Relatório ASA

Mihail Brinza 83533 Ricardo Brancas 83557

2 de Maio de 2017

1 Introdução

Este projeto tem como objetivo criar um sistema que permita ao utilizador conhecer a forma mais barata de ligar um conjunto de cidades com base numa rede de possíveis infra-estruturas. O utilizador dá como input ao sistema o número de cidades, o número de possiveis aeroportos, o custo de cada um deles, o número de estradas e o custo de cada uma. O software indica ainda caso não seja possível interligar todas as cidades. No caso de existirem duas formas de interligar as cidades com o mesmo custo, é indicada aquela que utiliza menos aeroportos.

2 Descrição da Solução

Para resolver o problema de encontrar a forma mais barata de interligar c cidades, considerando a possíveis aeroportos e e possíveis estradas, considerámos um grafo não dirigido $G = (V, E, w)$ em que:

1. $|V| = c + 1$
2. $|E| = a + e$
3. w é a função de pesos que faz corresponder a cada arco o custo de construir a estrada / aeroporto.

Neste grafo uma estrada entre as cidades i e j corresponde a um arco (v_i, v_j); e existir um aeroporto na cidade i corresponde a existir um arco (v_0, v_i), onde v_0 é um vértice especial que representa as ligações entre todos os aeroportos (como se fosse o céu).

Considerando este grafo, encontrar a solução do problema passa por descobrir a MST do grafo, sendo apenas necessário tomar especial atenção ao requerimento de que deve ser escolhida a MST que utiliza menos aeroportos (na realidade é necessário descobrir duas MST, uma considerando só as estradas e outra considerando todos os arcos).

3 Análise Teórica

Para encontrar a(s) MST utilizámos o Algoritmo de Kruskal. Consideramos que o algoritmo recebe como input duas priority queues de arcos (estradas num e aeroportos no outro) ordenadas segundo: 1) o peso dos arcos; 2) se o arco é um arco de aeroporto.

Nas seguintes complexiades considera-se que A = número de aeroportos, R = número de estradas.

1. Inicialização 1: $O(V)$
   Criamos um vetor de Disjoint-Sets que vai ser utilizado no algortimo.
2. Ciclo 1: $O(R \cdot log(V))$
   Executamos o ciclo principal do algoritmo de Kruskal, apenas em relação aos arcos de estradas. Sempre que escolhemos um arco, juntamo-lo à priority queue que contem os arcos de aeroporto. Ao longo da execução vamos computando o custo da MST e contando o número de arcos já selecionados.

3. Inicialização 2: $O(V)$
   Re-inicializamos o vetor dos Disjoint-Sets.

4. Ciclo 2: $O((V + A) \cdot log(V))$
   No segundo ciclo executamos novamente o algoritmo de Kruskal, mas apenas considerando os arcos escolhidos durante a ultima execução ($V - 1$ no máximo) e os arcos de aeroporto (ainda não considerados). Esta solução funciona porque os restantes arcos já nunca seriam escolhidos. Os arcos que vão sendo escolhidos são colocados num vetor. Ao longo da execução vamos contando, separadamente, o número de estradas e de aeroportos já selecionados.

5. Finalização: $O(V)$
   No fim é necessário verficar se todos os vértices fazem parte do mesmo Disjoint-Set. Computamos o custo da MST através dos arcos guardados no vetor (tendo em atenção ao detalhe que podemos ter apenas um arco-aeroporto selecionado). De seguida é necessário verificar se a primeira MST calculada é abrangente ao grafo todo e se o seu custo é inferior ao da segunda, se sim, essa é a solução ótima, se não, a solução ótima é a segunda MST calculada.

Complexidade total: $O(3V + R \cdot log(V) + (V + A) \cdot log(V)) = O((E + V) \cdot log(V))$
No caso em que o grafo é conexo ($E \geq V - 1$): $O(E \cdot log(V))$

```
KRUSKALS(V, R, A)
 1  MST = {}
 2  MST_roads = {}
 3  for each vertex u ∈ V
 4       MAKE-SET(u)

 5  while R ≠ {}
 6       e = (u, v) = HEAP-EXTRACT(R)
 7       if FIND-SET(u) ≠ FIND-SET(v)
 8            MST_roads = MST_roads ∪ {e}
 9            HEAP-INSERT(A, e)
10            UNION(u, v)

11  for each vertex u ∈ V
12       MAKE-SET(u)

13  while A ≠ {}
14       e = (u, v) = HEAP-EXTRACT(A)
15       if FIND-SET(u) ≠ FIND-SET(v)
16            MST = MST ∪ {e}
17            UNION(u, v)

18  if ∃ u, v ∈ V \ {v_0} : FIND-SET(u) ≠ FIND-SET(v)
19       error "Graph is not connected"
20  if IS-OPTIMAL(MST_roads)
21       return MST_roads
22  else
23       return MST
```

4 Avaliação Experimental

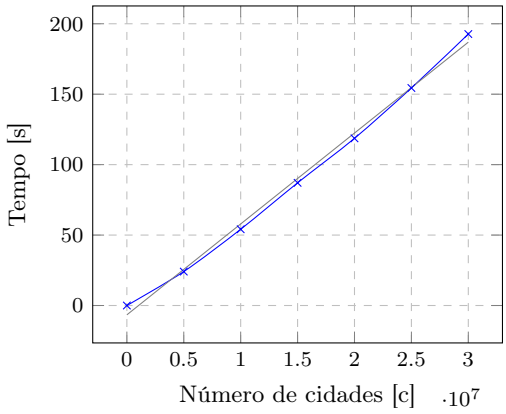
Os gráficos seguintes foram obtidos com recurso ao gerador de inputs. Considerámos dois casos:

1. Um "caso normal" em que $|E| = 8|V|$
2. Um "pior caso" em que $|E| = |V|^2$

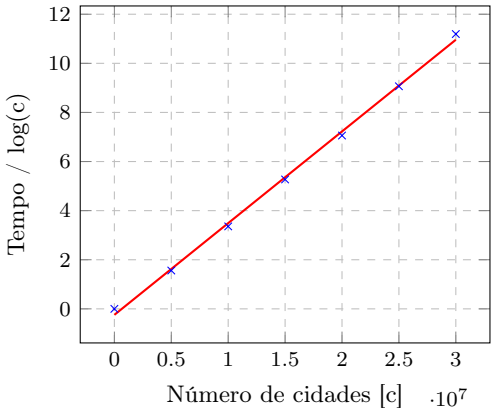
4.1 Caso normal

Concluímos que no "caso normal" o algoritmo comporta-se como esperado, tendo comportamento $O(x \cdot log(x))$, isto porque a complexidade teórica é $O(E \cdot log(V))$ e $|E| \simeq |V|$

Tempo de execução para problemas com solução



Tempo de execução dividido por $log(c)$



4.2 Pior caso

No "pior caso", verificámos que o algortimo tem comportamento $O(x^2)$, porque a complexidade teórica é $O(E \cdot log(V))$ e $|E| = |V|^2 \gg log(|V|)$

Tempo de execução para problemas com solução