



Universidad Católica del Norte
Facultad de Ingeniería y Ciencias Geológicas
Departamento de Ingeniería de Sistemas y Computación

Nota Cátedra: _____

Prom. Controles: _____

ESTRUCTURA DE DATOS – CÁTEDRA 2 (I-2019)

- Total: 240 pts (escala 60%) -

Pauta

Nombre - RUT:

Fecha: 08 de junio del 2019.

Competencias a evaluar: Programación de estructuras de datos utilizando árboles.

- 60 pts 1. Cree un algoritmo recursivo para:
- 30 a. En un ABB: Contar la cantidad de elementos mayores a X.
 - 30 b. En un AB: Indicar si existe algún nodo que sólo tenga un hijo izquierdo.
- 60 pts 2. Cree un algoritmo iterativo para recorrer un árbol binario por nivel.
`void recorridoPorNivel();`
- 60 pts 3. En un árbol AVL mostrar gráficamente paso a paso al insertar:
20, 30, 37, 10, 12, 29, 100, 90, 80, 85, 86, 5, 15, 13, 17, 1, 2
- 60 pts 4. En un árbol B+ de $M=4$ y $M_{HOJA}=4$, mostrar gráficamente paso a paso al:
- 30 a. Insertar: 7, 10, 15, 40, 20, 8, 100, 74, 50, 18, 4, 61, 16, 13
 - 30 b. Eliminar: 10, 13, 40, 61, 100

1. a) int contarMayor (Nodo p, int x) {

if (!p) {

return 0;

5 pts

}

if (p->data > x) {

5 pts

return 1 + contarMayor(p->Izq, x) + contarMayor(p->Der, x);

10 pts

} else {

5 pts

return contarMayor(p->Der, x);

5 pts

}

}

30 pts

b) bool buscarHIzq (Nodo p) {

if (!p) {

return FALSE;

5 pts

}

if (p->Izq && !p->Der) {

5 pts

{

return TRUE;

5 pts

}

return buscarHIzq(p->Izq) || buscarHIzq(p->Der);

15 pts

}

30 pts

2. void recorridoPorNivel () {

Queue q;

6 pts

Nodo aux = raiz;

6 pts

while (aux) { //mientras aux no apunte a NULL

8 pts

print (aux->data);

8 pts

if (aux->Izq) {

q.push (aux->Izq);

12 pts

}

if (aux->Der) {

q.push (aux->Der);

12 pts

}

aux = q.pop();

8 pts

}

}

60 pts

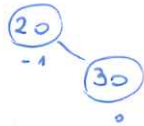
1

3.

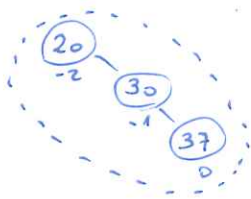
+20



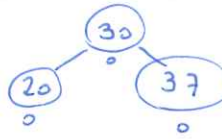
+30



+37

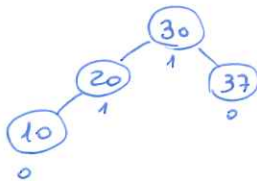


RR

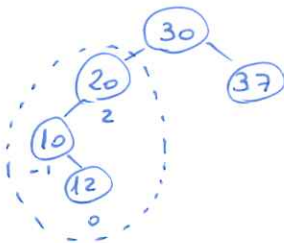


5 pts

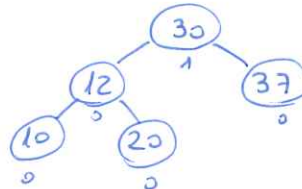
+10



+12

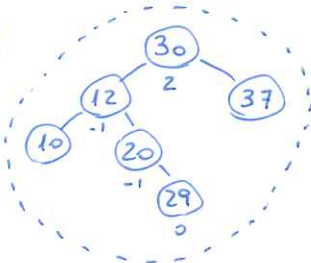


LR

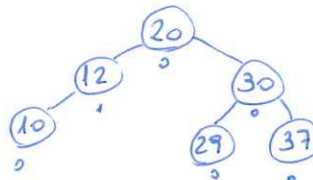


5 pts

+29

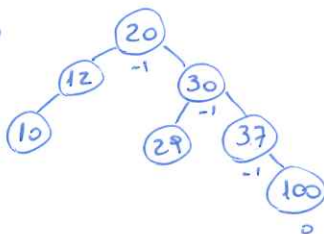


LR

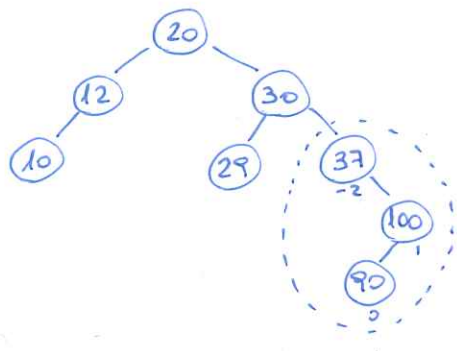


5 pts

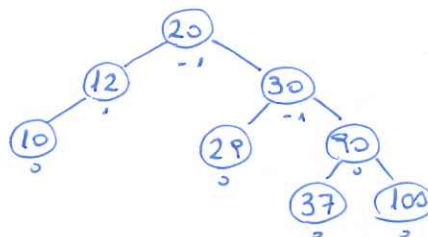
+100



+90

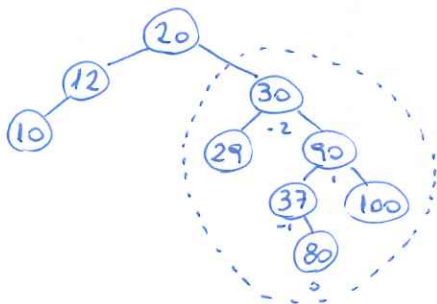


RL

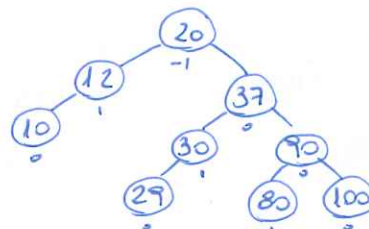


5 pts

+80

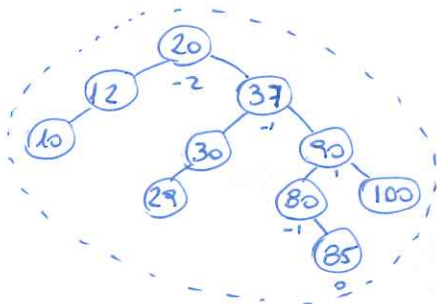


RL

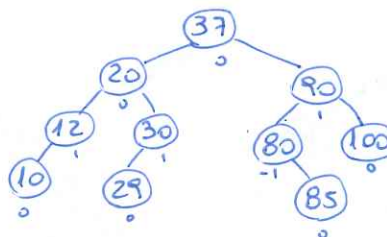


5 pts

+85

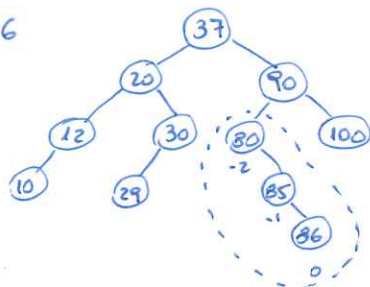


RR

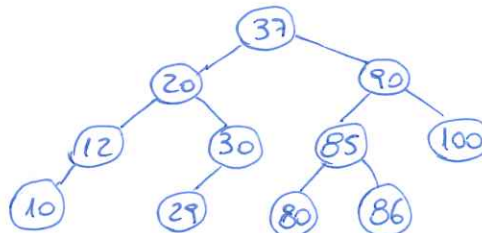


5 pts

+86

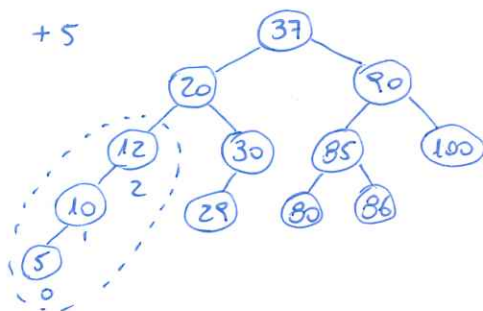


RR

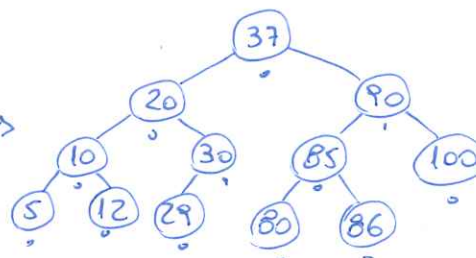


5 pts

+5



LL



5 pts

```

graph TD
    37((37)) --> 20((20))
    37 --> 90((90))
    20 --> 10((10))
    20 --> 30((30))
    10 --> 5((5))
    10 --> 12((12))
    12 --> 15((15))
    30 --> 29((29))
    90 --> 85((85))
    90 --> 100((100))
    85 --> 80((80))
    85 --> 86((86))
  
```

```

graph TD
    37((37)) --> 20((20))
    37 --> 90((90))
    20 --> 10((10))
    20 --> 30((30))
    10 --> 5((5))
    10 --> 13((13))
    13 --> 12((12))
    13 --> 15((15))
    30 --> 29((29))
    90 --> 85((85))
    90 --> 100((100))
    85 --> 80((80))
    85 --> 86((86))
    style 5 fill:none,stroke:none
    style 12 fill:none,stroke:none
    style 15 fill:none,stroke:none
    style 29 fill:none,stroke:none
    style 80 fill:none,stroke:none
    style 86 fill:none,stroke:none
    style 5_label[0] fill:none,stroke:none
    style 12_label[0] fill:none,stroke:none
    style 15_label[0] fill:none,stroke:none
    style 29_label[0] fill:none,stroke:none
    style 80_label[0] fill:none,stroke:none
    style 86_label[0] fill:none,stroke:none
    style 100_label[0] fill:none,stroke:none
  
```

5 pts

17

```
graph TD; 37((37)) --> 20((20)); 37 --> 90((90)); 20 --> 10((10)); 20 --> 30((30)); 10 --> 5((5)); 10 --> 13((13)); 13 --> 12((12)); 13 --> 15((15)); 15 --> 17((17)); 90 --> 85((85)); 90 --> 100((100)); 85 --> 80((80)); 85 --> 86((86));
```

5 pts

5 pts

+ 2

```

graph TD
    37((37)) --> 13((13))
    37 --> 90((90))
    13 --> 10((10))
    13 --> 20((20))
    10 --> 5((5))
    10 --> 12((12))
    20 --> 15((15))
    20 --> 30((30))
    15 --> 17((17))
    15 --> 29((29))
    90 --> 85((85))
    90 --> 100((100))
    85 --> 88((88))
    85 --> 86((86))
    5 --> 1((1))
    5 --> 2((2))
    style 1 stroke-dasharray: 5 5
    style 2 stroke-dasharray: 5 5
    style 2 stroke-width:4px
  
```

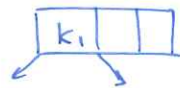
```

graph TD
    37((37)) --> 13((13))
    37 --> 90((90))
    13 --> 10((10))
    13 --> 20((20))
    10 --> 2((2))
    10 --> 12((12))
    20 --> 15((15))
    20 --> 30((30))
    90 --> 85((85))
    90 --> 100((100))
    85 --> 80((80))
    85 --> 86((86))
    30 --> 17((17))
    30 --> 29((29))
    2 --> 1((1))
    2 --> 5((5))
    12 --> 11((11))
    12 --> 13((13))
    15 --> 14((14))
    15 --> 16((16))
    17 --> 18((18))
    17 --> 19((19))
    29 --> 28((28))
    29 --> 30((30))
    80 --> 79((79))
    80 --> 81((81))
    86 --> 85((85))
    86 --> 87((87))
    100 --> 99((99))
    100 --> 101((101))
  
```

5 pts

60 pts

4. I) cant. máx. hijos = 4 \Rightarrow cant. máx. claves = 3
 cant. min. hijos = 2 \Rightarrow cant. min. claves = 1



II) cant. máx. claves = 4
 cant. min. claves = 2



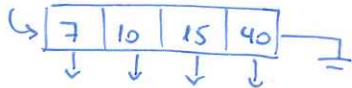
$$j = \left\lceil \frac{4+1}{2} \right\rceil = 3$$

a) +7

+10

+15

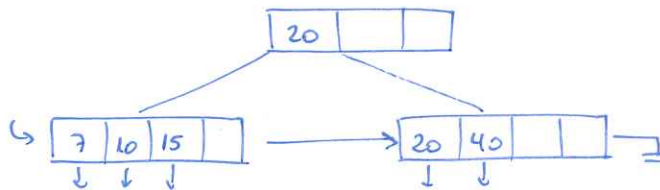
+40



+20 overflow: "j" entradas en el nodo original

7 10 15 || $\overset{\uparrow j+1}{(20)}$ 40

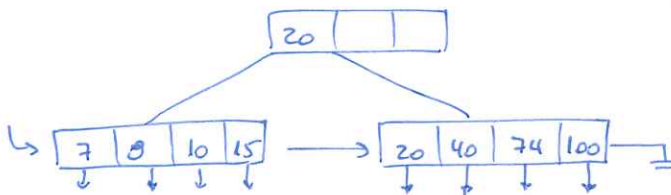
6 pts



+8

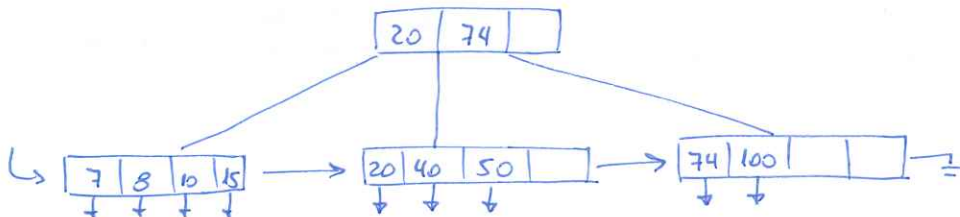
+100

+74



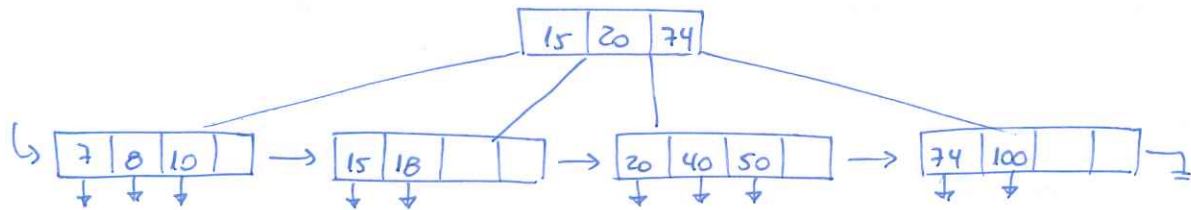
+50 overflow: 20 40 50 || $\overset{\uparrow}{(74)}$ 100

6 pts

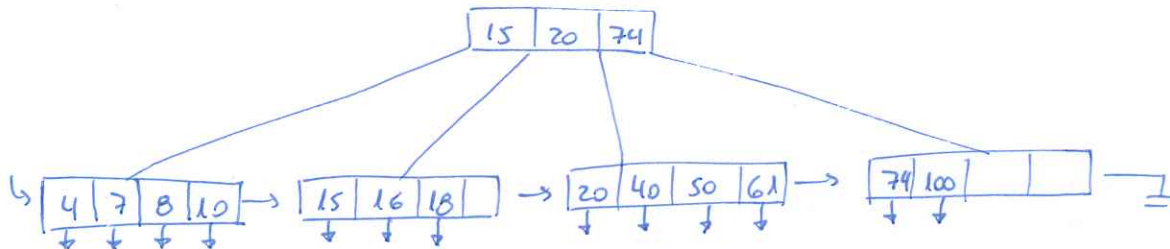


+18 overflow: 7 8 10 || \uparrow 15 18

6 pts



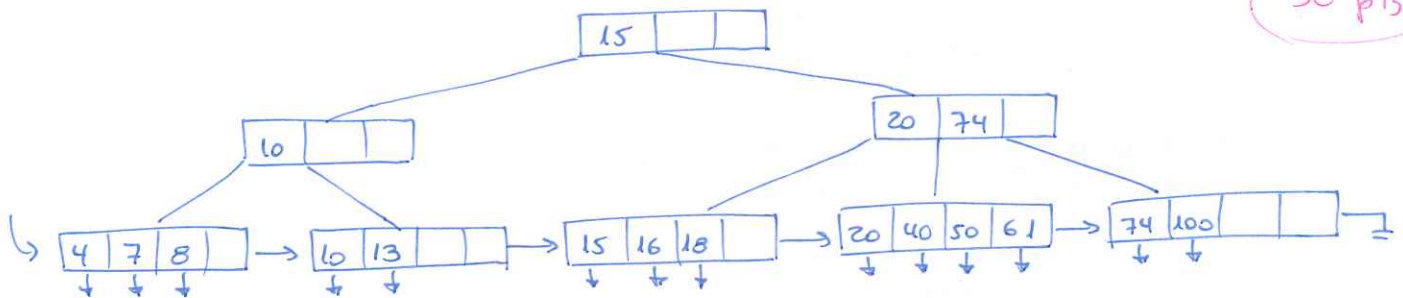
+4
+61
+16



+13 overflow: 4 7 8 || \uparrow 10 13 6 pts

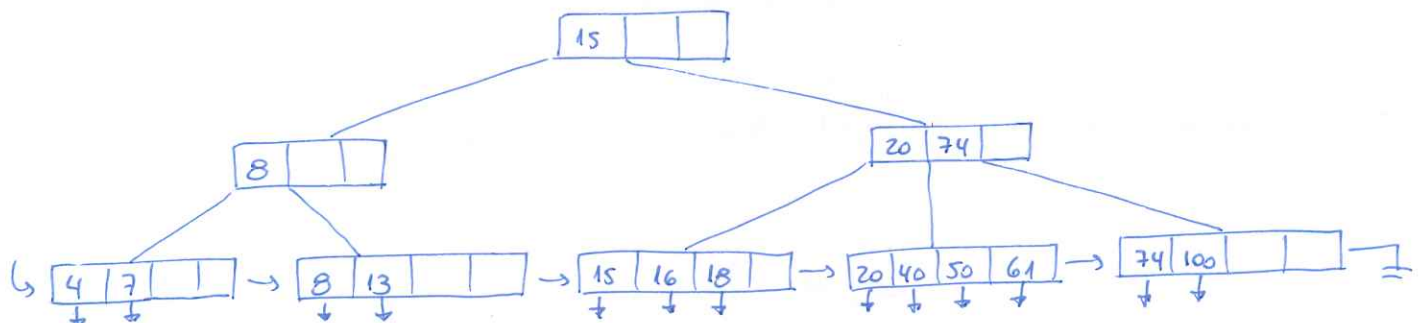
overflow (B): 10 \uparrow 15 20 74 6 pts
 $n=4 \Rightarrow \lceil \frac{n}{2} \rceil = 2$

30 pts



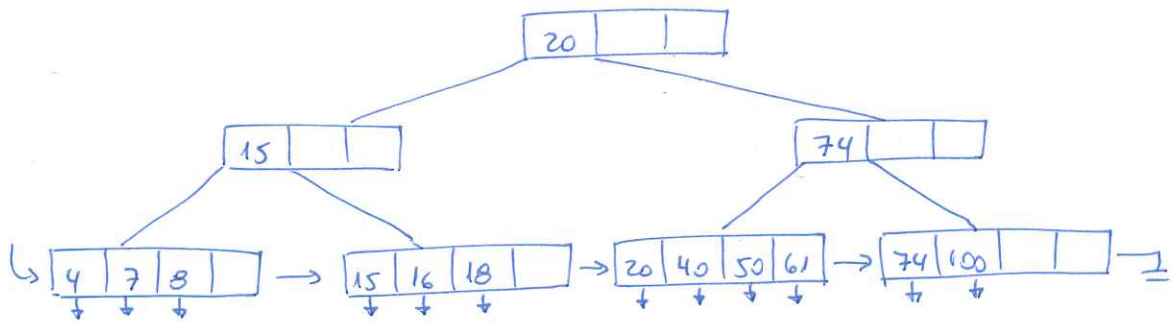
b) -10 underflow: 1) Redist ✓

6 pts

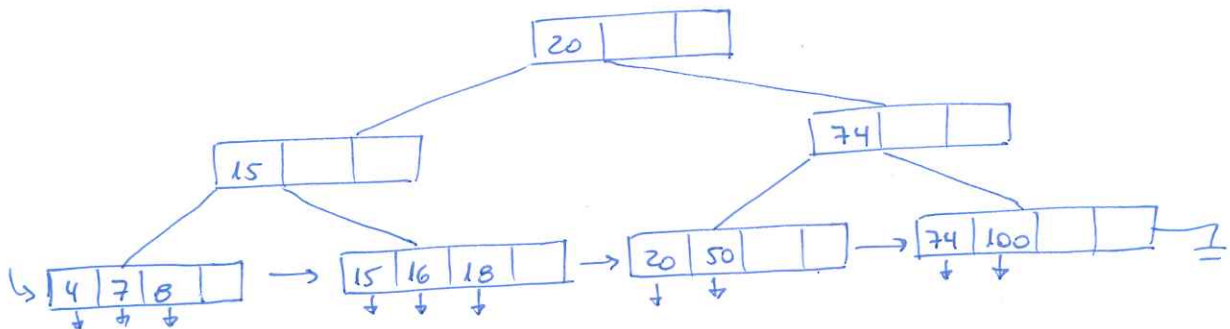


-13 underflow: 1) Redist X
2) Fusión: 4 7 8 6 pts

underflow (B): 1) Redist ✓ 6 pts

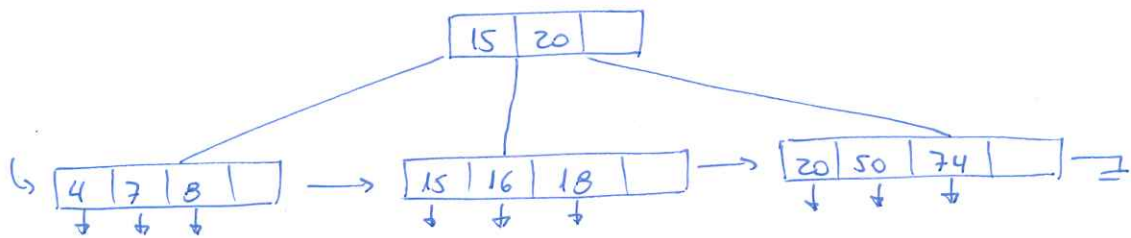


-40
-61



-100 underflow: 1) Redist X
2) Fusión: 20 50 74 6 pts

underflow (B): 1) Redist X
2) Mezcla: 15 20 6 pts



30 pts