BCC204 - Teoria dos Grafos

Marco Antonio M. Carvalho

Departamento de Computação
Instituto de Ciências Exatas e Biológicas
Universidade Federal de Ouro Preto

Conteúdo

1. Árvores
2. Árvores Geradoras
3. O Problema da Árvore Geradora de Custo Mínimo
4. O Algoritmo de Prim
5. O Algoritmo de Kruskal
6. Algoritmo de Prim vs. Algoritmo de Kruskal
7. Aplicações

Fonte

Este material é baseado no livro

- Goldbarg, M., & Goldbarg, E. (2012). *Grafos: conceitos, algoritmos e aplicações*. Elsevier.

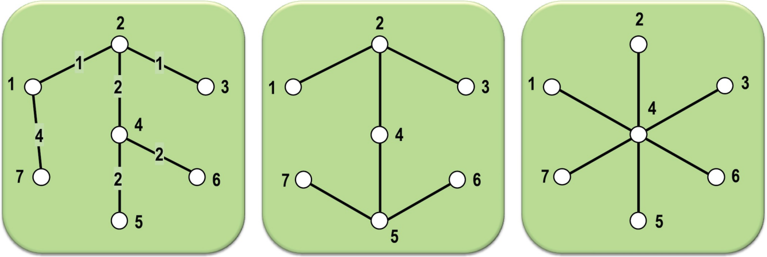
Licença

Árvores

Definição

Grafo conexo e sem ciclos em que há somente um caminho entre qualquer par de vértices.

Um subgrafo conexo e acíclico de uma árvore é denominado subárvore.

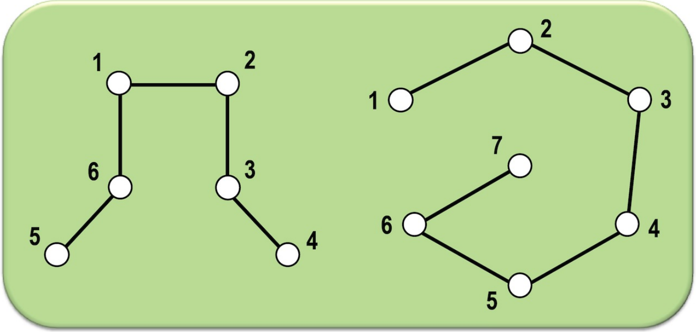


Árvore ponderada, árvore não ponderada e grafo estrela.

Grafo Caminho

Definição

Um grafo caminho (ou grafo linha) é um caso especial de árvore em que todos os vértices têm grau 2 ou 1, havendo apenas dois vértices com grau 1.



Grafos caminho.

Características

Seja T uma árvore com n vértices, então:

I. T é conexo e sem ciclos;
II. T possui $n - 1$ arestas;
III. Cada aresta de T é uma ponte;
IV. T é um grafo planar;
V. Se $n>1$, então T possui pelo menos dois vértices folhas (ou terminais).

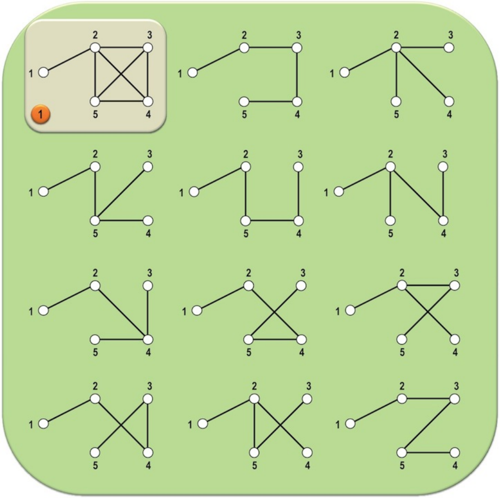
Definição

Todo grafo G conexo possui pelo menos uma árvore que contém todos os seus vértices.

Uma árvore geradora de um grafo G é um subgrafo conexo e acíclico que possui todos os vértices originais de G e um subconjunto das arestas originais de G.

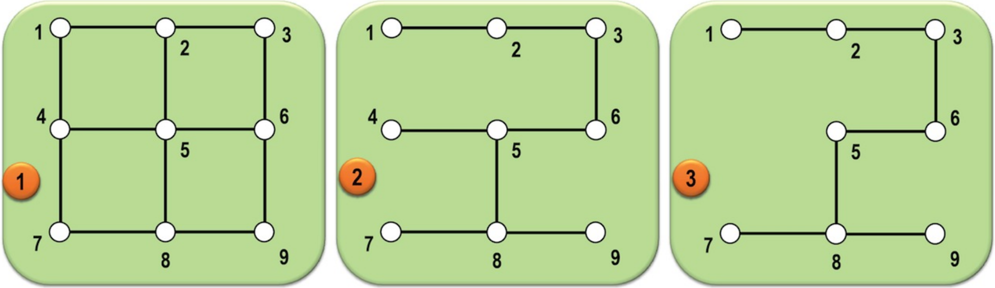
Como consequência das propriedades de uma árvore, todo grafo conexo possui pelo menos uma árvore geradora.

Árvores Geradoras



Grafo de exemplo e árvores geradoras.

Árvores Geradoras

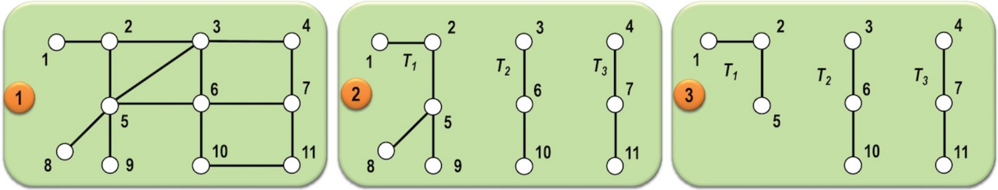


Grafo de exemplo, árvore geradora e uma árvore não geradora.

Definições

Uma floresta é um conjunto de árvores sem vértices em comum.

Uma floresta geradora é uma floresta que contém todos os vértices de um grafo.



Grafo de exemplo e florestas. A primeira floresta é geradora.

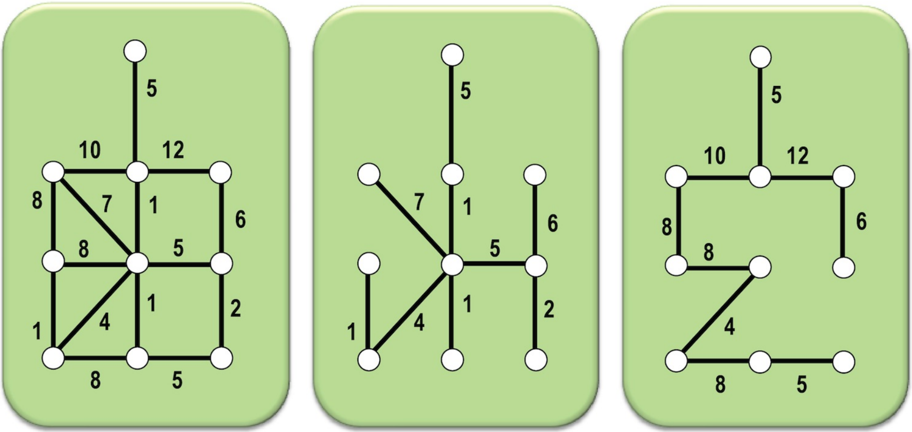
Definição

A árvore geradora de custo mínimo é a árvore geradora de menor custo dentre todas as possíveis em um grafo.

Analogamente, árvore geradora de custo máximo é a árvore geradora de maior custo dentre todas as possíveis em um grafo.

A determinação de ambas as árvores descritas pode ser feita em tempo **determinístico polinomial** por algoritmos gulosos.

Árvore Geradora de Custo Mínimo e Máximo



Grafo de exemplo, árvore geradora de custo mínimo e árvore geradora de custo máximo.

Evolução da Complexidade

Um dos primeiros algoritmos para determinação de árvores geradoras mínimas data do ano de 1928.

De lá para cá, a complexidade dos algoritmos evoluiu de $O(m \log n)$ para $O(m)$, cuja implementação data de 2008.

Os Básicos

Dois dos algoritmos mais populares para determinação de árvores geradoras mínimas, ambos gulosos, remetem ao final da década de 50: o algoritmo de **Prim** e o Algoritmo de **Kruskal**.

O Algoritmo de Prim

Histórico

Este algoritmo foi proposto originalmente em 1930 pelo matemático tcheco Vojtěch Jarník, posteriormente pelo cientista da computação americano Robert C. Prim ($\star$ 1921 † 2009) em 1957 e redescoberto posteriormente pelo holandês Edsger Dijkstra em 1959.

Princípio

Incluir, de forma gulosa, um a um, os vértices da árvore geradora mínima.

O algoritmo parte de qualquer vértice do grafo e, a cada passo, acrescenta a aresta de menor peso incidente ao conjunto de vértices que já foram selecionados e que possui uma extremidade em vértices no conjunto de não selecionados.

Algoritmo de Prim

Terminologia

- T_{min}: Conjunto de arestas que define a árvore geradora mínima;
- T: Conjunto dos vértices já selecionados pelo algoritmo;
- N: Conjunto dos vértices não selecionados pelo algoritmo;
- $\setminus$: subtração em conjuntos.

Algoritmo de Prim

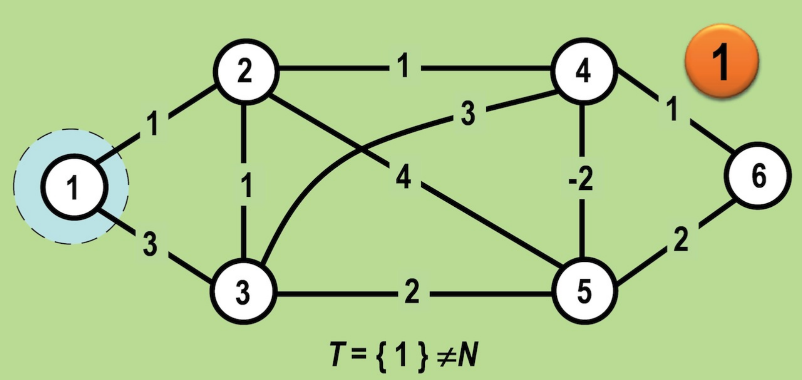
Entrada: Grafo $G = (V, A)$ e matriz de pesos D=$\{d_{ij}\}$ para todas as arestas $\{i, j\}$

Escolha qualquer vértice $i \in V$;
$T \leftarrow \{i\}$;
$N \leftarrow V \setminus i$;
$T_{min} \leftarrow \emptyset$;
enquanto $|T| \neq n$ **faça**
    Encontre a aresta $\{j, k\} \in A$ tal que $j \in T$, $k \in N$ e d_{jk} é mínimo;
    $T \leftarrow T \cup \{k\}$;
    $N \leftarrow N \setminus \{k\}$;
    $T_{min} \leftarrow T_{min} \cup \{j, k\}$;
fim

Algoritmo de Prim

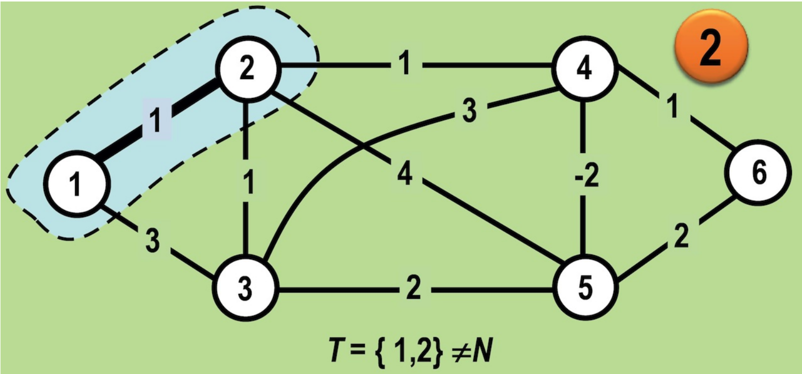
Complexidade

- Utilizando-se uma matriz de adjacências e uma busca linear na mesma, a complexidade é $O(n^2)$, por conta da aplicação repetidas vezes do procedimento que encontra a aresta de peso mínimo;
- Usando heaps binárias, o algoritmo pode ser implementado em $O(m \log n)$;
- Usando heaps de Fibonacci, o algoritmo pode ser implementado em $O(n \log n + m)$.



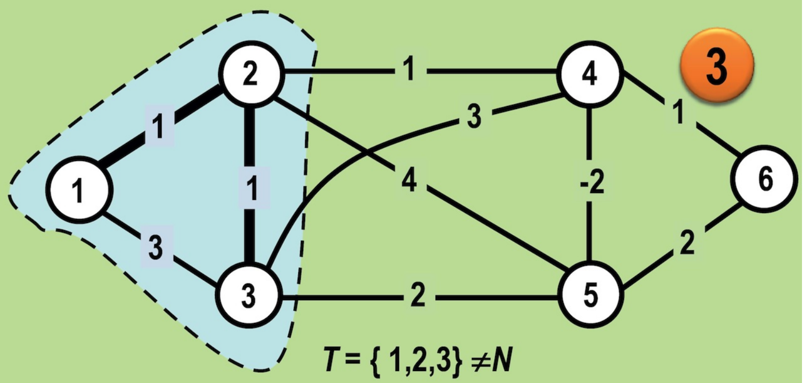
Grafo de exemplo. O vértice 1 é o primeiro a ser escolhido.

Exemplo



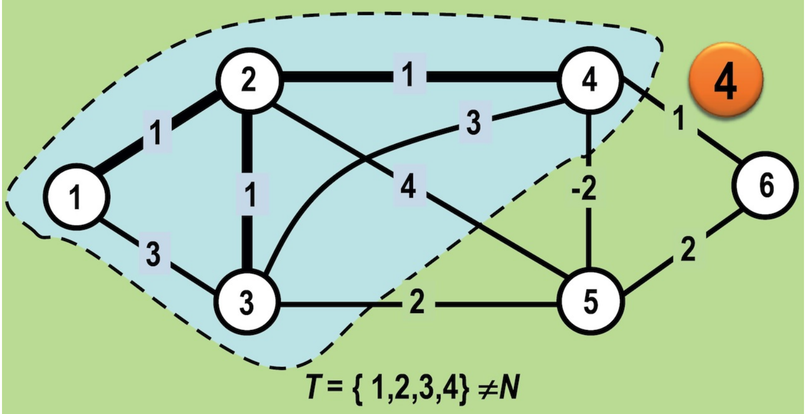
Inserção do vértice 2 e da aresta {1, 2}.
A região em azul indica os vértices escolhidos.

Exemplo



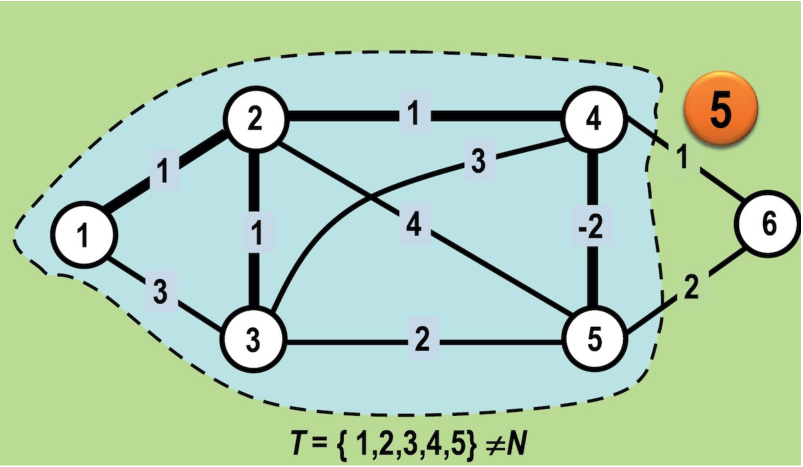
Inserção do vértice 3 e da aresta {2, 3}.

Exemplo



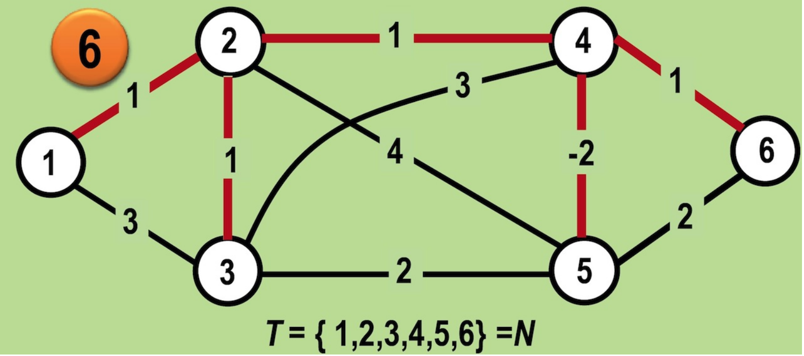
Inserção do vértice 4 e da aresta {2, 4}.

Exemplo



Inserção do vértice 5 e da aresta {4, 5}.

Exemplo



Inserção do vértice 6 e da aresta {4, 6}.
A árvore geradora mínima foi determinada.

O Algoritmo de Kruskal

Histórico

Este algoritmo foi proposto em 1956 por Joseph Bernard Kruskal Jr. ($\star$ 1928 $\dagger$ 2010), estatístico, matemático, cientista da computação e psicometrista americano.

Princípio

Incluir na árvore, a cada iteração, a aresta de menor custo que não formar ciclo.

Consequemente, processar $n - 1$ iterações;

O raciocínio está voltado para a formação da árvore a partir da inclusão de arestas, e não de vértices, como no algoritmo de Prim.

Algoritmo de Kruskal

Terminologia

- H: Vetor de arestas, ordenadas de acordo com os pesos;
- T: Conjunto de arestas que define a árvore geradora mínima;
- $\cup$: união em conjuntos.

Entrada: Grafo $G = (V, A)$ e matriz de pesos D=$\{d_{ij}\}$ para todas as arestas $\{i, j\}$

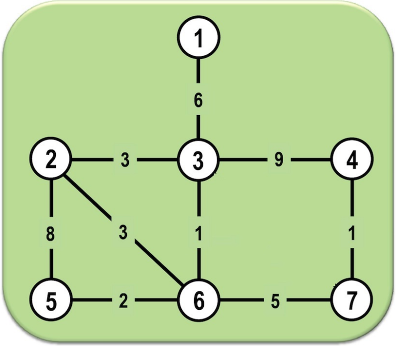
Ordene as arestas em ordem não decrescente de pesos d_{ij} no vetor H;
$T \leftarrow h_1$;
$i \leftarrow 2$;
enquanto $j < n - 1$ **faça**
    se $T \cup h_i$ *é um grafo acíclico* **então**
        $T \leftarrow T \cup h_i$;
        $j \leftarrow j + 1$;
    fim
    $i \leftarrow i + 1$;
fim

Algoritmo de Kruskal

Complexidade

- A ordenação das arestas pode ser feita em $O(m \log m)$;
- A escolha das arestas é realizada $O(m)$ vezes;
- A verificação se o grafo é acíclico exige complexidade $O(m)$;
- Logo, em problemas sem características particulares, a complexidade é $O(m \log m)$.

Exemplo

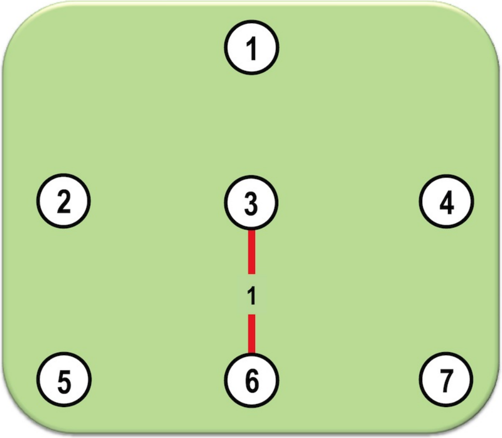


Vetor *H*

h_1 - (3,6) = 1; h_2 - (4,7) = 1; h_3 - (5,6) = 2;
h_4 - (2,3) = 3; h_5 - (2,6) = 3; h_6 - (6,7) = 5
h_7 - (1,3) = 6; h_8 - (2,5) = 8; h_9 - (3,4) = 9

Grafo de exemplo e vetor *H* desordenado.

Exemplo



Vetor *H*

h_1 - (3,6) = 1

h_2 - (4,7) = 1

h_3 - (5,6) = 2

h_4 - (2,3) = 3

h_6 - (6,7) = 5

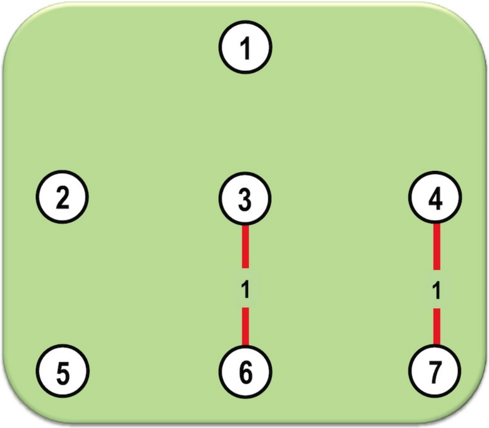
h_7 - (1,3) = 6

h_8 - (2,5) = 8

h_9 - (3,4) = 9

Inserção da primeira aresta em T.

Exemplo



Vetor H

h_1 - (3,6) = 1

h_2 - (4,7) = 1

h_3 - (5,6) = 2

h_4 - (2,3) = 3

h_5 - (2,6) = 3

h_6 - (6,7) = 5

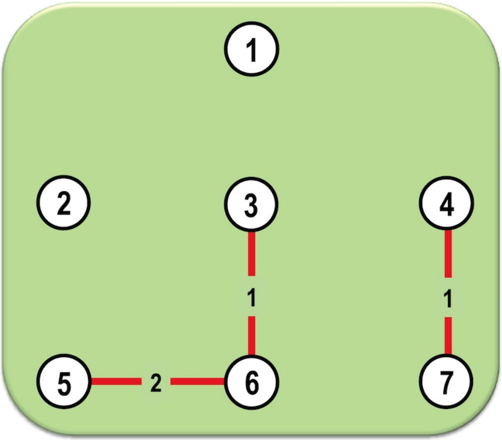
h_7 - (1,3) = 6

h_8 - (2,5) = 8

h_9 - (3,4) = 9

Inserção da segunda aresta em T.

Exemplo

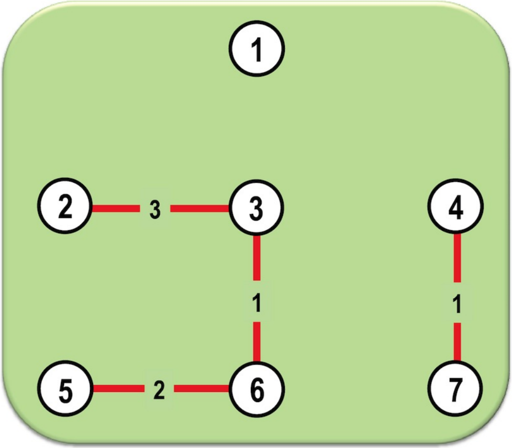


Vetor H

h_1 - (3,6) = 1
h_2 - (4,7) = 1
h_3 - (5,6) = 2
h_4 - (2,3) = 3
h_5 - (2,6) = 3
h_6 - (6,7) = 5
h_7 - (1,3) = 6
h_8 - (2,5) = 8
h_9 - (3,4) = 9

Inserção da terceira aresta em T.

Exemplo

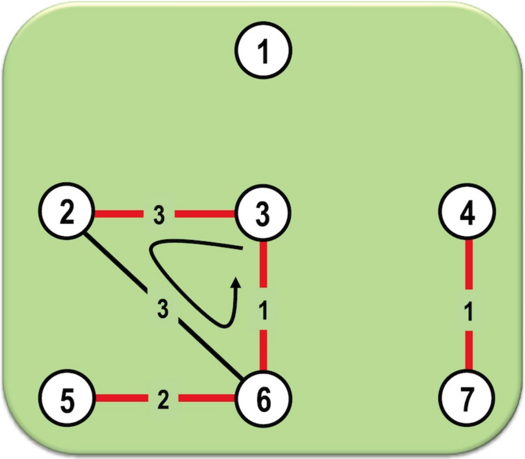


Vetor *H*

h_1 - (3,6) = 1
h_2 - (4,7) = 1
h_3 - (5,6) = 2
h_4 - (2,3) = 3
h_5 - (2,6) = 3
h_6 - (6,7) = 5
h_7 - (1,3) = 6
h_8 - (2,5) = 8
h_9 - (3,4) = 9

Inserção da quarta aresta em T.

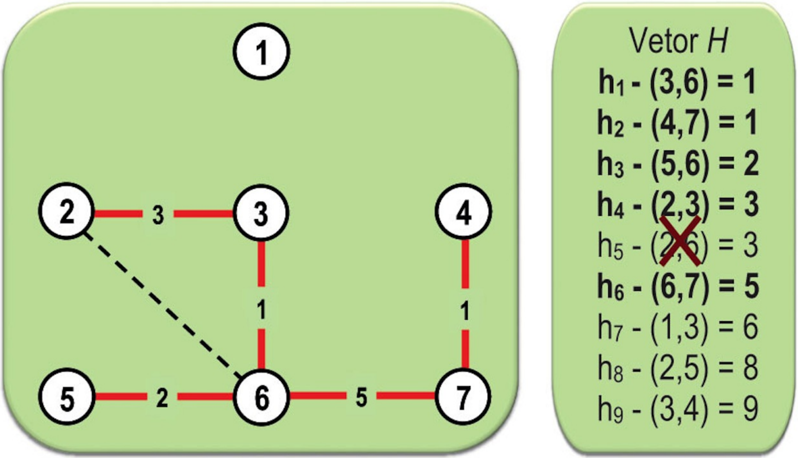
Exemplo



Vetor H

h_1 - (3,6) = 1
h_2 - (4,7) = 1
h_3 - (5,6) = 2
h_4 - (2,3) = 3
h_5 - (2,6) = 3
h_6 - (6,7) = 5
h_7 - (1,3) = 6
h_8 - (2,5) = 8
h_9 - (3,4) = 9

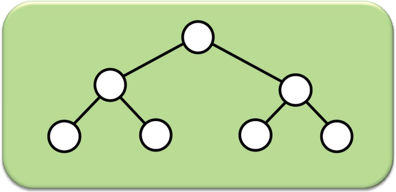
Tentativa de inserção da quinta aresta em T.

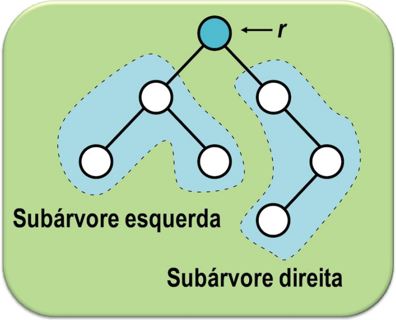


Inserção da quinta aresta em T.

Árvore Binária

Uma árvore binária possui um vértice especial chamado de raiz e os demais vértices são divididos em dois subconjuntos disjuntos: as subárvores esquerda e direita da raiz, que por sua vez, também são árvores binárias.





Exemplos de árvore binária e subárvores.

Aplicações reais

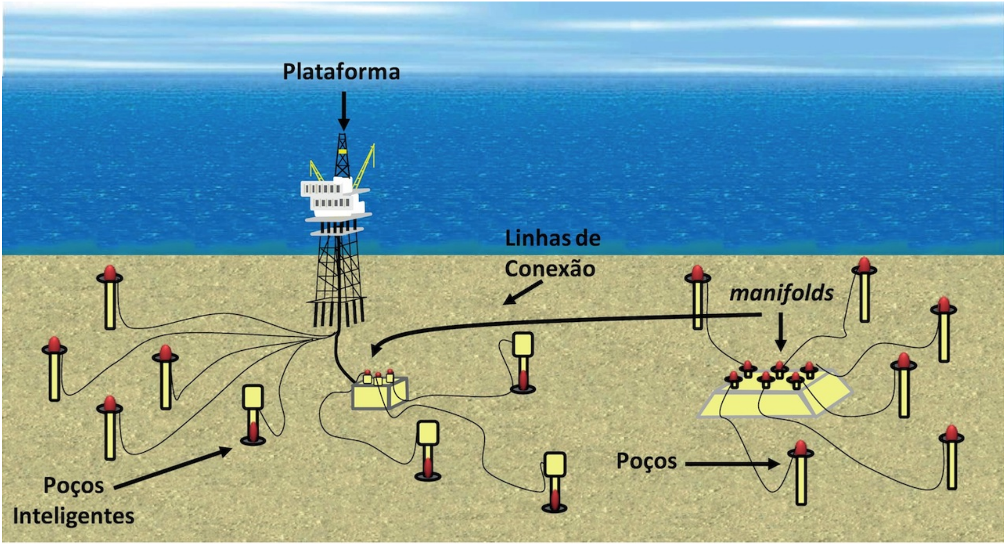
Otimização de Sistemas Submersos *Off-Shore*

Campos de petróleo *Off-Shore* são sistemas extremamente complexos que visam coletar e direcionar o óleo dos poços em solo submarino até um ponto de transporte.

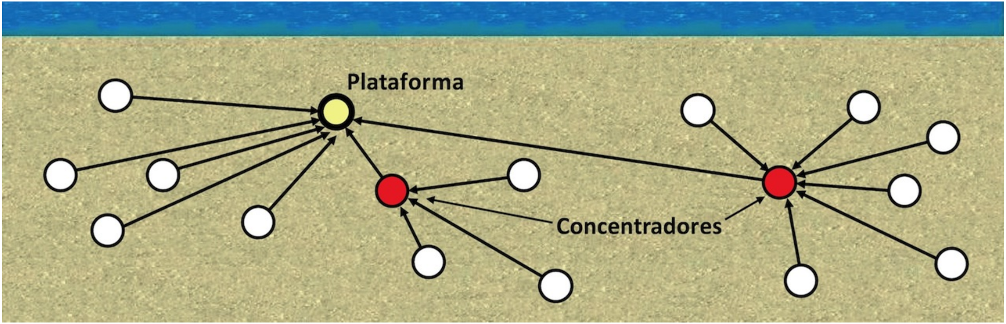
Os concentradores têm por função reunir a produção de diversos poços, de modo a minimizar o número de tubulações que acessam a plataforma e normalizar o fluxo de óleo, entre outros.

Um significativo problema de otimização é determinado pela necessidade de definir a localização e a capacidade destes concentradores.

Aplicações reais



Aplicações reais



Modelo em grafos associado à solução do esquema da figura anterior.
Vértices brancos representam o posicionamento dos poços e o vértice amarelo indica a plataforma.
Vértices vermelhos representam as posições viáveis para alocação dos concentradores.

Aplicações reais

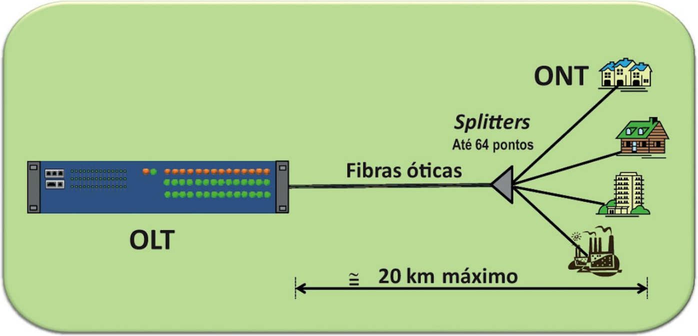
Redes Ópticas

Em um sistema de transmissão de uma rede óptica passiva, o sinal óptico é transmitido por uma rede de distribuição. Na fibra óptica são feitas derivações através do uso de *splitters* (divisores ópticos passivos).

Estas redes podem atingir um tamanho razoável e envolver custos significativos, tanto de implementação quanto de operação.

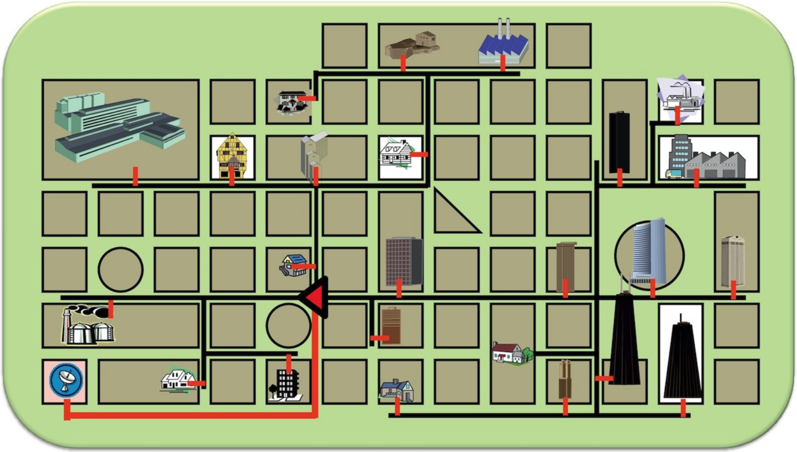
As figuras a seguir exibem a configuração de distribuição de uma rede óptica a partir dos *splitters*.

Conhecidas as posições dos *splitters*, a otimização da rede pode constituir uma árvore geradora sobre os pontos de demanda.



Arquitetura genérica de uma rede óptica.

Aplicações reais



Exemplo de rede de distribuição de fibra óptica.

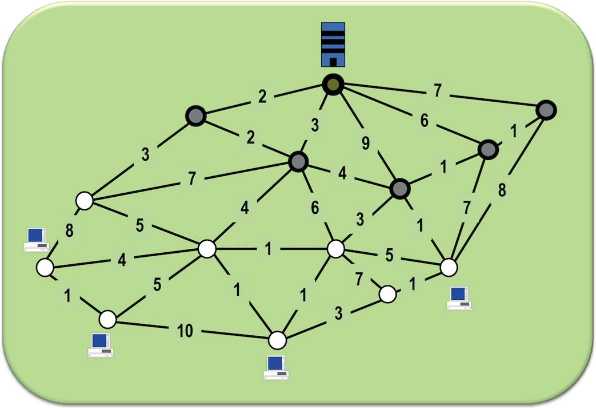
Otimização de Distribuição de Sinal em Redes

Um sinal é gerado em um ponto da rede, transita codificado até pontos de decodificação e é distribuído aos usuários.

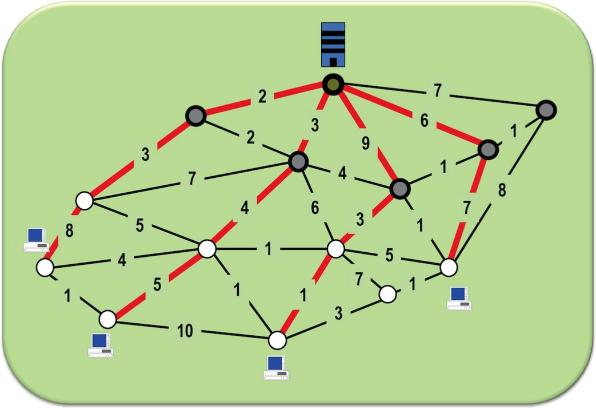
O serviço de decodificação tem custos diferentes em cada ponto, em virtude de características de demanda e operação nestes pontos.

Diferentes configurações de distribuição têm seu custo calculado através do custo do caminho percorrido mais o custo do serviço de decodificação.

Este problema pode ser modelado como uma árvore de *Steiner*.

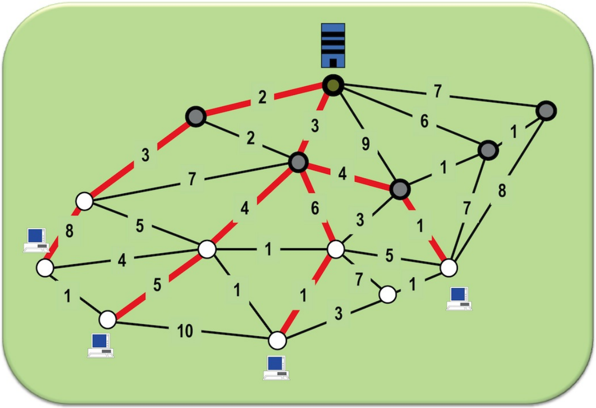


Rede com custos de distribuição nas arestas. Vértices da cor cinza indicam pontos de decodificação.



Solução com custo de 73 unidades.

Aplicações reais



Solução com custo de 48 unidades.

Outras Aplicações

- Projeto de redes de computadores e de comunicação;
- Instalações telefônicas, hidráulicas, elétricas, de petróleo e gás;
- Análise de agrupamentos;
- Análise genética;
- Análise de padrões de distribuição espacial de esporos;
- Astronomia (determinação de agrupamento de quasars);
- Geração de limites de problemas NP-Difíceis;
- Computação móvel;
- Modelos de localização de interação de partículas em fluxo turbulento de fluidos;
- etc.

Dúvidas?