Documentação do Projeto: Depósitos da Vacina

Augusto Carvalho Porto Pereira

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG) Belo Horizonte – MG – Brazil

1. Introdução

O objetivo deste trabalho é criar um sistema que receberá dados referentes a pequenas vilas agrícolas e trilhas terrestres para, a partir deles, determinar qual o número mínimo de vilas a receberem depósitos de vacinas para que qualquer vila possua um estoque de vacinas em seu terreno ou a somente uma trilha de distância. A solução do problema foi separada em duas partes diferentes: a primeira parte trata de uma solução em tempo polinomial para os casos nos quais não há caminhos cíclicos entre diferentes vilas e para a qual sempre conseguimos encontrar a melhor solução, enquanto a segunda parte compreende uma solução aproximativa para os problemas onde há ciclos formados pelas trilhas.

Nas próximas seções, serão descritos os algoritmos e a estrutura de dados do problema bem como a complexidade da solução e sua prova de corretude. Além disso, ao final haverá uma avaliação experimental para o cálculo do tempo gasto para diferentes valores de entrada.

2. Algoritmos e Estruturas de Dados

Buscando organizar o código e representar da melhor forma os conceitos reais em objetos, foi criada a classe Vila, que possui campos e métodos referentes aos dados de sua vizinhança direta recebidos na entrada, como o número e a lista de vizinhos para os quais estes vilões podem se dirigir passando somente por uma trilha. Esta classe é utilizada independentemente de qual das variações do problema a execução do programa busca responder.

Para as distribuições de vilas e trilhas que podem ser representadas por grafos acíclicos, a solução receberá o nome de “Tarefa 1”. Para esta tarefa, foi

criada a classe Arvore para representar o conceito teórico para organização do sistema, compondo uma ordenação das vilas com base em suas vizinhanças, de modo que cada nó da árvore corresponda a uma vila que tem como vizinhos todos os seus nó filhos e seu nó pai. A estratégia adotada foi realizar a montagem da árvore como apresentado anteriormente e, para cada nó, foi executada uma DFS *(depth-first search)* identificando se todas as ligações entre nós estavam cobertas e, se não, selecionar uma vila (sempre referente ao nó pai da relação) para a solução final. A execução funciona como no pseudocódigo a seguir:

```
vilas visitadas ← ∅
vilas selecionadas ← ∅
DFS: parâmetro ->  vila, vilas visitadas, vilas selecionadas
    Adiciona vila às vilas visitadas
    PARA CADA vizinho de vila
        SE vizinho não pertence a vilas visitadas
            DFS(vizinho, vilas visitadas, vilas selecionadas)
            SE nem vizinho nem vila pertence a vilas selecionadas
                Adiciona vila a vilas selecionadas (preferência paterna)
FIM ALGORITMO
```

O resultado deste algoritmo é dado pelo número de elementos presentes na lista de vilas selecionadas na execução da DFS. Como, para o escopo do problema, não é necessário informar quais vilas foram selecionadas, o sistema exibe somente o número de vilas onde deverão ser construídos depósitos de vacinas.

Já para a Tarefa 2, não foram criadas estruturas de dados auxiliares como classes ou TADs, somente uma matriz que armazena a lista de trilhas possíveis. A partir desta matriz, o algoritmo implementado funciona como a seguir:

```
TAREFA 2: vértices V, arestas E
    PARA CADA {a,b} em E
        Imprime a
        Imprime b
        PARA CADA v em V
            Remove {a,v} de E (se houver)
            Remove {v,a} de E (se houver)
            Remove {b,v} de E (se houver)
            Remove {a,b} de E (se houver)
FIM ALGORITMO
```

3. Complexidade Assintótica

Para analisar a complexidade de tempo e espaço da solução, será necessário separar as tarefas 1 e 2 em diferentes análises, pois cada uma utiliza diferentes estratégias para atingir a solução final. Iniciando a leitura do sistema linearmente, porém, é possível realizar uma análise dos pontos em comum entre as duas tarefas, sendo estes a leitura dos dados da entrada. Dessa forma, temos que ambas tarefas realizam a leitura e preenchimento das V vilas e das E trilhas em complexidade de tempo e espaço de O(|V|) e O(|E|) respectivamente..

Já para o caso de uso da primeira tarefa, o algoritmo que busca a cobertura mínima dos vértices é uma DFS e, portanto, tem sua complexidade ditada por esta que já é conhecida. Portanto, para a **Tarefa 1,** a complexidade assintótica de tempo e espaço da solução é **O(|V| + |E|)**.

Já para a Tarefa 2, o tempo e o espaço gastos para alocar e zerar os espaços da matriz de caminhos entre vilas é $O(|V|^2)$. Para encontrar cada aresta da lista ainda não coberta e adicionar ambos os vértices, é necessária a execução de 3 laços aninhados para o número de vilas, tendo portanto complexidade de tempo $O(|V|^3)$. Portanto, para a **Tarefa 2**, a **complexidade de tempo é $O(|V|^3)$** e a **complexidade de espaço é $O(|V|^2)$**.

4. Prova de Corretude

Tese: A solução apresentada para a Tarefa 1 é funcional.
Prova: É intuitivo perceber que a solução apresentada é uma solução válida pois todas as vizinhanças são verificadas na DFS. Portanto, se houver alguma trilha descoberta ela não pode pertencer à lista de trilhas E e, portanto, não existe.

Tese: A solução apresentada para a Tarefa 2 é no máximo o dobro da solução ótima
Prova: Suponhamos o caso ótimo para um grafo G contendo n vértices. Sabemos, pela premissa do problema, que cada um destes n vértices possui 1 ou mais caminhos ligados a ele e todos os caminhos estão ligados a pelo menos um dos n vértices. Para a solução apresentada, para cada caminho ainda não coberto são adicionados dois vértices, mesmo que somente um deles já fosse suficiente para o caso ótimo. Portanto, para cada vértice do conjunto ótimo, pode ou não existir um vértice a mais desnecessário para a solução. Dessa forma, S <= n + n.

5. Avaliação Experimental

Para avaliar o tempo gasto pela aplicação, foi utilizado o comando *time* nativo do Linux, tomando como resultado o tempo chamado de “real”, que contabiliza tanto o tempo gasto no processamento de dados quanto o tempo gasto por input e output. Os resultados seguiram como a seguir:

Tarefa 1

- **Caso de teste 0:** 10 vilas e 9 trilhas
  - **Média:** 0.022s
  - **Desvio padrão:** 0.007s
- **Caso de teste 1:** 25 vilas e 24 trilhas
  - **Média:** 0.017s
  - **Desvio padrão:** 0.004s
- **Caso de teste 2:** 50 vilas e 49 trilhas
  - **Média:** 0.018s
  - **Desvio padrão:** 0.002s
- **Caso de teste 3:** 500 vilas e 499 trilhas
  - **Média:** 0.021s
  - **Desvio padrão:** 0.002s

Tarefa 2

- **Caso de teste 0:** 10 vilas e 10 trilhas
  - **Média:** 0.025s
  - **Desvio padrão:** 0.012s
- **Caso de teste 1:** 25 vilas e 25 trilhas
  - **Média:** 0.027s
  - **Desvio padrão:** 0.017s
- **Caso de teste 2:** 50 vilas e 51 trilhas
  - **Média:** 0.017s
  - **Desvio padrão:** 0.003s
- **Caso de teste 3:** 500 vilas e 500 trilhas
  - **Média:** 0.068s
  - **Desvio padrão:** 0.053s