SCC0216 - Modelagem computacional em grafos
Professor: Alneu de Andrade Lopes
Estagiário PAE: Alan Valejo

Caminhos Mínimos - Dijkstra

1. Especificação

O laboratório consiste na implementação de um algoritmo para encontrar menores caminhos em dígrafos ponderados. Deverá ser utilizada a linguagem de programação C99. Os caminhos mínimos entre dois vértices deverão ser encontrados utilizando o algoritmo de Dijkstra com fila de prioridades. O programa terá um tempo limite de 3 segundos para execução e impressão dos resultados.

2. Descrição da entrada

O programa principal deverá ler da entrada padrão a lista de arestas do dígrafo e representá lo em memória – matriz ou lista de adjacência, conforme desejar. Em seguida, o programa deverá ler uma sequência de vértices de origem e de destino. Para cada par de vértices, o programa deverá imprimir o caminho mínimo entre eles na saída padrão.

Na primeira linha da entrada, haverá o descritor do dígrafo contendo 2 números separados por espaço. Os números indicam, nesta ordem, o número de vértices e de arestas do dígrafo. Nas linhas seguintes, as arestas do dígrafo serão representadas por três números indicando o vértice de origem, o vértice de destino e o peso da aresta (valores de 1 à 9999 inteiros). Por fim, nas demais linhas, pares de números indicarão os índices dos vértices de origem e de destino do caminho a ser procurado.

Os caminhos deverão ser impressos com os índices dos vértices seguidos de espaço. Ao final, insira uma quebra de linha. Quando não houver caminho, imprima apenas a quebra de linha.

3. Submissão

O exercício deverá ser entregue pelo sistema run.codes. Todos os alunos deverão submeter seus códigos no 'Caminhos Mínimos - Dijkstra' até o final da aula. Somente a última submissão será considerada. Todas as demais submissões serão desconsideradas,

incluindo aquelas dentro do período normal de submissão. Os exercícios deverão submetidos em um arquivo .zip contendo código-fonte do programa e um Makefile para compilação e teste do trabalho (verificar com o estagiário PAE, caso não saiba escrever um Makefile). Se necessário, incluam no .zip um arquivo chamado readme com informações que julgarem necessárias.

Os códigos deverão ser compilados pelo compilador gcc com a flag -std=c99. A não conformidade das implementações com a versão C estabelecida acarretará em nota zero. Atenção! Todos os códigos enviados passarão pelo sistema de verificação de plágio. Se forem identificados códigos duplicados, todos os alunos envolvidos receberão nota zero.

4. Correção e Avaliação

As implementações serão avaliadas por meio de casos de testes, com peso 7, e pela legibilidade e boas práticas de programação, com peso 3.

Os seguintes casos implicarão em nota zero:

- Não conformidade com a versão C99;
- Programas não estruturados como um TAD;
- Exercícios plagiados.

Atenção! Todos os códigos enviados passarão pelo sistema de verificação de plágio. Se forem identificados códigos duplicados, todos os alunos envolvidos receberão nota zero.

5. Dicas

- Implementem a fila de prioridades como um Heap binário.
- A cada atualização do vetor de distâncias, insiram novamente o vértice na fila de priori dades ao invés de atualizar sua posição na fila. Deste modo, um mesmo vértice poderá aparecer mais de uma vez na fila. Para resolver isto, utilizem um vetor auxiliar para indicar quais vértices já foram removidos da fila de prioridade para não processá-lo duas vezes.

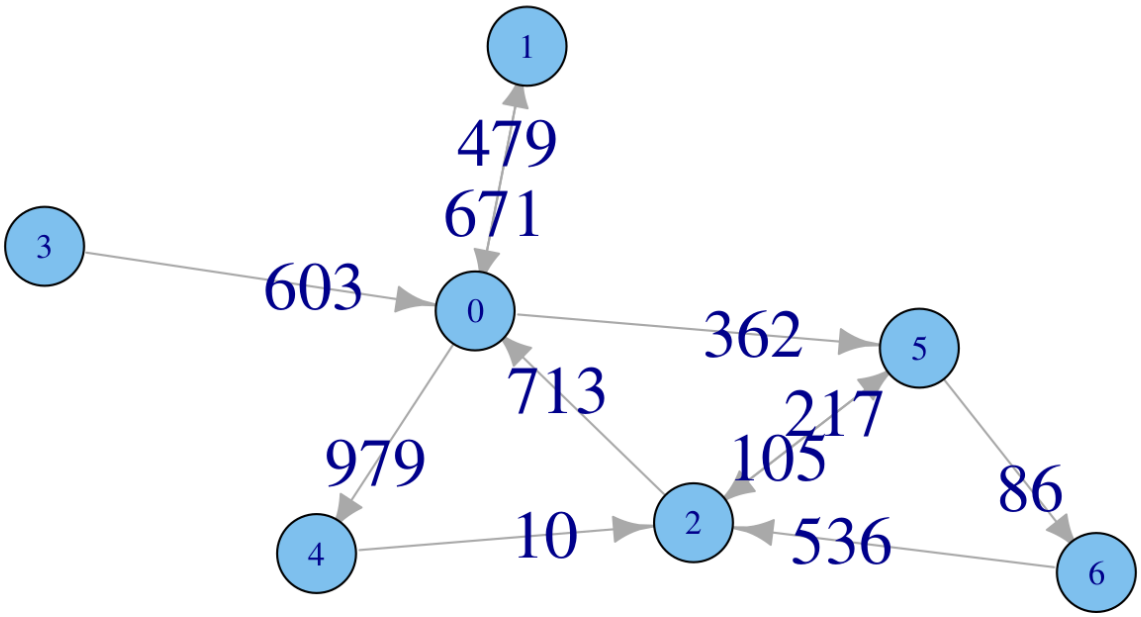
Apêndice: Exemplos

Os comentários são apenas descritivos. Estes não existirão nas

entradas e nem deverão ser impressos como saída.

Dado o digrafo:

```
7 11
5 2 105
5 6 86
2 0 713
3 0 603
0 5 362
0 1 479
1 0 671
4 2 10
0 4 976
6 2 536
2 5 217
2 3
1 4
```



Os vetores de predecessores com origem nos vértices 2 e 1, respectivamente, são

```
2 0 -1 -1 0 2 5
1 -1 5 -1 0 0 5
```

Deste modo, o programa terá como saída

```
                // Não existe caminho entre 2 e 3
1 0 4
```