Complementos de Matemática I
Matemática Discreta

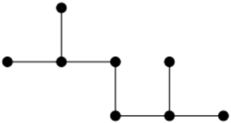
Depto de Matemática
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
UNR

2025

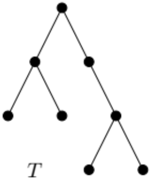
DEFINICIÓN

Un árbol es un grafo conexo sin ciclos. Notamos $T = (V, E)$. Cuando cada componente de un grafo es un árbol, se llama bosque.

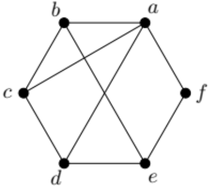
Un árbol es recubridor de un grafo G si es un subgrafo acíclico conexo (árbol) que contiene todos los vértices de G. Similar para los bosques recubridores.



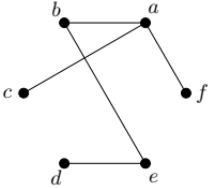
T



T



G



Árbol recubridor de G

Más ejemplos:

- P_n
- $K_{1,n}$

Todo árbol es bipartito. Vale la vuelta?

TEOREMA

En un árbol $T = (V, E)$ existe un único camino entre cualquier par de vértices distintos.

PROOF.

Hojitas □

TEOREMA

Dado $G = (V, E)$ un grafo no dirigido. G es conexo si y sólo si tiene un árbol recubridor.

PROOF.

Hojitas □

Teorema

En cualquier árbol $T = (V, E)$*,* $|V| = |E| + 1$*.*

Proof.

Hojitas □

Teorema

En cualquier árbol, si $|V| \geq 2$ *hay al menos dos vértices pendientes.*

Proof.

Hojitas □

Teorema

Los siguientes enunciados son equivalentes para un grafo $G = (V, E)$ sin bucles:

a) *G es un árbol*
b) *G es conexo y el borrado de cualquier arista lo desconecta en dos subgrafos que son árboles*
c) *G acíclico y $|V| = |E| + 1$*
d) *G conexo y $|V| = |E| + 1$*
e) *G es acíclico y si $a, b \in V$, $ab \notin E$, el grafo que se obtiene al agregar la arista ab a G tiene exactamente un ciclo*

Proof.

Hojitas $\square$

Corolario

Toda arista de un árbol es una arista de corte.

Ejercicio

Lema

Si T y T' son dos árboles recubridores en un grafo (conexo) G y $e \in E(T) - E(T')$ entonces existe $e' \in E(T') - E(T)$ tal que $(T \setminus e) \cup e'$ es un árbol recubridor.

Proof.

Hojitas $\square$

Lema

Si T y T' son dos árboles recubridores en un grafo (conexo) G y $e \in E(T) - E(T')$ entonces existe $e' \in E(T') - E(T)$ tal que $(T \cup e) \setminus e'$ es un árbol recubridor.

Ejercicio

ARBOLES RECUBRIDORES: BFS Y DFS

ALGORITMO BFS

Entrada: G grafo conexo y $u \in V(G)$.
Salida: Un árbol recubridor de T de G con una función predecesor p, y una función distancia $d(u,v)$ para todo $v \in V(G)$.

1: $Q := (u)$, $R = \{u\}$, $d(u,u) = 0$
2: **mientras** $R \neq V(G)$:
3: considerar x el primer vértice de Q
4: **mientras** existe $y \in N_G(x) \setminus R$ **hacer**:
5: agregar y atrás en Q
$R := R \cup \{y\}$
$p(y) = x$
$d(u,y) = d(u,x) + 1$
6: **fin mientras**
7: quitar x de Q
8: **fin mientras**
9: mostrar (p,d)

ALGORITMO BFS

Entrada: G grafo conexo y un orden de sus vértices $v_1, \ldots, v_n$.
Salida: Un árbol recubridor de T de G con una función predecesor p, y una función distancia $d(v_1, v_i)$ para todo $i \in [n]$.

$Q := (v_1)$, $R = \{v_1\}$, $d(v_1, v_1) = 0$
mientras $R \neq V(G)$:
  considerar x el primer vértice de Q
  for $i = 2$ hasta n **hacer**
    si $v_i \in N_G(x) \setminus Q$ **hacer**:
    agregar v_i atrás en Q
    $R := R \cup \{v_i\}$
    $p(v_i) = x$
    $d(v_1, v_i) = d(v_1, x) + 1$
    fin si
  fin for
  quitar x de Q
fin mientras
mostrar (p, d)

Algoritmo DFS

Entrada: G grafo conexo y $u \in V(G)$.
Salida: Un árbol recubridor de T de G con una función predecesor p.

$Q := (u)$, $R = \{u\}$
mientras $R \neq V(G)$:
  considerar x el primer vértice de Q
  si existe $y \in N_G(x) \setminus R$ **hacer**:
  agregar y adelante (primero) en Q
  $R := R \cup \{y\}$
  $p(y) = x$
  si no
  quitar x de Q
  fin si
fin mientras
mostrar p

Algoritmo DFS

Entrada: G grafo conexo y un orden de sus vértices $v_1, \ldots, v_n$.
Salida: Un árbol recubridor de T de G con una función predecesor p.

Ejercicio: Escribir DFS respetando el orden de los vértices.

We now apply this algorithm to the graph $G = (V, E)$ shown in Fig. 12.21(a). Here the order for the vertices is alphabetic: $a, b, c, d, e, f, g, h, i, j$.

First we assign the vertex a to the variable v and initialize T as just the vertex a (the root). Going to step 2, we find that the vertex b is the first vertex such that $\{a, b\} \in E$ and b has not been visited earlier. So we attach edge $\{a, b\}$ to T, assign b to v, and return to step 2.

At $v = b$ we find that the first vertex (not visited earlier) that provides an edge for the spanning tree is d. Consequently, the edge $\{b, d\}$ is attached to T, d is assigned to v, and we again return to step 2.

This time, however, there is no new vertex that we can obtain from d, because vertices a and b have already been visited. So we go to step 3. But here the value of v is d, not a, and we go to step 4. Now we backtrack from d, assigning the vertex b to v, and then we return to step 2. At this time we see that the edge $\{b, e\}$ can be added to T.

Continuing the process, we attach the edges $\{e, f\}$ and $\{e, h\}$ next. But now the vertex h has been assigned to v, and we must backtrack from h to e to b to a. When v is assigned the vertex a this (second) time, the new edge $\{a, c\}$ is obtained. Then we proceed to attach the edges $\{c, g\}$, $\{g, i\}$, and $\{g, j\}$. At this point all of the vertices in G have been visited, and we backtrack from j to g to c to a. With $v = a$ once again we return to step 2 and from there to step 3, where the process terminates.

The resulting tree $T = (V, E_1)$ is shown in part (b) of Fig. 12.21. Part (c) of the figure shows the tree T' that results for the vertex ordering: $j, i, h, g, f, e, d, c, b, a$.

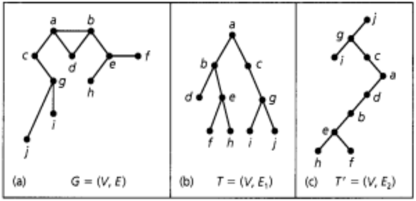
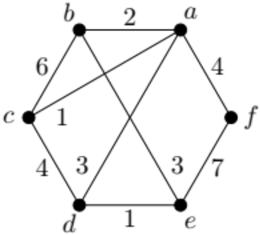


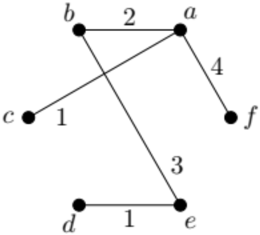
Figure 12.21

Árboles recubridores de costo óptimo (mínimo o máximo)

Definición: Un grafo ponderado es un grafo con etiquetas numéricas en sus aristas.

Ejemplo:





Veamos un algoritmo para hallar un árbol recubridor de peso mínimo en un grafo ponderado conexo.

Algoritmo de Kruskal

Algoritmo de Kruskal (para mínimo)

Entrada: Un grafo conexo G.

Idea: Mantener un subgrafo recubridor acíclico H e ir agregando aristas de peso mínimo.

Inicio: $E(H) = \emptyset$, $V(H) = V(G)$.

Iteración: Mientras existan aristas que unan dos componentes conexas de H, agregar la de mínimo peso a $E(H)$.

Algoritmo de Dijkstra (distancia mínima de un vértice a los restantes)

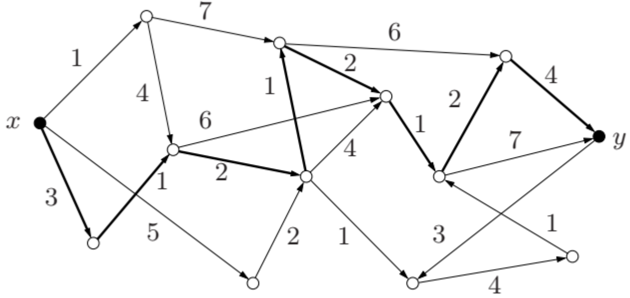
Algoritmo de Dijkstra

Entrada: Un grafo G ponderado con pesos no negativos y un vértice de inicio u. El peso de una arista xy es $\omega(xy)$ y si $xy \notin E(G)$, consideramos $\omega(xy) = \infty$.

Idea: Considerar un conj. S de vértices para los cuales hallamos un camino mínimo desde u, agrandando S hasta incluir todos los vértices. Tendremos una distancia arbitraria $t(z)$ desde u a cada $z \notin S$, hasta que la distancia mínima sea hallada.

Inicio: $S = \{u\}$, $t(u) = 0$, $t(z) = \omega(uz)\ \forall z \neq u$.

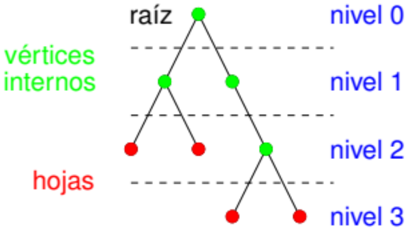
Iteración: Considerar $v \notin S$ con $t(v) = \min\{t(z) : \ z \notin S\}$.
Agregar v a S.
Explorar las aristas desde v y actualizar las etiquetas $t(z)$ para z vecino de v y $z \notin S$ con $t(z) = \min\{t(z), t(z) + \omega(vz)\}$.
Continuar la iteración hasta que $S = V(G)$ o hasta que $T(z) = \infty\ \forall z \notin S$.



x
y
1
7
6
2
4
4
1
2
6
1
7
1
4
2
3
1
5
2
1
3
1
4

Árboles binarios

Definición: Un árbol enraizado T es un árbol con un único vértice distinguido r, llamado raíz de T. Un árbol enraizado es binario si todo vértice tiene a lo sumo dos hijos. Si en particular todo vértice tiene grado 0 o 2 hijos el árbol es binario completo.



La altura del árbol enraizado es su máximo nivel.

TEOREMA

Si T es un árbol binario con i vértices internos entonces T tiene a lo sumo $i+1$ hojas.

Hojitas

TEOREMA

Si T es un árbol binario con altura H y l hojas entonces $log_2(l) \leq h$.

Ejercicio