Contenido

Contenido

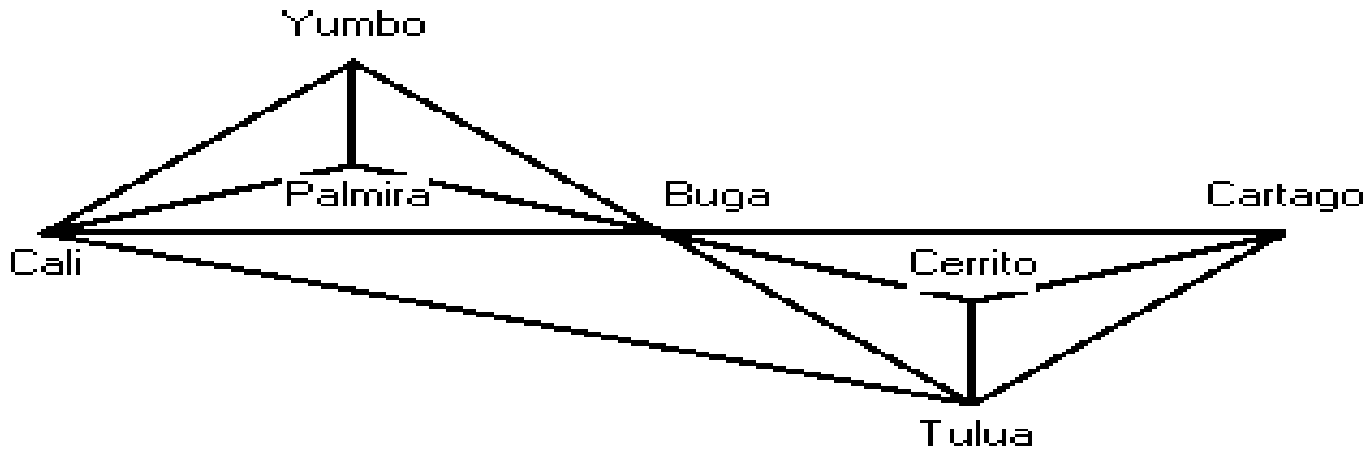

Figura: Ejemplo de un grafo

Problema: Que rutas pueden ser cerradas de tal manera que se pueda seguir viajando entre cualquier par de ciudades ?.

Árboles generadores (o recubridor)

Árboles generadores (o recubridor)

Sea *G* un grafo simple. Un árbol generador (o recubridor) de *G* es un subgrafo de *G* que es un árbol y contiene todos los vértice de *G*.

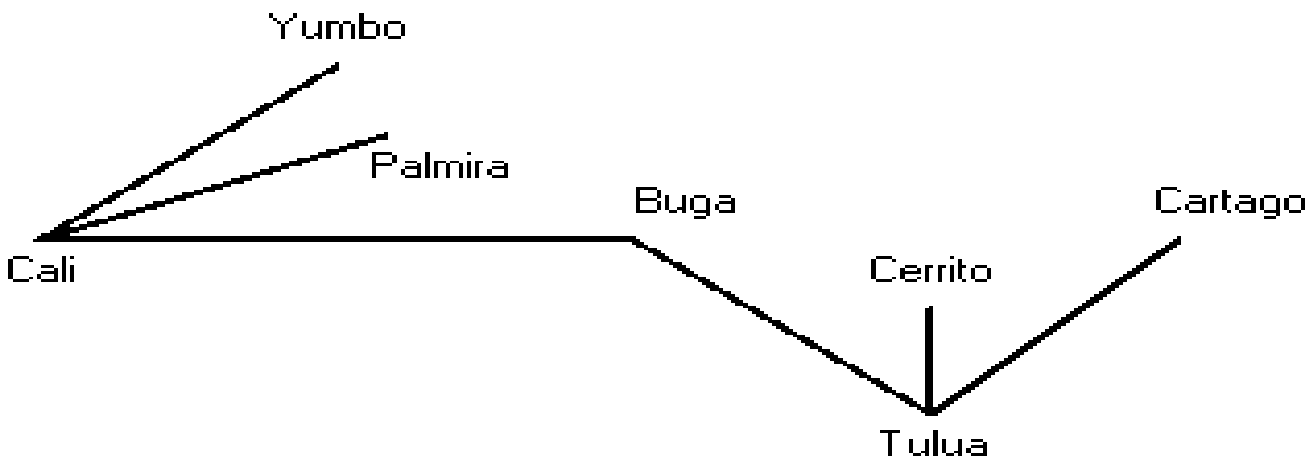

Figura: Ejemplo de un árbol generador

Ejemplo: El árbol anterior es un árbol generador del problema original.

Árboles generadores (2)

Nota: Un grafo simple con un árbol cobertor debe ser conexo, puesto que hay un camino en el árbol cobertor entre cualquier par de vértices.

Teorema 1

Un grafo simple es conexo si y solo si admite un árbol cobertor.

Los algoritmos mas usados para construir arboles generadores son: Búsqueda en profundidad y Búsqueda en amplitud.

Contenido

Algoritmo búsqueda por profundidad

Idea: Elegir un vértice arbitrario como raíz del árbol. Formar un camino que comienza en ese vértice añadiendo sucesivamente vértices y aristas, siendo cada nueva arista incidente con el último vértice del camino y un vértice que no está en el camino. Si el camino no pasa por todos los vértices del grafo, retroceder al penúltimo vértice e intentar un nuevo camino iniciando en él.

También llamado de vuelta atrás o retroceso (en inglés, *backtacking*) puesto que el algoritmo retorna a los vértices visitados previamente para adicionarlos al camino.

Algoritmo búsqueda por profundidad (2)

Procedimiento Profundidad (G: grafo conexo de vertices $v_1, v_2, \ldots, v_n$)
    T := arbol que consta solo del vertice v_1
    visitar(v_1)
Fin Procedimiento

Procedimiento visitar (v: vertice de G)
    Para cada vertice w adyacente a v y aun no en T
        Adicionar vertice w y arista $\{v, w\}$ a T
        visitar (w)
    Fin Para
Fin Procedimiento

Algoritmo búsqueda por profundidad (3)

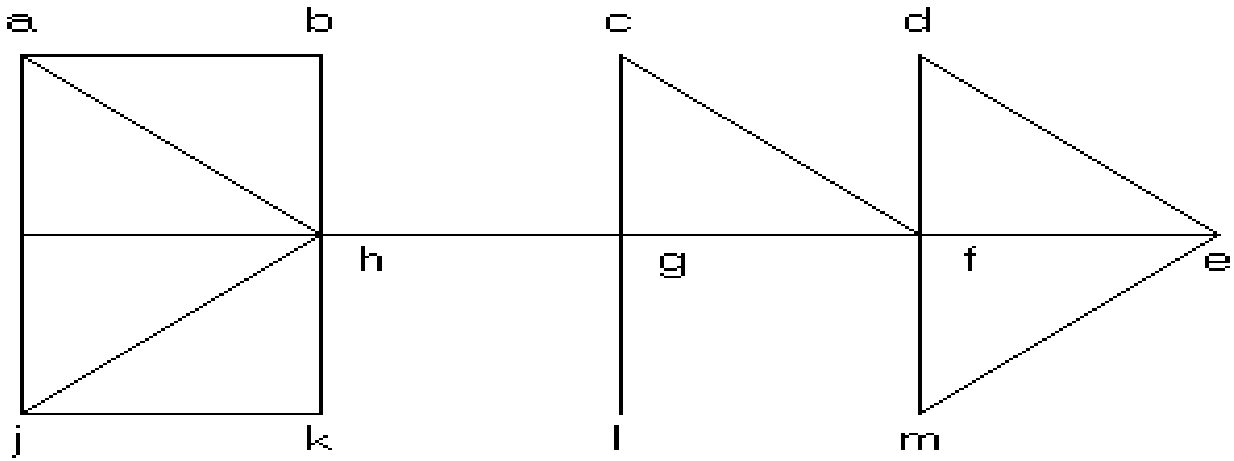

Figura: Ejemplo de un árbol generador

Ejercicio: Construir un árbol generador usando el algoritmo de búsqueda por profundidad, a partir del grafo anterior iniciando en el vertice *a*.

Contenido

Algoritmo búsqueda en anchura

Idea: Elegir un vértice arbitrario como raíz del árbol. Luego añadir todas las aristas incidentes con ese vértice (nivel 1 del árbol). Para cada vértice del nivel 1, visitados en orden, añadir todos los vertices incidentes con él siempre que no formen ciclos.
El procedimiento se repite hasta que se adicionen todos los vértices.

Algoritmo búsqueda en anchura (2)

Procedimiento Anchura (G: grafo conexo de vertices $v_1, v_2, \ldots, v_n$)

T := arbol que consta solo del vertice v_1

L := lista vacia

Mientras L no este vacia

borrar el primer vertice v de L

Para cada vertice w adyacente a v

Si w no esta en L y no esta en T

añadir el vertice w al final de la lista L

añadir el vertice w y la arista $\{v, w\}$ a T

Fin Si

Fin Para

Fin Mientras

Fin Procedimiento

Algoritmo búsqueda en anchura (3)

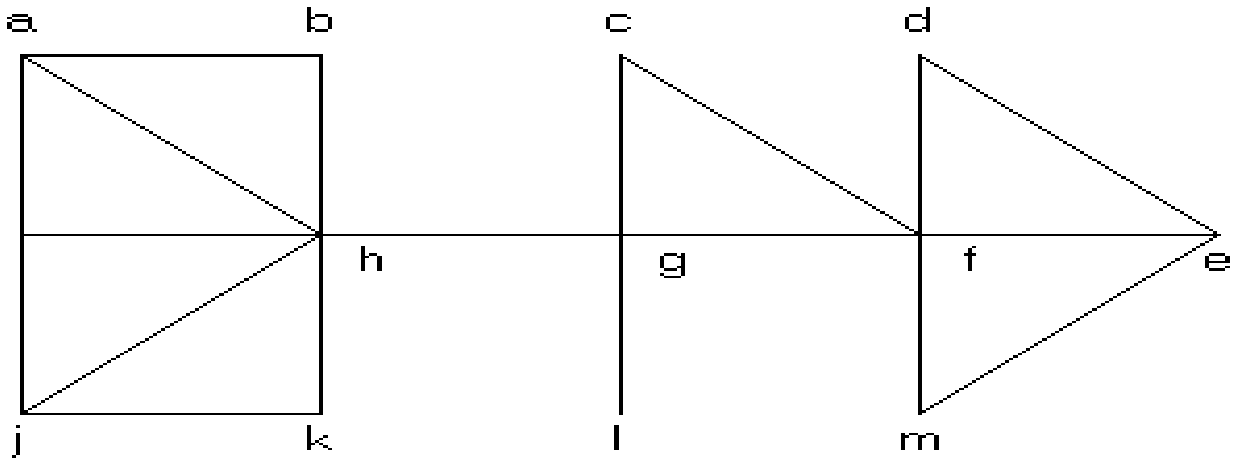

Figura: Ejemplo de un árbol generador

Ejercicio: Construir un árbol generador usando el algoritmo de búsqueda en anchura, a partir del grafo anterior iniciando en el vertice *a*.

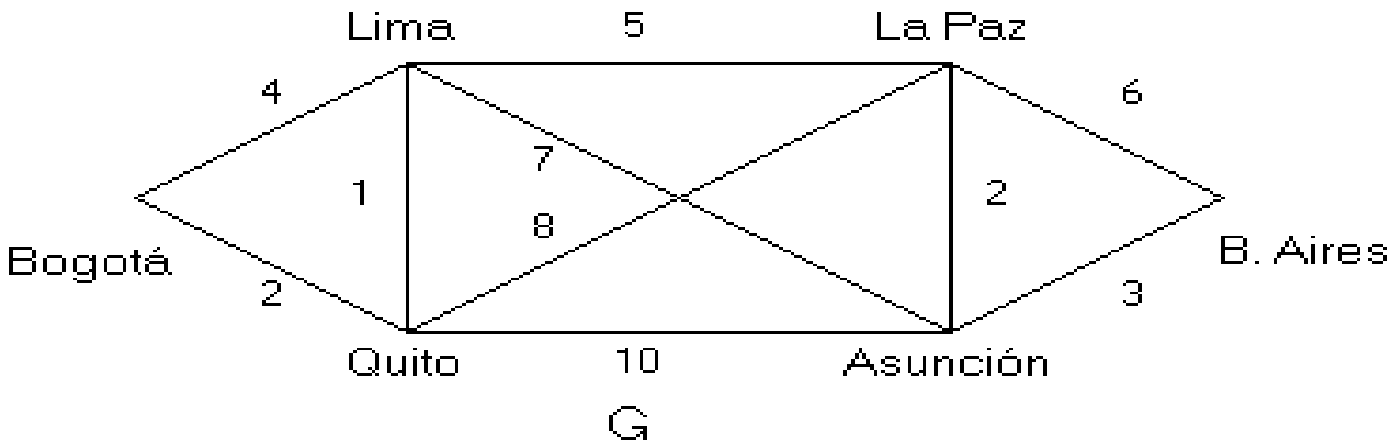

Figura: Ejemplo de un grafo ponderado

Problema: Que aristas deberían mantenerse, para asegurar que hay un camino entre cualquier par de ciudades y el costo total sea el menor posible ?.

Árboles generadores mínimos

Un árbol generador mínimo de un grafo ponderado es un árbol generador tal que la suma de los pesos de sus aristas es la mínima posible de entre todos los árboles generadores.

Existen dos algoritmos voraces (procedimientos que realizan una elección óptima en cada uno de sus pasos. La optimización de cada paso no garantiza una solución final optima) a saber:

- algoritmo de Prim
- algoritmo de Kruskal

Contenido

Algoritmo de Prim

Algoritmo propuesto por Robert Prim en 1957.
Idea: Elegir cualquier arista de peso mínimo y seleccionarla para el árbol (generador). Luego añadir sucesivamente aristas de peso mínimo que sean incidentes con un vértice que ya está en el árbol y que no formen ciclos con las que ya han sido incorporadas.
El algoritmo se ha de detener cuando hayamos añadido $n - 1$ aristas.

Algoritmo de Prim (2)

Procedimiento Prim (G: grafo ponderado conexo
y no dirigido de n vertices)
T := una arista de peso minimo
Para $i = 1$ **Hasta** $n - 2$
e := una arista de peso minimo incidente con
un vertice de T que no forme un ciclo en T
cuando se añada
T := T con e añadida
Fin Para
Fin Procedimiento (T es un arbol generador minimo de G)

Contenido

Algoritmo de Kruskal

Algoritmo descubierto por Joseph kruskal en en 1956.
Idea: Elegir cualquier arista de peso mínimo y seleccionarla para el árbol (generador). Luego añadir sucesivamente aristas de peso mínimo que no formen ciclos con las que ya han sido incorporadas.
El algoritmo se ha de detener cuando hayamos añadido $n - 1$ aristas.

Algoritmo de Kruskal (2)

Procedimiento Kruskal (G: grafo ponderado conexo
y no dirigido de n vertices)

T := grafo vacio

Para $i = 1$ **Hasta** $n - 1$

e := una arista de peso minimo de G que no forme un ciclo cuando se añada a T

T := T con e añadida

Fin Para

Fin Procedimiento (T es un arbol generador minimo de G)