Instituto Superior Técnico

**Relatório 1º Projecto de ASA**

*21 de Março de 2015*

Grupo 096

Gonçalo Fialho (ist179112)

Pedro Santos (ist178328)

**1 - Introdução:**

***1.1 - O problema:***

O problema consiste no cálculo do número de Erdõs de todos os cientistas que realizaram artigos com co-autores de Paul Erdõs ou mesmo com o próprio Paul Erdõs.

O matemático A tem número de Erdõs 1 se publicou pelo menos um trabalho em co-autoria com Erdõs, tem número de Erdõs número 2 se publicou em co-autoria com um matemático que tem número de Erdõs 1, e assim sucessivamente sendo que a Erdõs atribui-se o número 0.

***1.2 - O Input:***

O input consiste numa primeira linha com o número de pessoas N, e o número de relações de colaboração C, na segunda linha um número que identifica Paul Erdõs (entre 1 e N), e numa lista de C linhas em que cada linha contém dois inteiros de ***u*** e ***v*** que representam que a pessoa identificada por ***u*** tem pelo menos uma co-autoria com a pessoa identificada por ***v***.

Todas as pessoas são representadas por números inteiros entre 1 e N.

***1.3 - O Output:***

O output é representado com uma primeira linha M, sendo o maior valor de número de Erdõs dado pelo input. De seguida a sequência de M linhas em que a linha ***i*** (1 ≤ ***i*** ≤ M) corresponde ao número de pessoas com número de Erdõs igual a ***i***.

**2 - Descrição da Solução:**

***2.1 - Linguagem de programação:***

Para a implementação do projecto foi escolhida a linguagem C++ porque das três linguagem disponíveis para a realização do problema esta possui várias estruturas de dados já implementadas de raiz facilitando assim a criação de outras estruturas necessárias à implementação do problema.

***2.2 - Estruturas de dados:***

As estruturas de dados utilizadas no projecto foram:

* **std::vector** - vetor de memória dinâmica;
* **std::queue** - fila (FIFO);
* **graph** - estrutura criada para simular um grafo que contém os vértices e as suas ligações.

***2.3 - A solução:***

Após a análise do problema concluímos que seria necessário utilizar um ***Grafo*** (não dirigido) para a resolução do problema, pois existem os cientistas (vértices) que trabalham entre si (arcos), como a solução do problema passava por determinar o número de relações entre os cientistas foi necessário implementar o algoritmo ***Breadth-First Search*** (BFS).

***Grafo***: G = (V,E), definido por um conjunto V de vértices e um conjunto E de arcos.

***Breadth-First Search***:Dados G = (V,E) e vértice fonte *s*, o algoritmo BFS explora sistematicamente os vértices de G para descobrir todos os vértices atingíveis a partir de *s*.

***2.4 - Algoritmo:***

O programa começa por obter o ler o input e cria um grafo *G* a partir da informação recolhida, logo aseguir é executado o algoritmo de procura ***BFS*** em *G* e de seguida é dado os resultados do programa:

1. O programa começa por ler a primeira linha do input e cria um grafo **G** com N vértices e C arcos.
2. De seguida o programa guarda o índice que corresponde a Paul Erdõs (*var* **start**) para mais tarde calcular no algoritmo de procura.
3. São lidos as restantes linhas de input do programa ligando arcos aos vértices do grafo **G**.
4. É chamada a função BFS para o grafo **G** com início no vértice **start**.
   1. É criada uma *Queue* (**queue**) para auxiliar a procura, e dois *Arrays* (**color** **distance**).
   2. Os *Arrays* são inicializados com a cor WHITE (**color**) e distância -1 (**distance**).
   3. O vértice **start** é colocado com a cor GREY e à distância 0 (por ser o vértice inicial).
   4. É colocado na pilha o vértice **start**.
   5. O algoritmo BFS entra no seu ciclo principal até a *Queue* ficar vazia:
      1. É retirado o primeiro elemento que foi colocado na pilha para ser analisado.
      2. É realizada uma procura na lista de arcos para analisar os vértices adjacentes ao vértice que está a ser analisado.
      3. Se os vértices adjacentes ainda não foram analisados (cor WHITE) passam a ter a cor GREY e o valor da distância é incrementada. Estes vértices são também colocados na *Queue*
      4. Depois de os vértices adjacentes serem analisados coloca-se o vértice que estava a ser analisado com cor BLACK.
      5. Repete-se o processo para os restantes vértices.
   6. O algoritmo produz um output que diz o número máximo de Erdõs (**M**).
   7. É criado um *Array* **cnt** e é executado um algoritmo *Couting Sort* para determinar o número de cientistas que estão à distância 1 < *i* < **M** de Erdõs.
   8. O programa gera o output em que cada linha *i* corresponde ao número de cientistas com *i* Erdõs.

**3 - Análise Teórica:**

Este programa executa em tempo linear ao tamanho do problema. Analisando a complexidade de cada função do programa:

* Função *main* é ***O(V+E):***
  + Leitura do Input é***O(E)***
  + Construtor do grafo é ***O(V + E)***
  + Função BFS é ***O(V + E)***:
    - Inicializar array de cores e distâncias é ***O(V)***
    - Ciclo principal é ***O(V + E)***
    - Inicializar vector **cnt** é ***O(E)***
    - Contar vértices com número de Erdos i ≤ M é ***O(V)***
    - Output é ***O(E)***

**4 - Avaliação Experimental:**

A solução apresentada para este projecto, passou a todos os testes disponíveis no sistema Mooshak, com distinção. Sendo que os testes do Mooshak não são acessíveis para avaliar os tempos de execução optamos por usar os testes públicos e criar mais alguns testes para avaliar o desempenho do nosso programa.

Os resultados desses testes estão descritos na seguinte tabela e representados no gráfico.

|  |  |  |
| --- | --- | --- |
| Nº de Vértices | Nº de Arcos | Tempo (ms) |
| 8 | 15 | 003 |
| 20 | 50 | 004 |
| 250 | 945 | 005 |
| 5 000 | 10 000 | 017 |
| 10 000 | 25 000 | 042 |
| 15 000 | 50 000 | 062 |
| 30 000 | 90 000 | 118 |
| 50 000 | 100 000 | 162 |

