Universidade Estadual de Maringá
Departamento de Informática
Curso: Algoritmos em Grafos (6898) - Ciência da Computação
Professor: Marco A. L. Barbosa

RELATÓRIO 1 - ALGORITMO EM GRAFOS
IMPLEMENTAÇÃO E TESTES DE ALGORITMOS DE ÁRVORES ALEATÓRIAS

Thiago Issao Yasunaka RA: 103069
Otavio Hideki Gonçalves Kochi RA: 107635

Maringá
2019

Descrição da Experiência da Implementação

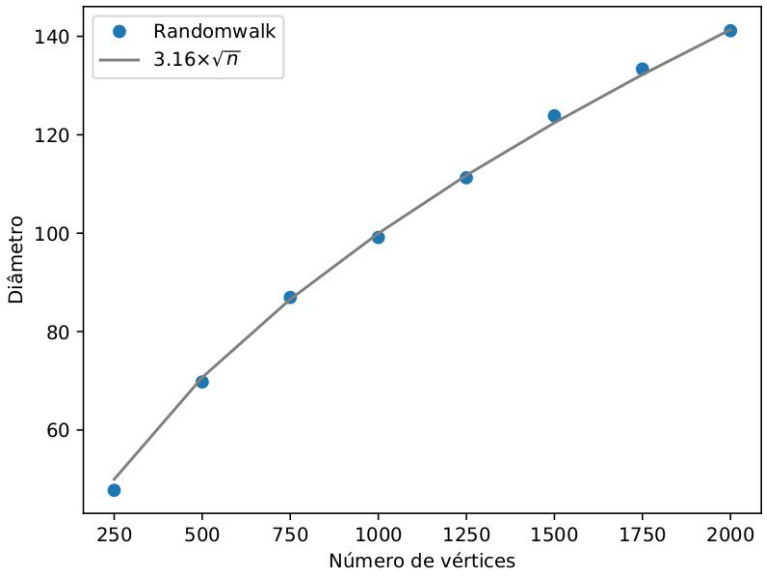
A primeira dúvida foi escolher uma linguagem eficiente e que não precisasse trabalhar com ponteiros, por exemplo, a linguagem C. De acordo com exemplos dados em sala de aula, a linguagem Python aparentava ser a ideal, porém durante a primeira parte do trabalho o desenvolvimento com ele foi um tanto quanto complicado, por algum motivo a linguagem não obedecia algumas atribuições feitas nos algoritmos (Claro, o erro era nosso por não ter um domínio sobre a linguagem). Após horas de tentativas de corrigir o algoritmo, não obtivemos sucesso, então tomamos a decisão de trocar a linguagem para Javascript, que é uma linguagem que já havíamos trabalhado em projetos fora da graduação. Agora com javascript, utilizando a mesma lógica o algoritmo funcionou, pelo menos até a parte que estava gerando o erro.

A linguagem javascript é muito flexível, ou seja, é possível atribuir tipos diferentes a mesma variável em tempo de execução, isso foi um fator positivo (Nem sempre é), pois em uma lista de adjacências que antes recebia apenas um vetor com vértices, que no nosso trabalho foi representado por inteiros, agora cada posição do vetor recebia um vetor de objetos, sendo que cada objeto possuía diversos atributos como d, pi, cor, adj. O algoritmo *breadth-first search* recebe um Grafo e um vértice inicial, representamos o Grafo como lista de adjacências, porém levou um tempo para nos darmos conta de que o segundo parâmetro do algoritmo bastava apenas passar um número que representasse o vértice, e não o vértice propriamente dito. Foi na primeira parte do trabalho na qual uma dúvida de extrema importância foi tirada, durante a execução do algoritmo bfs algumas atribuições a vetores estavam ocorrendo de maneira irregular, um problema semelhante na linguagem Python, foi aí então que descobrimos que Objetos em Javascript são passados por referência, ou seja, fazer `GAux = G(vetor)` diretamente não faz uma cópia do objeto, ele simplesmente recebe uma cópia da referência. A solução para este problema foi utilizar a classe Object do javascript, por esse motivo grande parte do código utiliza esta linha let `GAux = Object.assign({}, G).`

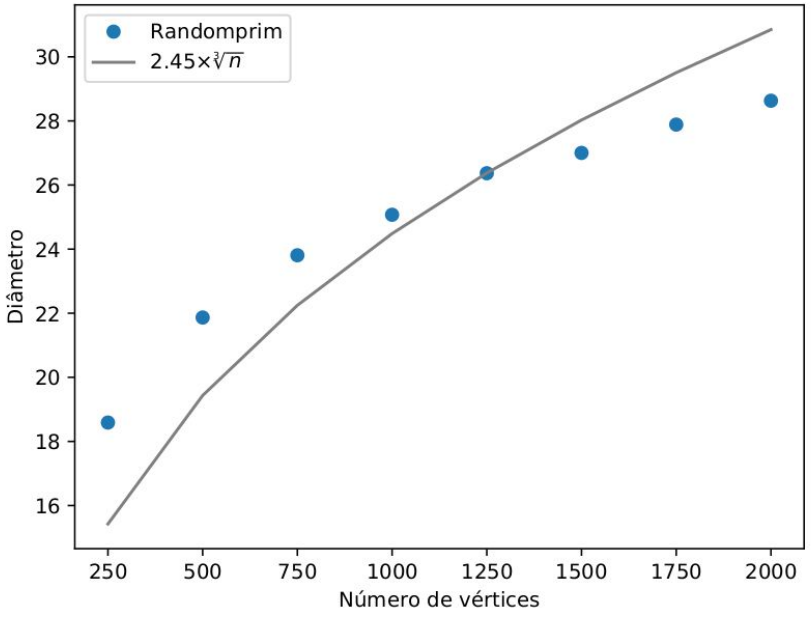
Durante a implementação do algoritmo de prim, ocorreu um erro na média do diâmetro das árvores, o teste para 250 vértices teve média 18, sendo que o esperado é uma média por volta dos 30. Comentando com outros alunos da turma, encontramos um trabalho que possuía a mesma média, porém este feito na linguagem C, no caso deste trabalho o erro ocorria na atribuição das chaves nos vértices da árvore, isso ocorria pois a lista implementada não era de posições. A correção deste erro foi criar uma lista de posições. No nosso caso, a lista já era de posições e devido a isso dificultou a depuração do código.

Análise dos resultados obtidos

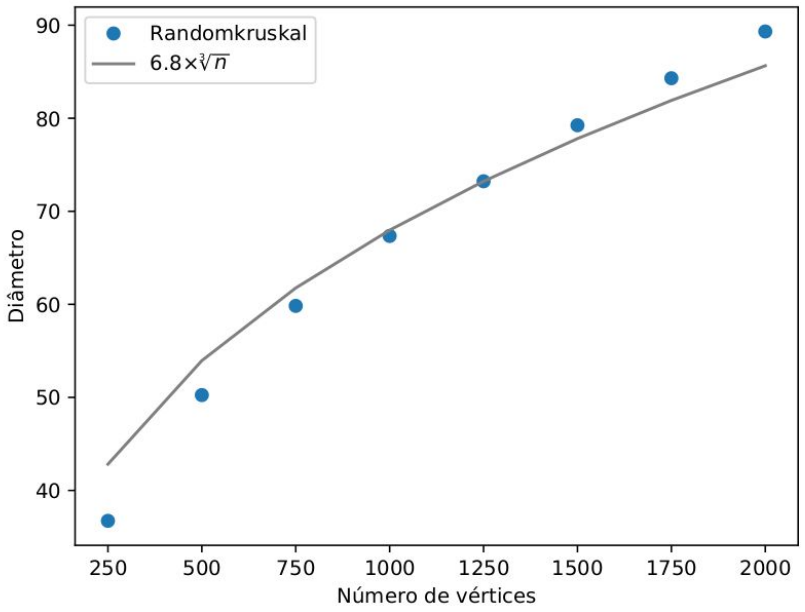
- Random Tree Walk - tempo de execução 0m22,707s



- Random Tree Prim - tempo de execução 13m10,437s



- Random Tree Kruskal - tempo de execução 3h21m24,002s



Explicação dos Resultados

Como esperado, o algoritmo do Random Tree Walk foi o mais rápido devido a sua complexidade ser menor totalizando 0m22,707s. O algoritmo do Random Tree Prim foi o segundo mais rápido, devido sua complexidade ser menor que a do Kruskal mas maior que a do Random Tree Walk, totalizando 13m10,437s. O algoritmo do Random Tree Kruskal foi o mais lento.