Aquæductus

GEI

Grau en Enginyeria Informàtica

2n Curs GEI
Grup 2

13/01/2021

Ton Llucià Senserrich
47125160T

Gerard Llubes Cano
49381025W

Disseny:

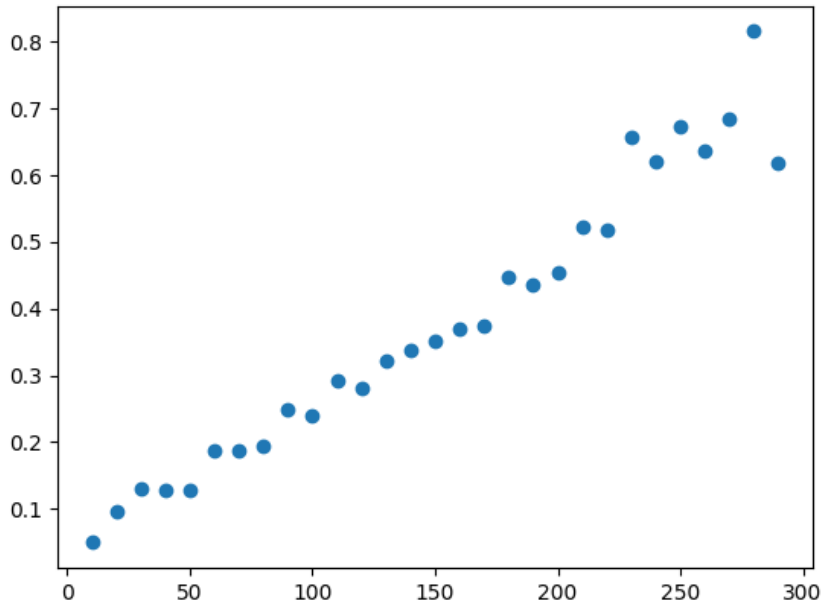
A l'hora de dissenyar els algorismes el primer que hem dut a terme ha estat el pseudocodi, en el qual hem obtingut el que creiem que és el millor disseny per poder-lo després implementar en python.

```
------------------ITERATIU-----------------
DEFINE FUNCTION its_posible_2pilars(height,rects):
    rads=calc_radius([rects[0],rects[-1]])[0]
    FOR i IN range(1,len(rects)):
        IF rects[i][1]>=height
           or not possible_pic(height,rads,[rects[0],rects[-1]],rects[i]):
            RETURN False
    RETURN True

DEFINE FUNCTION its_posible_multi_pilar(height,rects):
    rads=calc_radius(rects)
    hig=0
    FOR i,value IN enumerate(rects):
        IF i==0:
            hig=height-rads[0]
        ELSEIF i==len(rects)-1:
            hig=height-rads[-1]
        ELSE:
            hig=height-max(rads[i],rads[i+1])
        IF value[1]>=height-hig:
            RETURN False
    RETURN True
```

```
---------RECURSIU----------------
DEFINE FUNCTION its_posible_2pilars(height,rects):
    rads=calc_radius([rects[0],rects[-1]])[0]
    RETURN its_posible_2pilars_rec(height,rects[1:-1],rads,[rects[0],rects[-1]])

DEFINE FUNCTION its_posible_2pilars_rec(height,rects,rad,cols):
    IF len(rects)==1:
        RETURN not rects[0][1]>=height and  possible_pic(height,rad,cols,rects[0])
    RETURN ((not rects[0][1]>=height
            and possible_pic(height,rad,cols,rects[0]))
            and its_posible_2pilars_rec(height,rects[1:],rad,cols))

DEFINE FUNCTION its_posible_multi_pilar(height,rects):
    rads=calc_radius(rects)
    RETURN its_posible_multi_pilar_rec(height,rects,rads,True)

DEFINE FUNCTION its_posible_multi_pilar_rec(height,rects,rads,first=False):
    IF len(rects)==1:
        RETURN rects[0][1]>height-rads[0]
    IF first:
        RETURN rects[0][1]>height-rads[0] and its_posible_multi_pilar_rec(height,rects[1:],rads)
    RETURN (rects[0][1]>height-max(rads[0],rads[1])
            and its_posible_multi_pilar_rec(height,rects[1:],rads[1:]))
```

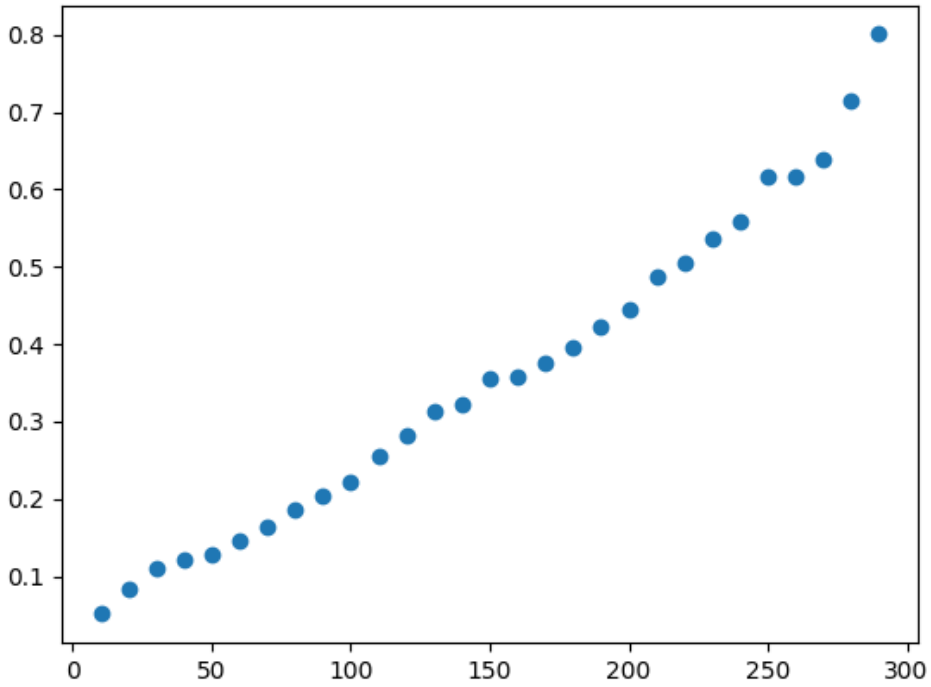
Cost

El cost d'aquest algorisme, com podem comprovar és O(n), ja que només recorrem un cop tot l'array de punts en cada una de les dues funcions utilitzades, per tant podem simplificar el cost de les dues a O(n), aquest és el cost teòric, per tant ens falta analitzaré el cost experimental, obtenim el gràfic:



Com veiem, el cost és més o menys similar a l'esperat. El cost mínim de l'algorisme veiem que és Ω(1), que seria el cas en el qual la primera comprovació de les dues funcions retornes fals, per tant només hauríem recorregut un element.

Per altra banda tenim l'algorisme recursiu el qual té un cost igual que l'iteratiu, ja que només anem escurçant l'array a mesura que fem les crides recursives, per tant tenim un cost màxim de O(n) i un cost mínim de Ω(1), ja que només que falli la primera comprovació ja retornarem fals, igualment hem generat la gràfica, que en teoria hauria de ser molt similar a l'anterior.



Podem veure en la gràfica es comporta com esperàvem, tot i que fins i tot podríem dir que ha donat un resultat més correcte que l'algorisme iteratiu.