Trabalho prático: Complexidade empírica de algoritmo

Sejam $G = (V, E)$ um grafo orientado e l_e a distância associada ao arco e, para todo $e \in E$. Sejam $m = |E|$ o número de arestas e $n = |V|$ o número de vértices de G. O trabalho consiste no estudo do comportamento empírico de dois algoritmos para resolver o Problema do Caminho Mínimo. Como entrada, os algoritmos recebem o grafo G e um nó s fonte. No nosso trabalho, o vértice fonte s é o primeiro vértice do grafo. Os algoritmos calculam a distância de s a todos os demais vértices do grafo. O enunciado do trabalho será apresentado em três etapas.

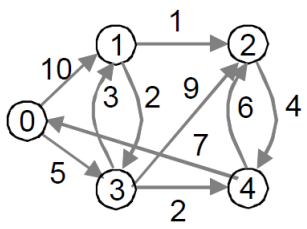
Primeira Etapa

1. A implementação deve ser feita nas linguagens `C`, `C++` ou `Java`.
2. Para representar o grafo G, deve-se implementar a estrutura de dados lista de adjacências (slide número 73 da aula de Grafos, página 87-89 do livro texto [5]).
3. Para a análise da complexidade empírica, utilize quatro classes de instâncias, disponíveis no Google Drive, conforme link disponibilizado no Classroom.
   (a) test_set1 - Classe de grafos completos e simétricos: dez instâncias, uma instância para cada valor de $n \in \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}$. Estas instâncias são simétricas, logo o custo das arestas $c(i, j) = c(j, i)$. Uma instância pequena com 5 nós, `check_v5_s1.dat`, é fornecida para teste.
   (b) Classe ALUE: conjunto de 14 instâncias simétricas de grafos esparsos.
   (c) Classe ALUT: conjunto de nove instâncias simétricas de grafos esparsos.
   (d) Classe DMXA: conjunto de 15 instâncias simétricas de grafos esparsos.

   A seguir, o formato das instâncias da classe test_set1:

```
NB_NODES   5
NB_ARCS    10
LIST_OF_ARCS   COSTS
0    1   10
0    3   5
1   2   1
1   3   2
2   4   4
3   1   3
3   2   9
3   4   2
4   2   6
4   0   7
END
```

   Esta instância corresponde ao grafo abaixo:



   O formato das instâncias das classes ALUE, ALUT e DMXA está disponível em `http://steinlib.zib.de/format.php`. A única seção que nos interessa é a "SECTION Graph" em que, por exemplo, `E 7 2 1` indica que existe aresta $(i, j) = (j, i) = (7, 2)$ com custo $c(i, j) = c(j, i) = 1$.

4. Note que nos arquivos das classes DMXA, ALUE e ALUT, os vértices são numerados de 1 até n e nas classes set os vértices são numerados de 0 até $n-1$. Sugestão, para as instâncias das classes DMXA, ALUE e ALUT, quando lendo o identificador do vértice do arquivo, diminua o identificador de uma unidade.

5. Data de entrega: 5 de junho. O grupo deverá levar um laptop e mostrar a implementação dos grafos usando lista de adjacências. Entregar um relatório impresso com uma tabela para cada classe de grafo. A tabela deverá ter uma coluna com o nome da instância, os valores de n, de m e outra coluna com a soma dos pesos de todas as arestas do grafo da instância. Para ilustrar, veja a tabela abaixo com os valores das somas das arestas de algumas instâncias de diferentes classes.

Instância	n	m	soma
check_v5_s1.dat	5	20	710
inst_v1000_s1.dat	1000	999000	50393710
alue7080.stp	34479	110988	976732
dmxa1010.stp	3983	14216	126232
alut2625.stp	36711	136234	1217122

Tabela 1: Valores da soma das arestas de alguns grafos de diferentes classes.

Segunda Etapa

1. Continuação da Primeira Etapa.

2. Implementar o Algoritmo Bellman-Ford, conforme apresentado na Seção 6.8 do livro texto do curso [5], com complexidade conforme Observação 6.5 e usando a melhoria de memória descrita nas páginas 295-96.

3. Data de entrega: 12 de junho. O grupo deverá levar um laptop e mostrar a implementação do algoritmo. Entregar um relatório impresso com uma tabela para cada classe de grafo. A tabela deverá ter uma coluna com o nome da instância, os valores de n, de m, o valor da menor distância entre o primeiro nó e o último nó $d(n)$ e uma última coluna com o tempo de execução do algoritmo (não considere o tempo de leitura dos dados de entrada e impressão de dados de saída). Para ilustrar, veja a tabela abaixo com valores de algumas instâncias:

Instância	n	m	$d(n)$	tempo
check_v5_s1.dat	5	20	32	
inst_v1000_s1.dat	1000	999000	3	
alue7080.stp	34479	110988	3254	
dmxa1010.stp	3983	14216	1133	
alut2625.stp	36711	136234	3191	

Tabela 2: Valores da distância do primeiro nó do grafo ao último nó $d(n)$ para alguns grafos de diferentes classes, produzido pelo Algoritmo Bellman-Ford.

Terceira Etapa

1. Continuação da Segunda Etapa.

2. Implementar o Algoritmo de Dijkstra para o Problema do Caminho Mínimo [5], páginas 137-142.

3. Seja $d(u)$ a distância do caminho mínimo de s até u, pede-se a implementação de duas versões do algoritmo:

   (a) Versão (v1) em que os valores $d(i), i = 1, \ldots, n$ são armazenados em um vetor.

   (b) Versão (v2) em que os valores $d(i), i = 1, \ldots, n$ são usados como chave em uma fila de prioridades que armazena os nos $V - S$, implementada como um *heap* binário (veja livro texto páginas 141-142: "(...) keeping the nodes $V - S$ in a priority queue with $d'(v)$ as their keys"). OBS: a estrutura de dados *heap* binário deve ser implementada (livro texto Seção 2.5 e outras referências do curso). Não é permitido usar estrutura *heap* pronta da linguagem.

4. Meça o tempo de CPU de execução do algoritmo para todas as instâncias, sem considerar a leitura dos dados de entrada e impressão dos dados de saída.

5. Para cada classe de grafo, produzir uma tabela com as seguintes informações: nome da instância, os valores de n, de m, o valor da menor distância entre o primeiro nó e o último nó $d(n)$, tempo de execução do Algoritmo de Bellman-Ford (coluna bf), complexidade teórica de bf, tempo de execução do Algoritmo de Dijkstra versão v1 (coluna djv1), complexidade teórica de djv1, tempo de execução do Algoritmo de Dijkstra versão v2 (coluna djv2), complexidade teórica de djv2. Para ilustrar, veja a tabela abaixo com valores de algumas instâncias:

Instância	n	m	$d(n)$	bf	$O(?)$	djv1	$O(?)$	djv2	$O(?)$
check_v5_s1.dat	5	20	32						
inst_v1000_s1.dat	1000	999000	3						
alue7080.stp	34479	110988	3254						
dmxa1010.stp	3983	14216	1133						
alut2625.stp	36711	136234	3191						

Tabela 3: Sugestão de tabela, para cada classe de instância. Essa tabela substitui a tabela da segunda etapa.

6. Para cada algoritmo, e para cada classe de instâncias, apresente graficamente a comparação do tempo de execução observado (implementação) versus o tempo esperado (complexidade teórica).

7. Elaborar relatório com análise dos resultados: observações, constatações, tabelas, gráficos, etc. No relatório também deve constar: (i) apresentação do problema; (ii) apresentação do algoritmo; (iii) análise do algoritmo; (iv) discussão sobre a implementação e complexidade; (v) informações sobre o ambiente computacional utilizado (processador, linguagem compilador, etc.); (vi) qual rotina foi usada para medir tempo; (vii) qualquer outra informação relevante. O relatório é tão importante quanto a implementação.

8. Outras referências [1, 3, 4]. Boas referências sobre estruturas de dados [7, 2, 6]. Exemplo de implementação de *heap* binário em `C`: `http://users.cs.fiu.edu/~weiss/dsaa_c2e/files.html`. Outra boa referência para discussão sobre a complexidade do algoritmo é [1], Seções 4.5 Dijkstra Algorithm e 4.7 Heap Implementations.

9. Data de entrega: veja no Classroom. Entregar o código fonte e o relatório em um arquivo zipado. O grupo deverá levar um laptop e mostrar a implementação do algoritmo. ENTREGAR TAMBÉM VERSÃO IMPRESSA do trabalho, no dia da apresentação.

Referências

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[2] W. Celes, R. Cerqueira, and J.L. Rangel. *Introdução a estruturas de dados: com técnicas de programação em C*. Elsevier, 2 edition, 2016.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, first edition, 1990.

[4] Sanjoy Dasgupta, Christos Papadimitriou, and Umesh Vazirani. *Algoritmos*. McGraw-Hill Interamericana do Brasil, 2009.

[5] Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[6] J.L. Szwarcfiter and L. Markenzon. *Estruturas de dados e seus algoritmos*. Livros Tecnicos e Cientificos, 1994.

[7] Mark Allen Weiss. Web page. http://users.cis.fiu.edu/ weiss/, 2017.