- Arboles binarios

```
struct Nodo{
    int dato;
    Nodo* iza;
    Nodo* der;
  Nodo(int x); dato(x), iza(Null), der(Null){}
};
```

Funcion para insertar

```
Nodo* insertar (Nodo* r, int x){
    if(r==Null) return new Nodo(x);
    if(x<r->dato) r->iza = insertar(r->iza, x);
    else if(x>r->dato) r->der = insertar(r->der, x);
    return r;
}
```

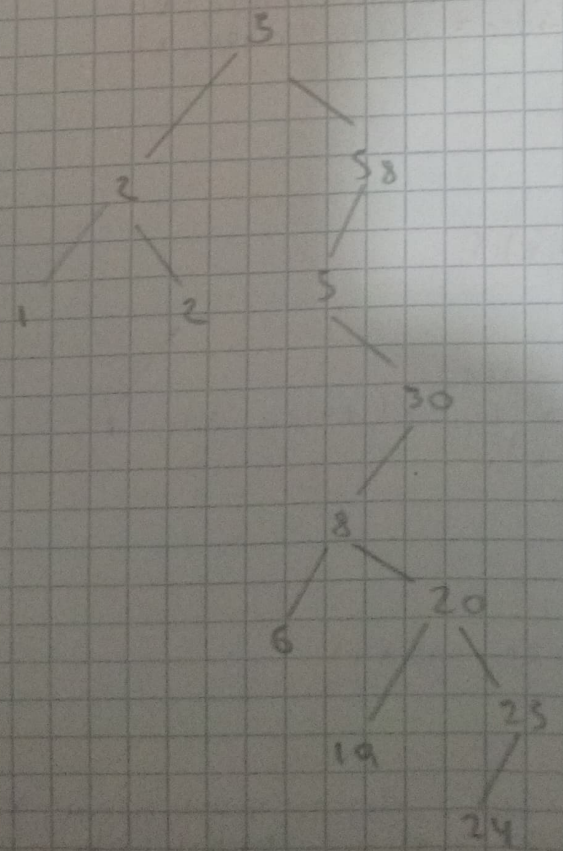
Funcion para buscar

```
bool buscar(Nodo* r, int x){
    if(!r) return false;
    if(x==r->dato) return true;
    if(x<r->dato) return buscar(r->iza, x);
    return buscar(r->der, x);
}
```

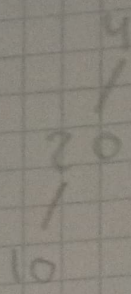
Funcion para Eliminar

```
Nodo eliminar Nodo(Nodo *r, int x){
    if(!r) return r;
  if(x<r->dato) r->iza = eliminarNodo(r->iza, x);
  else if(x>r->dato) r->der = eliminarNodo(r->der, x);
  else{
      if(!r->iza && !r->der){ delete r; return Null;}
  else if(!r->iza){ Nodo* t = r->der; delete r; return t;}
  else if(!r->der){ Nodo* t = r->iza; delete r; return t;}
    Nodo* t = minimo(r->der);
    r->dato = t->dato;
    r->der = eliminarNodo(r->der, t->dato);
```

```
void mostrarHorizontal(Nodo *r, string prefijo="", bool esIzq=false){
  if(r==NULL) return;
  if(r->der)
    mostrarHorizontal(r->der, prefijo+(esIzq?"|  ":"   "), false);
  cout<<prefijo
  if(esIzq)
    cout<<"L--";
  else
    cout<<"r--";
  cout<<r->dato<<endl;
  if(r->izq)
    mostrarHorizontal(r->izq, prefijo+(esIzq?"   ":"|  "), true);
}
```

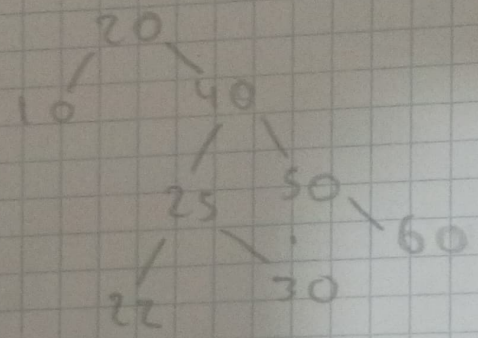


40, 20, 10, 25, 30, 22, 50, 60, 55, 52



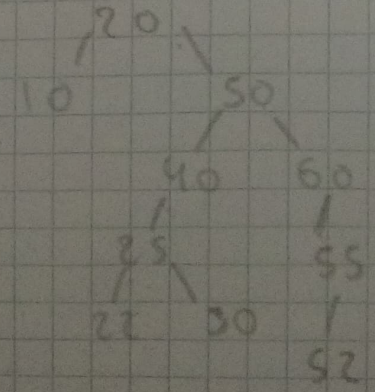
Fv = Iz - Derecha

Fv = 2 - 0 = 2



Fv = Iz - Derecha

Fv = 3 - 1 = +2



fv(60) = Derecha - sin

fv(60) = 0 - 2 = -2

20
10
30
40
35
25
32
60
22
30