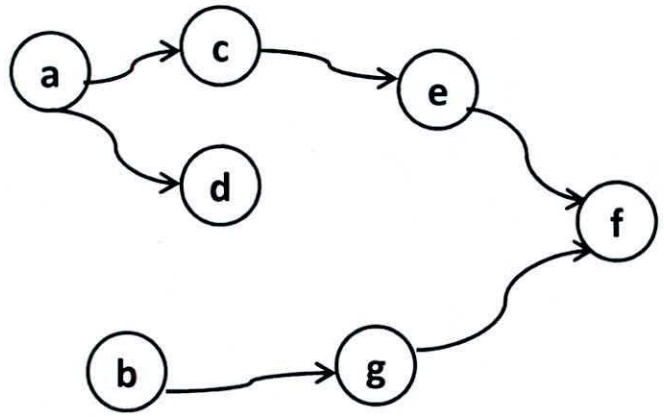
Algoritmos y Programación II, cátedra Calvo.

Final tomado en Julio/17

1. Realizar un recorrido topológico, en caso de ser posible, del grafo



   Explique, detalladamente, el algoritmo utilizado. Definir recorrido topológico

2. Defina AVL. Mostrar el alta de las siguientes claves:

   37 45 72 81 11 90 100

3. Defina Árbol B. Mostrar el alta de las siguientes claves:

   7 43 60 83 21 97

4. Hacer un algoritmo (no necesariamente uno visto en clase) con tiempo de referencia

$$T_{(n)} = T_{\left(\frac{n}{2}\right)} + n \qquad con\ T_{(1)} = 1$$

   y determinar su coste temporal por algún método de reducción por división.

5. Defina hashing, cómo implementaría una lista de dispersión con método de solución de colisiones por direccionamiento cerrado.

Para aprobar es necesario tener correctos y completos el 60% de cada ítem propuesto.
En la nota se ponderan, también, los resultados de los parciales y trabajos prácticos.
Duración del examen: 2 horas.

FINAL 1

① Realizar un recorrido topológico, en caso de ser posible, del sig. grafo. Explicar detalladamente (pero nearta precisión matemática) el algoritmo utilizado. Definir recorrido topológico

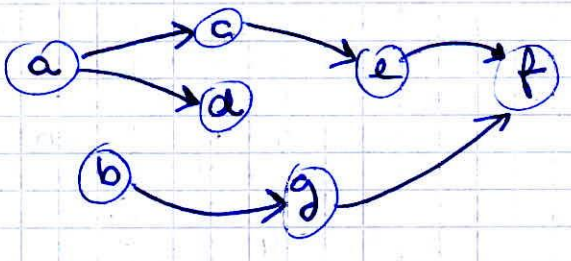
Recorrido topológico: Es un recorrido por los nodos de un grafo sin perder la unión entre ellos y respetando su precedencia. No puede tener ciclos. Se puede realizar en profundidad o en anchura

Algoritmo (en anchura):

1) Calcular los gr. de entrada de todos los nodos
2) Buscar nodo con gr. entr = 0 (sin predecesores)
   Marcarlo como visitado
3) Para todos los nodos adyacentes u a v, decrementar en 1 su grado de entrada
4) Repetir 2) y 3) hasta haber visitado todos los nodos

- Armar tabla con grados de entrada
- Acolar vértices con grado entr. = 0
- Mientras Cola $\neq \phi$

```
{v ← desacolar ()
 procesar v,
 ∀ u ady a v
   {decrementar grado de entrada de u
      si gr entr(u) == 0 → a colarlo
   }
}
```



No tiene ciclos así q se puede recorrer

	a	b	c	d	e	f	g
Tabla	0	0	~~X~~ 1	~~X~~ 1	~~X~~ 1	~~X~~ 1	~~X~~ 1
	0	0	0	0	0	0	0

Cola = ~~a, b, c, d, g, e, f~~

v = a, b, c, d, g, e, f

ady(b) = u(a) = ~~a~~
u(a) = ~~d~~
u(b) = ~~g~~
u(c) = ~~e~~
u(d) = ~~f~~
u(g) = ~~f~~
u(e) = f

↳ rec. en anchura:

{a, b, c, d, g, e, f}

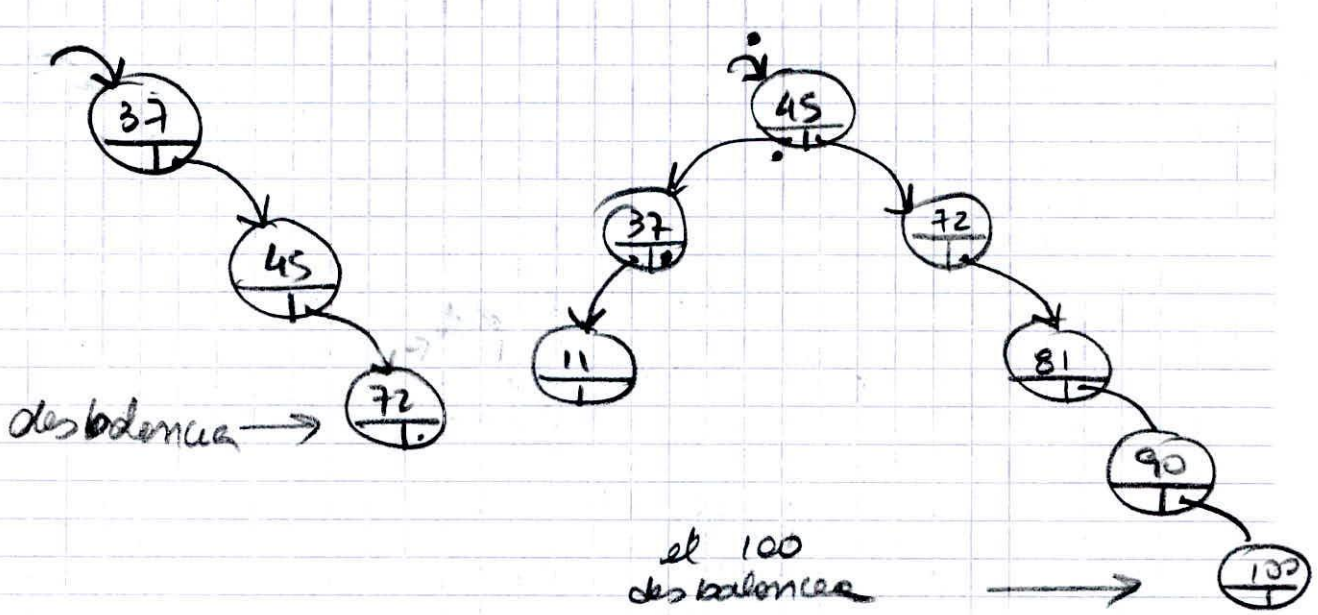
② Define AVL. Muestre el alta de las sig. claves

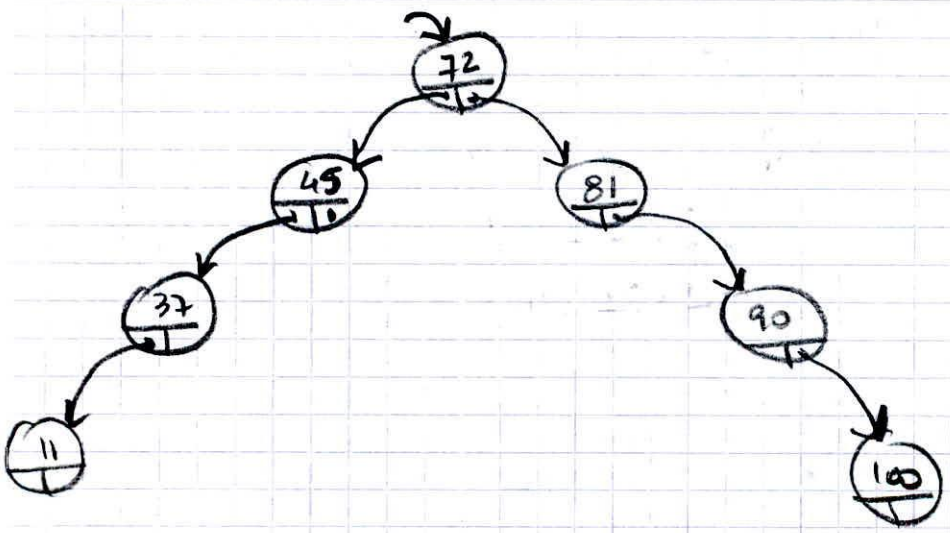
37 45 72 81 11 90 100

AVL: Es un árbol que siempre está balanceado, un dif. de balanceo entre subarbol izq y derecho en 1 o 0

Cada vez que al ingresar un dato se desbalancea, lo balancea haciendo rotaciones (dobles o simples según el caso) para q' el grado de balance sea 0 o 1

Es un arbol binario → Las claves de un nodo son mayores que toda clave del subarbol izq y menor que toda clave del subarbol derecho





HOJA N°

FECHA

Final 1

③ Defina Árbol B mostar el alta de las sig. claves

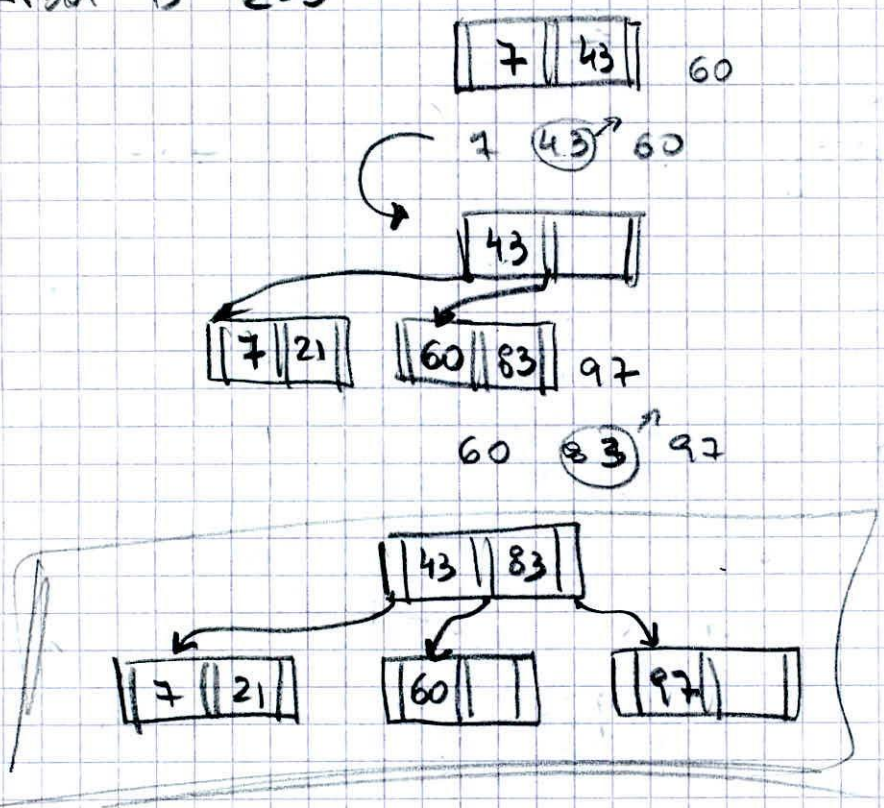
7 43 60 83 21 97

Árbol B (o multivías) ← m vías

Es un arbol ordenado, en el que se cumple:

- la raiz o es hoja o tiene, al menos, 2 hijos.
- cada nodo tiene m vias → tiene m-1 claves → si es hoja, tiene $m/2 \leq \#\text{claves} \leq m$
- todas las hojas estan al mismo nivel
- si un nodo no hoja tiene k claves, tiene k+1 hijos

Árbol B 2-3



NOTA

(4) Hacer un algoritmo (no necesariamente uno visto en clase) con tiempo de referencia $T(n) = T\left(\frac{n}{2}\right) + n$ y determinar su costo temporal por algún método de reduc. por división

$T(n) = T\left(\frac{n}{2}\right) + n$ ✗ teor. maestro → $a = 1$, $b = 2$, $k = 1$, $a < b^k$

$\in O(n)$

$T\left(\frac{n}{2}\right) = T\left(\frac{n}{4}\right) + \frac{n}{2}$

$T(n) = T\left(\frac{n}{4}\right) + \frac{n}{2} + n$

$T\left(\frac{n}{4}\right) = T\left(\frac{n}{8}\right) + \frac{n}{4}$

$T(n) = T\left(\frac{n}{8}\right) + \frac{n}{4} + \frac{n}{2} + n$

$T(n) = T\left(\frac{n}{2^i}\right) + \sum_{k=0}^{i-1} \frac{n}{2^k}$ → $n \sum_{0}^{i-1} \frac{1}{2^k}$

$T(n) = T(1) + n \cdot \sum_{0}^{\log_2(n)-1} \frac{1}{2^k}$

tomo $1 = \frac{n}{2^i} \rightarrow n = 2^i \rightarrow \boxed{i = \log_2(n)}$

$\in O(n)$

```
void quieroAprobar (int V[], int n) {
                                  ↖ largo del vector
    for (i = 0, i < n, i++)
        { cout << V[i] ; }
    quieroAprobar (V, n/2);

}
```

NOTA

Final 1

HOJA Nº

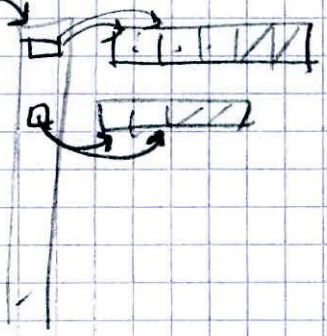
FECHA Julio/17

(3) Defina Hashing, cómo implementaría una lista de dispersión con método de solución de colisiones por direccionamiento cerrado

Hashing es una alternativa a los algoritmos de búsqueda basados en comparación entre claves. Se basan en una transformación de la clave.

Por direccionamiento cerrado implementaría una lista en un array que, ante colisiones, guarde el valor en la lista asociada

h(x)



h es la función hash y x es la clave

NOTA