PSI	DS d'informatique N°1	Corrigé

Exercice 1 : Base de données d'hotel

Q1 - Lister les numéros des chambres de l'hôtel ayant au moins trois couchages avec leur nombre de couchages.

```
SELECT CHB_NUMERO AS Numéro, CHB_COUCHAGE AS Nombre
  FROM T_CHAMBRE
  WHERE CHB_COUCHAGE>=3
```

Q2 - Donner la capacité moyenne des chambres de l'hôtel.

```
SELECT AVG(CHB_COUCHAGE) AS "Capacité moyenne des chambres"
  FROM T_CHAMBRE
```

Q3 - Lister les étages et leur nombre de chambres.

```
SELECT CHB_ETAGE AS Etage, COUNT(*) AS "Nombre de chambres"
  FROM T_CHAMBRE
  GROUP BY Etage
```

Q4 - Quel sont les étages qui ont une capacité moyenne des chambres supérieures ou égales à 2,5 ? Lister les étages et leur capacité moyenne des chambres.

```
SELECT CHB_ETAGE AS Etage, AVG(CHB_COUCHAGE) AS "Capacité moyenne"
  FROM T_CHAMBRE GROUP BY Etage
  HAVING "Capacité moyenne">=2.5
```

Q5 - Lister les noms et prénoms des clients ayant donné une adresse personnelle à PARIS.

```
SELECT C.CLI_NOM AS Nom, C.CLI_PRENOM AS Prénom
  FROM T_CLIENT AS C
  JOIN T_ADRESSE AS A  ON C.CLI_ID=A.CLI_ID
  WHERE A.ADR_VILLE="PARIS" and A.ADR_LOCAL="personnelle"
```

Q6 - Lister les noms et prénoms des clients qui ont donné une adresse personnelle et une adresse professionnelle.

```
SELECT C.CLI_NOM AS Nom, C.CLI_PRENOM AS Prénom
  FROM T_CLIENT AS C JOIN T_ADRESSE AS A  ON C.CLI_ID=A.CLI_ID
  WHERE A.ADR_LOCAL="professionnelle"
INTERSECT
SELECT C.CLI_NOM AS Nom, C.CLI_PRENOM AS Prénom
  FROM T_CLIENT AS C JOIN T_ADRESSE AS A  ON C.CLI_ID=A.CLI_ID
  WHERE A.ADR_LOCAL="personnelle"
```

Q7 - Donner la liste des personnes (Nom Prénom et numéro de téléphone) qui occupaient une chambre au premier étage de l'hôtel le 11 septembre 2025 ('2025-09-11')

```
SELECT C.CLI_NOM AS Nom, C.CLI_PRENOM AS Prénom
  FROM T_CLIENT AS C
  JOIN TJ_CHB_PLN_CLI AS TJ ON TJ.CLI_ID=C.CLI_ID
  JOIN T_CHAMBRE AS C ON C.CHB_ID=TJ.CHB_ID
  WHERE TJ.CHB_PLN_CLI_OCCUPE=1 AND TJ.PLN_JOUR='2025-09-11'
        AND C.CHB_ETAGE='1er'
```

Exercice 2 : Optimisation de la furtivité de la trajectoire d'un drône

Q1 - A quoi correspond le dictionnaire retourné par cette fonction ?

Ce dictionnaire est le dictionnaire des risques liés à tous les déplacements élémentaires de type 1 (Déplacement vers l'Est entre le centre d'un carré vers le centre du carré voisin vers l'Est)

Q2 - Compléter le code de la fonction `def dic_risques_type2(zone:list) -> dict` (lignes 4 à 6) qui retourne le dictionnaire de tous les déplacements élémentaires de type 2.

```
def dic_risques_type2(zone):
    Dico={}
    p,q=len(zone)-1,len(zone[0])-1
    for i in range(1,p):
        for j in range(0,q):
            Dico[((2*i-1,2*j+1),(2*i+1,2*j+1))]=zone[i][j]+zone[i][j+1]
    return Dico
```

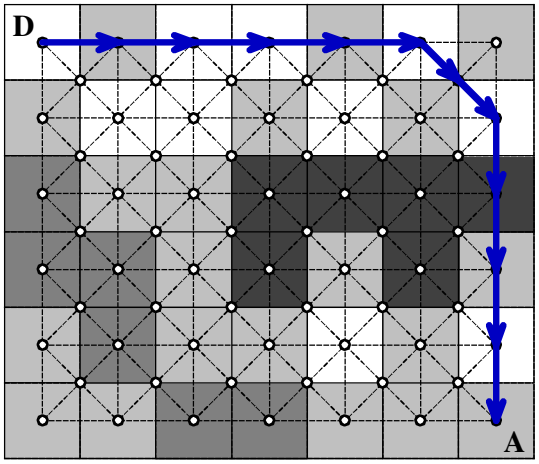
Q3 - Compléter le code de la fonction `def dic_risques_type3(zone:list) -> dict` (lignes 4 à 6) qui retourne le dictionnaire de tous les déplacements élémentaires de type 3.

```
def dic_risques_type3(zone):
    Dico={}
    p,q=len(zone)-1,len(zone[0])-1
    for i in range(p+1):
        for j in range(q):
            Dico[((2*i,2*j),(2*i,2*j+2))]=zone[i][j]+zone[i][j+1]
    return Dico
```

Q4 - Compléter le code de la fonction `def dic_risques_type5(zone:list) -> dict` (lignes 4 à 6) qui retourne le dictionnaire de tous les déplacements élémentaires de type 5.

```
def dic_risques_type5(zone):
    Dico={}
    p,q=len(zone)-1,len(zone[0])-1
    for i in range(p):
        for j in range(q):
            Dico[((2*i,2*j),(2*i+1,2*j+1))]=zone[i][j]*np.sqrt(2)
    return Dico
```

Q5 - Tracer sur le document réponse la trajectoire liée à la liste `['E', 'E', 'E', 'E', 'E', 'SE', 'SE', 'S', 'S', 'S', 'S']` puis calculer le risque lié à cette trajectoire.



Risque liée à la trajectoire :

1 + 1 + 0 + 1 + 1 + 0 + 0 + 3 + 4 + 1 + 1
= 13

Q6 - Implémenter la fonction `calcul_risque_trajectoire(L_directions :list, DR_deplac :dict)->float` qui prend en argument la liste `L_directions` d'une trajectoire, le dictionnaire `DR_deplac` des risques des déplacements élémentaires et qui retourne le risque lié à cette trajectoire.

```
def calcul_risque_trajectoire(L_directions,DR_deplac):
    risque_trajectoire=0
    i,j=0,0
    for direction in L_directions:
        if direction=='E':
            risque_trajectoire+=DR_deplac[((i,j),(i+2,j))]
            i+=2
        if direction=='S':
            risque_trajectoire+=DR_deplac[((i,j),(i,j+2))]
            j+=2
        if direction=='SE':
            risque_trajectoire+=DR_deplac[((i,j),(i+1,j+1))]
            i+=1
            j+=1
    return risque_trajectoire
```

Q7 - Quelle serait la complexité d'un algorithme utilisant cette stratégie ? Pourquoi va-t-on exclure cette stratégie algorithmique ?

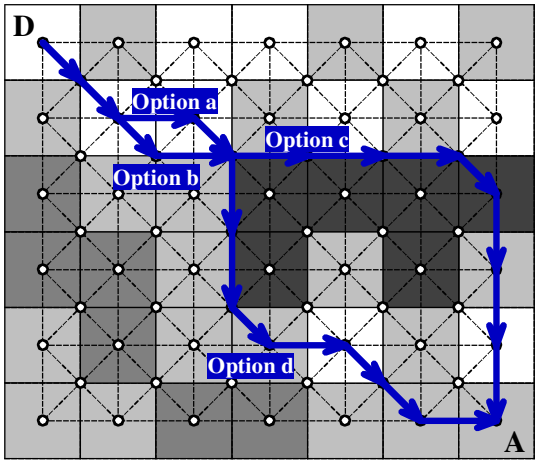
Il y a au total p+q déplacement élémentaires entre D et A. A chaque déplacement (sauf cas particuliers lorsqu'on est trop à l'Est ou au Sud) on a trois possibilités de direction à prendre (Est, Sud ou Sud-Est). Donc il y 3^{p+q} trajectoires possibles. Donc cet algorithme est en complexité $O(3^{p+q})$.

La complexité d'un tel algorithme étant de complexité exponentielle si les dimensions p et q augmentent la complexité sera rapidement trop importante pour finir l'algorithme dans un temps raisonnable.

Q8 - De quel type d'algorithme relève cette stratégie ?

C'est un algorithme glouton.

Q9 - Tracer sur le document réponse la trajectoire du drône obtenue avec cette stratégie. Donner également le risque total résultant de votre trajectoire (utiliser $\sqrt{2} \approx 1{,}4$). Remarque : Il peut y avoir plusieurs réponses possibles, une seule est demandée.



Risque liée à la trajectoire :

Trajectoire avec les options a et c

0 + 0 + 0 + 0 + 4 + 3 + 4 + 4,2 + 4 + 1 + 1 = 21,2

Trajectoire avec les options b et c

0 + 0 + 0 + 1 + 4 + 3 + 4 + 5,6 + 2 + 4 + 1 = 22,2

Trajectoire avec les options a et d

0 + 0 + 0 + 0 + 4 + 4 + 1,4 + 1 + 0 + 1,4 + 2 = 13,8

Trajectoire avec les options b et d

0 + 0 + 0 + 1 + 4 + 4 + 1,4 + 1 + 0 + 1,4 + 2 = 14,8

Q10 - Implémenter une fonction **`direction_moins_risquee(DR_deplac:dict, i:int,j:int,p:int,q:int)->str`** qui prend en argument un dictionnaire **`DR_deplac`** des risques de tous les déplacements élémentaires, une position donnée par les entiers **`i`** et **`j`**, la dimension de la zone ((2.p+1)×(2.q+1)) donnée par les entiers **`p`** et **`q`**, et qui retourne la direction (**`'E'`**,**`'S'`** ou **`'SE'`**) correspondant au déplacement élémentaire dont le risque est minimal.

```
def direction_moins_risquee(DR_deplac,i,j,p,q):
    risque_mini=np.inf
    if i<=2*p-2 and DR_deplac[((i,j),(i+2,j))]<=risque_mini:
        direction='E'
        risque_mini=DR_deplac[((i,j),(i+2,j))]
    if j<=2*q-2 and DR_deplac[((i,j),(i,j+2))]<=risque_mini:
        direction='S'
        risque_mini=DR_deplac[((i,j),(i,j+2))]
    if i<=2*p-1 and j<=2*q-1 and
                        DR_deplac[((i,j),(i+1,j+1))]<=risque_mini:
        direction='SE'
    return direction
```

Q11 - Compléter le code de cette fonction **`strategie1(zone:list)`**.

```
def strategie1(zone):
    DR_deplac=dic_risques(zone)
    trajectoire=[]
    i,j=0,0
    p,q=len(zone)-1,len(zone[0])-1
    for k in range(p+q):    # ou while (i,j)!=(2*p,2*q) :
        direction=direction_moins_risquee(DR_deplac,i,j,p,q)
        trajectoire.append(direction)
        if direction=='E':
            i+=2
        if direction=='S':
            j+=2
        if direction=='SE':
            i+=1
            j+=1
    risque =calcul_risque_trajectoire(trajectoire,DR_deplac)
    return risque,trajectoire
```

Q12 - Cette stratégie est-elle optimale ? Justifier la réponse.

On a vu à la question 10 que cette stratégie gloutonne donnait au mieux un risque minimal de 13,8. Or à la question 5 on a calculé le risque lié à une trajectoire dont le risque était de 13. Donc cette stratégie n'est pas optimale

Q13 - De quel type d'algorithme relève cette stratégie ?

C'est un algorithme de programmation dynamique.

Q14 - Que retourne cette fonction ? Donner le type des éléments retournés.

Cette fonction retourne un couple : (Risque lié à la trajectoire optimale, trajectoire optimale). C'est un couple (float,list)

Q15 - Quelles sont les lignes du code qui reprennent l'équation de Bellman de l'algorithme.

Les lignes de code qui reprennent l'équation de Bellman sont les lignes 8 à 17.

Q16 - De quel type est la sous fonction **`traj_opt(i,j)`** ?

C'est une fonction récursive car cette sous fonction fait appel à elle-même.

Q17 - Qu'est-ce qui justifie cela. Justifier votre réponse à partir du code de cette fonction donné ci-dessus.

A chaque déplacement élémentaire, la sous fonction fait appel à elle-même 3 fois (lignes 9, 12 et 15).

Donc comme il y a p+q déplacements élémentaire la complexité est exponentielle : $O(3^{p+q})$. Cela justifie donc que lorsque p+q augmente le temps d'exécution devient rapidement inacceptable.

Q18 - Implémenter une fonction **`strategie2_memoisation(zone)`** similaire à la fonction **`strategie2(zone)`** mais qui a une complexité linaire. Vous utiliserez pour cela un dictionnaire **`D_traj_opt`** dont les clefs sont les coordonnées des positions et les valeurs les trajectoires optimales pour rejoindre ces positions à partir du point D. La première clef de ce dictionnaire est **`(0,0)`** et la valeur correspondante **`[]`**. Soit : **`D_traj_opt[(0,0)]=[]`**.

En mémoïsant les trajectoires optimales dans le dictionnaire `D_traj_opt` nous n'aurons pas besoin de refaire ce calcul un nombre exponentiel de fois mais au maximum 2.p×2.q fois. Pour implémenter cela il faut d'abord mémoïser la trajectoire optimale de D à D : `D_traj_opt[(0,0)]`. Puis ensuite ne calculer une trajectoire optimale, que si elle n'est pas déjà dans le dictionnaire. On obtient alors le code ci-dessous (Les Modifications de ce code par rapport au code `strategie2(zone)` , sont soulignées :

```
def strategie2_memoisation(zone):
    DR_deplac=dic_risques(zone)
    p,q=len(zone)-1,len(zone[0])-1
    D_traj_opt={(0,0):[]}
    def traj_opt(i,j):
        if (i,j) in D_traj_opt:
            return D_traj_opt[(i,j)]
        Ropt_O,Ropt_N,Ropt_NO=np.inf,np.inf,np.inf
        if i>=2:
            if (i-2,j) not in D_traj_opt:
                D_traj_opt[(i-2,j)]=traj_opt(i-2,j)
            Topt_O=D_traj_opt[(i-2,j)]+['E']
            Ropt_O=calcul_risque_trajectoire(Topt_O,DR_deplac)
        if j>=2:
            if (i,j-2) not in D_traj_opt:
                D_traj_opt[(i,j-2)]=traj_opt(i,j-2)
            Topt_N=D_traj_opt[(i,j-2)]+['S']
            Ropt_N=calcul_risque_trajectoire(Topt_N,DR_deplac)
        if i>=1 and j>=1:
            if (i-1,j-1) not in D_traj_opt:
                D_traj_opt[(i-1,j-1)]=traj_opt(i-1,j-1)
            Topt_NO=D_traj_opt[(i-1,j-1)]+['SE']
            Ropt_NO=calcul_risque_trajectoire(Topt_NO,DR_deplac)
        Ropt=min(Ropt_O,Ropt_N,Ropt_NO)
        if Ropt==Ropt_O:
            Topt=Topt_O
        if Ropt==Ropt_N:
            Topt=Topt_N
        if Ropt==Ropt_NO:
            Topt=Topt_NO
        return Topt
    Topt_finale=traj_opt(2*p,2*q)
    Ropt_final=calcul_risque_trajectoire(Topt_finale,DR_deplac)
    return Ropt_final,Topt_finale
```