Análise e Síntese de Algoritmos

Estruturas de Dados para Conjuntos Disjuntos. Algoritmo de Kruskal.

Prof. Pedro T. Monteiro

IST - Universidade de Lisboa

2024/2025

Contexto

- Revisão [CLRS, Cap.1-13]
  - Fundamentos; notação; exemplos
- Técnicas de Síntese de Algoritmos [CLRS, Cap.15-16]
  - Programação dinâmica
  - Algoritmos greedy
- Algoritmos em Grafos [CLRS, Cap.21-26]
  - Algoritmos elementares
  - Caminhos mais curtos [CLRS, Cap.22,24-25]
  - Árvores abrangentes [CLRS, (Cap. 21 +) Cap.23]
  - Fluxos máximos [CLRS, Cap.26]
- Programação Linear [CLRS, Cap.29]
  - Algoritmos e modelação de problemas com restrições lineares
- Tópicos Adicionais [CLRS, Cap.32-35]
  - Complexidade Computacional

Resumo

Conjuntos Disjuntos

Definições

Conjuntos $\{S_1, ..., S_n\}$ sem elementos em comum, ou seja, a intersecção entre quaisquer dois conjuntos é o conjunto vazio $S_i \cap S_j = \emptyset, i \neq j$

- Cada conjunto caracterizado por um representante - elemento do conjunto

Estrutura de Dados

Permite manter uma colecção de conjuntos disjuntos dinâmicos

- Consultas à estrutura de dados não altera o representante

Conjuntos Disjuntos

Operações

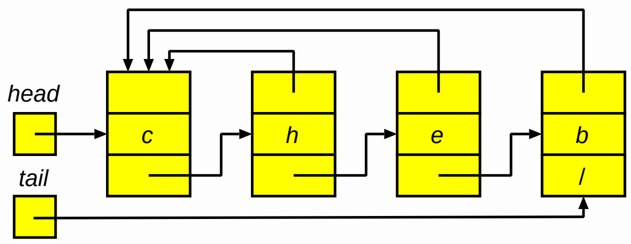
Cada elemento da estrutura é representado por objecto x

- MAKE-SET(X)
  - Cria novo conjunto que apenas inclui elemento x (representante)
  - x aponta para o único elemento do conjunto, o representante do conjunto
- UNION(X,Y)
  - Realiza a união dos conjuntos que contém x e y, respectivamente S_x e S_y
    - Novo conjunto criado: $S_x \cup S_y$
    - S_x e S_y eliminados (conjuntos disjuntos)
    - Novo representante será o representante de S_x ou S_y
- FIND-SET(X)
  - Retorna apontador para representante do conjunto que contém x

Uso de Lista ligada

Organização

- Elementos de cada conjunto em lista (simplesmente) ligada
- Primeiro elemento é o representante do conjunto
- Todos os elementos incluem apontador para o representante do conjunto



Uso de Lista ligada

Tempos de Execução

- MAKE-SET(x)
  - Criar nova lista com elemento x: $O(1)$
- FIND-SET(x)
  - Devolver ponteiro para representante: $O(1)$
- UNION(x,y):
  - Colocar elementos de x no fim da lista de y
  - Actualizar ponteiros de elementos de x para representante
- Operações sobre n elementos $x_1, x_2, \ldots, x_n$
  - n operações MAKE-SET(x_i)
    - $\Theta(n)$
  - $n-1$ operações UNION(x_{i-1}, x_i), para $i = 2, \ldots, n$
    - Cada operação UNION(x_{i-1}, x_i) actualiza $i-1$ elementos
    - Custo das $n-1$ operações: $\sum_{i=1}^{n-1} i = \Theta(n^2)$
  - Número total de operações é $m = 2n - 1$
    Em média, cada operação requer tempo $\Theta(n)$

Uso de Lista ligada

Heurística: União Pesada (union by size)

- A cada conjunto associar o número de elementos
- Para cada operação UNION, juntar lista com menor número de elementos à lista com maior número de elementos
  - Necessário actualizar menor número de ponteiros para representante
- Custo total de m operações é melhorado
- Sequência de m operações de MAKE-SET, UNION e FIND-SET (que incluem $O(n)$ operações UNION) é: $O(m + n \log n)$

Tempos de Execução (com Heurística) (Prova Teorema 21.1 CLRS)

- Sempre que o ponteiro para o representante de x é actualizado, x encontra-se no conjunto com menor número de elementos
  - Da 1ª vez, conjunto resultante com pelo menos 2 elementos
  - Da 2ª vez, conjunto resultante com pelo menos 4 elementos
  - ...
  - Após representante de x ter sido actualizado $\lceil log \ k \rceil$ vezes, conjunto resultante tem pelo menos k elementos
- Maior conjunto tem n elementos
  - Cada ponteiro (de qualquer elemento) actualizado não mais do que $\lceil \lg \ n \rceil$ vezes
- Tempo total para actualizar n elementos é $O(n \ log \ n)$
- MAKE-SET e FIND-SET têm tempos de execução $O(1)$ e há $O(m)$ destas operações
- Tempo total para m operações (com n UNION) é $O(m + n \ log \ n)$

Exemplo (usar heurística União Pesada)

for $i \leftarrow 1$ **to** 16 **do** MAKE-SET(x_i) **end for**	{1},{2},{3},{4},{5},{6},{7},{8},{9},{10},{11},{12},{13},{14},{15},{16}
for $i \leftarrow 1$ **to** 15 **by** 2 **do** UNION(x_i, x_{i+1}) **end for**	{1,2},{3,4},{5,6},{7,8},{9,10},{11,12},{13,14},{15,16}
for $i \leftarrow 1$ **to** 13 **by** 4 **do** UNION(x_i, x_{i+2}) **end for**	{1,2,3,4}, {5,6,7,8}, {9,10,11,12}, {13,14,15,16}
UNION(x_1, x_{13})	{1,2,3,4, 13,14,15,16}, {5,6,7,8}, {9,10,11,12}
UNION(x_6, x_9)	{1,2,3,4, 13,14,15,16}, {5,6,7,8, 9,10,11,12}
FIND-SET(x_7)	5
UNION(x_3, x_{11})	{1,2,3,4, 13,14,15,16, 5,6,7,8, 9,10,11,12}
FIND-SET(x_{14})	1

Organização

- Cada conjunto representado por uma árvore
- Cada elemento aponta apenas para antecessor na árvore
- Representante da árvore é a raíz
- Antecessor da raíz é a própria raíz

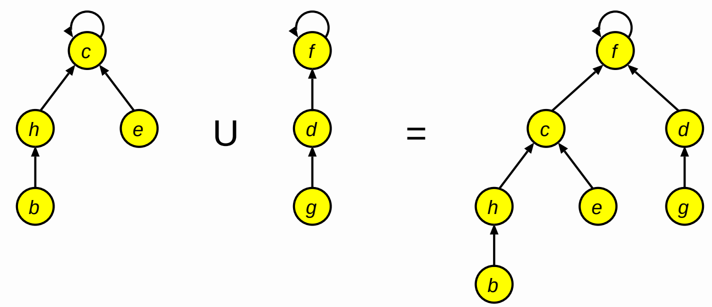
Operações

- FIND-SET: Percorrer antecessores até raíz ser encontrada $O(n)$
- UNION: raíz de uma árvore aponta para raíz da outra árvore $O(1)$

Complexidade

- Sequência de $O(m)$ operações é $O(m \, n)$
- Pior caso ocorre quando as árvores que são apenas listas dos n elementos

Exemplo



Uso de (floresta de) árvores

Heurística: União por Categoria (union by rank)

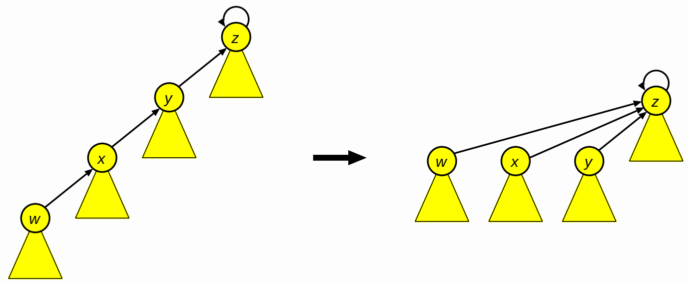
Numa união de dois conjuntos, colocar árvore com menos elementos a apontar para árvore com mais elementos

- Utilizar estimativa da altura de cada sub-árvore
- Categoria (*rank*): aproxima logaritmo do tamanho da sub-árvore e é um limite superior na altura da sub-árvore
- Numa união, raíz da árvore com menor *rank* aponta para raíz da árvore com maior *rank*

Uso de (floresta de) árvores

Heurística: Compressão de Caminhos (path compression)

Em cada operação FIND-SET coloca cada nó visitado a apontar directamente para a raíz da árvore (representante do conjunto)



Uso de (floresta de) árvores

```
Make-Set(x)
  p[x] ← x
  rank[x] ← 0

Find-Set(x)
  if x ≠ p[x] then
    p[x] ← Find-Set(p[x])
  end if
  return p[x]

Union(x, y)
  Link(Find-Set(x), Find-Set(y))

Link(x, y)
  if rank[x] > rank[y] then
    p[y] ← x
  else
    p[x] ← y
    if rank[x] == rank[y] then
      rank[y] ← rank[y] + 1
    end if
  end if
```


Uso de (floresta de) árvores

Complexidade

Execução de m operações sobre n elementos: $O(m\,\alpha(n))$

- $\alpha(n) \leq 4$ para todos os efeitos práticos

Prova

Capítulo 21.4 do livro CLRS (análise amortizada - método do potencial)

Turing Awardee Clips: Tarjan on analyzing the "union-find" data structure

https://www.youtube.com/watch?v=Hhk8ANKWGJA

Uso de (floresta de) árvores

Exemplo (usar heurística União por categoria & compressão caminhos)

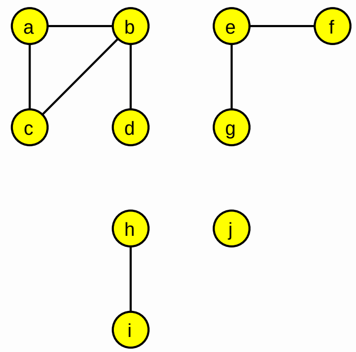
for $i \leftarrow 1$ **to** 16 **do**
  MAKE-SET(x_i)
end for
for $i \leftarrow 1$ **to** 15 **by** 2 **do**
  UNION(x_i, x_{i+1})
end for
for $i \leftarrow 1$ **to** 13 **by** 4 **do**
  UNION(x_i, x_{i+2})
end for
UNION(x_{11}, x_{13})
UNION(x_6, x_{16})
FIND-SET(x_9)

Conjuntos Disjuntos: Aplicações

Problema
Identificar os componentes ligados de um grafo não dirigido $G = (V, E)$

Connected-Components(G)

for each v $\in$ G.V **do**
  MAKE-SET(v)
end for
for each (u, v) $\in$ G.E **do**
  if FIND-SET(u) $\neq$ FIND-SET(v) **then**
    UNION(u, v)
  end if
end for



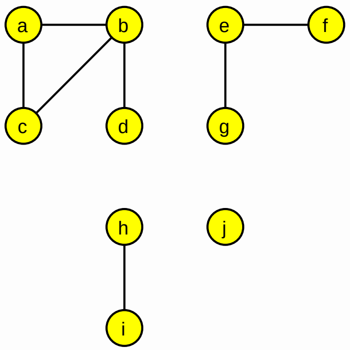
Same-Component(u, v)
  return FIND-SET(u) == FIND-SET(v)

Conjuntos Disjuntos: Aplicações

Problema
Identificar ciclos num grafo não dirigido $G = (V, E)$

Cycle-detection(G)

for each v $\in$ G.V **do**
  MAKE-SET(v)
end for
for each (u, v) $\in$ G.E **do**
  if FIND-SET(u) $\neq$ FIND-SET(v) **then**
    UNION(u, v)
  else
    return "Cycle found"
  end if
end for



Conjuntos Disjuntos: Aplicações

Problema
Considere que está na equipa de projecto de uma rede de distribuição entre um conjunto de cidades. Foram efectuados estudos que calcularam o custo $c(u, v)$ associado a cada ligação possível da nova rede. Pretende-se saber qual o menor custo total de uma rede que interligue todas as cidades

Solução

- Representar a rede como um grafo não-dirigido e pesado
- Função de pesos é definida pelo custo entre as possíveis ligações
- Rede de menor custo será a Árvore Abrangente de Menor Custo (MST) do grafo

Algoritmo de Kruskal

Características

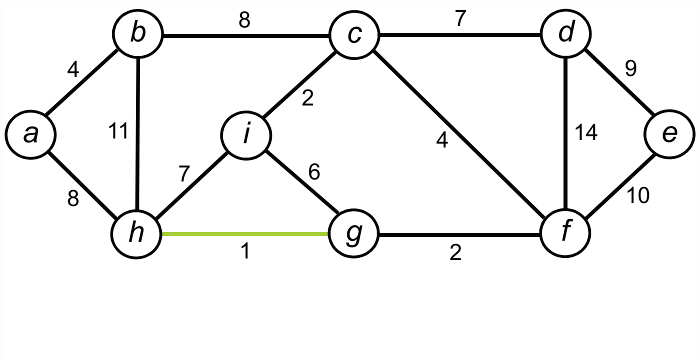
- Algoritmo greedy
- Mantém floresta (de árvores) A
- Utilização de uma estrutura de dados para representar conjuntos disjuntos
- Cada conjunto representa uma sub-árvore de uma MST
- Em cada passo é escolhido um arco leve, seguro para A

Algoritmo de Kruskal

MST-Kruskal(G,w)

```
A = ∅
for each v ∈ G.V do
  MAKE-SET(v)                           Cria conjunto para cada v
end for
sortedE ← sortNonDecreasing(G.E)
for each (u,v) ∈ sortedE do
  if FIND-SET(u) ≠ FIND-SET(v) then
    A = A ∪ {(u, v)}                    (u,v) é arco leve, seguro para A
    UNION(u, v)
  end if
end for
return A
```


Algoritmo de Kruskal

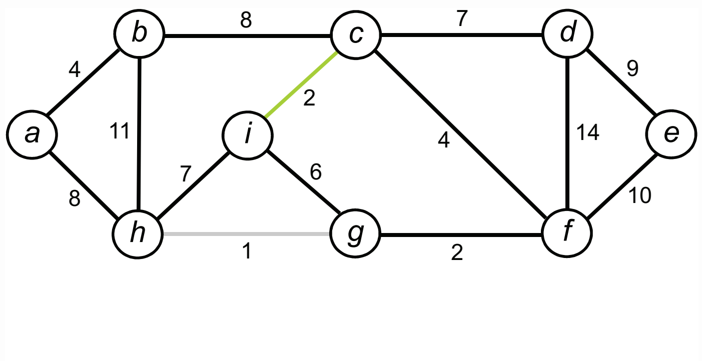


$A = \{ (h,g) \}$

$E =$ (h,g),(i,c),(g,f),(a,b),(c,f),(i,g),(c,d),(i,h),(a,h),(b,c),(d,e),(e,f),(b,h),(d,f)

1 2 2 4 4 6 7 7 8 8 9 10 11 14

Algoritmo de Kruskal

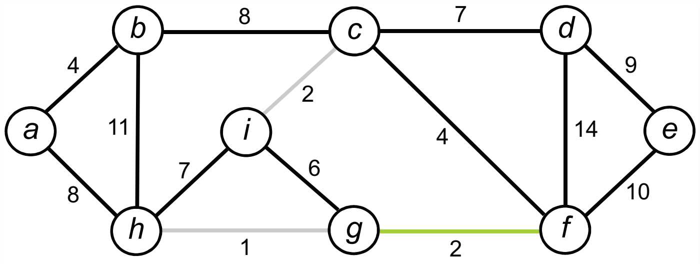


$A = \{ (h,g), (i,c) \}$

$E =$ (h,g),(i,c),(g,f),(a,b),(c,f),(i,g),(c,d),(i,h),(a,h),(b,c),(d,e),(e,f),(b,h),(d,f)

1 2 2 4 4 6 7 7 8 8 9 10 11 14

TÉCNICO LISBOA

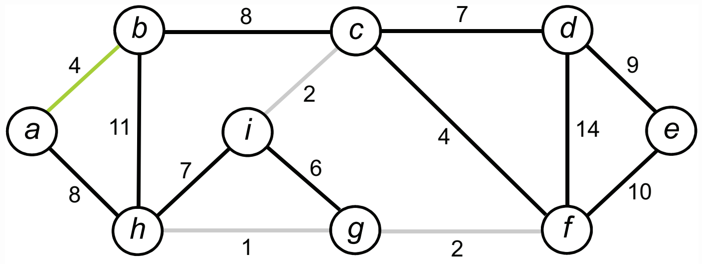


$A = \{ (h,g), (i,c), (g,f) \}$

$E =$ (h,g),(i,c),(g,f),(a,b),(c,f),(i,g),(c,d),(i,h),(a,h),(b,c),(d,e),(e,f),(b,h),(d,f)
1 2 2 4 4 6 7 7 8 8 9 10 11 14

Algoritmo de Kruskal

TÉCNICO LISBOA

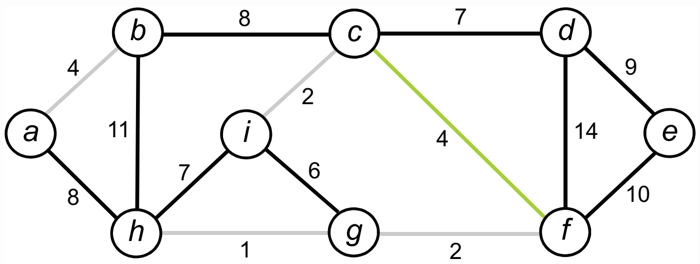


$A = \{ (h,g), (i,c), (g,f), (a,b) \}$

$E =$ (h,g),(i,c),(g,f),(a,b),(c,f),(i,g),(c,d),(i,h),(a,h),(b,c),(d,e),(e,f),(b,h),(d,f)
1 2 2 4 4 6 7 7 8 8 9 10 11 14

Algoritmo de Kruskal

TÉCNICO LISBOA

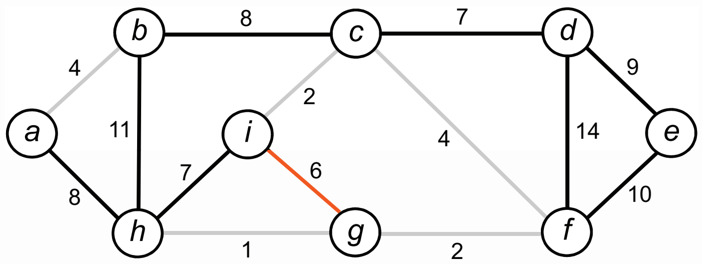


$A = \{ (h,g), (i,c), (g,f), (a,b), (c,f) \}$

$E =$ (h,g),(i,c),(g,f),(a,b),(c,f),(i,g),(c,d),(i,h),(a,h),(b,c),(d,e),(e,f),(b,h),(d,f)
1 2 2 4 4 6 7 7 8 8 9 10 11 14

Algoritmo de Kruskal

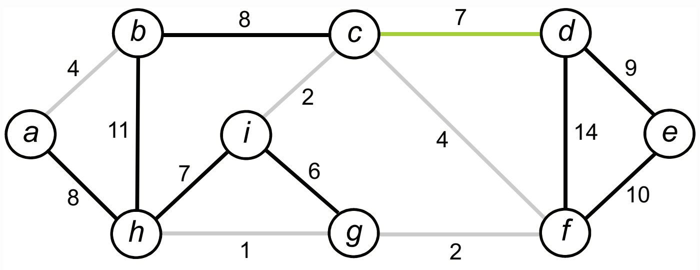
TÉCNICO LISBOA



$A = \{ (h,g), (i,c), (g,f), (a,b), (c,f) \}$

$E =$ (h,g),(i,c),(g,f),(a,b),(c,f),(i,g),(c,d),(i,h),(a,h),(b,c),(d,e),(e,f),(b,h),(d,f)
1 2 2 4 4 6 7 7 8 8 9 10 11 14

Algoritmo de Kruskal

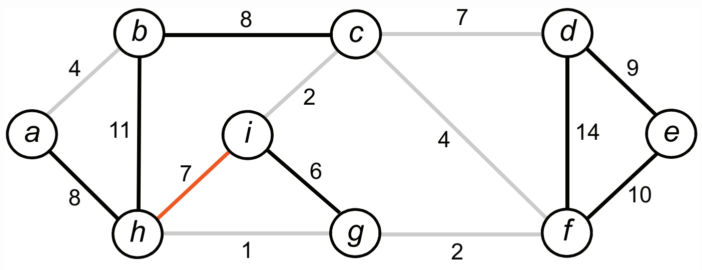


$A = \{ (h,g), (i,c), (g,f), (a,b), (c,f), (c,d) \}$

$E =$ (h,g),(i,c),(g,f),(a,b),(c,f),(i,g),(c,d),(i,h),(a,h),(b,c),(d,e),(e,f),(b,h),(d,f)

1 2 2 4 4 6 7 7 8 8 9 10 11 14

Algoritmo de Kruskal

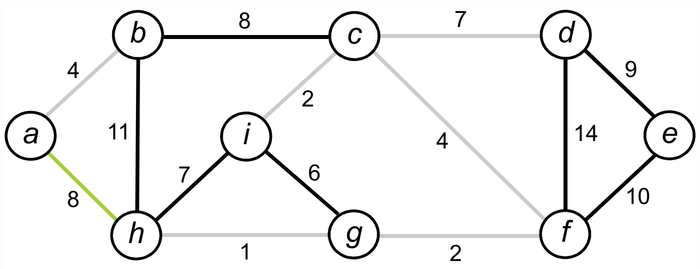


$A = \{ (h,g), (i,c), (g,f), (a,b), (c,f), (c,d) \}$

$E =$ (h,g),(i,c),(g,f),(a,b),(c,f),(i,g),(c,d),(i,h),(a,h),(b,c),(d,e),(e,f),(b,h),(d,f)

1 2 2 4 4 6 7 7 8 8 9 10 11 14

Algoritmo de Kruskal

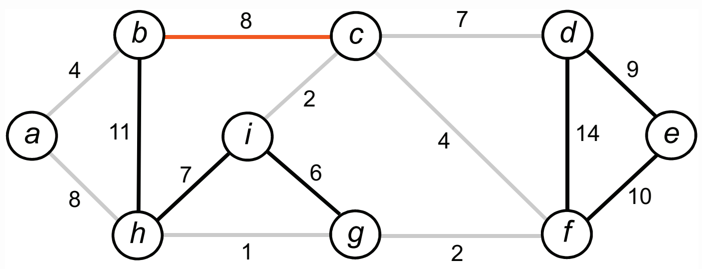


$A = \{ (h,g), (i,c), (g,f), (a,b), (c,f), (c,d), (a,h) \}$

$E =$ (h,g),(i,c),(g,f),(a,b),(c,f),(i,g),(c,d),(i,h),(a,h),(b,c),(d,e),(e,f),(b,h),(d,f)

1 2 2 4 4 6 7 7 8 8 9 10 11 14

Algoritmo de Kruskal

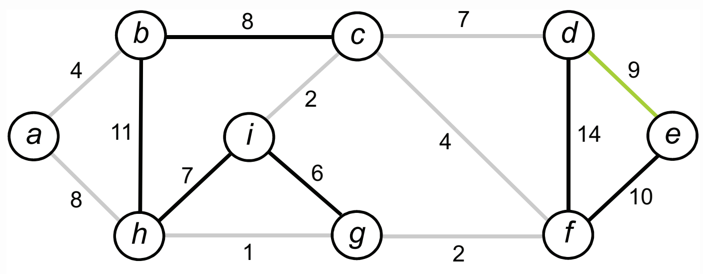


$A = \{ (h,g), (i,c), (g,f), (a,b), (c,f), (c,d), (a,h) \}$

$E =$ (h,g),(i,c),(g,f),(a,b),(c,f),(i,g),(c,d),(i,h),(a,h),(b,c),(d,e),(e,f),(b,h),(d,f)

1 2 2 4 4 6 7 7 8 8 9 10 11 14

Algoritmo de Kruskal

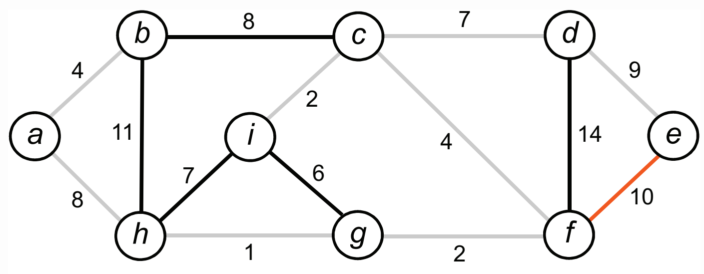


$A = \{\ (h,g),\ (i,c),\ (g,f),\ (a,b),\ (c,f),\ (c,d),\ (a,h),\ (d,e)\ \}$

$E =$ (h,g),(i,c),(g,f),(a,b),(c,f),(i,g),(c,d),(i,h),(a,h),(b,c),(d,e),(e,f),(b,h),(d,f)

1 2 2 4 4 6 7 7 8 8 9 10 11 14

Algoritmo de Kruskal

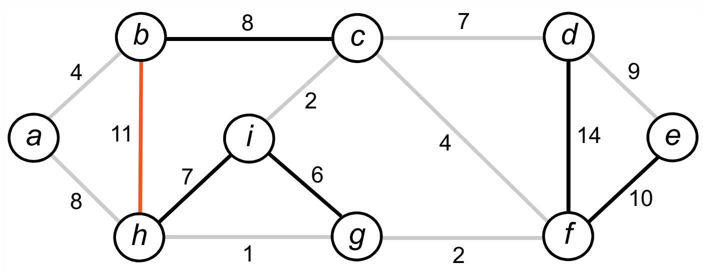


$A = \{\ (h,g),\ (i,c),\ (g,f),\ (a,b),\ (c,f),\ (c,d),\ (a,h),\ (d,e)\ \}$

$E =$ (h,g),(i,c),(g,f),(a,b),(c,f),(i,g),(c,d),(i,h),(a,h),(b,c),(d,e),(e,f),(b,h),(d,f)

1 2 2 4 4 6 7 7 8 8 9 10 11 14

Algoritmo de Kruskal

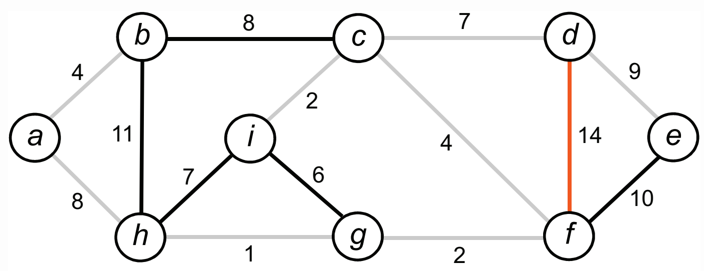


$A = \{\ (h,g),\ (i,c),\ (g,f),\ (a,b),\ (c,f),\ (c,d),\ (a,h),\ (d,e)\ \}$

$E =$ (h,g),(i,c),(g,f),(a,b),(c,f),(i,g),(c,d),(i,h),(a,h),(b,c),(d,e),(e,f),(b,h),(d,f)

1 2 2 4 4 6 7 7 8 8 9 10 11 14

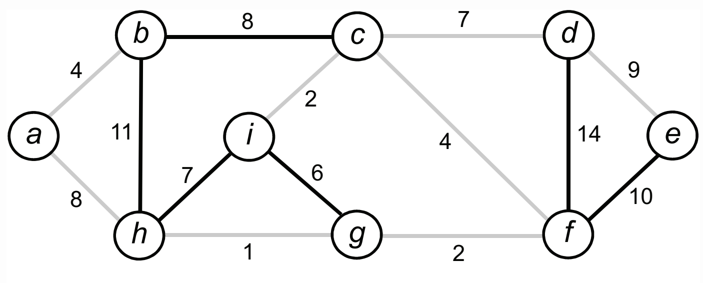
Algoritmo de Kruskal



$A = \{\ (h,g),\ (i,c),\ (g,f),\ (a,b),\ (c,f),\ (c,d),\ (a,h),\ (d,e)\ \}$

$E =$ (h,g),(i,c),(g,f),(a,b),(c,f),(i,g),(c,d),(i,h),(a,h),(b,c),(d,e),(e,f),(b,h),(d,f)

1 2 2 4 4 6 7 7 8 8 9 10 11 14



A = { (h,g), (i,c), (g,f), (a,b), (c,f), (c,d), (a,h), (d,e) }

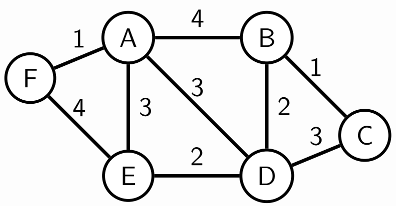
E = (h,g),(i,c),(g,f),(a,b),(c,f),(i,g),(c,d),(i,h),(a,h),(b,c),(d,e),(e,f),(b,h),(d,f)

1 2 2 4 4 6 7 7 8 8 9 10 11 14

Complexidade

- Depende da implementação das operações sobre conjuntos disjuntos
- Inicialização: $O(E \lg E)$ devido à ordenação dos arcos
- Operações sobre os conjuntos disjuntos
  - $O(V)$ operações de Make-Set
  - $O(E)$ operações de Find-Set e Union
  - Com estruturas de dados adequadas (árvores com compressão de caminhos e união por categorias) para conjuntos disjuntos é possível estabelecer que $O((V + E)\ \alpha(V))$
  - Como $|E| \geq |V| - 1$ porque o grafo é ligado, então temos $O(E\ \alpha(V))$
- Logo, é possível assegurar $O(E\ log\ E)$ (maior termo)
  - Dado que $|E| < |V|^2$, obtém-se também $O(E\ log\ V)$
    $log|E| < log|V|^2 \Leftrightarrow log|E| < 2log|V|$

Exercício: Calcule a MST usando o algoritmo de Kruskal

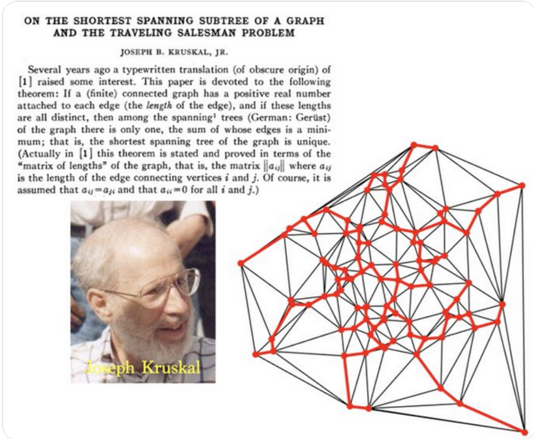


	A	B	C	D	E	F
rank[v]						
π[v]						

Peso das MSTs:	
Número de MSTs:	

Gabriel Peyré
@gabrielpeyre

Oldies but goldies: J. B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, 1956. Computes the minimum spanning tree in n*log(n) operations. en.wikipedia.org/wiki/Prim%27s_...


ON THE SHORTEST SPANNING SUBTREE OF A GRAPH AND THE TRAVELING SALESMAN PROBLEM

JOSEPH B. KRUSKAL, JR.

Several years ago a typewritten translation (of obscure origin) of [1] raised some interest. This paper is devoted to the following theorem: If a (finite) connected graph has a positive real number attached to each edge (the *length* of the edge), and if these lengths are all distinct, then among the spanning[1] trees (German: Gerüst) of the graph there is only one, the sum of whose edges is a minimum; that is, the shortest spanning tree of the graph is unique. (Actually in [1] this theorem is stated and proved in terms of the "matrix of lengths" of the graph, that is, the matrix $\|a_{ij}\|$ where a_{ij} is the length of the edge connecting vertices i and j. Of course, it is assumed that $a_{ij} = a_{ji}$ and that $a_{ii} = 0$ for all i and j.)

Questões?

Dúvidas?