Análise e Desenho de Algoritmos 2024/25 – Trabalho 2

Maurício Pólvora (Nº 64879) e Alexandre Marques (Nº 65370)

NOVA School of Science and Technology - Universidade NOVA de Lisboa

1 Resolução do Problema

O problema "Middle Ages"apresenta um cenário onde viajantes na Idade Média precisam determinar rotas entre locais, considerando o tempo para percorrer cada estrada. Os viajantes "frágeis"viajam exatamente uma estrada por dia, e a "dureza"de uma jornada é definida como o tempo máximo de viagem diária. Uma "boa jornada"é aquela que minimiza esta dureza.

A nossa solução para este problema baseia-se na observação de que, numa jornada ótima entre dois locais, o caminho deve minimizar o tempo máximo de qualquer estrada. Esta característica alinha-se perfeitamente com as propriedades de uma Minimum Spanning Tree - MST modificada, onde o peso a ser minimizado não é a soma total, mas o máximo peso individual em qualquer caminho.

1.1 Abordagem da Solução

A nossa abordagem utiliza dois algoritmos clássicos da teoria dos grafos:

1. **Algoritmo de Kruskal**: Utilizado para construir uma Minimum Spanning Tree (MST) do grafo ponderado que representa as estradas e locais.
2. **Busca em Profundidade (DFS)**: Aplicada sobre a MST para encontrar a hardness entre o local de partida e o destino.

A intuição por trás desta abordagem é que, ao construir uma MST ordenando as estradas por tempo de viagem, garantimos que, para qualquer par de locais, existe um único caminho entre eles na MST, e este caminho minimiza o peso máximo de qualquer aresta. Esta propriedade é exatamente o que precisamos para encontrar as "boas viagens"como definido no problema.

1.2 Implementação

A nossa implementação é composta por duas etapas principais. Na classe Problem, definimos uma série de estruturas de dados para representar o grafo e sua MST resultante, incluindo uma lista para armazenar todas as estradas (roads) e um array de listas ligadas para representar a MST.

Na primeira etapa, implementamos a construção da MST utilizando o algoritmo de Kruskal. Este processo começa ordenando todas as estradas por tempo de viagem em ordem crescente, garantindo que as estradas mais rápidas sejam consideradas primeiro. Em seguida, utilizamos uma estrutura de dados Union-Find (UnionFindInArray) com path compression para detectar e evitar ciclos durante a construção da MST. Para cada estrada na lista ordenada, verificamos se os dois locais que ela conecta já estão ligados por algum caminho. Se não estiverem, adicionamos a road à MST e unimos as componentes. Esta operação é realizada através do método buildMST(), que é executado apenas uma vez e reutilizado para todas as consultas subsequentes.

Na segunda etapa, após a construção da MST, utilizamos o algoritmo de Busca em Profundidade (DFS) implementado no método findHardness() para encontrar o caminho com menor dureza entre o local de partida e o destino. Iniciamos a busca a partir do local de origem, marcando-o como visitado (utilizando um array visited) e atribuindo zero ao seu valor máximo de tempo de viagem.

Para cada local atual no topo da pilha, exploramos seus vizinhos não visitados. Quando encontramos um vizinho não visitado, marcamos-o como visitado, calculamos o valor máximo entre o tempo atual e o tempo necessário para percorrer a nova estrada, e adicionamos o vizinho à pilha. Continuamos a exploração a partir deste novo local. Se em algum momento não conseguirmos encontrar vizinhos não visitados para o local atual, realizamos um "backtracking"removendo-o da pilha e retornando ao local anterior. Este processo continua até encontrarmos o destino ou até que todos os locais acessíveis tenham sido explorados.

O array maxRoadTime armazena, para cada local, o tempo máximo de qualquer estrada no caminho desde a origem até aquele local. Esta solução funciona corretamente para o problema porque estamos a trablhar com uma MST, onde existe exatamente um caminho entre quaisquer dois locais. O valor final em maxRoadTime para o destino representa exatamente a dureza da melhor jornada conforme definido no problema.

2 Complexidade Temporal

A complexidade temporal da nossa solução pode ser dividida nas duas etapas principais:

2.1 Construção da MST

- Ordenação das estradas: $O(R \log R)$, onde R é o número de estradas
- Processamento das estradas com Union-Find: $O(R \cdot \alpha(L))$, onde L é o número de locations e α é a função inversa de Ackermann (praticamente constante para os valores dos bounds do trabalho.)

A complexidade nesta etapa fica $O(R \log R)$.

2.2 DFS para Encontrar o Caminho

- Percorrer a MST usando DFS: $O(L + R')$, onde R' é o número de estradas na MST (que é $L - 1$ para uma árvore)

A complexidade desta etapa permanece $O(L)$, pois estamos a visitar cada nó e cada aresta da MST no máximo uma vez.

2.3 Complexidade Total

Considerando que a MST é construída apenas uma vez e reutilizada para todas as consultas, temos:

- Construção inicial da MST: $O(R \log R)$
- Para cada consulta: $O(L)$
- T consultas

Complexidade Total: $O(R \log R + T \cdot L)$

3 Complexidade Espacial

A complexidade espacial da nossa solução é determinada pelas estruturas de dados utilizadas:

- Lista de estradas: $O(R)$
- Representação da MST como listas de adjacência: $O(L + R')$, onde R' é $L - 1$ para uma árvore
- Estruturas auxiliares para DFS (array visitados, stack, array maxRoadTime): $O(L)$
- Estrutura Union-Find: $O(L)$

A complexidade espacial total é, portanto, $O(L + R)$.

4 Conclusão

A solução proposta para o problema "Middle Ages"utiliza uma abordagem baseada em algoritmos de grafos, combinando a construção de uma Minimum Spanning Tree (MST) com técnicas de busca para otimizar a jornada entre locais. A eficiência do método é garantida pela redução do espaço de busca através da MST, que é construída uma única vez e reutilizada para todas as consultas, assegurando desempenho consistente mesmo em cenários com múltiplos casos de teste. Entretanto, a abordagem assume implicitamente a conectividade total dos locais (como previsto no enunciado) e pode enfrentar limitações de memória em grafos grandes devido à representação da MST com listas ligadas.