MC202 — ESTRUTURAS DE DADOS

Lab 12 — Grafos II

Mais uma vez, o Abutre está à solta! Ele colocou uma bomba na cidade e também sequestrou a MJ. Ela está presa em uma sala fora do alcance da explosão da bomba, mas um gás letal será liberado. A detonação da bomba e a liberação do gás ocorrerão exatamente à meia-noite. Situação impossível, na qual nosso herói, o cabeça-de-teia, provavelmente terá que escolher entre salvar os cidadãos da cidade e salvar a MJ.

O herói, também conhecido como Amigão da Vizinhança, tem menos de uma hora antes da detonação da bomba e da liberação do gás, então solicita a Karen, a inteligência artificial de seu traje, para que determine a trajetória mais rápida para que ele possa salvar a todos. Karen utilizará uma sub-rotina (i.e., um programa) projetada por você.

Problema

A sua tarefa consiste em desenvolver o programa utilizado por Karen. O programa deve levar em conta as observações abaixo.

1. O método de locomoção do cabeça-de-teia consiste basicamente em saltar do topo de um prédio para o outro e disparar suas teias para se balançar, como um pêndulo. Entretanto, a cada salto, é necessário certificar-se de que haja um ponto alto em que as teias possam grudar e de que a distância entre os prédios não seja maior que **duas vezes** o comprimento máximo de um "fio de teia".
2. A velocidade média da "pendulada" do Amigão da Vizinhança é de 20m/s e ele precisa de 10 minutos (600 segundos) para desarmar a bomba e de 2 minutos (120 segundos) para desativar o mecanismo de liberação do gás.

Os prédios da cidade podem ser modelados como vértices de um grafo ponderado, cujas arestas possuem como peso a distância, dada em metros, entre os prédios correspondentes.

Entrada

A primeira linha contém dois inteiros positivos n e m que indicam, respectivamente, o número de vértices e o número de arestas. Os vértices são numerados de 0 a n-1. Cada uma das m linhas seguintes contém três inteiros positivos: i e j ($1 \leq i, j \leq n$) que correspondem a dois prédios próximos; e d que corresponde à distância entre os prédios i e j. Na linha seguinte, são dados quatro inteiros positivos: k que indica o comprimento máximo de um fio de teia; e a, b e c que indicam, respectivamente, os prédios onde se encontram o Amigão da Vizinhança, a bomba e a MJ.

Saída

O seu programa deve produzir uma única linha de saída indicando os vértices que devem ser percorridos pelo nosso herói. Se for possível salvar a todos em menos de uma hora, então a saída deve conter o caminho que **minimiza o tempo total**. Caso contrário, a saída deve conter o caminho de **menor tempo total** entre desarmar a bomba e desativar o mecanismo de liberação do gás. Considere o tempo necessário para se desarmar a bomba e desativar o mecanismo de liberação do gás na escolha de qual local ir. Existe **um único caminho** mais curto para cada caso de teste.

Exemplo

Entrada

```
10 22
0 2 16
0 3 22
0 4 28
1 3 25
1 4 17
1 5 25
2 3 16
2 4 24
2 6 28
2 7 24
3 4 8
3 5 22
3 7 28
3 8 25
4 5 17
4 8 24
5 8 11
5 9 22
6 7 16
7 8 24
7 9 17
8 9 11
10 7 9 1
```

Saída

```
7 9 8 5 4 1
```

Obs.: não há um espaço em branco no final da saída, mas há um caractere '\n' (nova linha).

Observações

- Neste laboratório, é obrigatório representar o grafo com listas de adjacências e resolver o problema a partir de uma adaptação do algoritmo de Dijkstra.
- Para as turmas A, B e C, esse laboratório tem peso 4.
- Deverão ser submetidos os seguintes arquivos: **lab12.c**, **fila_prio.h** e **fila_prio.c**. Os arquivos **fila_prio.h** e **fila_prio.c** serão fornecidos e podem ser alterados caso julgar necessário. Outros dois arquivos **opcionais e de sua escolha** podem ser submetidos, desde que contribuam para a solução e que mantenham ou melhorem a qualidade do código.
- Observações sobre SuSy:
    - Versão do GCC: 4.4.7 20120313 (Red Hat 4.4.7-17).
    - Flags de compilação:
        - `-ansi -Wall -pedantic-errors -Werror -g -lm`
    - Utilize comentários do tipo `/* comentário */`;
      comentários do tipo `//` serão tratados como erros pelo SuSy
    - Tempo máximo de execução: 1 segundo.
    - Evite usar o sistema como compilador ou apenas para testar as instâncias.
    - Sempre use o software **valgrind** para detectar erros decorrentes do uso incorreto da memória dinâmica **em todos os testes abertos**. Relembrando:
        - `valgrind --leak-check=full ./lab12 < arq01.in`
- Além das observações acima, esse laboratório será avaliado pelos critérios:
    - Indentação de código e outras boas práticas, tais como:
        - uso de comentários (apenas quando são relevantes);
        - código simples e fácil de entender;
        - sem duplicidade (partes que fazem a mesma coisa).
    - Organização do código:
        - arquivo de cabeçalho (.h) contendo definições de constantes, tipos de dados criados pelo usuário, protótipo de funções etc;
        - arquivos de implementação (.c) que implementam as funções definidas no arquivo de cabeçalho;
        - tipos de dados criados pelo usuário e funções bem definidas e tão independentes quanto possível.
    - Corretude do programa:
        - programa correto e implementado conforme solicitado no enunciado;
        - desalocar toda memória alocada dinamicamente durante a execução do programa;
        - se não realiza leitura ou escrita em blocos de memória não alocados;
        - inicialização de variáveis sempre que for necessário;
        - dentre outros critérios.