N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction.
Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

Le sujet est d'une durée de 3 h. Il est composé de 3 parties indépendantes

Optimisation de rendement d'une entreprise de livraison

Une entreprise de livraison dispose de plusieurs locaux en France et chacun possède plusieurs camions de livraisons. Celle-ci souhaite optimiser le chargement de ses camions pour diminuer ses frais de fonctionnement.

Partie I - Optimisation du chargement

Chaque camion de l'entreprise peut charger une cargaison jusqu'à un poids maximal noté P_{max}. L'entreprise dispose de différentes informations provenant de ses clients :
- le poids de chaque produit p_i (chaque client propose un seul produit) ;
- la valeur v_i associée au transport de chaque produit : c'est-à-dire l'argent gagné par l'entreprise si elle réalise le transport de ce produit.

En considérant que l'entreprise dispose de n clients, l'entreprise cherche donc à trouver une liste d'indices notée I contenue dans $\{1, \dots, n\}$ telle que :

$$\sum_{i \in I} p_i \leq P_{max} \quad \text{: respect du poids maximal} \qquad [\boldsymbol{eq.1}]$$

et

$$\arg max \{\sum v_i \,|i \in I\} \quad \text{: optimisation du profit pour l'entreprise} \qquad [\boldsymbol{eq.2}]$$

Dans toute la suite, les poids seront donnés en centaines de kilogrammes et les valeurs en centaines d'euros.

I.1 – Modélisation du problème : un exemple

On suppose que $n = 4$ et que $P_{max} = 8$ centaines de kilogrammes. On stocke alors les différentes informations dans trois listes :
- `Pr` est la liste des produits proposés par les clients numérotés de 1 à 4 : `Pr = [1, 2, 3, 4]` ;
- P est la liste des poids associés : P = [3, 2, 1, 4] ;
- V est la liste des valeurs associées : V = [4, 3, 1, 9].

Par exemple, le produit 2 a un poids de deux centaines de kilogrammes et une valeur de trois centaines d'euros.

Q1. /2 Donner toutes les cargaisons de trois produits respectant le poids maximal. On donnera à chaque fois le profit fait par l'entreprise.

Q2. /1 Quelle est la cargaison maximisant le profit de l'entreprise ? Que vaut le profit dans ce cas ?

I.2 - Une méthode intuitive pour la résolution du problème

Gardons les notations de la partie précédente, soit :
- P_r est la liste des produits (numérotés de 1 à n inclus),
- $P = [p_1, \dots, p_n]$ est la liste des poids associés aux produits,
- $V = [v_1, \dots, v_n]$ est la liste des valeurs associées aux produits.

Q3. /2 Définir une fonction `ListeProduits` ayant pour argument un entier naturel non nul `n` et qui retourne la liste `Pr`.

Une méthode intuitive pour tenter d'optimiser le profit de l'entreprise est la suivante : on calcule les ratios $\frac{v_i}{p_i}$, puis on trie les objets par ordre décroissant suivant ces valeurs. Les produits sont alors classés par rentabilité : le premier produit devient le plus rentable " au poids " et ainsi de suite. On ajoute progressivement chaque produit dans la cargaison, dans cet ordre, sans dépasser la limite du poids maximal.

Q4. /2 Définir une fonction `Ratio` ayant pour arguments deux listes `P`, `V` où `P` et `V` correspondent respectivement à la liste des poids et des valeurs, renvoyant la liste des ratios $\frac{v_i}{p_i}$.

La fonction suivante est associée à une méthode de tri :

```
def Tri(L):
    '''L est une liste de nombres réels'''
    for i in range(1,len(L)):
        x=L[i]
        j=i
        while j>0 and x<L[j-1]:
            L[j]=L[j-1]
            j=j-1
        L[j]=x
    return L
```

Q5. /2 On exécute `Tri(L)` avec `L=[3, 5, 2, 1]`. Combien y a-t-il d'itérations de la boucle for ? Donner la valeur de L à la fin de chaque itération de la boucle for.

Q6. /1 Ce tri fonctionnerait-t-il pour une chaîne de caractères dont chaque caractère est un entier ? Justifier.

Q7. /2 Quelle est la méthode de tri utilisée dans la fonction `Tri` ? Donner sa complexité en nombre de comparaisons et en utilisant la notation de Landau dans le pire des cas. On justifiera soigneusement ces résultats.

Q8. /2 Définir une fonction `Inverse` ayant pour argument une liste de nombres réels L et renvoyant l'inverse de celle-ci. Par exemple, l'inverse de $[1, 5, 3, 4]$ est $[4, 3, 5, 1]$.

La syntaxe `L[::-1]` n'est pas autorisée.

Q9. /1 On souhaite trier une liste de poids P et une liste de valeurs V associées à une liste de produits en suivant l'ordre décroissant de la liste des ratios $\frac{v_i}{p_i}$. Justifier que les fonctions `Ratio`, `Tri` et `Inverse` ne permettent pas de répondre simplement au problème posé.

Q10. /4 Écrire, à l'aide des fonctions `Ratio` et `Inverse`, une fonction `Tri2` ayant pour arguments une liste de poids P et une liste de valeurs V associées à une liste de produits. Cette fonction renverra 2 listes ; les listes des poids et des valeurs triées par ordre décroissant de la liste des ratios.

La construction de cette fonction peut s'inspirer avantageusement de la fonction `Tri` rappelée à la question Q3.

Q11. /3 Ecrire une fonction `Vmax` qui a pour arguments les listes des poids P, des valeurs V et le poids maximal `Pmax` du chargement et qui renvoie la valeur maximale du profit de l'entreprise en suivant la méthode intuitive proposée.

Q12. /2 On souhaite appliquer cette méthode en utilisant les listes des poids et des valeurs définies en I.1. Donner la liste des ratios, les listes des poids et des valeurs obtenues à l'aide de la fonction `Tri2` ainsi que le profit obtenu. La méthode intuitive proposée est-elle optimale ou sous-optimale ? Commenter et justifier votre réponse.

I.3 - Une méthode récursive

Nous gardons les notations de la partie I.2 soit `Pr`, P et V Afin de simplifier l'étude, considérons que les poids des produits sont des entiers, ainsi que P_{max}.

Nous introduisons une méthode récursive pour résoudre ce problème d'optimisation :

- pour chacun des produits, deux choix sont possibles : il fait partie de la cargaison ou non,
- la récursivité s'effectuera sur la liste des indices de `Pr` : le premier appel de la fonction se fera en utilisant l'indice n, puis l'indice $n - 1$ et ainsi de suite jusqu'à l'indice 0 (correspondant au cas où il n'y a plus de produits),
- pour $i \in \{0, 1, \ldots, n\}$ et $\omega \in \{0, 1, \ldots, P_{max}\}$, on note $S(i, \omega)$ la valeur maximale cumulée des produits que l'on peut placer dans un camion d'une capacité maximale (en poids) de ω avec la liste constituée des i premiers produits de P_r.

On pose alors la relation de récursivité suivante :

$$S(i,\omega) = \begin{cases} 0 & si\ i = 0 \\ S(i-1,\omega) & si\ i > 0\ et\ p_i > \omega \\ max(S(i-1,\omega), v_i + S(i-1,\omega-p_i)) & si\ i > 0\ et\ p_i \leq \omega \end{cases} \qquad [\boldsymbol{eq.3}]$$

Q13. /2 Justifier les relations précédentes dans les trois cas.

Q14. /1 Justifier la terminaison de l'algorithme associé à la relation de récursivité précédente, sachant que la première valeur donnée pour i sera n et la première valeur pour ω sera `Pmax`.

Q15. /2 Définir une fonction `Max` ayant pour arguments deux réels et renvoyant le maximum parmi ces deux valeurs. Il est interdit d'utiliser la fonction `max` prédéfinie dans Python.

Q16. /2 En vous basant sur $[\boldsymbol{eq.3}]$, compléter la définition de la fonction récursive `recur` ci-dessous ayant pour arguments les listes de poids et de valeurs P et V, un indice i, un poids ω, et renvoyant $S(i, \omega)$.

```
def recur(P,V,i,w) :
    if i==0:
        return..................
    if P[i-1]>w:
        return recur(P,V,i-1,w)
    else :
        ...................
```

Q17. /1 Donner une série d'instructions utilisant la fonction `recur` et permettant de déterminer le profit de la sous-partie I.1

Partie II - Données liées aux livraisons conservées par l'entreprise

À chaque livraison, l'entreprise stocke des données relatives à celle-ci. L'entreprise dispose de 20 locaux, numérotés de 1 à 20, disposant chacun d'un certain nombre de camions. Pour faciliter ses livraisons, l'entreprise découpe la France en 30 zones et associe à chaque local, trois zones possibles de livraisons.
Le client est identifié par un nombre codé en binaire sur 8 bits. Ainsi, le code '00010111' est associé au client dont l'identifiant est le numéro 23

Q18. /1 Donner le codage associé au client 39.

Afin de retrouver l'identifiant de chaque client à l'aide de son code binaire Bin typé str, la fonction suivante est proposée :

```
1    def Identifiant(Bin):
2        '''Bin est est une chaine de caractères constituée de 0 et 1'''
3        S=0
4        for i in range(len(Bin)) :
5            S = S + Bin[i]*2**(len(Bin)-i)
6        return S
```

Q19. /2 Trouver les deux erreurs dans le code de la fonction précédente.

On suppose maintenant la fonction précédente corrigée. La ligne 5 pose un problème de complexité : à chaque itération de boucle, la puissance de 2 est recalculée entièrement.

Q20. /4 Écrire une fonction `Identifiant2`, qui donne le même résultat que la fonction Identifiant avec une meilleure complexité. Le nombre de multiplications devra être linéaire suivant la longueur de `Bin`.

Chut, il faut connaître le schéma de Horner présenté en cours, …

Q21. /1 Si l'entreprise souhaite aussi stocker la zone de chaque client en codage binaire, donner le nombre de bits minimal nécessaire.

Partie III - Optimisation du trajet

Le chargement une fois effectué, il s'agit ensuite de déterminer le trajet optimal qui minimise la distance parcourue. Le graphe ci-dessous est le modèle du réseau routier des différents sites à livrer. Ses sommets représentent les lieux de livraison et les arêtes, les distances exprimées en kilomètres entre ces différents lieux.

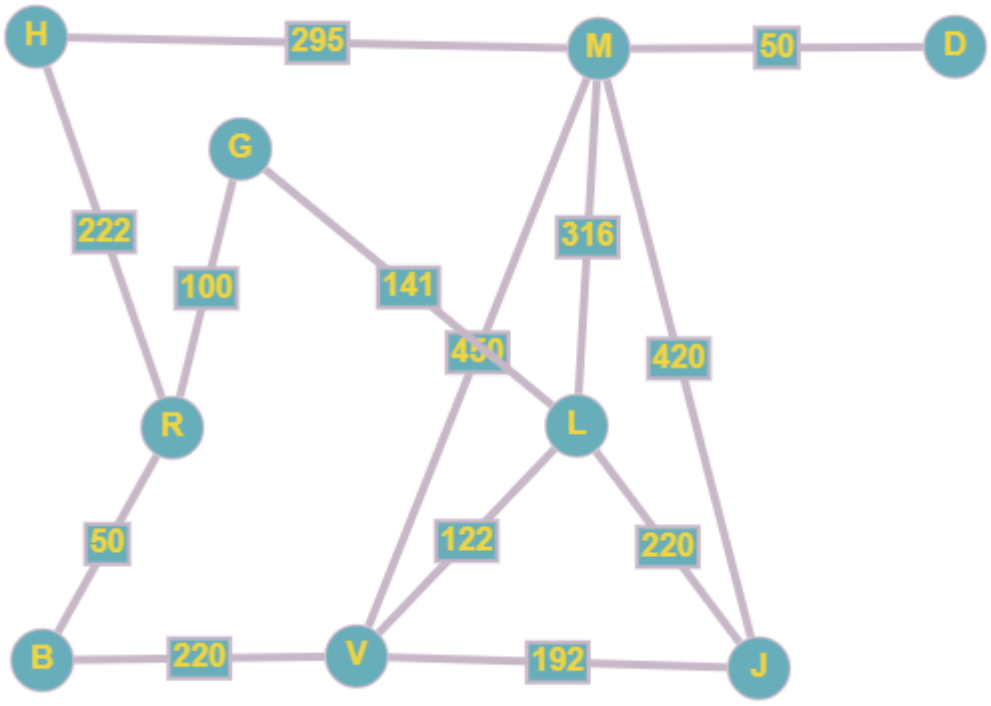


Figure 1 : modèle routier des sites à livrer

Lorsque le livreur termine son parcours en D, il souhaite revenir en B le plus rapidement possible.

On rappelle l'algorithme de Dijkstra :

Algorithme $[Dist,Pred]$=$\boldsymbol{Dijkstra}(G,s)$

Données : Soit un graphe orienté valué graphe G = (V, E, ω), d'ordre n =Card(V) et de taille m=Card(E), et x un sommet de G. L'algorithme construit deux matrices de taille 1×n :
$Dist$: matrice des distances telle que $Dist(y)$= distance optimale de x à y
$Pred$: matrice des prédécesseurs telle que $Pred(y)$= prédécesseur de y dans le chemin optimal depuis x

$[Dist,Pred]$= $\boldsymbol{Dijkstra}\ (G,s)$
    n est le nombre de sommets de G
    $Pred$: tableau des prédécesseurs initialisé à 0
    $Dist$: tableau des distances initialisé à +∞ (sauf $Dist(s)=0$)
    ω tableau des poids des arcs initialisé à +∞ si l'arc n'existe pas
    $C=\{1,\dots,n\}$: liste des sommets restant à traiter
    $D=\emptyset$: liste des sommets déjà traités
    tant que $C\neq 0$ faire
        $x\leftarrow$ sommet de C le plus proche de s
        Retirer le sommet x de C et le placer dans D
        pour chaque sommet $\boldsymbol{y\in C}$ et plus proche voisin de x faire
            si $Dist[y]>Dist[x]+\omega(x,y)$ alors
                $Dist[y]\leftarrow Dist[x]+\omega(x,y)$
                $Pred[y]\leftarrow x$
            fin si
        fin pour
    fin tant que

y∈C sont les ppv de x

Q22. /5 Déterminer ce plus court chemin en utilisant l'algorithme de Dijkstra. Compléter pour cela le document réponse DR1 en précisant les valeurs de $\boldsymbol{x, D, C, Dist}$ et $\boldsymbol{Pred}$ à la fin de chaque itération de la boucle «tant que»

Q23. /1 Quelle est la distance parcourue en km pour le trajet de retour ?

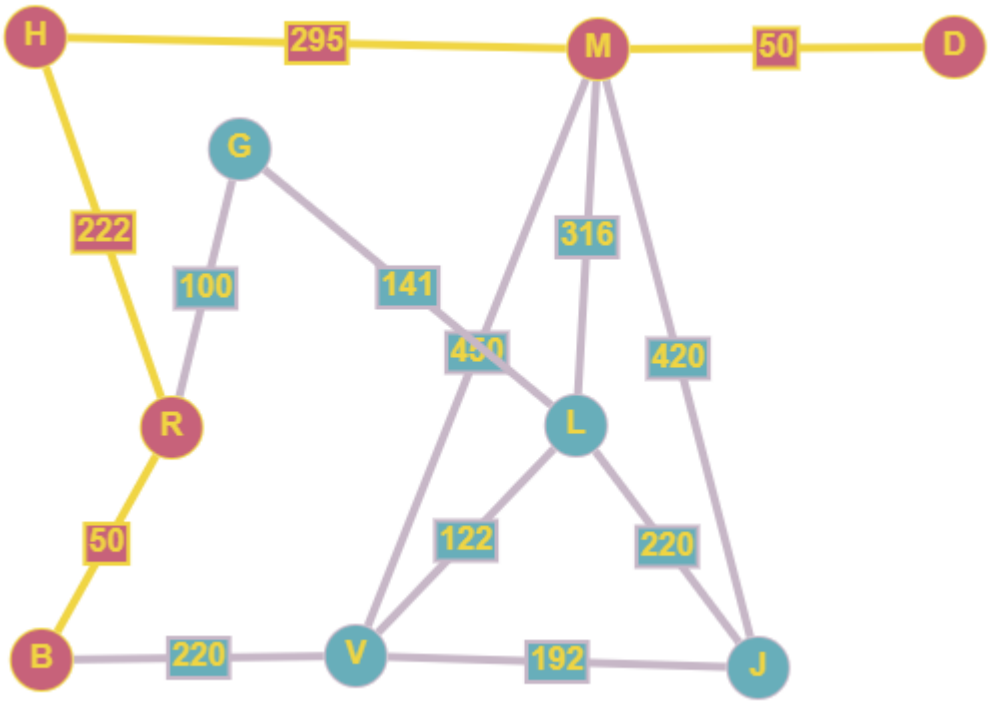
DR1 : correction

x	D	C	$Dist$									$Pred$								
			B	D	G	H	J	L	M	R	V	B	D	G	H	J	L	M	R	V
0	$\{\ \}$	$\{B,D,G,H,J,L,M,R,V\}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	0	0	0	0	0	0	0	0
D	$\{D\}$	$\{B,G,H,J,L,M,R,V\}$	∞	0	∞	∞	∞	∞	50	∞	∞	0	0	0	0	0	0	D	0	0
M	$\{D,M\}$	$\{B,G,H,J,L,R,V\}$	∞	0	∞	345	470	366	50	∞	500	0	0	0	M	M	M	D	H	M
H	$\{D,M,H\}$	$\{B,G,J,L,R,V\}$	∞	0	∞	345	470	366	50	567	500	0	0	0	M	M	M	D	H	M
L	$\{D,M,H,L\}$	$\{B,G,J,R,V\}$	∞	0	507	345	470	366	50	567	488	0	0	L	M	M	M	D	H	L
J	$\{D,M,H,L,J\}$	$\{B,G,R,V\}$	∞	0	507	345	470	366	50	567	488	0	0	L	M	M	M	D	H	L
V	$\{D,M,H,L,J,V\}$	$\{B,G,R\}$	708	0	507	345	470	366	50	567	488	V	0	L	M	M	M	D	H	L
R	$\{D,M,H,L,J,V,G\}$	$\{B,R\}$	708	0	507	345	470	366	50	567	488	V	0	L	M	M	M	D	H	L
G	$\{D,M,H,L,J,V,G,R\}$	$\{B\}$	617	0	507	345	470	366	50	567	488	R	0	L	M	M	M	D	H	L
B	$\{D,M,H,L,J,V,G,R,B\}$	$\{\ \}$	**617**	0	507	345	470	366	50	567	488	R	0	L	M	M	M	D	H	L

Coût optimal : 617

Chemin optimal : $D \rightarrow M \rightarrow H \rightarrow R \rightarrow B$

(On lit dans la dernière ligne que le prédécesseur de B est R, dont le prédécesseur est H, dont le prédécesseur est M, dont le prédécesser est D)


H
295
M
50
D
G
222
316
100
141
450
420
R
L
50
122
220
B
220
V
192
J