta verificar um número constante de vizinhos na ordem Y.

Análise de Complexidade

Parte 1

- **Tempo:** O algoritmo percorre o vetor de entrada três vezes (uma para L, uma para R e uma para o máximo final). Sendo N o número de pilhas, cada iteração realiza operações constantes O(1). Logo, a complexidade temporal é \textbf{linear, O(N).
- **Espaço:** Foram utilizados dois vetores auxiliares de tamanho N. Logo, a complexidade espacial é O(N).

Parte 2

- **Tempo:** A ordenação inicial leva *O(N log N)*. A relação de recorrência do algoritmo é *T(N) = 2T(N/2) + O(N)*. O termo *O(N)* vem da construção da faixa e da ordenação dos pontos na faixa pela coordenada Y. Pelo Teorema Mestre, isso resulta em uma complexidade de *O(N log N)*.A verificação na faixa é *O(N)* pois o loop interno executa um número constante de vezes devido à limitação geométrica do perímetro de d.
- **Espaço:** A complexidade espacial é O(N) devido ao uso de vetores auxiliares para a recursão e armazenamento da faixa.

Considerações Finais

A realização deste trabalho permitiu a aplicação prática de conceitos teóricos importantes. A Parte 1 (Guloso) foi mais direta, exigindo apenas uma intuição correta sobre como as restrições locais se propagam. A Parte 2 (Divisão e Conquista) apresentou maior dificuldade, principalmente na etapa de combinação e no tratamento correto dos casos de empate para garantir a ordem lexicográfica mínima exigida. A implementação cuidadosa das estruturas de dados e a otimização

das comparações foram essenciais para atingir a eficiência desejada nos casos de teste maiores.

Referências

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., \& Stein, C. (2009). \textit{Introduction to Algorithms} (3rd ed.). MIT Press. (Capítulo sobre Geometria Computacional e Par de Pontos Mais Próximo).
- Material de aula da disciplina de Algoritmos I - DCC/UFMG.
- Tutoriais do GeekforGeeks para consulta dos algoritmos MergeSort e Closest Pair of Points.https://www.geeksforgeeks.org/dsa/closest-pair-of-points-onlogn-implementation/ e https://www.geeksforgeeks.org/dsa/merge-sort/.