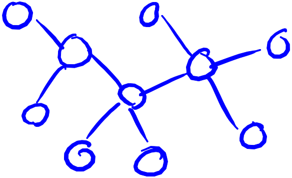
Árvores

Árvores

Definição (Árvore)

*Uma **árvore** é um grafo conexo e sem ciclos (acíclico)*



Árvores

Um resultado auxiliar que utilizaremos

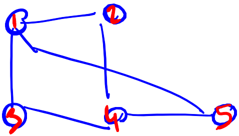
Lemma

Considere um grafo G. Se G tem um ***passeio que começa e termina no mesmo nó*** *(passeio fechado), então ele tem um* ***ciclo****.*

Exemplo:

passeio

5 – 1 – 2 – 4 – 3 – 1 – 5



Lemma

Considere um grafo G. Se G tem um ***passeio que começa e termina no mesmo nó*** *(passeio fechado), então ele tem um* ***ciclo****.*

Prova: Considere um passeio $W = (v_0 e_1 v_1 \ldots e_k v_0)$ começando e terminando em v_0

Se não tem nó repetido (a não ser v_0), é um ciclo, OK

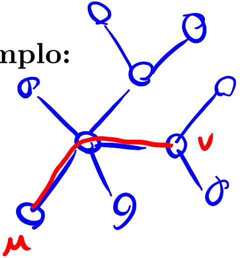
Caso contrário, olhe para o primeiro nó v_i que se repete em W, e seja v_j sua primeira repetição (v_j está depois de v_i)

Árvores

Theorem

Sejam u e v dois vértices distintos de uma árvore T. Então só existe um único caminho entre u e v em T.

Exemplo:



Theorem

Sejam u e v dois vértices distintos de uma árvore T. Então só existe um único caminho entre u e v em T.

Prova: Tem pelo menos um caminho entre u e v, pois é grafo conexo

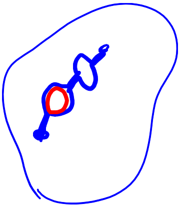
Theorem

Sejam u e v dois vértices distintos de uma árvore T. Então só existe um único caminho entre u e v em T.

Prova: Tem pelo menos um caminho entre u e v, pois é grafo conexo

Tem no máximo um caminho entre u e v: exercício (usando lemma anterior)

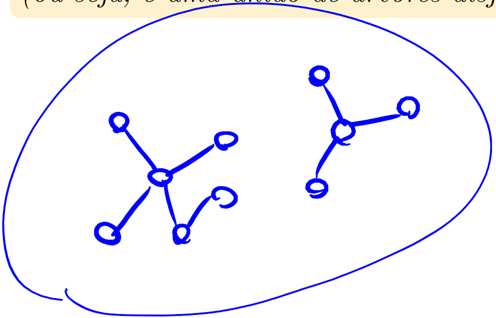
G =



Árvores

Definição (Floresta)

Uma floresta é um grafo G em que todos os componentes conexos são árvores
(ou seja, é uma união de árvores disjuntas)



$= G$

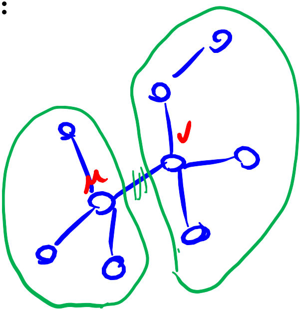
≡ grafo sem ciclo

Árvores

Proposição

Seja T uma árvore e seja (u, v) uma aresta de T. O grafo $T - (u, v)$ (removendo aresta (u, v)) é uma floresta com dois componentes conexos, um contendo o nó v e outro contendo o nó u

Exemplo:



Proposição

Seja T uma árvore e seja (u, v) uma aresta de T. O grafo $T - (u, v)$ (removendo aresta (u, v)) é uma floresta com dois componentes conexos, um contendo o nó v e outro contendo o nó u

Prova: 1) Não existe caminho entre u e v em $T - (u, v)$, e portanto esses nós estão em componentes conexos diferentes:

Pela proposição anterior, só tem um caminho entre u e v em T, então tem que ser o caminho "direto" que só usa aresta (u, v) $\Rightarrow$ esse único caminho desapareceu em $T - (u, v)$

2) Não tem mais do que 2 componentes conexos em $T - (u, v)$, pois tem no máximo um comp. conexo a mais que T:

Mencionamos na ultima aula que remover uma aresta aumenta o número de componentes conexos em no máximo 1

Ótimo exercício provar isso formalmente

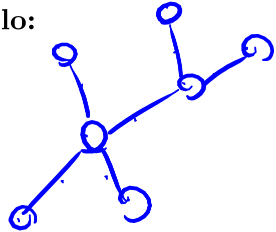
Por contradição

Árvores

Theorem

Se G é uma árvore não vazia, então $\#arestas = \#nós - 1$

Exemplo:



#arestas = 6

#nós = 7

Theorem

Se G é uma árvore não vazia, então $\#arestas = \#nós - 1$

Prova: Indução forte no número de arestas.

Árvores

Theorem

Se G é uma árvore não vazia, então $\#arestas = \#nós - 1$

Prova: Indução forte no número de arestas.

Caso base: 0 arestas. Então G tem exatamente 1 nó, e temos $\#\text{arestas} = \#\text{nós} - 1$

Theorem

Se G é uma árvore não vazia, então $\#arestas = \#nós - 1$

Prova: Indução forte no número de arestas.

Caso base: 0 arestas. Então G tem exatamente 1 nó, e temos #arestas = #nós −1

Passo indutivo: Suponha que o resultado válido para todas as árvores com no máximo m arestas

Theorem

Se G é uma árvore não vazia, então $\#arestas = \#nós - 1$

Prova: Indução forte no número de arestas.

Caso base: 0 arestas. Então G tem exatamente 1 nó, e temos #arestas = #nós −1

Passo indutivo: Suponha que o resultado válido para todas as árvores com no máximo m arestas

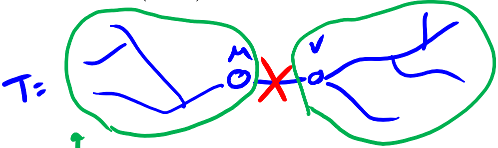
Vamos provar pra árvore com $m + 1$ arestas

Theorem

Se G é uma árvore não vazia, então #arestas = #nós -1

Seja T uma árvore com $m+1$ arestas

Seja (u, v) uma aresta de T e sejam T_u e T_v os componentes conexos de $T-(u, v)$ contendo u e v, respectivamente



m+1 arestas

pelo hip. ind

Árvores

Na verdade essa propriedade (+ conexidade) caracteriza árvores

Proposição

Se $G = (V, E)$ é um grafo conexo com $|E| = |V| - 1$, então G é uma árvore

(Não vamos provar isso)

Árvores Geradoras de Custo Mínimo

Árvores Geradoras de Custo Mínimo

Problema: Suponha que temos várias cidades, e queremos construir ruas de forma a conectar todas elas (ou seja, dá pra ir de qualquer uma a qualquer outra)

Suponha que tem custos diferentes de construir ruas entre cada duas cidades

Queremos computar a forma mais barata de conectar essas cidades

Árvores Geradoras de Custo Mínimo

Podemos representar esse problema em termos de grafos da seguinte forma

Entrada. Um grafo conexo $G = (V, E)$, com um custo c_e para cada aresta e

Saída. Um subconjunto de arestas $E' \subseteq E$ que satisfaz:

i Essas arestas formam uma árvore conectando todos os nós de G, ou seja, o grafo $G' = (V, E')$ é uma árvore

ii O custo total de E' (ou seja, $\sum_{e \in E'} c_e$) é menor possível

Árvores Geradoras de Custo Mínimo

Podemos representar esse problema em termos de grafos da seguinte forma

Entrada. Um grafo conexo $G = (V, E)$, com um custo c_e para cada aresta e

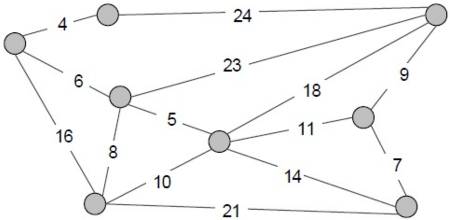
Saída. Um subconjunto de arestas $E' \subseteq E$ que satisfaz:

i Essas arestas formam uma árvore conectando todos os nós de G, ou seja, o grafo $G' = (V, E')$ é uma árvore

ii O custo total de E' (ou seja, $\sum_{e \in E'} c_e$) é menor possível

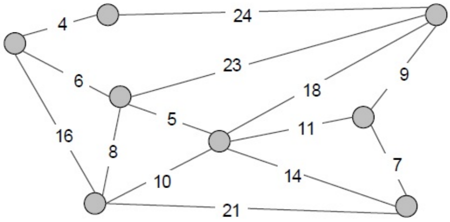
Chamada Árvore Geradora de Custo Mínimo (AGM)

Exemplo:



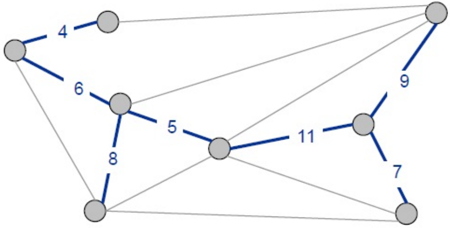
G = (V, E)

Exemplo:



G = (V, E)

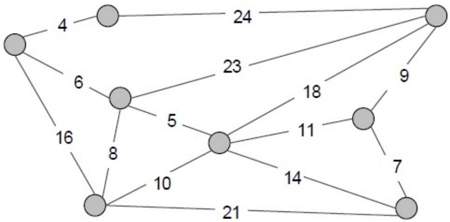
Árvore Geradora de Custo Mínimo:



Árvores Geradoras de Custo Mínimo

Note que "não estamos perdendo nada" em pedir uma árvore para conectar:

Se conectássemos os nós formando ciclo, poderíamos deletar arestas e obter uma solução mais barata



G = (V, E)

Árvores Geradoras de Custo Mínimo

Pergunta: Algoritmo para encontrar AGM?

Árvores Geradoras de Custo Mínimo

Pergunta: Algoritmo para encontrar AGM?

Vamos provar um lema que nos ajudará...

Árvores Geradoras de Custo Mínimo

Lema

Seja C um ciclo no grafo G e seja e uma aresta de custo máximo no ciclo C. Então existe uma AGM que não contém a aresta e

Exemplo:

Lema

Seja C um ciclo no grafo G e seja e uma aresta de custo máximo no ciclo C. Então existe uma AGM que não contém a aresta e

Prova: Considere uma AGM T para G. Se $e \notin T$, o lema está provado

Lema

Seja C um ciclo no grafo G e seja e uma aresta de custo máximo no ciclo C. Então existe uma AGM que não contém a aresta e

Prova: Considere uma AGM T para G. Se $e \notin T$, o lema está provado

Assuma então que T contém a aresta e. Seja $e = (u, v)$

Olhe para os componentes conexos de $T - (u, v)$ contendo u e v

O ciclo C tem um aresta (u', v') diferente de (u, v) com uma ponta em cada componente conexo de $T - (u, v)$ [exercício]

Pegue T e troque a aresta (u, v) pela aresta (u', v') (chame $T' = T - (u, v) + (u', v')$ esse novo grafo)

O novo grafo T' obtido é uma árvore:

1) É conexo
2) Mesmo número de arestas que T, então

$$\#\text{arestas de } T' = (\#\text{nós de } T') - 1$$

$\Rightarrow$ é árvore por proposição anterior

Como arestas (u, v) e (u', v') pertencem ao ciclo C e (u, v) é a mais cara, a árvore nova T' é mais barata:

$$\text{custo } T' = \text{custo } T \text{ - } c_{uv} + c_{u'v'} \leq \text{custo } T$$

Portanto T' também é árvore geradora mínima!

Então encontramos uma árvore geradora mínima sem a aresta (u, v), provando o resultado

Árvores Geradoras de Custo Mínimo

Baseado nesse lema, temos o seguinte algoritmo para encontrar AGM:

`Procedimento CalculaAGM(`G`)`

$T \leftarrow G$.

Enquanto existir um ciclo em T

Escolha um ciclo em T e remova de T a aresta de maior custo do ciclo

Fim Enquanto

Return T

Baseado nesse lema, temos o seguinte algoritmo para encontrar AGM:

`Procedimento CalculaAGM(`G`)`
  $T \leftarrow G$.
  Enquanto existir um ciclo em T
    Escolha um ciclo em T e remova de T a aresta de maior custo do ciclo
  Fim Enquanto
  Return T

Pra implementar essa algoritmos, temos que detectar ciclo, encontrar aresta mais cara, etc.

Árvores Geradoras de Custo Mínimo

Baseado nesse lema, temos o seguinte algoritmo para encontrar AGM:

`Procedimento CalculaAGM(`G`)`
    $T \leftarrow G$.
    Enquanto existir um ciclo em T
        Escolha um ciclo em T e remova de T a aresta de maior custo do ciclo
    Fim Enquanto
    Return T

Pra implementar essa algoritmos, temos que detectar ciclo, encontrar aresta mais cara, etc.

Tem algoritmos mas simples: Kruskal, Prim, Boruvka

Árvores Geradoras de Custo Mínimo

Theorem

Para todo grafo conexeo G, o algoritmo $CalculaAGM(G)$ obtém uma árvore geradora de custo mínimo para G.

Vamos provar por indução no número de arestas que o algoritmo funciona corretamente para todo grafo G com n vértices.

Caso Base. G tem $n-1$ arestas. Como G é conexo, nos provaramos que G é uma árvore (exercicio). Como a única AGM para G é o próprio G, o algoritmo age corretamente.

Passo Indutivo.

Nos queremos provar que se o procedimento CalculaAGM obtém a AGM para todo grafo conexo G com k arestas, então CalculaAGM também devolve a AGM para todo grafo conexo G' com $k+1$ arestas, para todo $k \geq n-1$.

Prova do Passo. Seja G' um grafo conexo com $k+1$ arestas e seja e a aresta removida na primeira iteração de CalculaAGM(G'). Devemos observar que o algoritmo CalculaAGM(G') devolve a mesma solução que CalculaAGM($G'-e$) já que ele executa os mesmos passos depois de remover a aresta e. Por hipótese de indução, CalculaAGM($G'-e$) devolve a AGM para $G'-e$. Como e é aresta mais cara de um ciclo de G' segue do lema anterior que a AGM para $G'-e$ é também uma AGM para G'. Portanto, CalculaAGM(G') devolve uma AGM para G'.

Exercícios

Exercício 1: Prove que em uma árvore T, existe no máximo um caminho entre cada par de nós u, v

Dica:

1) Suponha por contradição que tem mais de um caminho entre u e v

2) Deduza que existem um passeio frechado

3) Utilize o lema que provamos no começo da aula (passeio fechado $\Rightarrow$ ciclo) pra obter contradição

Exercícios

Exercício 2: Uma **folha** de uma árvore é um nó de grau 1. Prove que toda árvore T não vazia tem pelo menos uma folha

Dica: Uma proposição sobre existência de ciclos que provamos na última aula

Obs: Folhas são muito importantes para fazer prova por indução no número de nós de uma árvore, pois removendo uma folha obtemos uma árvore com 1 nó a menos