Relatório de Atividade – Implementação do Método Kruskal para Árvores Geradoras de Custo Mínimo

Nome: Raul Augusto Teixeira; *RA:* 205177; 1º semestre – 2020

1. Implementação

O programa em anexo *Implementacao_Kruskal.m* é uma implementação em MATLAB do Algoritmo de Kruskal para Árvores Geradoras de Custo Mínimo. Recebe como entrada um grafo com m nós e n arestas não-direcionadas, e os custos de cada aresta. Depois de reorganizar os nós em ordem crescente, cria um vetor de m casas para armazenar as raízes de cada nó, chamadas no programa de "nó-mãe", ou "node.mae". Também cria uma matriz "node.filhos" para armazenar os filhos de cada nó de tal maneira que a(i,j) = 1 significa que o nó j é filho do nó i. Neste programa, cada nó ou é mãe e possui vários filhos, ou é filho e não possui nenhum filho.

Prosseguindo com o Algoritmo de Kruskal, o programa verifica, a cada nova aresta da lista de arestas armazenadas na matriz E, se o nó-origem e o nó-destino possuem o mesmo nó-mãe. Se eles possuírem, a adição desta aresta criará um ciclo na árvore, então ela não será adicionada à árvore. Se não possuírem o mesmo nó-mãe, todos os filhos da mãe do nó-destino incluindo ele mesmo e sua mãe se tornarão irmãos do nó-origem, ou seja, filhos da mãe do nó-origem.

O programa percorre todas as arestas da lista até que o número de arestas da árvore solução seja igual a m-1, ou até que se tenha percorrido toda a lista. No primeiro caso, o programa encontra uma Árvore Geradora de Custo Mínimo, e indica sua matriz de adjacências. No segundo caso, não existe Árvore Geradora de Custo Mínimo para o problema, e o programa termina.

2. Entradas

Arquivo .txt da seguinte forma:
Número de nós número de arestas
Vetor b m x m
Número da aresta nó-origem nó-destino limite inferior limite superior custo

Obs.: o programa ignora o vetor b e os limites inferior e superior.

3. Saídas

A cada aresta analisada, imprime o seu índice e a porcentagem de nós atuais na árvore em relação a uma AGCM completa.

Caso tenha achado uma AGCM, o programa imprime o custo dessa AGCM e armazena na matriz M a matriz de adjacências dela.

Caso não tenha achado uma AGCM, o programa imprime essa informação e diz quantas arestas foram incluídas pelo algoritmo em relação a quantas seriam necessárias para se obter uma AGCM (= m-1). A matriz de adjacências M armazenará a árvore encontrada.

Ao final, o programa imprime quantas arestas foram testadas para se construir a árvore obtida pelo algoritmo.

Também é impresso o tempo de execução em segundos.

4. Resultados

Minha implementação do Algoritmo de Kruskal foi testado para 5 grafos pequenos que possuíam exemplos de resolução à mão encontrados na internet ou nas notas de aula. Para todos esses grafos, o programa rodou perfeitamente em menos de 1 segundo e devolveu a AGCM corretamente.

Para os arquivos-teste disponibilizados em http://www.ime.unicamp.br/~ aurelio/problemas/ , o programa devolveu as seguintes saídas:

- Ken-07.txt:

  O grafo não possui Árvore Geradora de Custo Mínimo
  Número de arestas incluídas pelo algoritmo: 2352/2400 = 98% do necessário
  Número de arestas testadas: 3577/3577 = 100% do total
  Tempo de execução: 3.37373 segundos

- Ken-11.txt:

  O grafo não possui Árvore Geradora de Custo Mínimo
  Número de arestas incluídas pelo algoritmo: 14520/14640 = 99.1803% do necessário
  Número de arestas testadas: 21296/21296 = 100% do total
  Tempo de execução: 50.0038 segundos

- Ken-13.txt:

  O grafo não possui Árvore Geradora de Custo Mínimo
  Número de arestas incluídas pelo algoritmo: 28392/28560 = 99.4118% do necessário
  Número de arestas testadas: 42588/42588 = 100% do total
  Tempo de execução: 160.199 segundos

5. Conclusões

Apesar de estar rodando normalmente para grafos mais simples de até 9 nós e 14 arestas, é possível que minha implementação não cubra alguns casos muito específicos de adição de arestas, pois não foram encontradas árvores geradoras para os problemas-teste “ken”. Tentei rodar o programa com vários grafos diferentes da internet para encontrar o erro, inclusive um dos grafos possuía arestas paralelas e laços, porém mesmo assim não consegui encontrar o problema de minha implementação.