

Relatório - 2º Projeto ASA

1- Introdução

O 2º Projeto de ASA é sobre uma rede, constituída por fornecedores, estações de abastecimento e um hipermercado. O programa que foi desenvolvido recebe como input a representação de uma rede, num grafo, com vértices(estacoes, fornecedores e o hiper) e arestas(ligações) e, deve ter como output o fluxo máximo, bem como os identificadores(id) das estações de abastecimento e das ligações que devem ser aumentadas.

O input é:

- Uma linha com os valores $f > 0$, $e \geq 0$ e $t \geq 0$ separados por 1 espaço, que representam, respectivamente, o número de fornecedores, o número de estações de abastecimento e o número de ligações que existem na rede do Sr. Caracol;
- Uma linha com f inteiros, separados por 1 espaço, que representam a produção de cada fornecedor;
- Uma linha com e inteiros, separados por 1 espaço, que representam a capacidade de cada estação de abastecimento;
- Uma sequência de t linhas com 3 inteiros $o \geq 2$, $d \geq 1$ e $c \geq 1$ que representam a origem de uma ligação o , o destino de uma ligação d e a capacidade c da ligação em causa;
- Para identificação de uma ligação é utilizada a seguinte numeração dos vértices:
- O Hiper é sempre o número 1;
- Os números de 2 a $f + 1$ representam fornecedores;
- Os restantes números representam estações de abastecimento e controlo;

O output é:

- Uma linha com a capacidade máxima da rede;
- Uma linha com os números das estações de abastecimento que devem ser aumentadas separadas por um espaço e por ordem crescente. Caso não haja estações a ser aumentadas, deve ser apresentada uma linha apenas com o caracter de fim de linha;
- Uma sequência de linhas com os valores o e d das ligações que devem ser aumentadas. Esta sequência deve ser ordenada por ordem crescente, considerando primeiro o e em caso de empate d . Caso não haja ligações a ser aumentadas, nada deve ser apresentado;

2- Descrição da solução

Como linguagem de programação decidimos usar C. C tem nível baixo o suficiente para ser eficaz e eficiente na implementação deste projeto, onde tempo e uso de memória importam.

Como abstração face ao problema descrito, consideramos usar um nó do grafo como representação de um fornecedor, estação de abastecimento ou hipermercado e para cada ligação uma passagem de mercadoria com a sua devida capacidade.

Assim, temos um vetor de nós e cada nó possui uma lista ligada de arestas, as quais saem do mesmo para se ligarem a outros nós.

João Lopes 90732
Tomás Gomes 90782

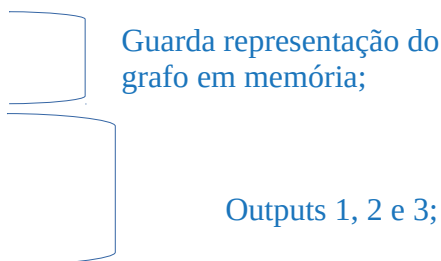
O objetivo é encontrar os identificadores(id) das estações de abastecimento e das ligações que devem ser aumentadas. Assim, temos como solução sugerida do nosso problema:

- 1) Construção do grafo a partir de inputs lidos no stdin.
- 2) Executar push relabel to front no grafo construido para encontrar o fluxo maximo.
- 3) Executar DFS a partir do hipermercado.

3- Análise Teórica

(Contextualização):

```
int main(){  
----- PASSO 1 -----  
    lerGrafo();  
----- PASSO 2 -----  
    pushRelabel();  
----- PASSO 3 -----  
    DFS();  
}
```

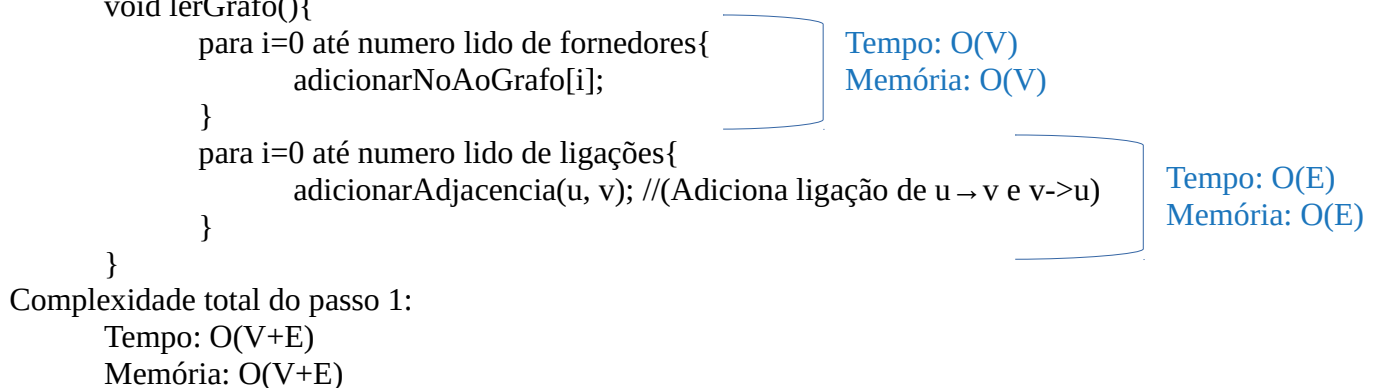


Guarda representação do grafo em memória;

Outputs 1, 2 e 3;

Passo 1: Construção do grafo

```
void lerGrafo(){  
    para i=0 até numero lido de fornecedores{  
        adicionarNoAoGrafo[i];  
    }  
    para i=0 até numero lido de ligações{  
        adicionarAdjacencia(u, v); //(Adiciona ligação de u → v e v->u)  
    }  
}
```



Tempo: $O(V)$
Memória: $O(V)$

Tempo: $O(E)$
Memória: $O(E)$

Complexidade total do passo 1:
Tempo: $O(V+E)$
Memória: $O(V+E)$

João Lopes 90732
Tomás Gomes 90782

Passo 2 e 3: DFS, marcação de pontos de articulação.

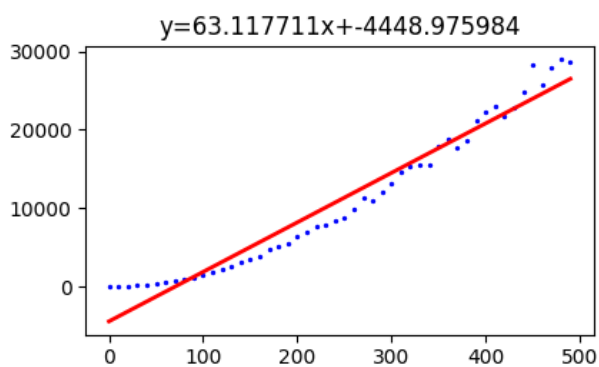
```
pushRelabel(){
    inicializarPreFlow();
    criarLista();
    PARA CADA VERTICE  $u \in V[G] - \{s, t\}$ 
         $current[u] \leftarrow head[N[u]]$ 
     $u \leftarrow head[L]$ 
    ENQUANTO  $u \neq NIL$ 
         $old-height \leftarrow h[u]$ 
        DISCHARGE( $u$ )
        SE  $h[u] > old-height$ 
            moveR  $u$  PARA A FRENTE DA lista  $L$ 
         $u \leftarrow next[u]$ 
}
```

Tempo: $O(V^2E)$

Assim, a complexidade do algoritmo é: $O(V^2E)$

4- Avaliação experimental dos resultados

Tempo



Memória

