

Relatório do 1º projecto de ASA

Rodrigo André Moreira Bernardo
ist178942

Instituto Superior Técnico

23 de Março de 2015

1 O Problema

1.1 Introdução

Paul Erdős, reconhecido matemático do século XX, colaborou com mais de 500 matemáticos na co-autoria de artigos científicos. O número de Erdős é definido como a distância colaborativa de uma pessoa a Paul Erdős. Paul Erdős tem número 0, um co-autor de artigos com Paul Erdős tem número de Erdős 1, um co-autor de artigos com co-autores de Paul Erdős tem número 2 e assim por adiante [1].

1.2 Objectivo

Dado um input que identifique Erdős, um conjunto de colaboradores seus e as colaborações entre si, determinar os números de Erdős de cada um dos colaboradores. O output deverá indicar o maior valor, M , de número de Erdős identificado, assim como informação do número de pessoas com número de Erdős i , com $1 \leq i \leq M$.

2 A Solução

O programa foi implementado em linguagem C. A solução passa por executar uma procura em largura primeiro (BFS) sobre um grafo cujos vértices representam Erdős e os colaboradores, e cujos arcos representam as colaborações. Com a BFS conseguimos obter a distância colaborativa entre cada

colaborador e Erdős. Por fim, apenas é necessário efectuar duas passagens pelos colaboradores: uma para determinar M e outra para determinar os valores i (ver *Objectivo*).

2.1 A Representação

Tanto Paul Erdős como os colaboradores são representados por um inteiro. A representação do grafo é em listas de adjacências. Mais pormenorizadamente, este é representado através de um vector de listas ligadas simples, com tamanho igual ao número de vértices. Cada lista tem a informação relativa ao vértice correspondente, utilizada no algoritmo BFS (cor, distância e predecessor), assim como um apontador para o primeiro elemento da lista. Cada elemento da lista contém um inteiro representativo do colaborador e um apontador para o próximo elemento.

2.2 O Algoritmo

O algoritmo BFS utilizado é um adaptado da página 595 da terceira edição do livro *Introduction to Algorithms* [2].

3 Análise Teórica

3.1 Avaliação

Do programa destacam-se os seguintes blocos:

- i. A inicialização do vector de listas de adjacências, nas linhas 211-214;
- ii. A inserção dos arcos no vector, nas linhas 217-233;
- iii. A BFS, nas linhas 239 e 150-189;
- iv. A determinação de M (ver *Objectivo*), nas linhas 242-247;
- v. A determinação dos valores i (ver *Objectivo*), nas linhas 254-258;
- vi. A impressão do output, nas linhas 261-265;
- vii. As operações de alocação e libertação de memória.

Sejam V o número de vértices e E o número de arcos do grafo. A complexidade das operações realizadas nos pontos **i.**, **iv.**, **v.** é $O(V)$ (ciclos *for* no tamanho do número de vértices). Em relação ao ponto **ii.**, deve ser referido

que a operação de inserção em lista é $O(1)$, pois cada novo elemento é inserido no início da lista. Assim, como a operação *scanf* também é constante, conclui-se que o tempo de execução deste ponto é $O(E)$, visto que é um ciclo *for* no tamanho do número de arcos. O tempo de execução do algoritmo BFS é analisado em [2], na página 597, sendo este $O(V + E)$ (ponto **iii.**). De facto, o tempo gasto em operações relacionadas com a *queue* é $O(V)$, assim como na inicialização. Para além disso, cada lista de adjacências é passada apenas uma vez, sendo que o tempo para passar cada uma é $O(E)$. Como $M \leq E$, sai que a complexidade relativa ao ponto **vi.** é $O(E)$. Por fim, relativamente ao ponto **vii.**, alocou-se memória para o vector de listas de adjacências, para as listas em si, para os seus elementos e para a *queue* utilizada no algoritmo BFS. Tem-se que o número de listas é igual a V , que o número de elementos que cada uma tem é $O(E)$ e que o tamanho da *queue* é $O(V)$.

3.2 Conclusões

Conclui-se que o tempo de execução do programa é $O(V + E)$. A memória ocupada é, também, $O(V + E)$.

4 Avaliação Experimental

Após correr alguns testes, foram obtidos os seguintes tempos (totais) de execução com a ferramenta *Unix time*.

Vértices	Arcos	Tempo (s)
500	5 000	0,005
15 000	60 000	0,063
30 000	60 000	0,064
30 000	90 000	0,080
30 000	120 000	0,105
45 000	60 000	0,067
45 000	90 000	0,088
45 000	120 000	0,109
45 000	150 000	0,133

Também foi obtida a memória utilizada, com recurso à ferramenta *valgrind*.

Vértices	Arcos	Memória (bytes)
15 000	60 000	2 637 892
30 000	60 000	3 360 036
30 000	90 000	4 320 036
30 000	120 000	5 280 036
45 000	60 000	4 080 036
45 000	90 000	5 040 036
45 000	120 000	6 000 036
45 000	150 000	6 960 036

Com recurso às tabelas, é possível identificar que o programa corre linearmente no tamanho do input, como seria de esperar. Destaca-se, no entanto, que o programa nota uma sobrecarga maior se aumentarmos os arcos do que se aumentarmos os vértices na mesma quantidade. Isto verifica-se porque no algoritmo BFS existem mais passagem sobre os arcos do que sobre os vértices, no caso do tempo. O impacto na memória do aumento dos arcos já seria de esperar, obviamente.

5 Referências

- [1] Enunciado do primeiro projecto de ASA.
- [2] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald. L; Stein, Clifford (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.