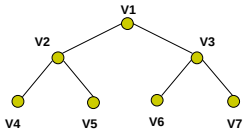
Matemáticas Discretas

Capítulo 6: Arboles

Introducción

- **Definición:**
  - **Un árbol (libre) T** es una gráfica que satisface:
    - Si v y w son vértices en T, entonces existe un único camino simple de v a w .
  - **Arbol con raiz** es un árbol en el cual un vértice particular se designa como la raíz.

Cont…



Cont…

- Un **árbol** es un grafo conexo y sin ciclos.

- **Propiedades:**
  - Si G = (V,A) es un árbol de n vértices, entonces:
    a) Para todo par de vértices x e y existe un único camino de x a y.
    b) Todas las aristas de G son puentes.
    c) $|A| = n - 1$.
    d) Todo árbol tiene al menos dos hojas (vértices de grado uno).

Cont…

- **Uso típico de los arboles:**
  - representación de estructuras jerárquicas
  - organizaciones
  - árbol genealógico
  - directorios
  - acceso rápido a datos ordenados en número desconocido
  - como vectores ordenados respecto a vectores normales
  - sistemas de ficheros avanzados
  - elementos de representación en juegos
  - sistemas de compresión

Cont…

- **La terminología**
  - **Hijo:** X es hijo de Y, sí y solo sí el nodo X es apuntado por Y. También se dice que X es descendiente directo de Y.
  - **Padre:** X es padre de Y sí y solo sí el nodo X apunta a Y. También se dice que X es antecesor de Y.
  - **Hermano:** Dos nodos serán hermanos si son descendientes directos de un mismo nodo.
  - **Hoja:** Se le llama hoja o terminal a aquellos nodos que no tienen ramificaciones (hijos).

Cont…

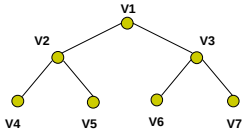
- **Cont...**
  - **Nodo Interior:** Es un nodo que no es raíz ni terminal.
  - **Grado:** Es el número de descendientes directos de un determinado nodo.
  - **Grado del Arbol:** Es el máximo grado de todos los nodos del árbol.
  - **Nivel de un vértice:** Es el número de arcos que deben ser recorridos para llegar a un determinado nodo. Por definición la raíz tiene nivel 1. Es la longitud del camino simple de la raíz al nodo.

Cont…

- **Cont...**
  - **Altura:** Es el número máximo de nivel, de todos los nodos, que aparece en el árbol.
  - **Peso:** Es el número de nodos del árbol sin contar la raíz.
  - **Longitud de Camino:** Es el número de arcos que deben ser recorridos para llegar desde la raíz al nodo X. Por definición la raíz tiene longitud de camino 1, y sus descendientes directos longitud de camino 2 y así sucesivamente.

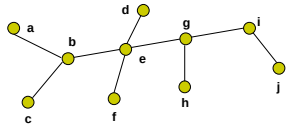
Cont...

- **Cont...**
  - **Ejemplo 1:**
    - Para el árbol dado los vértices v1, v2, v3, v4, v5,v6, v7 tienen los niveles 0, 1, 1, 2, 2, 2, 2
    - La altura del árbol es 2



Cont...

- **Cont...**
  - **Ejercicio en clases:**
    - Para el árbol T considere que la raíz es e, g, d



Cont…

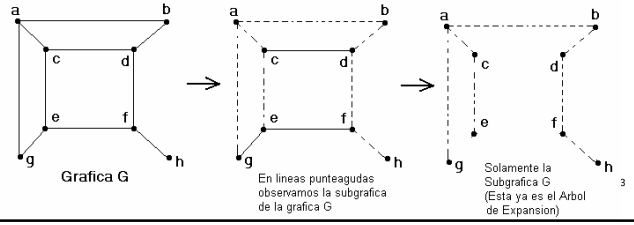
- **Terminología y Caracterizaciones**
  - **Definición:**
    - Sea T un árbol con raíz v_0. Suponga que x, y y z son vértices en T y que (v_0, v_1, ..., v_n) es un camino simple en T. Entonces:
      a) V_{n-1} es el padre de v_n
      b) v_0, v_1, ..., v_{n-1} son ancestros de v_n.
      c) V_n es un hijo de v_{n-1}.
      d) Si x es un ancestro de y, y es un descendiente de x.
      e) Si x y y son hijos de z, x y y son hermanos.
      f) Si x no tiene hijos, x es vértice terminal (o una hoja)
      g) Si x no es un vértice terminal, x es un vértice interno (o una rama).
      h) El subárbol de T con raíz en x es la gráfica con conjunto de vértices V y conjunto de aristas E, donde V es x junto con los descendientes de xy

Cont...

- **Cont...**
  - **Teorema:**
    - Sea T una gráfica con n vértices. Las siguientes afirmaciones son equivalentes:
      a) T es un árbol
      b) T es conexa y acíclica
      c) T es conexa y tiene n-1 aristas
      d) T es acíclica y tiene n-1 aristas

Arboles de expansión

- **Definición:**
  - Un árbol T es un árbol de expansión de **una gráfica G** si **T es una subgrafica de G** que contiene a **todos los vértices de G.**
    - Observemos la definición del siguiente modo: Sea una grafica G, encontramos una sub. grafica de G del tal modo que esta sub. grafica contenga todos los vértices de la grafica original, ósea de G, a esto se le llama árbol de expansión.



Arboles de expansión

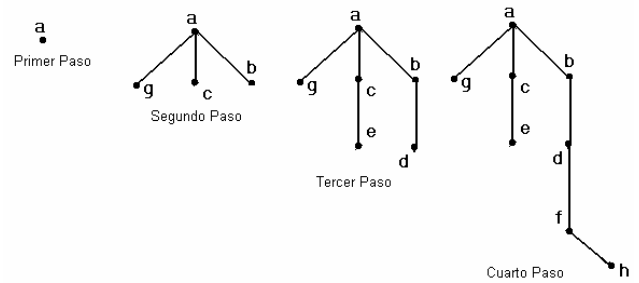
- **Teorema:**
  - Una gráfica G tiene un árbol de expansión si y sólo si G es conexa .
  - **Métodos Para Identificar los árboles de expansión**
    - Búsqueda a lo Ancho
    - Búsqueda a Profundidad ó Retroceso

Cont…

- **Búsqueda a lo Ancho**
  - La idea **es procesar todos los vértices de un nivel dado** antes de pasar al siguiente **nivel superior**.
  - Tomemos la grafica G del ejemplo:
    - Elegimos un orden (cualquiera) digamos abcdefgh, de los vértices de G
    - Elegimos el primer vértice **a** y lo etiquetamos como la raíz, siendo T la grafica formada por el único vértice **a** sin aristas.
    - Luego, agregamos a T todas las aristas (a,x) y los vértices sobre los cuales son incidentes, desde x = b hasta h, que no produzcan un ciclo al agregarse a T.
    - Repetimos este procedimiento con los vértices del nivel 1, examinándolos en orden.
      - b: Incluir (b,d)
      - c: Incluir (c,e)
      - g: Ninguno
    - Repetimos el procedimiento con los vértices del nivel 2:
      - d: Incluir (d,f)
      - e: Ninguno
    - Repetimos el procedimiento con los vértices del nivel 3:
      - f: Incluir (f,h)

Cont…

- **Cont...**

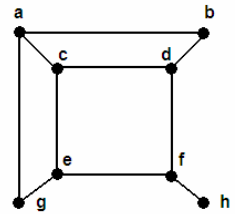


Cont…

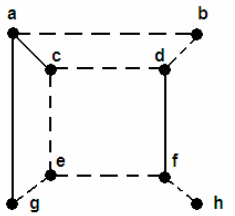
- **Búsqueda a Profundidad ó Retroceso**
  - Este método pasa a los niveles sucesivos de un árbol en la primera oportunidad posible.
    a) Elegimos el primer vértice **a** y lo llamamos la raíz
    b) Agregamos a nuestro árbol la arista (a, x) con x mínimo. En nuestro caso, agregamos la arista (a, b)
    c) Repetimos este proceso. Agregamos las aristas (b, d) (d, c) (c, e) (e, f) y (f, h)
    d) Aquí nos damos cuenta que no podemos agregar (h, x), así que retrocedemos al padre f de h e intentamos agregar una arista de la forma (f, x)
    e) Nuevamente, no podemos lograr esto, de modo que regresamos al padre e de f.
    f) Ahora podemos agregar la arista (e, g).
    g) En este momento no se pueden agregar mas aristas, de modo que finalmente retrocedemos hasta la raíz y el procedimiento concluye

Cont…

- **Cont...**



GRAFICA 1



GRAFICA 2

Cont...

- **Ejemplo:**
  - **Problema de las cuatro reinas**
    - El problema consiste en colocar 4 fichas en cuadriculas 4*4 de modo que no haya 2 fichas en el mismo renglón, la misma columna o en diagonal.
    - La idea es colocar las fichas de manera sucesiva en las columnas. Cuando no sea posible colocar las fichas en la una columna, retrocedemos y ajustamos la ficha de la columna anterior.
    - Entrada: Un arreglo row de tamaño 4.
    - Salida: true, si existe una solución
      false, si no existe solución

Cont…

- **Algoritmo.**

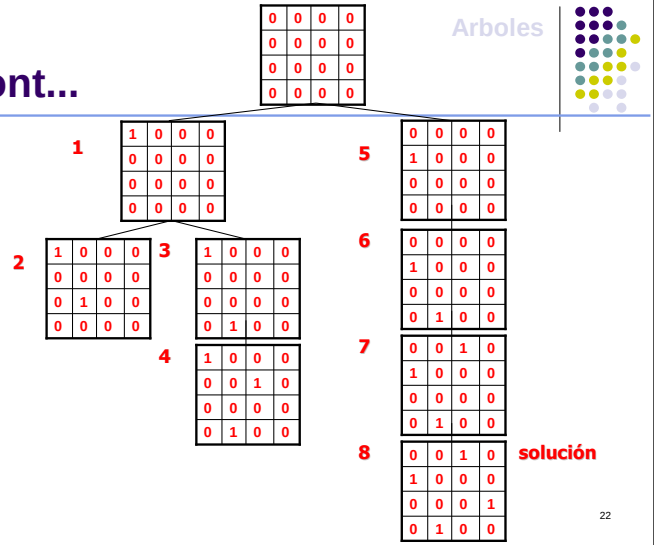
```
proceedure four_queens ( row )
   K := 1 // se comienza en la columna 1
   // row (k) se incrementa antes de utilizarse, de modo que comenzamos en
   el reglón 1
   row (1):= 0
   While k >0 do
    Begin
    Row (k): = row (k) +1
    // se busca un movimiento válido en la columna k
    While row  (k )  ≤ 4 and hay conflicto en  la columna k , renglón row (k) do
      // se intenta  con el siguiente renglón
      row (k) := row (k) +1
```

Cont…

- **Cont...**

```
If row (k) ≤4 then
    //se ha determinado un movimiento válido en la columna k
    if k = 4 then   //solución concluida
      return(true)
    else    //siguiente columna
      begin
      k : = k +1
      row (k) : =0
      end
  else   // se retrocede  a la columna anterior
     k : = k - 1
  End
 return(false)   // no existe solución
End four_ queens
```


Cont...

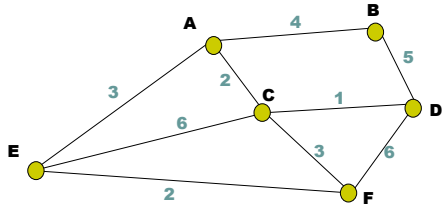


Arboles de Expansión Mínima

- **Definición:**
  - Sea G una gráfica con pesos. Un árbol de expansión mínima de G es un árbol de expansión de G con peso mínimo.
  - **Métodos de Identificación de árboles de Expansión Mínima**
    - Metodo de PRIM
    - Metodo de Kruskal

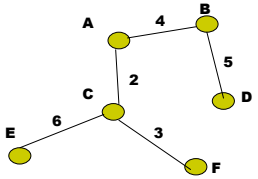
Cont...

- **Ejemplo:**
  - La gráfica con pesos G muestra seis ciudades y los costos de construcción de carreteras entre ciertos pares de ellas. Debemos construir el sistema de carreteras de menor costo .

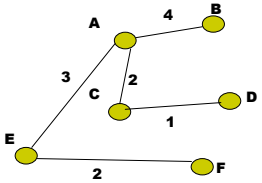


Cont...

- **Solución:**
  - La solución se puede representar por medio de una subgrafica que debe ser un árbol de expansión, pues debe contener a todos los vértices, debe ser conexa para llegar a una ciudad desde cualquier otra y debe tener un único camino simple entre cada par de vértices.



Un árbol de expansión de peso 20 para la grafica G



Un árbol de expansión de peso 12 para la grafica G

Cont...

- **Algoritmo de Prim**
  - Es utilizado para determinar un árbol de expansión mínimo, agregando aristas de peso mínimo que no formen un ciclo en el árbol en cuestión, esto se hace de manera iterativa.
    - Entrada: Una grafica conexa con pesos con vértices 1,....n y vértice inicial s. Cada arista (i,j) tiene un peso w(i,j), si no existe arista, entonces w(i,j) = inf.
    - Salida: Un conjunto de aristas E en un árbol de expansión mínimo.

Cont…

- **Algoritmo.**

```
procedure prim (w,n,s)
//v(i)=1 si el vértice se agregó al Árbol de expansión mínimo
//v(i)=0 si el vértice no se agregó
1. for i:=1 to n do
2.      v(i)=0     //todos los vertices no han sido agregados
3. v(s):=1 //se agrega el vértice inicial al árbol de exp. minim.
4. E:=Ø  //Se comienza con un conjunto de aristas vacío
   //Se colocan n-1 aristas en el árbol de expansión mínimo.
5. for i:=1 to n-1 do
6.    begin
        //se agrega la arista de peso mínimo que tenga un vértice en el
          a.e.m. y otro vértice que no este en el a.e.m
7.      min:=inf
```


Cont…

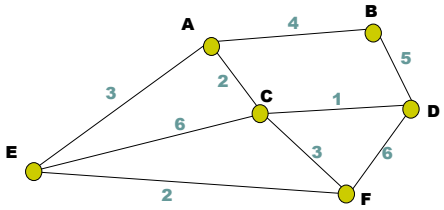
- **Algoritmo.**

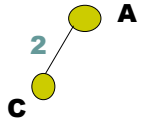
```
8.    for j:=1 to n do
9.      if v(j)=1 then //j es un vertice en el a.e.m.
10.       for k=1 to n do
11.         if v(k)=0 and w(j,k)<min then
12.             begin
13.               add_vertex:=k
14.               e:=(j,k)
15.               min:=w(j,k)
16.             end
      //se agregan el vertice y la arista al arbol de expansion minima
17.   v(add_vertex):=1
18.   E:=E U {e}
19.  end
20.  return(E)
21. end prim
```

Cont...

- Vértice Inicial A
- Las aristas con un vértice en el árbol y otro fuera del árbol son:
  - (A,B) 4
  - (A,C) 2
  - (A,E) 3

Escogemos la arista A,C pues tiene el menor peso y la agregamos a E.

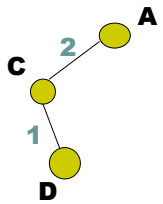




Cont...

- Ejecutamos nuevamente el ciclo for de las líneas 8-16, las aristas con un vértice en el árbol y otro fuera del árbol son:
  - (A,B) 4
  - (A,E) 3
  - (C,D) 1
  - (C,E) 6
  - (C,F) 3

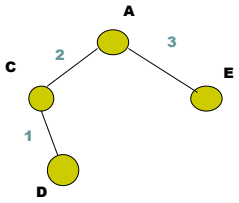
Escogemos la arista (C,D) y la agregamos a E



Cont...

- Ejecutamos el ciclo 8-16 y tenemos dos aristas con el mismo peso, sin importar la arista elegida obtendremos el a.e.m.
  - (A,B) 4
  - (A,E) 3
  - (D,B) 5
  - (C,E) 6
  - (C,F) 3
  - (D,F) 6

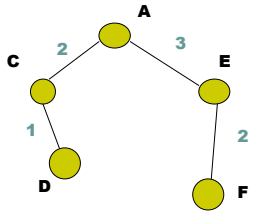
Escogemos la arista (A,E) y la agregamos a E



Cont...

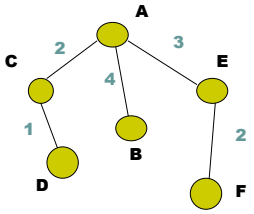
- Ejecutamos el ciclo 8-16 para escoger otra arista con peso mínimo y con un vértice en el a.e.m. y otro fuera.
  - (A,B) 4
  - (D,B) 5
  - (C,F) 3
  - (D,F) 6
  - (E,F) 2

Escogemos la arista (E,F) y la agregamos a E

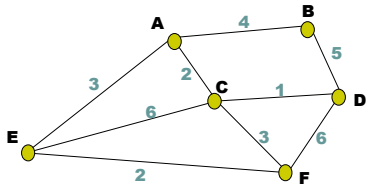


Cont...

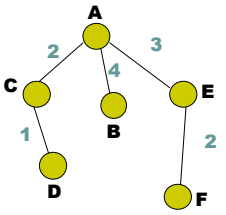
- Ejecutamos el ciclo 8-16 para escoger otra arista con peso mínimo y con un vértice en el a.e.m. y otro fuera.
- En las líneas 17 y 18 se agrega el vertice B al a.e.m y la arista (A,B) se agrega al conjunto E
  - (A,B) 4 ← **Escogemos la arista (A,B)**
  - (D,B) 5



Cont...



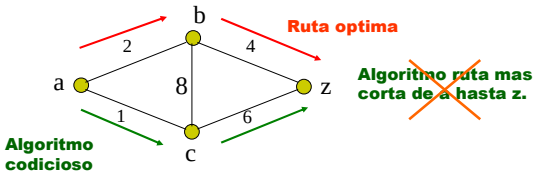
GRÁFICA G(V,E)



ÁRBOL DE EXPANSIÓN MINIMA DE G(V,E)

Cont...

- **Cont...**
  - El Algoritmo de Prim es un algoritmo codicioso, pues, optimiza la elección en cada iteración sin considerar las elecciones anteriores.
  - Un algoritmo codicioso no es necesariamente un algoritmo correcto (produce la solución adecuada u optima).
  - El algoritmo del camino mas corto es un algoritmo codicioso y no nos da una solución optima.



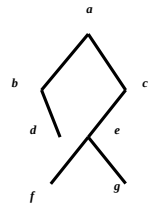
Árboles Binarios

- **Definición:**
  - Un árbol binario es un árbol con raíz en el cual cada vértice tiene cero, uno o dos hijos.
  - Si un vértice tiene un hijo, ese hijo se designa como un hijo izquierdo o un hijo derecho (pero no ambos).Si un vértice tiene dos hijos, uno de ellos se designa como un hijo izquierdo y el otro se designa como un hijo derecho.

Cont...

Ejemplo:

- En el árbol binario de la figura, el vértice *b* es el hijo izquierdo del vértice *a* y el vértice *c* es el hijo derecho del vértice *a*.
- El vértice *d* es el hijo derecho del vértice *b*; el vértice *b* no tiene hijo izquierdo.
- El vértice *e* es el hijo izquierdo del vértice *c*; el vértice *c* no tiene hijo derecho.



Cont…

- **Arbol Binario Completo**
  - Un árbol binario completo es un árbol binario en el cual cada vértice tiene dos o cero hijos.
  - **Teorema:**
    - Si T es un árbol binario completo con *i* vértices internos, entonces T tiene *i*+1 vértices terminales y 2*i*+1 vértices en total.

Cont…

- **Demostración:**
  - Existe un vértice que no es hijo de nadie: la raíz.
  - Existen *i* vértices internos, cada uno de los cuales tienen dos hijos, existen 2*i* hijos.
  - Así la cantidad total de vértices de T es 2*i*+1 y el número de vértices terminales es

$$(2i+1) - i = i + 1$$

Cont…

- **Teorema:**
  - Si un árbol binario de altura h tiene t vértices terminales, entonces

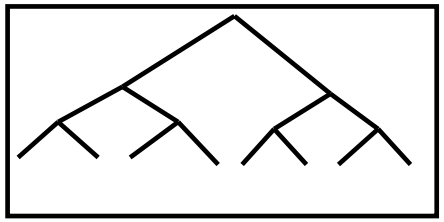
$$\lg_2 t \leq h$$

- **Ejemplo:**
  - El árbol binario de la siguiente figura tiene altura $h = 3$ y el número de vértices terminales es $t = 8$.
  - Para este árbol, la desigualdad del teorema anterior se convierte en una igualdad

Cont…

- Cont...

$$\lg_2 t = h$$



Cont…

- **Arbol de Búsqueda Binaria**
  - **Definición:**
    - Un árbol de búsqueda binaria es un árbol binario T en el cual se asocian ciertos datos con los vértices.
    - Los datos están ordenados de modo que, para cada vértice v en T, cada elemento de dato en el subarbol izquierdo de v sea menor que el elemento de dato en v y cada elemento de dato en el subarbol derecho de v es mayor que el elemento de dato en v.

Cont...

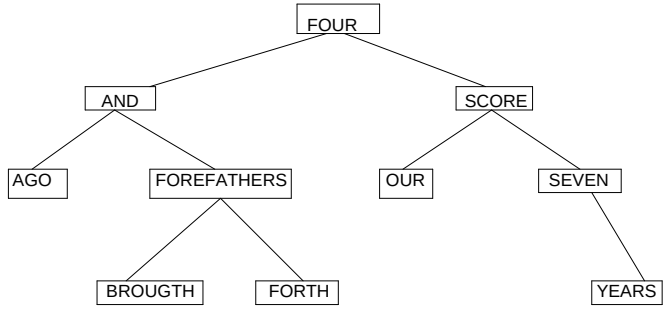
- **Construcciòn de un algoritmo de bùsqueda binaria**
  - Este algoritmo construye un árbol de búsqueda binaria. La entrada se lee en el orden con el cual fue enviada. Después de leer cada palabra, ésta se inserta en el árbol.
  - **Entrada**: Una sucesión $w_1,\ldots,w_n$ de palabras distintas y la longitud n de la sucesiòn.
  - **Salida**: Un árbol de búsqueda binaria T

```
procedure make_bin_search_tree(w,n)
Sea T el àrbol con un vèrtice, root
Guardar w1 en root
For i := 2 to n do
      begin
      v:= root
      search:= true //encontrar un lugar para w1
      while search do
          begin
          s:= palabra en v
          if wi < s then
```


Cont...

```
          if v no tiene hijo izquierdo then
              begin
              agregar un hijo izquierdo l a v        HIJO
              guardar wi en l                        IZQUIERDO
              search:= false //fin de la bùsqueda
              end
          else
              v:= hijo izquierdo de v
      else   //wi > s
          if v no tiene hijo derecho then
              begin
              agregar un hijo derecho r a v          HIJO
              guardar wi en r                        DERECHO
              search:= false //fin de bùsqueda
              end
          else
              v:= hijo derecho de v
      end   //mientras
  end    // para
return (T)
end make_bin_ search_ tree
```

Cont...



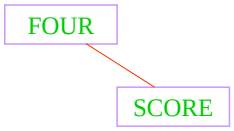
Cont...

- **Ejercicio:**
  - Coloque las palabras FOUR SCORE AND SEVEN YEARS AGO OUR FOREFATHERS BROUGHT FORTH, en orden de aparición, en un árbol de busca binaria.

Cont...

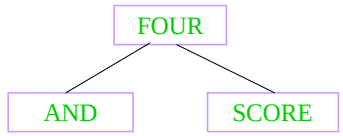
- **Solución**:

Root = FOUR
for i: = 2 **to** n **do**
v: = root =FOUR
search: = **true**
While search **do**
begin
s: =palabra en v
s : =FOUR =ROOT=V
W_2 = SCORE
if W_i < s (Score < FOUR) falso
else // W_i > s
si no tiene hijo derecho entonces
W_2 =Hijo derecho



Cont...

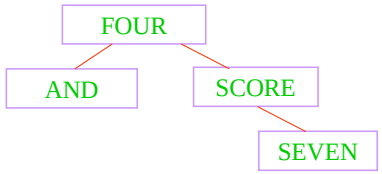
- Root = FOUR
for i: = 3 **to** n **do**
v: = root =FOUR
search: = **true**
while search **do**
begin
s: =palabra en v
s : =FOUR =ROOT=V
W_3 = AND
if W_i < s (AND < FOUR) verdadero
si no tiene hijo izquierdo
W_3 =hijo izquierdo



Cont...

- Root = FOUR
  for i: = 4 **to** n **do**
  v: = root =FOUR
  search: = **true**
  while search **do**
  begin
  s: =palabra en v
  s : =FOUR =ROOT=V
  W_4 = SEVEN
  if W_i < s (SEVEN < FOUR) falso
  else // W_i > s
  si no tiene hijo derecho agregamos
  pero como v si tiene hijo derecho entonces
  end
  else
  v: = hijo derecho
  v: = SCORE (Return to while)
  while search **do**
  begin

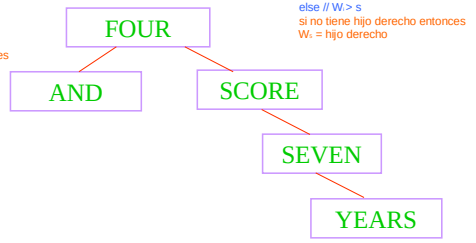
s: =palabra en v
s : = SCORE =ROOT=V
W_4 = SEVEN
if W_i < s (SEVEN < SCORE) falso
else // W_i > s
si no tiene hijo derecho entonces
W_4 = hijo derecho



Cont...

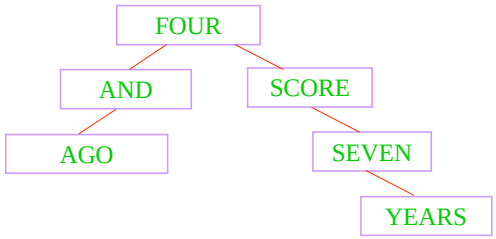
- Root = FOUR
  for i: = 5 **to** n **do**
  v: = root =FOUR
  search: = **true**
  while search **do**
  begin
  s: =palabra en v
  s : =FOUR =ROOT=V
  W_5 = YEARS
  if W_i < s (YEARS < FOUR) falso
  else // W_i > s
  si no tiene hijo derecho agregamos
  pero como v si tiene hijo derecho entonces
  end
  else
  v: = hijo derecho
  v: = SCORE (Return to while)
  while search **do**
  begin
  s: =palabra en v
  s : = SCORE =ROOT=V
  W_5 =YEARS
  if W_i < s (YEARS < SCORE) falso
  else // W_i > s
  si no tiene hijo derecho agregamos, pero
  como v si tiene hijo derecho entonces:
  end

else
v: = hijo derecho
v: = SEVEN (Return to while)
while search **do**
begin
s: =palabra en v
s : = SEVEN =ROOT=V
W_5 =YEARS
if W_i < s (YEARS < SEVEN) falso
else // W_i > s
si no tiene hijo derecho entonces
W_5 = hijo derecho



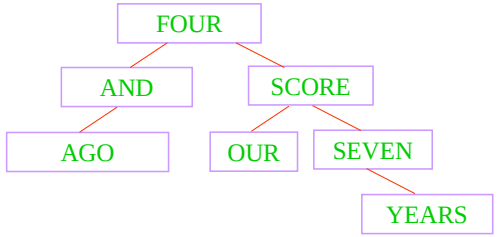
Cont...

- Root = FOUR
  for i: = 6 **to** n **do**
  v: = root =FOUR
  search: = **true**
  while search **do**
  begin
  s: =palabra en v
  s : =FOUR =ROOT=V
  W_6 = AGO
  if W_i < s (AGO < FOUR) verdadero
  si no tiene hijo izquierdo agregamos
  pero como v si tiene hijo izquierdo
  entonces
  end
  else
  v: = hijo izquierdo
  v: = AND (Return to while)
  while search **do**
  begin
  s: =palabra en v
  s : = AND =ROOT=V
  W_6 =AGO
  if W_i < s (AGO < AND) verdadero
  Si v no tiene hijo izquierdo entonces
  W_6 = hijo izquierdo



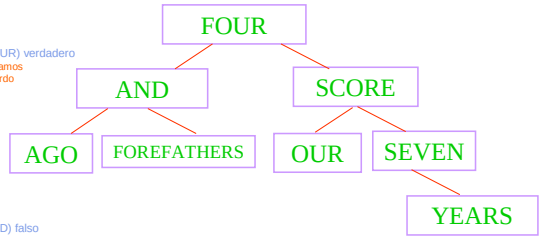
Cont...

- Root = FOUR
  for i: = 7 **to** n **do**
  v: = root =FOUR
  search: = **true**
  while search **do**
  begin
  s: =palabra en v
  s : =FOUR =ROOT=V
  W_2 = OUR
  if W_i < s (OUR < FOUR) falso
  else // W_7 > s
  si no tiene hijo derecho agregamos
  pero como v si tiene hijo derecho
  entonces
  end
  else
  v: = hijo derecho
  v: = SCORE (Return to while)
  while search **do**
  begin
  s: =palabra en v
  s : = SCORE =ROOT=V
  W_7 = OUR
  if W_i < s (OUR < SCORE) verdadero
  si no tiene hijo izquierdo entonces
  W_7 = hijo izquierdo



Cont...

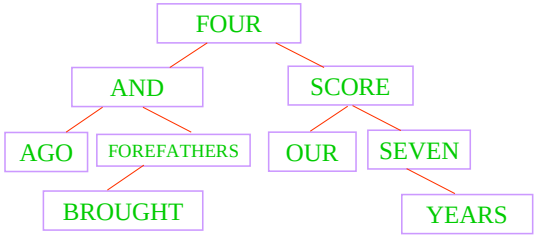
- Root = FOUR
  for i: = 8 **to** n **do**
  v: = root =FOUR
  search: = **true**
  while search **do**
  begin
  s: =palabra en v
  s : =FOUR =ROOT=V
  W_8 = FOREFATHERS
  if W_i< s (FOREFATHERS < FOUR) verdadero
  si no tiene hijo izquierdo agregamos
  pero como v si tiene hijo izquierdo
  entonces
  end
  else
  v: = hijo izquierdo
  v: = AND (Return to while)
  while search **do**
  begin
  s: =palabra en v
  s : = AND =ROOT=V
  W_8 = FOREFATHERS
  if W_i< s (FOREFATHERS < AND) falso
  else // W_i> s
  si no tiene hijo derecho entonces
  W_8 =Hijo derecho



Cont...

- Root = FOUR
  for i: = 9 **to** n **do**
  v: = root =FOUR
  search: = **true**
  while search **do**
  begin
  s: =palabra en v
  s : =FOUR =ROOT=V
  W_9 = BROUGHT
  if W_i< s (BROUGHT < FOUR) verdadero
  si no tiene hijo izquierdo agregamos
  pero como v si tiene hijo izquierdo entonces
  end
  else
  v: = hijo izquierdo
  v: = AND (Return to while)
  while search **do**
  begin

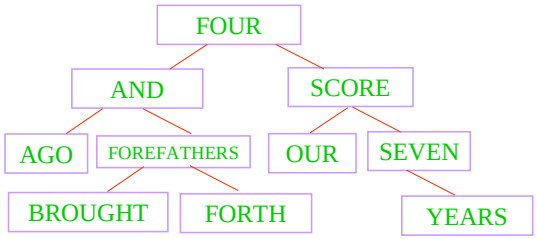
s: =palabra en v
s : = AND =ROOT=V
W_9 = BROUGHT
if W_i< s (BROUGHT < AND) falso
else // W_i> s
si no tiene hijo derecho agregamos
pero como v si tiene hijo derecho entonces
end
else
v: = hijo derecho
v: = FOREFATHERS (Return to while)
while search **do**
begin
s: =palabra en v
s : = FOREFATHERS =ROOT=V
W_9 = BROUGHT
if W_i< s (BROUGHT < FOREFATHERS) verdadero
si no tiene hijo izquierdo entonces
W_9= hijo izquierdo



Cont...

- Root = FOUR
  for i: = 10 **to** n **do**
  v: = root =FOUR
  search: = **true**
  while search **do**
  begin
  s: =palabra en v
  s : =FOUR =ROOT=V
  W_{10} = FORTH
  if W_i< s (FORTH < FOUR) verdadero
  si no tiene hijo izquierdo agregamos
  pero como v si tiene hijo izquierdo entonces
  end
  else
  v: = hijo izquierdo
  v: = AND (Return to while)
  while search **do**
  begin

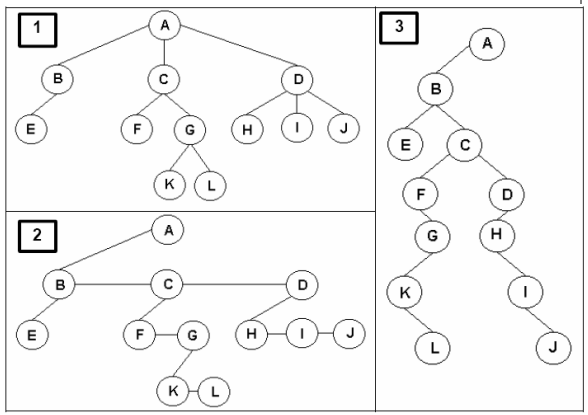
s: =palabra en v
s : = AND =ROOT=V
W_{10} = FORTH
if W_i< s (FORTH < AND) falso
else // W_i> s
si no tiene hijo derecho agregamos
pero como v si tiene hijo derecho entonces
end
else
v: = hijo derecho
v: = FOREFATHERS (Return to while)
while search **do**
begin
s: =palabra en v
s : = FOREFATHERS =ROOT=V
W_{10} = FORTH
if W_i< s (FORTH < FOREFATHERS) falso
else // W_i> s
si no tiene hijo derecho entonces
W_{10}= hijo derecho
END



Algoritmo para la Conversión de un Árbol General a un Árbol Binario

- Siendo A un árbol general y B el árbol binario, el algoritmo de conversión es el siguiente:
  1. La raíz de B es la raíz de A.
  2. Seguimos el siguiente proceso para la creación:
     a) Enlazar al nodo raíz con el hijo más izquierdo.
     b) Enlazar este nodo con los restantes descendientes del nodo raíz en un camino, con lo que se forma el nivel 1.
     c) A continuación, repetir los pasos a) y b) con los nodos del nivel 2, enlazando siempre en un mismo camino todos los hermanos – descendientes del mismo nodo –. Repetir estos pasos hasta llegar al nivel más alto.
  3. Girar el diagrama resultante 45 grados, para diferenciar entre los subárboles izquierdo y derecho.

Cont…



Recorrido de un Arbol

- La búsqueda a lo ancho y a profundidad proporcionan formas de “recorrer” un árbol, recorrerlo de forma sistemática de modo que cada vértice sea visitado exactamente una vez.
- Otros métodos para recorrer un árbol son:
  - Preorden
  - Entreorden
  - Posorden

Cont...

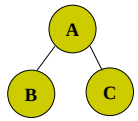
- **Recorrido en Preorden**
  - Este algoritmo procesa los vértices de un árbol binario utilizando el recorrido en preorden.

procedure ***preorden(PT)***
1. if **PT es vacío** then
2. return
3. **procesar PT**
4. **l:=hijo izquierdo de PT**
5. ***preorden(l)***
6. ***r:=hijo derecho de PT***
7. ***preorden(r)***

end ***preorden***

Cont...

- **Cont...**
  - Examinaremos el algoritmo para este caso sencillo:
  - Si el árbol binario es vacío (no tiene hijos) el algoritmo no procesa nada.
  - Decimos que PT es igual a la raíz (A) procesamos la raíz en la línea 3.
  - En la línea 5 llamamos a preorden con PT igual al hijo Izquierdo de la raíz. Vimos que si la entrada de preorden consta de un único vértice preoden procesa ese vértice, así a continuación procesamos el vértice B y luego en la línea 7 procesamos el vértice C.

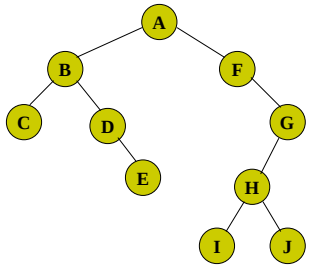


7.6.1

Cont...

- **Cont...**
  - **Ejemplo:**
    - **El preorden para este árbol es:**

ABCDEFGHIJ



Cont...

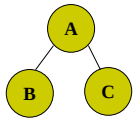
- **Recorrido en Entreorden**
  - Este algoritmo procesa los vértices de un árbol binario utilizando el recorrido en entreorden

procedure **entre*orden(PT)***
1. if **PT es vacío** then
2. return
3. **l:=hijo izquierdo de PT**
4. ***entreorden(l)***
5. ***procesar PT***
6. ***r:=hijo derecho de PT***
7. ***entreorden(r)***

end ***entreorden***

Cont...

- Cont...
- Examinaremos el algoritmo para este caso sencillo:
  - Si el árbol binario es vacío (no tiene hijos) el algoritmo no procesa nada.
  - Decimos que PT es igual a la raíz (B) procesamos la raíz en la línea 3.
  - En la línea 4 llamamos a entreorden con PT igual al hijo Izquierdo de la raíz. Vimos que si la entrada de entreorden consta de un único vértice entreorden procesa ese vértice, así a continuación procesamos el vértice A y luego en la línea 7 procesamos el vértice C.

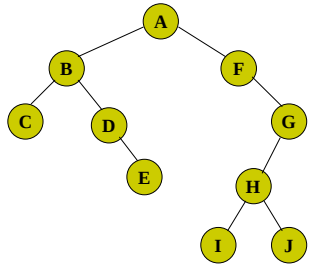


7.6.1

Cont...

- Cont...
  - Ejemplo:
    - El entreorden para este árbol es:

CBDEAFIHJG



Cont...

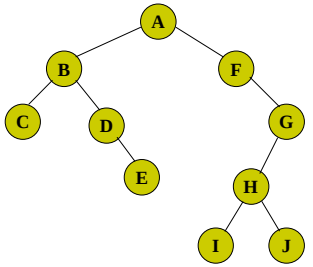
- Recorrido en Posorden
  - Este algoritmo procesa los vértices de un árbol binario utilizando el recorrido en posorden

```
procedure posorden(PT)
1.  if PT es vacío then
2.      return
3.  l:=hijo izquierdo de PT
4.  posorden(l)
5.  r:=hijo derecho de PT
6.  posorden(r)
7.  procesar PT
end posorden
```


Cont...

- Cont...
  - Ejemplo:
    - El posorden para este árbol es:

CEDBIJHGFA



Cont...

- **Representación de Expresiones Aritméticas**
  - **Consideraremos la representación de Expresiones aritméticas mediante árboles.**
  - **Restringiremos nuestros operadores a**

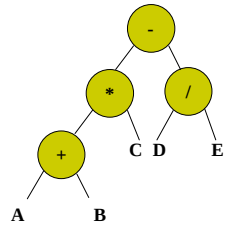
    +,-,*,/ Ej.: (A+B)*C-D/E

  - **Las variables A,B,C,D,E se conocen como operandos y los operadores trabajan sobre pares de operandos o expresiones**

Cont...

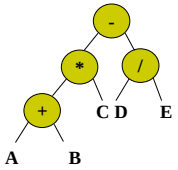
- **Cont...**
  - Una expresión se puede representar como un árbol binario.

Esta seria la representación mediante un árbol binario de la expresión (A+B)*C-D/E



Cont...

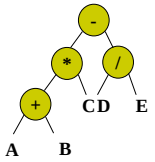
- **Cont...**
  - En el subárbol raíz el operador de división opera sobre los operandos D y E.
  - En el subárbol cuya raíz es * el operador de la multiplicación opera sobre el subárbol encabezado por + que a su vez representa una expresión y C.



Cont...

- **Cont...**
  - Si recorremos el árbol en entreorden, obtenemos (e insertamos un par de paréntesis para cada operación)

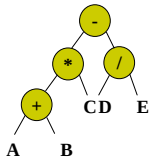
    (((A + B) * C) – (D / E))
  - Esta es la forma con todos los paréntesis de la expresión.
  - No es necesario especificar cuales operaciones deben realizarse antes que las demás, pues los paréntesis indican el orden.



Cont...

- **Cont...**
  - Si recorremos el árbol en posorden (notación polaca inversa), obtenemos

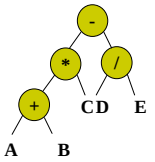
    AB + C * DE / -
  - Esta es la forma posfija de la expresión, aquí el operador se coloca después de los operandos.
  - La ventaja de la forma posfija es que no se necesitan paréntesis en relación con el orden de las operaciones.



Cont...

- **Cont...**
  - Se puede obtener una tercera forma de representación aplicando el recorrido en forma prefija (notación polaca).

    - * + ABC / DE
  - Esta forma al igual que la anterior no necesita de paréntesis acerca del orden de las operaciones



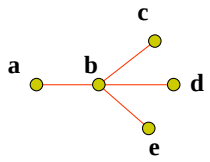
Isomorfismo de Arboles

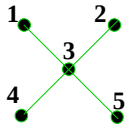
- Dos gráficas (G_1 y G_2) son isomorfas si y sólo si existen una función, f, uno a uno y sobre el conjunto de vértices de G_1 al conjunto de vértices de G_2 que preserva la relación de adyacencia en los sentidos de los vértices v_i y v_j son adyacentes en G_1si y sólo si los vértices $f(v_i)$ y $f(v_j)$ son adyacentes en G_2.

Cont...

- **Ejemplo:**
  - La función f del conjunto de vértices del árbol T1 del ejemplo anterior al conjunto de vértices del árbol T2 está definido por la relación.

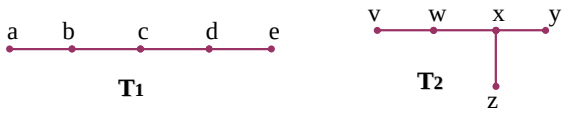
    f(a)=1 ; f(b)=3 ; f(c)=2 ; f(d)=4 ; f(e)=5





Cont...

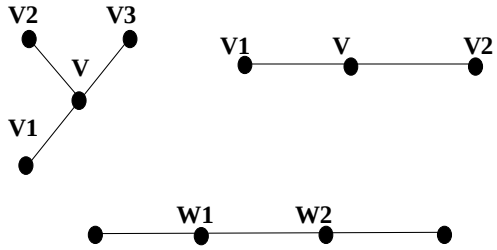
- **Ejemplo:**
  - Los árboles T1 y T2 de la siguiente figura no son isomorfos, pues T2 tiene un vértice (X) de grado 3, pero T1 no tien un vértice de grado 3.



Cont...

- **Cont...**
  - Existen árboles no isomorfos con cinco vértices.
  - El árbol tiene cuatro aristas.
  - Si tuviera un vértice v mayor que 4, v incidiría en más de cuatro aristas.
  - Cada vértice en el árbol tiene 4 aristas como máximo.
  - Determinaremos todos los árboles no isomorfos con cinco vértices, en que el grado máximo de los vértices sea 4.
  - Luego determinaremos todos los árboles no isomorfos con cinco vértices en los cuales el grado máximo de los vértice sea tres.

Cont...

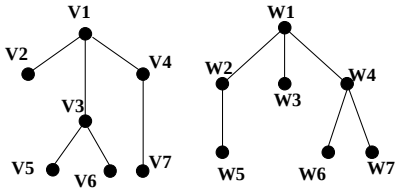


Cont...

- **Árboles con Raíz.**
  - Sea T_1 un árbol con raíz r_1 y T_2 con raíz r_2.
  - Los árboles de raíces r_1 y r_2 son isomorfos si existe una función, *f*, uno a uno y sobre el conjunto de vértices de T_1 en el conjunto de vértices T_2.
  - Siempre y cuando se cumplan las siguientes condiciones.
    - a) Los vértices v_1 y v_2 son adyacentes en T_1 si y sólo si los vértices $f(v_i)$ y $f(v_j)$ son adyacentes en T_2
    - b) $f(r_1) = r_2$

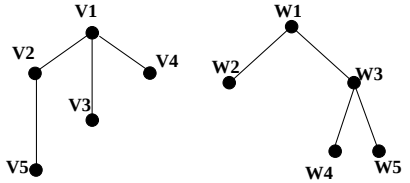
Cont...

- **Ejemplo:**
  - Árboles con raíz T_1y T_2, son isomorfos.



Cont...

- **Ejemplo:**
  - Árboles con raíz T_1y T_2, **NO** son isomorfos.

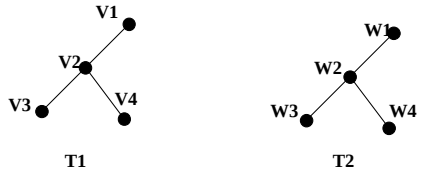


Cont...

- **Arboles Binarios Isomorfos**
  - Sea T1 un árbol binario de raíz r1y sea T2 un árbol binario con raíz r, los árboles binarios T1 y T2 son isomorfos si tienen una función uno a uno y sobre el conjunto de vértices T1 al conjunto de vértices T2.
    - a) Los vértices v_1 y v_2 son adyacentes en T_1 si y sólo si los vértices $f(v_i)$ y $f(v_j)$ son adyacentes en T_2.
    - b) $f(r_1) = r_2$
    - c) v es un hijo izquierdo de w en T1, si y solo si f(v) es un hijo izquierdo de f(w) en T2.
    - d) v es un hijo derecho de w en T1, si y solo si f(v) es un hijo derecho de f(w) en T2.

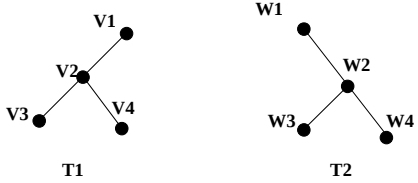
Cont...

- **Ejemplo:**



Cont...

- **Cont...**
  - **Ejemplo:**
    - Los árboles binarios T1 y T2 no son isomorfos. La raíz de v1 de T1 tiene un hijo izquierdo pero la raíz w1 de T1 no tiene un hijo derecho.



85