(1) extraido del campus

Algo 2
Calvo

1) Considere estos datos: 25, 23, 3, 13, 33, 31, 11, 39, 10

i) Muestre gráficamente la evolución de las siguientes estructuras de datos, inicialmente vacías, hasta graficar su estado final luego de haber hecho todas las altas:
a) ABB b) AVL c) árbol Heap de mínimo d) Trie (con la implementación que Ud. elija)

ii) Para el ABB, AVL y Heap, indique cual es el costo temporal del alta, **justificando las respuestas.** (puede justificar sin formalización matemática)

2) Considere dos implementaciones diferentes del TDA Conjunto:
En la primera implementación se usa una estructura de lista ligada ordenada sin repeticiones, con un único puntero de entrada al primer nodo de la misma. El Alta en el Conjunto es un alta ordenada en la lista (sólo si el dato no estaba), la Baja en el conjunto es la Baja en la lista, la Búsqueda en el conjunto es la búsqueda habitual en la estructura de lista ligada.
En la segunda implementación se usa una estructura de árbol AVL; el Alta en el conjunto es un alta en al AVL, la Baja en el conjunto es una baja en el AVL, y la Búsqueda es la búsqueda habitual en el AVL.
Se pide comparar ambas implementaciones en términos de coste temporal, indicando (puede hacerlo coloquialmente) cual es el tiempo del peor caso para cada operación según cada implementación, en función del número de elementos almacenados en el Conjunto, justificando la respuesta. ¿Cuál es la mejor implementación en términos de coste temporal?

3) Explique en qué consiste la estrategia Divide y Vencerás; describa detalladamente algún algoritmo que use esa estrategia, indicando en que parte del algoritmo está aplicando cada etapa de la estrategia. Indique cual es la complejidad temporal del algoritmo que describió, justificando su afirmación.

4) a) ¿Qué es una cola con prioridades?
b) Elija una estructura eficiente para implementar una cola con prioridades y describa como se lleva a cabo el alta (describa el algoritmo detalladamente). Los elementos a almacenar serán: 10, 32, 43, 23, 12, 12, 23, 54. Luego, describa cómo se realiza una baja.
c) Justifique la elección de la estructura en base a la eficiencia temporal (explique por qué la estructura tiene, para el alta y la baja, la eficiencia que usted indique.).

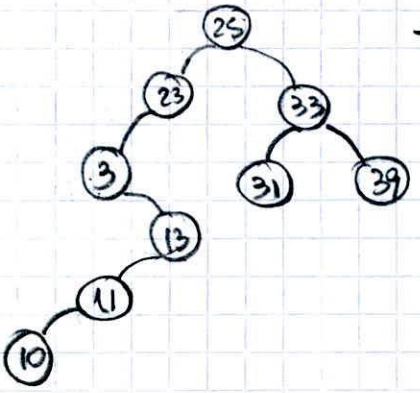
5) a)Defina árbol binario de búsqueda, y árbol AVL.
b)Describa algún algoritmo que permita determinar la altura de un árbol binario.
c)Considere un árbol AVL inicialmente vacío y muestre gráficamente la evolución del mismo al realizar estas altas: 12,20,29,11,5,8,67,70,80,90
d)Explique qué mejoras representa el árbol AVL con respecto al árbol binario de búsqueda y justifique.

- ①

① Considerar estos datos 25 23 3 13 33 31 11 39 10

i) Mostrar gráficamente la evolución de los sig. estr. de datos, inicialmente vacíos hasta graficar su estado final luego de haber hecho todos los altas

a) ABB

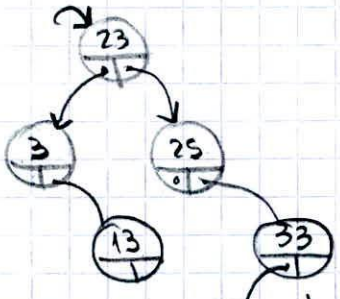


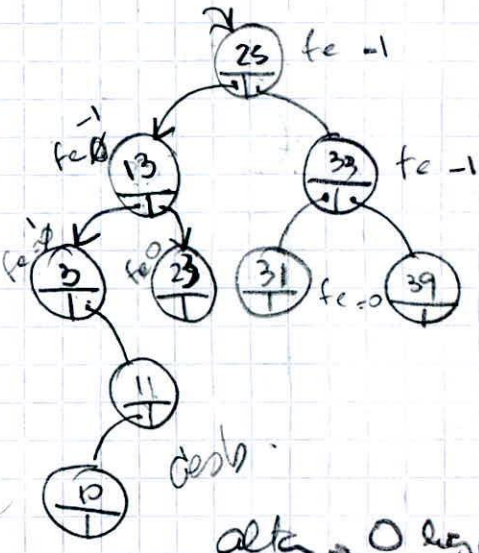
→ alta → Si está balanceado $O(\log(n))$ altura

Si degenerá en lista $O(n)$ (como vector o lista)

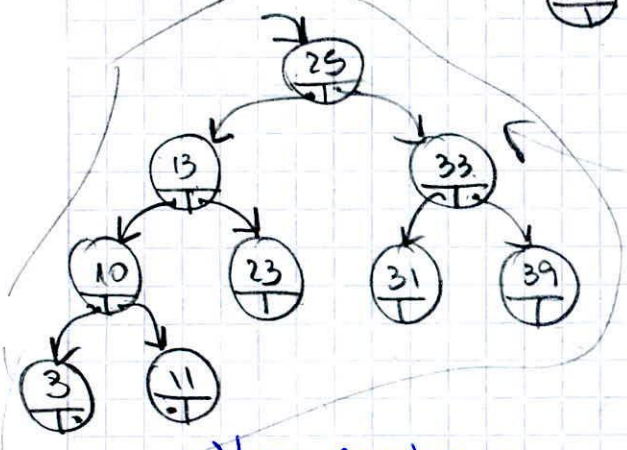
b) AVL

→ desbal.





desb.



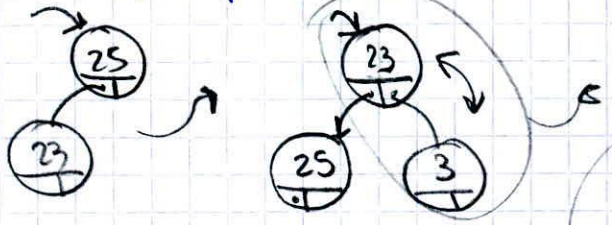
alta → $O \log(n)$

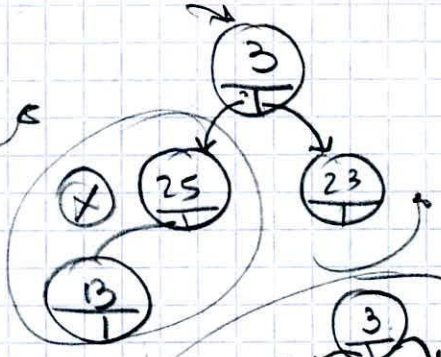
está balanceado, recorre la altura hasta ubicarse

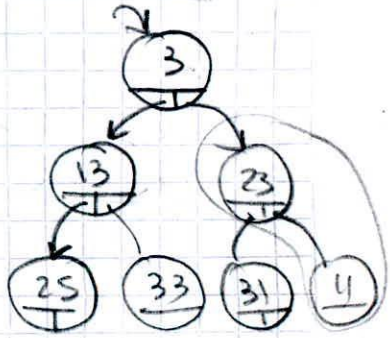
altura max = $1.44 \log(n)$

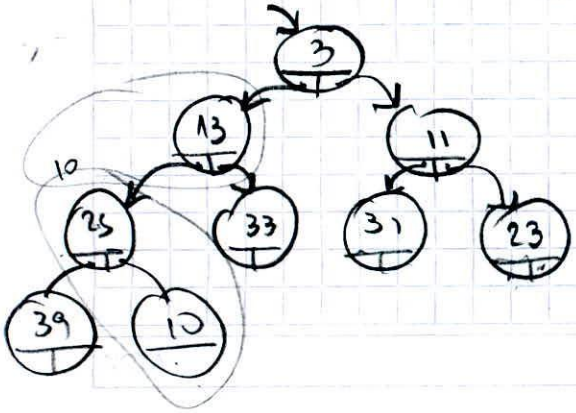
$\in \log(n)$

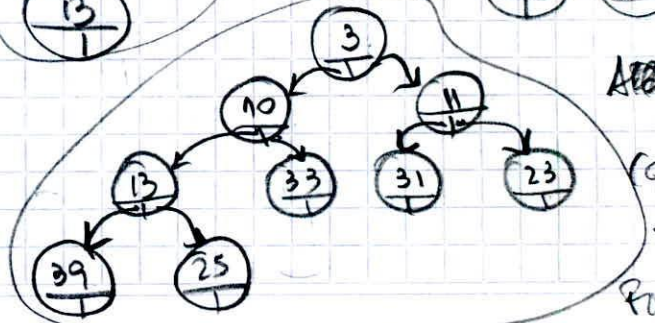
c) heap de mínimo







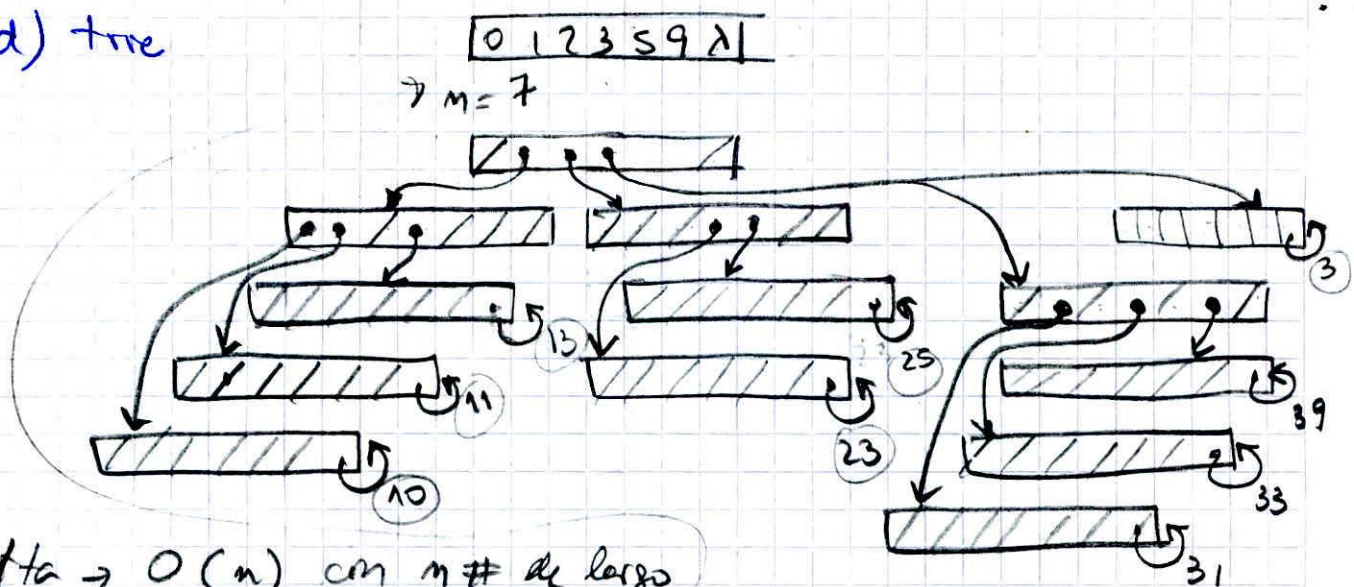




Alta $O \log(n)$ (queda bal. → altura)

Final

d) trie



Alta → O(n) con n # de largo del mazo

ii)

(2) En una lista ordenada, para recorrerla (para buscar si el dato está y, si no está, ubicarlo) tiene un orden de O(n) pues, en el peor caso, tiene que recorrer toda la lista. Con la baja y búsqueda es similar (tiene q' recorrer la lista) → O(n)

En AVL recorrer los nodos depende de la altura y en el caso más desfavorable (árbol de Fibonacci) la altura es de 1,44 log(n) .°. recorrer un AVL para encontrar un dato ∈ O(log(n)) lo mismo para baja y alta.

En términos de costo temporal, conviene la implementación de árbol AVL

(3) Estrategia D y V

Dado un problema de tamaño N > L (valor arbitrario) se lo divide en m subproblemas independientes entre sí y con la misma estructura que el problema original

Luego, si N ≤ L → se lo resuelve de otra manera.

La solución general es una combinación de las soluciones de los subproblemas

El Quick Sort, o Merge Sort, entre otros, utiliza esta estrategia.

Divide un vector dado en 2 vectores, que vuelve a dividir y así hasta llegar a 1 elemento y luego los ordena.

HOJA N°

FECHA

```
void MS (int V[], int primero, int ultimo) {
    int medio = (primero + ultimo) / 2;
    Si (primero < ultimo) {
        MS (V, primero, medio);
        MS (V, medio + 1, ultimo);
        intercalar (V, primero, ultimo);
    }
};
```

acá divido el problema en 2 partes (y asi sucesivamente)

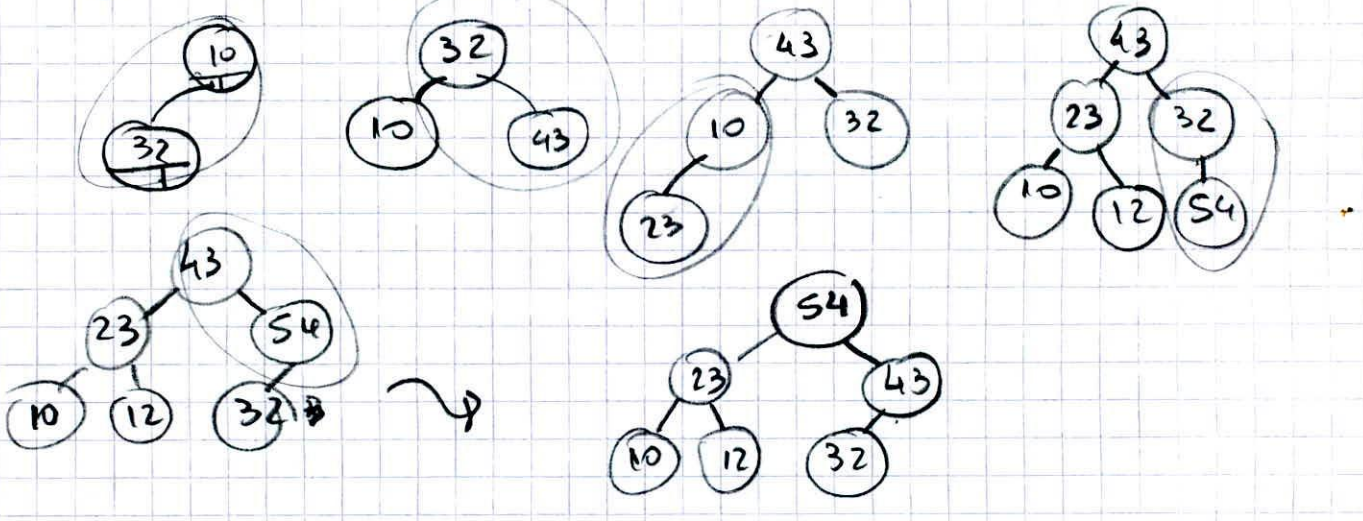
costo : $T(n) = 2^{a} T\left(\frac{n}{2}\right)^{b} + n^{k=1}$

x teor. Maestro → a = 2, b = 2, k = 1 → $a = b^k$ → $T(n) \in n \cdot \log(n)$

(4) a) ¿Qué es cola con prioridades?

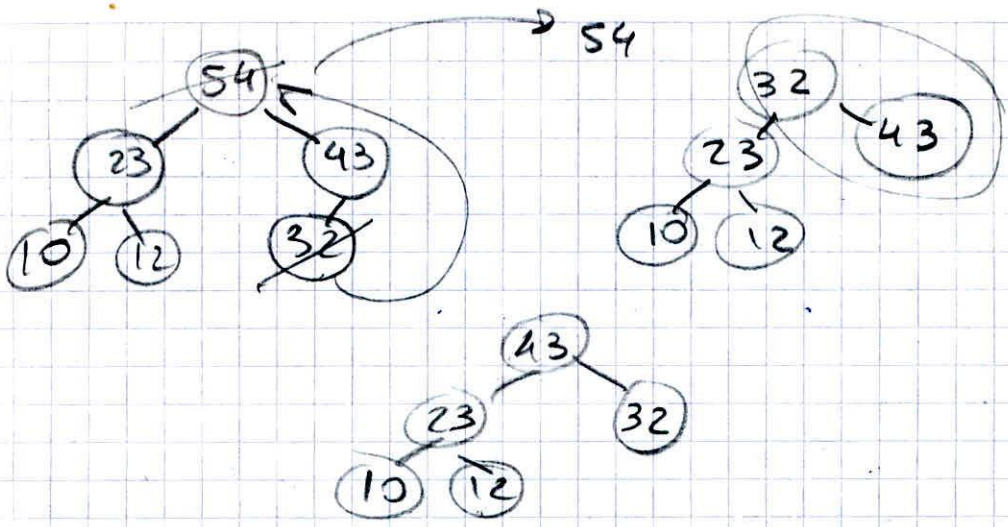
Es una implementación en donde se almacenan los datos según una prioridad dada.
Para esto se utiliza heap (de máx. o mínimo) ya que almacena en su raíz el valor que se extraerá primero (según la prioridad dada)

b) 10 32 43 23 12 12 23 54 (heap máx)



NOTA

baja



es un arbol completo o casi completo. La baja que se produce es la raíz → se lo reempl. x ultima hoja y se ajusta para q' quede ordenado → Coste → altura del heap → $\in O(\log(n))$

(5) a) **ABB**: es un árbol binario ordenado → o bien la raíz es hoja o tiene, a lo sumo, 2 hijos. todas las claves del subárbol izq. son menores que la clave del nodo y todas las del subárbol derecho son mayores

b) **AVL**. es un árbol binario (ABB), también ordenado, que está siempre balanceado. Si se ingresa una clave y queda desb. → se balancea. Balanceo → dif altura subarb izq y subarbol derecho ≤ 1

b) algoritmo que determine altura de un arbol binario

```
altura (arbol A) {
  si (A es hoja) devuelve 1.
  sino
      para cada hijo i de A {
          h = 1+ altura (A.hijo (i));
      }
  devuelve max (h[]);
}
}
```