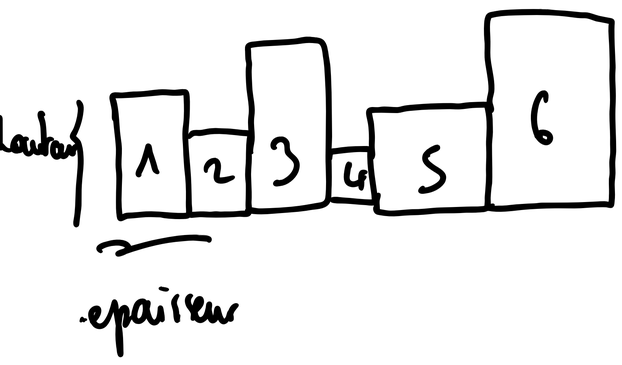
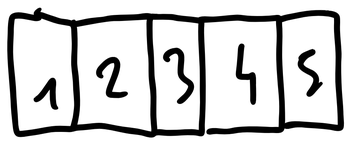
// Exercice 1



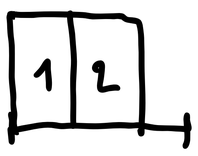
H[1 .. n]

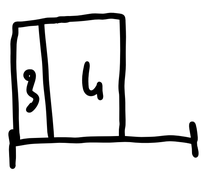
E[1 .. n]

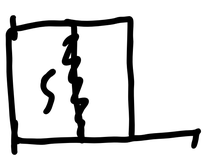
① 

Algo: Pour i=1 à n

- Si il y a la place sur l'étagère actuelle ajouter le livre i-sinon commencer une autre étagère.



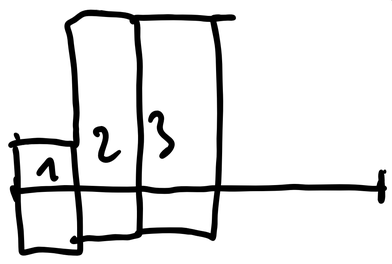




// Si ∃ sol opti != que celle donnée par algo glouton

↳ cad → on peut remonter le livre d'étagère pour retrouver cette sol optimale

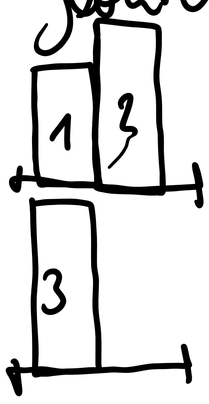
exemple où le glouton ne marche pas:



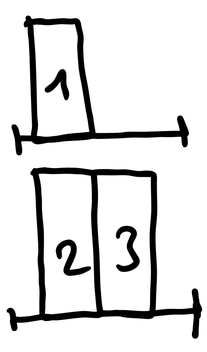
H[1,3,3]

E[1,1,1]

glouton:



solution:



cout(i) = "le cout de ranger les i premiers livres, dans les bacs 1 à i"

def:

$$\begin{cases} cout(0) = 0 \\ cout(i) = \min_{cout(j)} \left(cout(j-1) + \max\left(H[j..i]\right) \right) \text{ pour } i \geq 1 \\ cond(j) = 1 \leq j \leq i \text{ et } \sum_{k=j}^{i} E[k] \leq L \end{cases}$$

j - i

```
def cout(h,e,l):
  n = len(h)
  cout = [0]*n
  cout[0] = h[0]
  for i in range(1,n):
    cout[i] = h[i] + cout[i-1]
    ha = h[i]
    ep = e[i]
    j = i-1
    while j>0 and ep+e[i] <= L
      ep + e[j]
      ha = max(ha, h[j])
      cout[i] = min(cout[i], cout[j-1]+ha)
      j -= 1
  print(cout)
  return cout[n-1]
```

$O(n^2)$ //sans analyser plus finement. // {Prog dynamique.

Exercice 2 : 13 26

commence par le min — max

	20	15	30	5
22	20	2	0	0
13	0	13	0	0
4	0	0	4	0
31	0	0	26	5

$$\sum_{i=1}^{n} L[i] = \sum_{j=1}^{m} C[j]$$

$$\sum_{i=1}^{n} L[i] = \sum_{i=1}^{n}\sum_{j=1}^{m} M[i][j]$$

$$= \sum_{j=1}^{m}\sum_{i=1}^{n} M[i][j] = \sum_{j=1}^{m} C[j]$$

On commence à remplir M[i][i]

On y met min (C[i], L[i])

Si $M[i][j] = C[j]$, alors
on remplit la premier colonne avec 0 et on l'enlève
on met à jour $L[i] = L[i] - C[j]$

Si $M[i][j] = L[i]$ alors
On remplit la premier ligne avec 0, et on l'enlève
on met à jour $C[j] = C[j] - L[i]$

On continue récursivement et (i) reste vrai

// Rq: on peut commencer par ∀ la colonne.

Exercice 3 :

1. ex qui ne marche pas :

	0	2
0		
2		

⇒ ne peut pas donner de matrice binaire comme :

	1	1
0	0	0
2	1	1

La condition a. ajouter (2 cond symétrique) :
- Si $L[i] = k$ il doit y avoir au moins k colonnes avec $C[j] > 0$
- Si $C[j] = k$ il doit y avoir au moins k lignes avec $L[i] > 0$

en plus des cond de l'ex 2

	2	0	2	2
0	0	0	0	0
1	1	0	0	0
2	1	0	1	1
3	1	0	1	1

// oblige cette ligne : il faut prendre le max des val et soustraire 1

Algo : On commence avec la première ligne :
on met L[1] "1" dans la ligne
Le premier "1" est mis dans la colonne j où C[j] est maximum
on met à jour $C[j] = C[j] - 1$ et on continue
on fait la même chose avec les autres lignes
A la dernière ligne on met directement $C[i] = C[n]$