DOCUMENTAÇÃO – TRABALHO PRÁTICO 03

Nome: Lucca Silva Medeiros

Matrícula: 2019054773

Disciplina: Algoritmos I

Introdução:

O trabalho prático consiste em desenvolver um programa, capaz de processar um conjunto numérico de entrada, que representa um conjunto de vilas agrícolas que precisam receber infraestrutura para vacinação, e com isso realizar duas tarefas:
- tarefa 1: encontrar o menor número de depósitos de forma que todos os caminhos sejam atendidos, considerando que os caminhos não formam ciclos.
- tarefa 2: encontrar uma solução aproximada de no máximo duas vezes pior que a solução ótima, considerando que agora podem conter ciclos entre os caminhos.

Modelagem computacional do problema:

Para resolver o problema proposto utilizei a modelagem em grafos, mais especificamente grafos não direcionados e não ponderados, onde cada vértice representa uma vila e as arestas os caminhos existentes entre elas. Para resolver o problema não foi necessário ponderar as arestas.

Pensando agora no objetivo do problema 1, que é encontrar o menor número de vértices de forma que todas as arestas sejam alcançáveis, considerando que os grafos não formam ciclos, apliquei paradigma da programação dinâmica resultando na seguinte solução:

Para qualquer solução deste problema temos duas opções:

Incluir o vértice v em questão no conjunto solução resulta em ter que decidir se os vértices que são alcançáveis por v vão ser também incluídos ou serão excluídos do conjunto solução.

ou

Não incluir v no conjunto solução mas obrigatoriamente os vértices filhos de v devem estar incluídos para que todas as arestas ainda sejam alcançáveis.

Agora basta passar o algoritmo acima para cada vértice e verificar qual das duas decisões minimiza o número de vértices do conjunto solução. Porém, para evitar que meu programa

processe repetidamente as mesmas sub-soluções é necessário armazenar também os resultados das operações e resolver o problema de maneira bottom-up.

Para cada vértice v do grafo:
    criar vetor de decisão entre incluir ou não incluir;

DFS no grafo (para acessar de maneira bottom-up todos os vértices) e para cada chamada:
    computar os valores de inclusão ou exclusão do vértice;
    retornar o mínimo entre as decisões de incluir ou excluir o vértice;

imprime a solução;

Já para a tarefa 2, cujo objetivo é encontrar uma solução aproximada de no máximo duas vezes pior que a solução ótima, considerando que agora podem conter ciclos entre os vértices. A abordagem utilizada foi com base no paradigma guloso.

Para toda aresta ainda não visitada:
    Seja A uma aresta qualquer do grafo:
    Inclui-se os vértices x e y de A no conjunto solução;
    Marca-se como visitada todas as arestas alcançáveis por x e por y;

imprime a solução;

Prova de corretude:

Segue a prova de corretude de que o algoritmo desenvolvido para solucionar a tarefa 2 sempre resultará no máximo duas vezes a solução ótima:

Como descrito acima a cada iteração o algoritmo seleciona uma aresta, de um conjunto de tamanho X. Para solucionar o problema são necessários pelo menos X vértices para que toda aresta do grafo seja alcançável.

A aproximação proposta faz com que para cada uma das X arestas do grafo, 2 vértices são adicionado ao conjunto solução, e com isso meu resultado aproximado igual a 2*X.

Como Solução Ótima >= X e o resultado aproximado é 2X, logo o resultado aproximado é pelo menos 2 vezes menor que o resultado ótimo.

Estruturas de Dados:

Foi utilizada a seguinte estrutura de dado para solução do problema:

Graph: representa a malha de vilas, modelada em grafo.

Atributos: int V, int E, list <int> adj*, vector<Edge> edgesList;

Obs: O grafo foi representado por filas onde V é o número de vértices do grafo, E é o número de arestas e cada vértice possui um fila adj para representar os vértices adjacentes. Além disso, o grafo possui também um vetor de arestas.*

Métodos:

Construtor: cria um grafo com o número de vértices e o número de arestas passados por parâmetro;

AddEdge: adiciona no grafo a aresta passada por parâmetro, colocando na lista de adjacência dos vértices correspondentes e no vetor de arestas;

PrintEdge: função usada para testar o funcionamento do vetor de arestas;

DFS: procura em dfs no grafo para realizar a decisão de incluir ou não incluir um vértice conforme paradigma de programação da tarefa 1;

minimalVertexCover: função que inicializa o vetor de decisão para cada vértice do grafo e realiza a tarefa 1 através da chamada de DFS;

aproxMinimalVertexCover: função que percorre todas as arestas do vetor grafo e realiza a tarefa 2 conforme algoritmo guloso descrito acima;

Complexidade Assintótica de Tempo e Avaliação Experimental:

O programa começa com a leitura do arquivo de entrada. Haja vista que **m** é o número de arestas e **n** o número de vértices a complexidade assintótica de tempo dessa parte do programa é **O (m)**. A construção do grafo com os dados de entrada, complexidade **O (1).**

Após isso o programa executa a tarefa desejada:

Caso for a tarefa 1, a complexidade é de **O (m + n),** haja vista que a maior complexidade dentre os passos do algoritmo é a busca em DFS.

Já a tarefa 2, a complexidade no pior caso é **O (m^2),** haja vista que o programa itera cada aresta existente no grafo e após cada decisão de solução ele precisa conferir todas as arestas restantes para excluir as que já foram visitadas.

Para a avaliação temporal inicialmente utilizarei os 4 casos de testes fornecidos via fórum pelos monitores da disciplina. Em todos os casos os grafos são acíclicos. Segue os resultados individuais de cada teste:

CASO TESTE - CT00		Estatística	T1	T2
10 vértices		Tempo Médio	1,2	4,8
9 arestas		Desvio Padrão	0,45	4,09

Tarefa 1	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5
Tempo em ms	1	1	1	2	1

Tarefa 2	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5
Tempo em ms	4	2	3	3	12

CASO TESTE - CT01		Estatística	T1	T2
25 vértices		Tempo Médio	1,2	13,6
24 arestas		Desvio Padrão	0,45	11,93

Tarefa 1	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5
Tempo em ms	1	2	1	1	1

Tarefa 2	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5
Tempo em ms	4	5	6	24	29

CASO TESTE - CT02		Estatística	T1	T2
50 vértices		Tempo Médio	1,8	39,2
49 arestas		Desvio Padrão	0,45	19,19

Tarefa 1	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5
Tempo em ms	2	2	1	2	2

Tarefa 2	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5
Tempo em ms	9	31	52	51	53

CASO TESTE - CT03		Estatística	T1	T2
500 vértices		Tempo Médio	3	542,4
499 arestas		Desvio Padrão	0,71	35,29

Tarefa 1	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5
Tempo em ms	3	4	3	2	3

Tarefa 2	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5
Tempo em ms	485	579	560	548	540

Em resumo temos:

	Desempenho Médio (ms)	
Vértices	Tarefa 1	Tarefa 2
10	1,2	4,8
25	1,2	13,6
50	1,8	39,2
500	3	542,4

Gráficos que resumem o desempenho: