Análise comparativa entre diferentes algoritmo de enumeração de ciclos em um grafo

Homenique Vieira Martins[1], **Lucas Santiago de Oliveira**[2],

[1]Instituto de Ciências e Informática
Pontifícia Universidade Católica de Minas Gerais (PUC-MG)

Abstract. *This work aims to make a comparative analysis of time between two algorithms, in the enumeration of cycles existing in an undirected graph. For this work, a walking and a permutation algorithm were implemented, and the results of the same were analyzed in the search for cycles in some graphs of different sizes.*

Resumo. *Este trabalho tem como objetivo fazer uma análise comparativa de tempo entre dois algoritmos, na enumeração de ciclos existente em um grafo não direcionado. Para esse trabalho foram implementados um algoritmo de caminhamento e outro de permutação, e analisados os resultados dos mesmo na busca em de ciclos em alguns grafos de tamanho diversos.*

1. Explicando a classe grafos

A classe grafo que escrevemos é um simples modelo que liga dois pares em C++, o primeiro valor é um número inteiro o segundo valor é um par de inteiros. Sendo o primeiro valor o peso das arestas, o próximo é um par de inteiros que representam a ligação entre os vértices.

```
1         std::vector<std::pair<int, std::pair<int, int>>> arestas;
2
```

2. Explicando o algoritmo de Kruskal

A ideia da implementação do algoritmo de Kruskal é primeiramente ordenar todos os pesos de arestas e ir ligando eles a partir de Disjoint Sets. Disjoint sets (ou em português União-Busca), é uma estrutura matemática em que grupos não possuem elementos que se intersectam, ou seja, não possuem elementos iguais entre eles. Dessa forma, é possível se deslocar dentro dessa estrutura em C++ de forma simples e descobrir se algum elemento está em ambas e se estiver, não conectar no grafo final. Resultado em uma MST, a partir de um algoritmo simples.

3. Explicando a função de Benchmark

Para a execução do benchmark deve levar em consideração que existem outros processos concorrente que disputar a disponibilidade do processador e por isso a execução de somente um teste pode não trazer o custo real de uma operação, pois por um azar a trend que que ficou responsável por rodar o programa já estava ocupada com outro processo e desta forma impactaria negativamente no resultado,para mitigar esse problema definido que seria executado 10^5 de um mesmo algoritmo com uma configuração de grafo, todo esse tempo e somando a uma variável e após isso é feito uma média do tempo de execução.

O que traria um resultado mais preciso do tempo médio de execução,também para um comparativo de dados peguei o maior e menor tempo de execução de um algoritmo em cima de um grafo.

4. Apresentação dos valores

4.1. Grafo 1

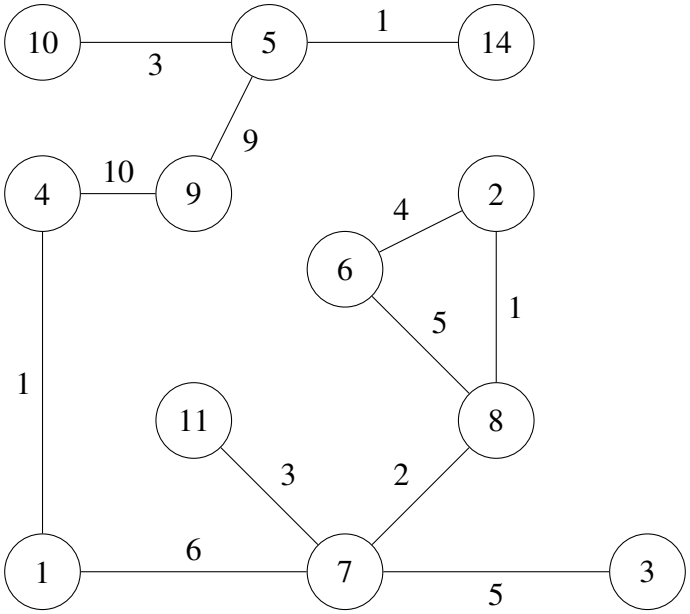

Figure 1. Grafo 1 - 12 vértices e 11 arestas

Table 1. Testes nos algoritmos implementados usando o grafo 1

Medida	**Tempo médio(em segundos)**	
	Kruskal	***DFS***
Media de tempo	0.002	0.001
Tempo minimo	0000	0000
Tempo Máximo	0.0599	0.0570
Tempo total (10^5)	2896.44	1935.28

4.2. Grafo 2

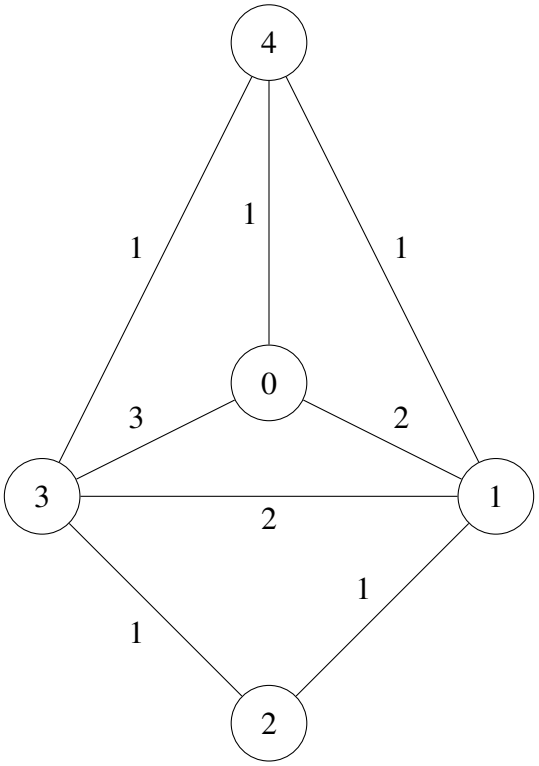

Figure 2. Grafo 2 - 5 vértices e 8 arestas

Table 2. Testes nos algoritmos implementados usando o grafo 2

Medida	**Tempo médio(em segundos)**	
	Kruskal	***DFS***
Media de tempo	0.002	0.007
Tempo minimo	0000	0.005
Tempo Máximo	0.136	0.0940
Tempo total (10^5)	2522.054	7646.743

4.3. Grafo 3

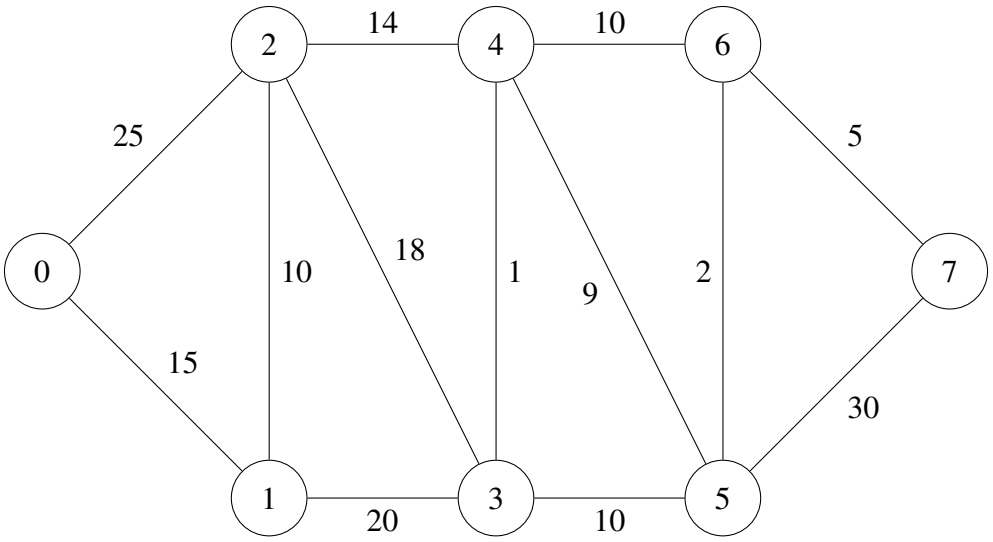

Figure 3. Grafo 3 - 8 vértices e 13 arestas

Table 3. Testes nos algoritmos implementados usando o grafo 3

Medida	**Tempo médio(em segundos)**	
	Kruskal	***DFS***
Media de tempo	0.003	0.012
Tempo minimo	0000	0.010
Tempo Máximo	0.1380	0.0459
Tempo total (10^5)	3717.456	12823.205