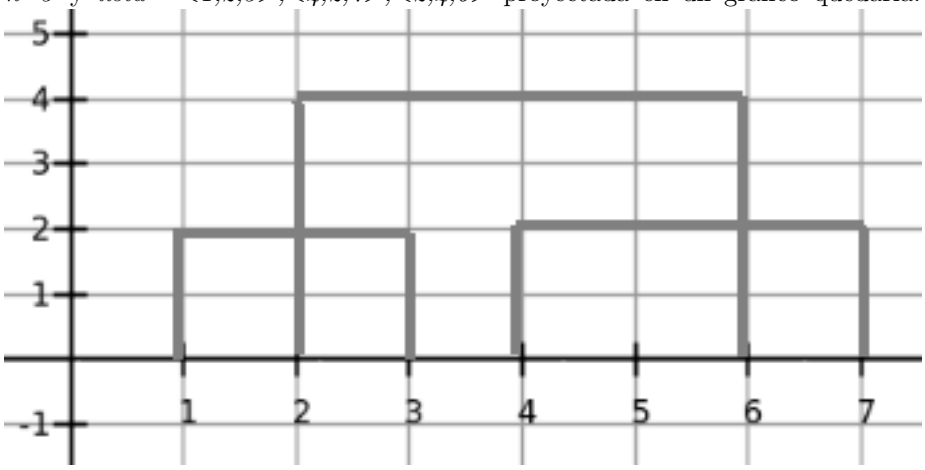
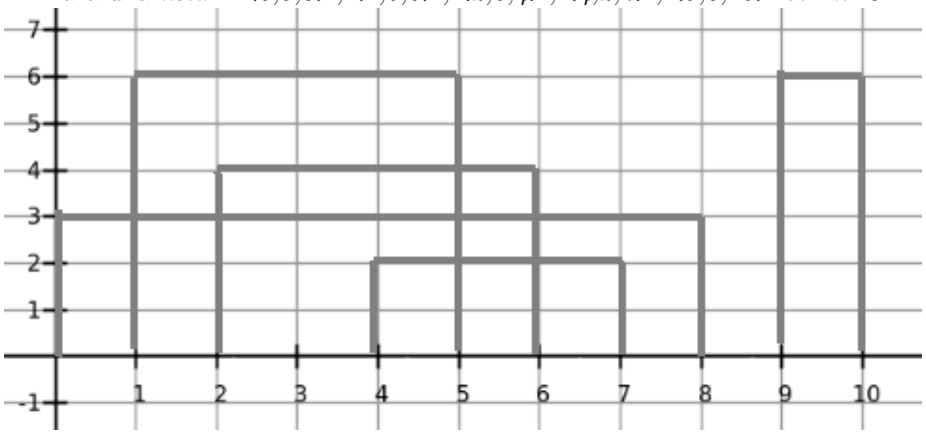
Problema a resolver:

El problema esta dado por la siguiente situación: tenemos en un "lista" con una cantidad $3*n$ de números(n un número fijo).
Para i desde *0* a *n-1*, vamos a decir la posición i en la lista va a ser *Izq* del edificio *i-ésimo*, *i+1* va a ser *Alt* del edificio *i-ésimo* e *i+2* va a ser *Der* del edificio i-ésimo.
A grandes rasgos vamos a tener una lista de n edificios (interpretamos a un edificio como una tupla *<Izq,Alt,Der>*) con una base en común implícita que es 0.
Por ejemplo para un entrada de la forma:
n=3 y *lista= <1,2,3>,<4,2,7>,<2,4,6>* proyectada en un gráfico quedaría:

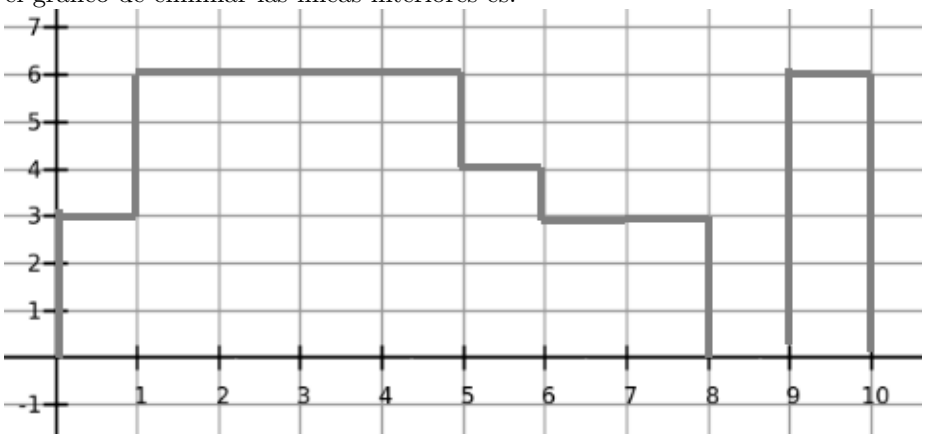


Lo que queremos hacer es "eliminar todas las lineas interiores del gráfico", quedarnos con su contorno se obtiene el mismo resultado "siguiendo con el dedo el gráfico".

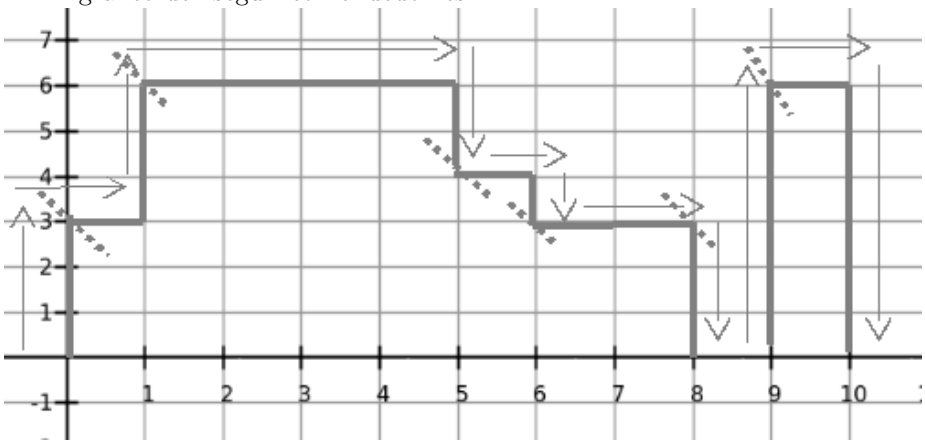
Para una *lista*= *<0,3,8>,<1,6,5>,<2,6,4>,<4,2,7>,<9,6,10>* con *n=5*



el gráfico de eliminar las lineas interiores es:



El gráfico de "seguir con el dedo" es:



Lo que hago cuando "sigo con el dedo" es:
Empezar con el primer edificio y seguimos el trazo, si me interseco con otro edificio seguir el trazo del edificio con el que me intersequé desde ese punto.
Si no me interseco con nadie pero hay más edificios adelante "siguir con el dedo" los otros. Si no hay mś edificios terminé.
Luego de ese contorno voy a obtener la solución final que son los puntos donde hay cambios($\uparrow \longrightarrow$ y $\downarrow \longrightarrow$).

Sea lista: lista(<Izq,Alt,Der>) y n la cantidad de tuplas en la lista.

Resolución:

```
1:  procedure ResolverEdificios(lista,n)
2:      comparo ← lista[0]
3:      heap ← vacio
4:      imprimo el primer punto
5:      for (i ← 1, n − 1) do //voy recorriendo los edificios
6:          siguiente ← lista[i]
7:          if (se intersecan comparo y siguiente*0) then
8:              if siguiente > comparo en altura *1 then
9:                  imprimir solución
10:                 if (el heap no está vacio) then
11:                     heap.encolar(comparo)
12:                 else
13:                     if (comparo no está en el tope del heap) then
14:                         heap.encolar(comparo)
15:                     end if
16:                 end if
17:                 comparo ← siguiente
18:             end if
19:             if siguiente == comparo en altura *2 then
20:                 imprimir solución
21:                 if (el heap no está vacio) then
22:                     heap.encolar(comparo)
23:                 else
24:                     if (comparo no está en el tope del heap) then
25:                         heap.encolar(comparo)
26:                     end if
27:                 end if
28:             end if
29:             if siguiente < comparo en altura *3 then
30:                 heap.encolar(comparo)
31:             end if
32:         end if(no se intersecan comparo y siguiente*4)
33:         if el heap no está vacio then
34:             while heap no vacio do
35:                 if primero.heap termina antes que siguiente *5 then
36:                     desencolar.heap
37:                 else(primero.heap termina despues que siguiente *6)
38:                     imprimir interseccion entre comparo y primero.heap
39:                 end if
40:             end while
```

```
41:         else //no pasé a nadie que cortaría a comparo, y si no corta a comparo tampoco a siguiente, como no se intersecan imprimo ambos puntos
42:             imprimir comparo.Der y 0
43:             imprimir siguiente.Izq y siguiente.Alt
44:             comparo ← siguiente
45:         end if
46:     end for(no hay siguiente,puede que hayan quedado cosas en el heap *7)
    //uso a comparo que es el último edificio con el que haya en el heap
47:     while el heap no sea vacio do
48:         if comparo termina antes que primero.heap *8 then
49:             imprimir intersección comparo y primero.heap
50:             comparo ← heap.primero
51:         else(comparo termina después que el primero.heap *9)
52:             desencolar.heap
53:         end if
54:     end while(ultimo punto no lo imprimo nunca*9)
    //lo imprimo acá
55:     imprimir ultimo punto
56: end procedure
```