Documentação TP2

Giovanna Louzi Bellonia
Algoritmos I

22 de maio de 2019

1 Descrição do Problema

Busca-se criar um algoritmo que, dado um conjunto de n posiçoões de telescópios ao redor da Terra, onde cada posição corresponde a um par $(lat, long)$, encontra-se a menor distância máxima entre esse pontos. Para isso, deve se modelar um grafo com vértices representando os telescópios e arestas representando a distância entre os pares desses telescópios. Dado esse grafo, retorna a aresta que minimiza a distância máxima necessária para conectar todos os pares de cidades sem ciclos com o comunicador mais barato possível, dado que todos os comunicadores precisam ser idênticos para comunicarem uns com os outros.

2 Estruturas de dados e algoritmos utilizados

2.1 TADs Utilizados

Foi criado um tad **grafo_t** com os seguintes componentes:

- **int quant_vert:** indica a quantidade de vertíces que o grafo possui;
- **int quant_arestas:** indica a quantidade de arestas que o grafo possui;
- **floresta_t *floresta:** indica os componentes conexos do grafo, que são encontradas através da função DFS;
- **aresta_t *arestas:** guarda as arestas do grafo em um vetor de arestas;
- **lista_t **lista_adj:** um vetor de listas onde a posição n do vetor indica o vértice n, e os nós da lista dessa posição indica os vértices conectados ao vértice n e o peso da aresta que conecta ambos;

2.2 Algoritmos utilizados

2.2.1 Mediana das Medianas

O algoritmo da mediana das medianas retorna a mediana de um vetor sem a necessidade de ordena-lo. Ele encontra uma mediana aproximada a partir das repartições

recursivas desse vetor em vetores menores e, a partir dela, separa o vetor em valores maiores ou menores que esse encontrado.

Dessa forma, recursivamente chama a função novamente para o pedaço do vetor em que com certeza a mediana está, até que a mediana aproximada encontrada esteja na posição exata da mediana real (passada como parâmetro k).

2.2.2 DFS

A DFS (*Depth-First Search*) é um algoritmo de busca em profundidade. Ela é utilizada no código para encontrar os componentes conexos de um grafo. Para cada iteração, inicia-se de um vértice ainda não visitado e encontra-se todos os vértices que ele consegue alcançar pela busca em largura, ou seja, que participam do mesmo componente conexo.

Vértices que participam de um mesmo componente, terão mesmo número na posição do vetor utilizado na DFS que representa esses vértices. Ou seja, se ao retornar a DFS temos um vetor [0 0 1 2 3 2], indica que temos quatro componentes conexos (0,1,2 e 3) e que os vértices 0 e 1 participam do mesmo componente 0, os vértices 3 e 5 do mesmo componente 2, e que os vértices 2 e 4 estão conectados a nenhum outro vértice.

2.2.3 Camerini

O algoritmo de Camerini original retorna uma MBST (*Minimum Bottleneck Spanning Tree*) que é uma árvore geradora que tem sua aresta máxima como a menor possível dentre todas as possíveis árvores geradoras do grafo. No caso do tp, o algoritmo está adaptado para retornar apenas essa aresta, que é nossa resposta final.

O algoritmo possui os seguintes passos:

1. Retorna a aresta do grafo G passado como parâmetro caso ela seja única (será a resposta);
2. Encontra a mediana dos pesos das arestas do grafo G;
3. Cria o grafo A e B tal que o grafo B tenha metade da quantidade de arestas do grafo G, sendo elas menores ou iguais a mediana, e que o A tenha a outra metade maior ou igual a mediana;
4. Encontra os componentes conexos de B através da DFS;
5. Se o grafo B tem apenas um componente conexo, é chamado recursivamente a função para o grafo B; se não:
6. É criado um grafo C nos quais os vértices são os componentes conexos de B que viram supervértices e as arestas são as de A que conectam esses componentes e a função é chamada recursivamente pra esse grafo C.

2.3 Makefile

Para gerar o compilável tp2 é só utilizar o comando *make*.

3 Análise de Complexidade

3.1 Espaço

Os grafos utilizados possuem listas de adjacência. Como temos, para o grafo inicial (que é o maior), V vértices que se conectam a V-1, temos como complexidade de espaço $O(V^2)$.

3.2 Tempo

O algoritmo das medianas utiliza da lógica do quickselect para funcionar. O quickselect pode gastar um tempo $O(n^2)$ quando o pivot não é tão bem escolhido, o que faz com que a quantidade do conjunto de elementos sendo analisados reduza devagarosamente, levando a esse tempo excessivo. Entretanto, quando o pivô é bem escolhido, a quantidade de elementos analisados no proximo passo reduz rapidamente, tendo um tempo O(n). Sabemos que o pivot da mediana é bem escolhido já que é garantido que a mediana esteja entre 30% e 70% da posição do vetor, logo temos o problema reduzido em no mínimo 30% sempre a cada iteração. Logo, para E arestas, teremos o tempo de O(E).

O algoritmo de Camerini a cada recursão chama o algoritmo para $E/2^i$ arestas, logo, teremos $O(E + E/2 + E/4... + 1) = O(E)$.

Já a DFS gasta um tempo relacionado a quantidade de arestas e a quantidade de vértices (V), $O(E + V)$. Entretanto, como a cada recursão é chamado para um número cada vez mais reduzido de vértices, esse não influncia tanto no tempo final.

Então, somando o tempo de todos os algoritmos, podemos concluir que o tempo final gasto é O(E).

4 Prova de Corretude

Se o grafo B é um grafo induzido conexo de G, então a MBST de G também é a MBST de B, por isso chamamos recursivamente para o grafo B no passo 5. Isso é verdade pois, temos que a menor aresta máxima de todos as arvóres geradoras de G estará dentro do conjunto de arestas de B (já que ele possui uma das possibilidades de árvores geradoras e já que tem o conjunto de menores arestas de G).

Agora, se o grafo B é um grafo induzido desconexo de G, então a MBST de G é a MBST de C que criamos no passo 6 do algoritmo. Isso funciona pois sabemos que o valor da aresta máxima pertence ao grafo A (já que ela estaria em B se o B fosse conexo como provado em cima) e sabemos que só podem ser aquelas que conectam os componentes desconexos de B (que são as que acrescentamos em C), já que as arestas de B que ligam componentes conexos a outros serão menores ou iguais do que qualquer aresta de A que poderia substituí-las. Logo, para formar C, desconsideramos as arestas que pertenciam a B e acrescentamos apenas as arestas de A que conectam esses componentes desconexos e chamamos recursivamente a função para o grafo C, que retornará a mesma resposta para o grafo G.

5 Avaliação Experimental

Para a análise experimental foram criados 700 testes aleátorios de forma que houvesse uma quantidade crescente de arestas a cada teste feito. Cada teste foram rodados 10 vezes e tirado a média do tempo gasto para a resolução do mesmo. No fim os valores dos tempos médios gastos em cada teste foram plotados no gráfico a baixo de acordo com a quantidade de arestas.

Figura 1: Gráfico para Arestas X Tempo

Com isso, pode-se perceber, pelo formato do gráfico, que o tempo gasto realmente é linear em relação a quantidade de arestas ($O(E)$).