Instituto Superior Técnico

# Relatório do 1º Projecto de ASA

Grupo 162
André Silva (ist175455)
Francisco Baio (ist75328)
23 de Março de 2015

# Introdução

## Problema

Sendo considerado um dos maiores matemáticos do século XX, Paul Erdõs muito contribuiu para áreas como a teoria dos conjuntos, teoria dos números ou teoria do grafos, chegando a colaborar com mais de 500 outros matemáticos, publicando mais de 1500 artigos. Com isto, a distancia entre co-autores até Paul poderia ser calculada como num numero finito de saltos, de co-autor a co-autor.

## Input

O input consiste numa sequência de linhas com 3 propriedades.
A primeira linha contem dois números separados por um espaço, N e R, em que N corresponde ao numero de autores e R ao numero total de relações entre os mesmos.
A segunda linha consiste no índice em que se encontra Paul Erdõs no posterior grafo das relações.
As seguintes P linhas são compostas por um par de números, separados por um espaço, que contem informação das relações entre autores.

## Output

No output estará presente uma lista números, sendo o primeiro, a maior distancia no grafo processado entre um co-autor e Paul Erdôs, seguido de uma lista, ordenada pela distancia a Paul, com a quantidade de autores com a mesma distancia.

# Descrição da Solução

## Linguagem de Programação

Foi escolhido C++ para a implementação do projecto. Das três linguagens disponíveis para escolha (C, C++ e Java) esta revelou-se a mais intuitiva para implementar um algoritmo com restrições de tempo e memória. Os factores que mais pesaram na escolha foram: possuir várias estruturas previamente implementadas e permitir um grande controlo dos recursos.

## Estruturas de Dados

Da Standard Library usamos a std::queue e std::vector
std::vector -> Um vector dinâmico de memória contigua.
std::queue -> uma lista FIFO.

Cada vértice do nosso grafo é uma instância da classe Author que tem como atributo o `vector<int> coauthors;` que regista os ids dos co-autores desse autor.

Nas variáveis globais temos:

- `int n;` - número de nós (pessoas)
- `int r;` - número de relações entre autores
- `int paul;` - guarda o valor do vértice inicial do grafo
- `vector<Author*> author;` - é o nosso grafo, serve para guardarmos um número de pessoas igual ao N. As pessoas são identificadas pelo número do índice que têm no vector
- `queue<int> q;` - funciona como pilha para o algoritmo BFS
- `bool *visited;` - array de booleanos onde registamos se cada nó já foi visitado ou não
- `int *dist;` - array de inteiros onde registamos a distância de cada nó à origem

## A Solução

Pelo enunciado percebemos que se tratava de um grafo não dirigido, visto que se tratava das relações entre um autor ter um ou mais co-autores (vértices, arestas). Como o principal problema que queríamos resolver era saber as distancias entre autores, uma BFS é a solução mais indicada visto que o propósito

de uma Breadth First Search, parecido com o algoritmos Dijkstra, é procurar os caminhos mais curtos, sendo na BFS, as arestas mantêm todas o mesmo comprimento.

# Analise Teórica

## O Algoritmo BFS

Todos os vértices devem estar desmarcados (bool *visited)
Escolher um vértice onde começar (int paul)
Marcar o vértice como visitado e adicioná-lo à queue
Enquanto queue não estiver vazia
    remover o primeiro da queue e procurar por todos os adjacentes do removido
    se algum dos adjacentes do removido não tiverem sido visitados
        então marcá-lo como visitado e inseri-lo na queue
        regista distancia do mesmo igual à do retirado anteriormente da queue + 1

# Avaliação Experimental

Após lermos o input, criamos **n** nodes, adicionando por cada um, um novo **Author** ao vector, marcando a posição correspondente dos **visited** a false e a **dist** a máximo. De seguida, registamos as relações em cada **Author**.
Assim que temos o grafo criado, começamos o algoritmos BFS por colocar o **paul** na **pilha** e, enquanto esta não estiver vazia, vamos retirando Authors, e visitando os seus co-autores um a um, sendo que quando ainda não tinham sido visitados, os adicionamos à pilha e registamos a sua **dist** como a do autor actual mais um.

O nosso programa tem a seguinte complexidade:

- Criação de autores: O(N)
- Leitura de relações: O(R)
- BFS: O(N+R)