Trabalho Prático N.01

Gabriel Alejandro Figueiro Galindo, Athos Geraldo Salomon Dolabela da Silveira, Gustavo Menezes Barbosa

[1]PUC-Minas
Praça da Liberdade

Abstract. *This report describes details of the implementations, experiments and results obtained from the codes developed for the practical assignment number 1 of Graph Theory and Computability. In accordance to the teacher's requirements, two programs were developed, a program that creates a random directed graph and prints the base, anti-base and transitive closure of this graph through the Naive method, and another program that performs the same process through the Warshall method. Through a series of tests, it was concluded that the Floyd-Warshall algorithm is more efficient than the Naive*

Resumo. *Esse relatório descreve detalhes das implementações, dos experimentos e resultados obtidos dos códigos desenvolvidos para a realização do trabalho prático numero 1 de Teoria dos Grafos e Computabilidade. De acordo com os requisitos do professor, dois programas foram desenvolvidos, um programa que cria um gráfico direcionado aleatório e que imprime a base, anti-base e fecho transitivo desse gráfico por meio do método Naive, e um outro programa que realiza o mesmo processo por meio do método Warshall. Através de uma serie de testes, foi concluído que o algoritmo de Floyd-Warshall é mais eficiente que o Naive.*

1. Introdução

Este relatório descreve o desenvolvimento de dois programas de computador. O primeiro programa cria um gráfico direcionado de forma aleatória e busca a sua base, anti-base, fecho transitivo direto e inverso por meio do Naive Bayes, enquanto que o segundo programa realiza um processo semelhante mas por meio do algoritmo de Warshall. Esses dois softwares foram desenvolvidos utilizando a metodologia ágil e com a participação ativa de todos os membros do grupo durante todo o processo. O objetivo principal deste projeto foi descobrir qual dos dois programas possui um tempo de execução médio menor por meio da realização de testes. Esse relatório também descreve os desafios encontrados e as soluções implementadas ao longo do processo de desenvolvimento, bem como as ferramentas e tecnologias utilizadas pela equipe de desenvolvimento.

2. Desenvolvimento

Durante o desenvolvimento do software, reuniões foram realizadas periodicamente para entender melhor as necessidades e objetivos do trabalho. Uma vez que os requisitos e as funcionalidades necessárias foram bem definidas e entendidas, dois programas de computador foram desenvolvidos separadamente. O primeiro programa desenvolvido foi chamado de ”Naive”, que cria um gráfico direcionado de forma aleatória utilizando duas variáveis que representam o número exato de vértices e arestas que o gráfico gerado deve

ter. Uma vez que o gráfico é criado, o método Naive Bayes é utilizado para procurar a base, a anti-base, o fecho transitivo direto e inverso do gráfico e então os resultados são imprimidos junto com o tempo de execução. O segundo programa, chamado de ”Warshall”, também cria um gráfico direcionado aleatório, utilizando a mesma função usada no programa ”Naive”. Entretanto, o programa ”Warshall”, realiza a busca da base, anti-base e dos fechos transitivos do gráfico por meio do algoritmo de Floyd-Warshall. Para o melhor entendimento de como o método Warshall e Naive deviam ser implementados, ferramentas de auxilio como a monitoria fornecida pela PUC-Minas e ChatGPT foram utilizados.

2.1. Métodos Implementados

Como mencionado anteriormente, os algoritmos implementados nos programas desenvolvidos são o método Floyd-Warshall e o Naive Bayes. O Warshall armazena uma matriz de distâncias entre cada par de vértices do grafo, sendo que, a cada iteração, o algoritmo atualiza a matriz para incluir a distância mais curta possível entre cada par de vértices, utilizando uma abordagem de ”relaxamento” para encontrar caminhos mais curtos através do grafo.[Leuenhagen et al. 2016] O outro algoritmo é o Naive, que normalmente é utilizado para análise de texto e outros tipos de dados não estruturados, não sendo diretamente aplicável a problemas de busca em grafos, além disso, ele também pode ser caracterizado por estabelecer a probabilidade de um evento ocorrer com base em informações prévias.[Buescher et al. 2016]

2.2. Requisitos

Os requisitos do software foram pré-definidos e descritos no documento de especificação do trabalho prático disponível na disciplina de Teoria dos Grafos e Computabilidade. Neste documento, as seguintes necessidades foram levantadas:

- Gerador de grafos aleatórios direcionados.
- Dois métodos para identificação do fecho transitivo direto e inverso de todos os vértices.
- Implementação do método Naive.
- Implementação do método Warshall.
- Determinar base e anti-base dos grafos direcionados gerados.

2.3. Explicação do código

2.3.1. Gerador de Grafos

```java
//Função que gera um gráfico direcionado aleatório;
public static List<List<Integer>> GeradorDeGrafo(int vertices, int arestas){
    List<List<Integer>> graph = new ArrayList<>();
    Random rand = new Random();
    for(int i = 0; i < vertices; i++){
        graph.add(new ArrayList<>());
    }
    int numarestas = 0;
    while(numarestas < arestas){
        int u = rand.nextInt(vertices);
        int v = rand.nextInt(vertices);
        if(u != v && !graph.get(u).contains(v)){
            graph.get(u).add(v);
            numarestas++;
        }
    }
    return graph;
}
```

Figure 1. Função utilizada para gerar os gráficos direcionados no trabalho

O código acima pega duas variáveis que representam o número de vértices e arestas do gráfico que vai ser criado. Depois, um ArrayList de inteiros é utilizado para representar o gráfico, com cada índice do ArrayList externo representando um vértice e cada índice do ArrayList interno contendo os vizinhos desse vértice. Depois disso, arestas são adicionadas randomicamente até que o número desejado seja atingindo.

2.3.2. Busca da Base em Naive

O código abaixo implementa o algoritmo Naive para achar a base do gráfico. Ele itera sobre todos os vértices do grafo e verifica se existe algum outro vértice que tenha uma aresta apontando para ele, caso o contrário, o vértice é adicionado ao conjunto base.

```java
//Procura a base do grafo com o método Naive;
public static Set<Integer> AcharBase(List<List<Integer>> graph){
    Set<Integer> base = new HashSet<>();
    int vertices = graph.size();
    for(int i = 0; i < vertices; i++){
        boolean flag = true;
        for(int j = 0; j < vertices; j++){
            if(graph.get(j).contains(i)){
                flag = false;
                break;
            }
        }
        if(flag){
            base.add(i);
        }
    }
    return base;
}
```

Figure 2. Função que busca a base usando Naive

2.3.3. Busca do Fecho Transitivo Direto e Inverso

3. Conclusão

Os programas “Naive” e “Warshall” foram entregues dentro do prazo estabelecido e atenderam a todos os requisitos definidos pelo trabalho. Além disso, eles foram testados e avaliados, garantindo sua qualidade e segurança, os testes requisitados e seus resultados podem ser vistos abaixo.

3.1. Testes

Como requisitado pelo professor, experimentos foram realizados para avaliar o tempo médio gasto pelos dois programas com grafos aleatórios contendo 100, 1.000, 10.000 e 100.000 vértices. Como não existe nenhum requisito em relação ao número de arestas necessárias para os grafos dos testes, o número de arestas é equivalente ao número de vértices mais um para todos os casos de testes abaixo.

3.1.1. Método Naive

Table 1. Testes do método Naive Bayes

Dados de Entrada	Ações Executadas	Tempo médio gasto para a aplicação
100	Método Naive aplicado a grafos contendo 100 vértices	2:36 minutos
1000	Método Naive aplicado a grafos contendo 1000 vértices	9:53 minutos
10000	Método Naive aplicado a grafos contendo 10000 vértices	21:14 minutos
100000	Método Naive aplicado a grafos contendo 100000 vértices	33:29 minutos

3.1.2. Método Warshall

Table 2. Testes do método Floyd-Warshall

Dados de Entrada	Ações Executadas	Tempo médio gasto para a aplicação
100	Método Warshall aplicado a grafos contendo 100 vértices	167 milissegundos
1000	Método Warshall aplicado a grafos contendo 1000 vértices	6247 milissegundos
10000	Método Warshall aplicado a grafos contendo 10000 vértices	9576 milissegundos
100000	Método Warshall aplicado a grafos contendo 100000 vértices	246675 milissegundos

3.2. Resultados

Pegando os resultados acima, pode ser notado que o programa ”Naive” possui um tempo médio de execução de 1008 segundos, com o menor tempo sendo em torno de 2 minutos e 36 segundos, e o maior tempo sendo aproximadamente 33 minutos e 29 segundos. Enquanto isso, o programa Warshall possui um tempo médio de execução de aproximadamente 66 segundos, com o menor tempo sendo igual a 167 milissegundos e o maior sendo 9576 milissegundos. Com esses resultados pode se concluir que o algoritmo de Floyd-Warshall é mais eficiente que o Naive Bayes para encontrar o caminho mais curto entre todos os pares de vértices em um grafo. Isso se deve ao fato que o Warshall possui uma complexidade de tempo de $O(n^3)$, onde n é o número de vértices no grafo, enquanto que o

algoritmo Naive tem uma complexidade de tempo de $O(n^4)$. Além disso, o método Warshall usa programação dinâmica para evitar a repetição de cálculos que já foram feitos anteriormente, enquanto que o método Naive recalcula todos os caminhos mais curtos em cada iteração.[Viloria et al. 2019]

4. Referências

4.1. Link do Projeto

Link do projeto no Overleaf: https://pt.overleaf.com/read/fsfcfbrhxsdr

References

Buescher, N., Kretzmer, D., Jindal, A., and Katzenbeisser, S. (2016). Scalable secure computation from ansi-c. In *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6.

Leuenhagen, S., Burghaus, L., Kukolja, J., Rosenkranz, S., Kabbasch, C., Fink, G., and Onur, O. (2016). Das infektiöse pseudoaneurysma – eine seltene komplikation bei patienten mit septischem krankheitsbild: aktuelle datenlage und vorschlag eines diagnostischen und therapeutischen algorithmus:. *Fortschritte der Neurologie · Psychiatrie*, 84:411–418.

Viloria, A., Pineda Lezama, O. B., and Mercado, N. (2019). Model and simulation of structural equations for determining the student satisfaction. *Procedia Computer Science*, 160:527–531. The 10th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2019) / The 9th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2019) / Affiliated Workshops.