Universidade Federal do Rio Grande do Norte
Estruturas de Dados e Algoritmos
Programa de Pós-Graduação em Sistemas e Computação - PPgSC
Departamento de Informática e Matemática Aplicada - DIMAp

Projeto 2 - Programação Dinâmica e Caminhos Mais Curtos em Grafos

Valmiro Ribeiro da Silva

Natal-RN

2017.1

Sumário

1 Algoritmo de Floyd-Warshall

1.1 Programa que implementa o algoritmo de Floyd-Warshall

O Algoritmo de Floyd-Warshall é um algoritmo usado para calcular os caminhos mais curtos em um grafo com pesos positivos e negativos que não possua ciclos negativos. A execução deste algoritmo calcula os valores dos caminhos mais curtos entre todos os pares de vértices.

O programa implementado, utilizando a linguagem C++, recebe um arquivo que contém um número na primeira linha, que indica o número N de vértices no grafo, seguido de $N \times N$ valores que indicam os pesos associados as arestas que conectam dois vértices.

O programa é composto por 3 funções:

- *void* printSolution (int V, int **D)
  - Essa função imprime a solução calculada através do algoritmo de Floyd-Warshall, recebendo como parâmetro a matriz D com a solução calculada e o número V de linhas/colunas dessa matriz.
- *void* FloydWarshall(int V, int **G)
  - Função que calcula o resultado do algoritmo de Floyd-Warshall. Recebe como parâmetros a matriz de adjacência que representa um grafo G e o seu número V de vértices.
- *int* main(int argc, char*argv[])
  - Função principal do programa, que recebe o arquivo contendo as informações do grafo via linha de comando, processa essas informações para criar o grafo G e executa e exibe a solução do algoritmo de Floyd-Warshall.

1.2 Complexidade do algoritmo

O algoritmo de Floyd-Warshall roda em tempo $\Theta(N^3)$, pois ele é limitado por 3 laços aninhados, onde cada laço executa sempre N vezes, e esse custo é maior do que copiar os valores dos pesos das arestas para a matriz D antes de executar os laços aninhados.

1.3 Calcular e armazenar os caminhos dos menores caminhos para cada vértice

Os caminhos podem ser calculados enquanto calculamos a matriz D no algoritmo computando uma matriz Π de tamanho $N \times N$ com os predecessores nos caminhos. Inicialmente, a matriz Π será inicializada com Nil quando $i = j$ ou $w_{ij} = \infty$, e i quando $i \neq j$ e $w_{ij} < \infty$.

Para calcular os valores de Π enquanto calculamos D, basta fazer $\Pi[i][j] = \Pi[k][j]$ quando $D[i][j] > D[i][k] + D[k][j]$ a cada iteração dentro dos loops que calculam D. Com isso, ao final da execução do algoritmos teremos a matriz Π contendo o endereços dos predecessores.

Essa modificação não alteraria a complexidade do algoritmo.

1.4 Detecção de ciclos negativos

Para detectar ciclos negativos, basta verificar se algum valor da diagonal principal da matriz D é menos que 0 após a execução dos laços aninhados. Quando um grafo não possui ciclos negativos o caminho de um vértice para ele mesmo possui custo 0, e caso existam ciclos negativos, o caminho dele para ele mesmo será menor que 0, indicando que foi possível sair dele e chegar a ele mesmo por um ciclo negativo.

Como verificar a diagonal de uma matriz quadrata tem custo (N), e essa verificação é feita fora dos laços aninhados, a complexidade do algoritmo permanece a mesma.

2 Instruções de compilação

Como o programa faz uso de funções que só estão presentes a partir da versão 11 do $C++$, é necessário compilar o programa especificando qual versão do $C++$ estamos utilizando. O comando abaixo deve ser utilizado para compilar o arquivo *RBTree.cpp*:

g++ -std=c++11 floydwarshall.cpp -o floydwarshall

Para rodar o programa, basta executar o seguitne comando:

./floydwarshall <arquivo.txt>

Caso o arquivo contendo as informações do grafo não seja passado via linha de comando ao rodar o programa, o programa retornará um erro.

Referências

CORMEN, T. H. *Introduction to algorithms*. [S.l.]: MIT press, 2009.