1/2

Even
Rescousse
TSI2

Informatique.

20 -

1) • Pour une ou deux cargaisons
On peut trouver un produit
à rajouter pour augmenter la
valeur du chargement sans dépasser p_{max}

Exemple: $\{1, 4\}$, Poids : 7, Val. = 13
On $\{1, 3, 4\}$, poids : $8 \leq p_{max}$, val.

• La somme des 4 cargaisons fait un
poids de $10 > p_{max}$

2) $\{1, 2, 3\}$ profit de 800 €
$\{2, 3, 4\}$ profit de 1300 €
$\{1, 3, 4\}$ profit de 1400 €

3) La cargaison $\{1, 3, 4\}$ maximise le profit
pour un total de 1400 €

4)
```
def Listes_Produits(n):
    return list(range(1, n+1))
```

5)
```
def Radio(P, V)
    L = []
    for i in range(len(P)):
        L.append(V[i]/P[:])
    return L
```

6) Il y a 3 itérations

1 : L = [3, 5, 2, 1]

2 : L = [2, 3, 5, 1]

3 : L = [1, 2, 3, 5]

7)

```
def Inverse(L):
    n = len(L)
    return [L[n-1-i] for i in range(n)]
```

8) Ces fonctions ne travaille qu'avec des listes individuelle.
Il n'y a pas de lien entre l'indice et le produit donc on ne sait pas quel nom correspond à quel prod.

9)

```
def Tri2(P,V):
    R = Recio(P,V)
    for i in range(1, len(R)):
        x = R[i]      | y = P[i]
        j = i         | z = V[i]
        while j>0 and x < R[j-1]
            R[j] = R[j-1]
            P[j] = P[j-1]
            V[j] = V[j-1]
            j = j-1
        R[j] = x ; P[j] = y, V[j] = z
    return Inverse(P), Inverse V
```

10)

```
def Vmax(P, V, Pmax):
    P2, V2 = Tri2(P, V)
    SP = 0
    SV = 0
    i = 0
    while SP + P2[i] ≤ Pmax and i ≤ len(P2)
        SP += P2[i]
        SV += V2[i]
        i += 1
    return SV
```

1

11)

Ratio = $[\frac{4}{3}, \frac{3}{2}, 1, \frac{9}{4}] \approx [1,33, 1,5, 1, 2,$

Après Tri2 :

$P_2 = [4, 2, 3, 1]$

$V_2 = [9, 3, 4, 1]$

1,5

D'après Vmax le profit maximal est de 12 ce qui n'est pas vrai car on a trouvé 1400 €.

12)

Si i = 0 alors il n'y a pas de profit donc S(i, w) = 0

Si le poids de l'objet i est plus grand que la capacité w alors on ne prend pas donc S(i, w) = S(i-1, w)

Si le poids de l'objet i est plus petit que la charge restante alors on choisit entre le prendre et donner la charge

2 $S(i,w) = V_i + S(i-1, w-p_i)$

ou le laisser :

$S(i,w) = S(i-1, w)$

On recherche le maximum de ces deux choix d'où l'égalité.

13) A chaque appel, le programme donne la valeur de i jusqu'à 0 où il retourne 0 c'est sa terminaison.

1

14)

```
def max(a, b):
    if a >= b:
        return a
    return b
```

1

15)

```
def recu(p, V, i, w):
    if i == 0:
        return 0
    if p[i-1] > w:
        return recu(p, V, i-1, w)
    else:
        return max(recu(p, V, i-1, w),
                   V[i] + recu(p, V, i-1, w-
```

1

p

Évan
Récursivité
TSI2

Informatique

16) La profondeur maximale de I.1 est

1 recon([3,2,1,4], [4,3,1,9], 4, 8)

17)

1 Mémoire = [[-1 for j in range(P_max + 1)] for i in range

18)

```
def recon2(P, V, i, w, Mémoire):
    if i == 0:
        return 0
    if Mémoire[i][w] > -1:
        return Mémoire[i][w]
    if P[i-1] > w:
        Mémoire[i][w] = recon2(P, V, i-1, w, Mémoire)
        return Mémoire[i][w]
    else:
        if Mémoire[i-1][w] == -1
            Mémoire[i-1][w] = recon2(P, V, i-1, w,
        if Mémoire[i-1][w-P[i-1]] == -1
            Mémoire[i-1][w-P[i-1]] = recon(_____)
2   a = max(_____)
    Mémoire[i][w] = a
    return a
```

19) id_client est la clé primaire pour la table livraison

0 (id_client, date)

20)
```
SELECT id_client
FROM livraison
WHERE date = "10-01-2021"
```
1

21)
```
SELECT date, heure
FROM livraison as l, client as c
JOIN client ON l.id_client = c.id
WHERE zone = 5 AND l.date = "02-03-
```
2

22)
```
SELECT count(*)
FROM livraison as li, ~~livreur as lv~~
Join livreur as lv ON li.id_livreur = lv.id
WHERE li.date = "03-02-2021"
      AND lv.jour1 <= 10
      AND lv.jour2 <= 10
      AND lv.jour3 <= 10.
```
1,5

23) 00100111

0,5

24) $\forall i \in [\![0, \text{len}(\text{Bin}) - 1]\!]$

Donc on retourne un nombre 2x trop grand

1

- On doit convertir Bin[i] en entier avant de l'ajouter

25)

```
def Identifiant2 (Bin)
    s = 0
    K = 1
    for i in range (len (Bin)):
        s = s + int (Bin[len(Bin) - 1 - i]) * K
        K = K * 2
    return s
```

26) Il y a 30 gnous donc 30 combinaisons possibles.

$2^5 = 32$ donc 5 bits suffisent.