Petrópolis-RJ, 28 de Janeiro de 2025

PAÍSES EM GUERRA

Augusto Freitas Franco Gomes

1

AEDs-II
Engenharia de Computação
Centro de Educação Tecnológica Celso Suckow da Fonseca - CEFET/RJ
Petrópolis - RJ

Resumo. *Este documento apresenta a solução do problema de encontrar o menor caminho entre duas cidades em 2050, onde há uma guerra entre os países, usando o algoritmo de Dijkstra. Ele simula o transporte de cartas entre cidades, considerando pesos que representam o tempo de entrega. As cartas só podem ser entregues se houver um acordo de envio de cartas entre elas. O código foi implementado em C.*
Palavras-Chave. *Menor, Caminho, Dijkstra, Cartas, Pesos, Tempo.*

Abstract. *This document presents the solution to the problem of finding the shortest path between two cities in 2050, where there is a war between the countries, using Dijkstra's algorithm. It simulates the transport of letters between cities, considering weights that represent delivery time. Letters can only be delivered if there is a letter-sending agreement between them. The code was implemented in C.*
Keywords. *Shortest, Path, Dijkstra, Letters, Weights, Time.*

Petrópolis-RJ, 28 de Janeiro de 2025

1. Introdução

O cenário simula uma guerra entre países, e a comunicação entre os espiões dos países só é possível por meio de cartas. O problema propõe encontrar o menor caminho entre as centrais de correio de duas cidades e verificar se é possível entregar as cartas, considerando a existência de um acordo de envio de cartas entre as cidades.

A solução pode ser implementada utilizando algoritmos de caminhos mínimos. No grafo utilizado para resolver o problema, os acordos postais entre as cidades correspondem às arestas, enquanto cada cidade é representada por um vértice. Assim, o objetivo é determinar o menor caminho entre os vértices, respeitando os acordos. Além disso, é necessário verificar a conectividade entre as cidades para determinar se a entrega da carta é ou não possível.

2. Implementação do Algoritmo

A seguir, será apresentada a implementação do algoritmo utilizado para determinar o menor caminho entre duas cidades, considerando o tempo de entrega das cartas como o peso das arestas no grafo.

2.1. Main

```
int main(){
    int n, e, h, k, x, y;
    scanf("%d %d", &n, &e);
while (n != 0 || e != 0){
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= n; j++){
            matriz[i][j] = INF;
        }
    }
```

Listing 1. Início da Função Main e Primeiro laço while

Lê o número de vértices (n) e arestas (e), após isso, guarda os resultados da leitura e inicia um ciclo de repetição enquanto um dos valores é diferente de 0. Após isso, inicializa cada posição da matriz com um valor ”infinito”definido, assim como proposto por Dijkstra em seu algoritmo.

```
    for(int i = 0; i < e; i++){
        scanf("%d %d %d", &x, &y, &h);
        if(matriz[y][x] != INF){
            matriz[x][y] = matriz[y][x] = 0;
        }
        else{
            matriz[x][y] = h;
        }
    }
}
```

Listing 2. Adiciona as arestas, analisa se a aresta é bidirecional e adiciona os pesos

Lê os vértices de origem (x), os vértices de destino (y) e as horas/peso (h), que servem para representar a existência de uma aresta entre os nós e o peso da aresta.

Para cada aresta, se já existe uma aresta no sentido contrário (1 para 2 e 2 para 1), é considerada uma aresta bidirecional, e conforme pedido pelo autor do exercício, o peso da aresta, nesse caso, é 0. Caso contrário, insere o peso da aresta no sentido x para y.

```
    scanf("%d", &k);
    while(k != 0){
        scanf("%d %d", &x, &y);
        dijkstra(n, x, y);
        --k;
    }
```

Listing 3. Laço da função de consultas

Lê o número de consultas (k) e inicia um laço de repetição while, enquanto a variável for diferente de 0. Para cada consulta, chama a função Dijkstra para calcular a menor distância entre o par de vértices fornecido.

2.2. Função Dijkstra

```
void dijkstra(int n, int o, int d){
    int i, j;
    int visitas[n+1];
    int distancia[n+1];
    for(i = 0; i <= n; i++){
        visitas[i] = 0;
        distancia[i] = INF;
    }
    distancia[o] = 0;
```

Listing 4. Início da Função Dijkstra e iniciação dos vetores ”distância”e ”visitas”

Inicia a função ”Dijkstra”, inicializa as variáveis que serão utilizadas e preenche o vetor de distância com ”infinito”, e o de visitas com 0, representando que nenhuma cidade foi visitada ainda. Após isso, no vetor de visitas, na posição referente ao nó de origem, define a distância como 0.

```
for(i = 0; i < n; ++i){
    int v = -1;
    for(j = 1; j <= n; ++j){
        if(visitas[j] == 0 && (v == -1 || distancia[j] < distancia[v])){
            v = j;
        }
    }
    if(distancia[v] == INF) break;
    visitas[v] = 1;
```

Listing 5. Início do laço para encontrar os caminhos mínimos

Implementa a escolha do próximo vértice a ser processado no algoritmo. O laço percorre até que todos os vértices sejam visitados ou até que o vértice mais próximo não seja acessível. A variável v é definida como -1 para indicar que nenhum vértice foi selecionado.

O laço interno verifica todos os vértices não visitados e seleciona o vértice v com a menor distância a partir da origem. Caso todos os vértices restantes tenham distância infinita, o algoritmo interrompe a execução com o comando break.

Após determinar v, ele é marcado como visitado no vetor visitas.

```
for(j = 1; j <= n; ++j){
    if(matriz[v][j] != INF && distancia[v] + matriz[v][j] < distancia[j]){
        distancia[j] = distancia[v] + matriz[v][j];
    }
}
```

Listing 6. Função para o cálculo da distância mínima

Esse for percorre todos os vértices do grafo (j de 1 até n) e verifica se o caminho atual até j passando por v é menor do que o registrado em distancia[j]. Se essa condição se cumprir, então há necessidade de atualizar as distâncias dos vértices adjacentes ao vértice atual v.

```
    if(distancia[d] < INF){
    printf("%d\n", distancia[d]);
} else {
    printf("Nao e possivel entregar a carta\n");
}
```

Listing 7. Print dos resultados

Caso a distância entre a origem ”o”até o destino ”d”exista, ou seja, seja diferente de infinito, a menor distância é exibida. Se não for possível alcançar o destino, é exibida a mensagem ”Nao e possivel entregar a carta”.

3. Conclusão

Este relatório abordou a implementação de um algoritmo de caminhos mínimos usando o algoritmo de Dijkstra para simular o esquema de envios de cartas por meio de centrais de correios seguindo uma série de restrições no sistema de envio em um cenário fictício de uma terceira guerra mundial.

O uso do algoritmo de Dijkstra mostrou-se adequado para calcular o menor tempo de entrega de cartas entre as cidades, dado que considera os pesos das arestas como tempos de entrega. Além disso, a implementação assegura a verificação de conectividade entre as cidades, garantindo que as entregas sejam realizadas apenas quando há um acordo de envio de cartas.

Referências

Cormen, T. H (2012). *Algoritmos - Teoria e Prática.* Elsevier 3ª edição, Rio de Janeiro