1 Correction du TD 10

Exercice 1

```
vil = { }
for row in df.values :
    vil [row[0]] = 0
    vil [row[1]] = 1
vil = list(vil.keys())
print (len(vil))
```

Exercice 2

```
dist = { }
for row in df.values :
    a = row[0]
    b = row[1]
    dist[a,b] = dist[b,a] = row[2]
print (len(dist))
```

```
print ( dist["Charleville-Mezieres","Bordeaux"] )
```

Seconde demi-heure : L'algorithme du plus court chemin (Dikjstra)

Exercice 3

On peut remplr facilement toutes les cases correspondant aux villes reliées à Charleville-Mezieres.

```
d = { }
d['Charleville-Mezieres'] = 0
for v in vil : d[v] = 1e10
for v,w in dist :
    if v == 'Charleville-Mezieres':
        d[w] = dist[v,w]
```

Exercice 4

Si on découvre que $d[w] > d[v] + dist[w, v]$, cela veut dire qu'il faut mettre à jour le tableau d car il ne contient pas la distance optimale. On passe en revue toutes les distance connue :

```
for v,w in dist :
    d2 = d[v] + dist[v,w]
    if d2 < d[w] :
        d[w] = d2
```

On trouve 829140 mètres pour la distance (Charleville-Mezieres, Bordeaux). Ce n'est pas forcément la meilleure. Pour être sûr, il faut répéter la même itération autant de fois qu'il y a de villes (car le plus long chemin contient autant d'étapes qu'il y a de villes).

```
for i in range(0,len(d)) :
    for v,w in dist :
        d2 = d[v] + dist[v,w]
        if d2 < d[w] :
            d[w] = d2
```

On trouve 795670 mètres.

Troisième demi-heure : La répartition des skis

Exercice facultatif

Pour montrer que l'algorithme suggéré permettra d'obtenir la solution optimale, il faut montrer qu'il n'est pas nécessaire d'envisager aucun autre ordre que celui des skieurs et des paires triés par taille croissante. On considère donc un appariement σ qui associé le skieur t_i à la paire $s_{\sigma(i)}$. Il suffit que montrer que :

$$\forall i, j,\ t_i \leqslant t_j \iff s_{\sigma(i)} \leqslant s_{\sigma(j)}$$

Pour montrer cela, on fait un raisonnement par l'absurde : pour i et j quelconques, on suppose qu'il existe un appariement optimal tel que $t_i > t_j$ et $s_{\sigma(i)} \leqslant s_{\sigma(j)}$. Le coût $C(\sigma)$ de cet appariement est :

$$C(\sigma) = \sum_{k=1}^{N} \left|t_k - s_{\sigma(k)}\right| = \alpha + \left|t_i - s_{\sigma(i)}\right| + \left|t_j - s_{\sigma(j)}\right| \tag{1}$$

Le coût de l'appariement en permutant les skieurs i et j (donc en les rangeant dans l'ordre croissant) est :

$$C(\sigma') = \sum_{k=1}^{N} \left|t_k - s_{\sigma(k)}\right| = \alpha + \left|t_j - s_{\sigma(i)}\right| + \left|t_i - s_{\sigma(j)}\right| \tag{2}$$

On calcule :

$$C(\sigma) - C(\sigma') = \left|t_i - s_{\sigma(i)}\right| + \left|t_j - s_{\sigma(j)}\right| - \left|t_j - s_{\sigma(i)}\right| - \left|t_i - s_{\sigma(j)}\right| \tag{3}$$

Premier cas : $t_j \geqslant s_{\sigma(i)}$ et $t_i > t_j \geqslant s_{\sigma(i)}$

$$\begin{aligned}
C(\sigma) - C(\sigma') &= \left|t_i - s_{\sigma(i)}\right| + \left|t_j - s_{\sigma(j)}\right| - \left(t_j - s_{\sigma(j)} + s_{\sigma(j)} - s_{\sigma(i)}\right) - \left|t_i - s_{\sigma(j)}\right| && (4)\\
&= t_i - s_{\sigma(i)} + \left|t_j - s_{\sigma(j)}\right| - \left(t_j - s_{\sigma(j)}\right) - \left(s_{\sigma(j)} - s_{\sigma(i)}\right) - \left|t_i - s_{\sigma(j)}\right| && (5)\\
&= t_i - s_{\sigma(i)} - \left|t_i - s_{\sigma(i)}\right| + \left|t_j - s_{\sigma(j)}\right| - \left(t_j - s_{\sigma(j)}\right) && (6)\\
&= \left|t_j - s_{\sigma(j)}\right| - \left(t_j - s_{\sigma(j)}\right) && (7)\\
&\geqslant 0 && (8)
\end{aligned}$$

Second cas : $t_j \leqslant s_{\sigma(i)}$ et $t_j \leqslant s_{\sigma(i)} \leqslant s_{\sigma(j)}$

$$\begin{aligned} C(\sigma) - C(\sigma') &= \left|t_i - s_{\sigma(i)}\right| + s_{\sigma(j)} - t_j - \left(s_{\sigma(i)} - t_j\right) - \left|t_i - s_{\sigma(j)}\right| && (9)\\ &= \left|t_i - s_{\sigma(i)}\right| + s_{\sigma(j)} - s_{\sigma(i)} - \left|t_i - s_{\sigma(j)}\right| && (10)\\ &\geqslant \left|t_i - s_{\sigma(j)}\right| - \left|s_{\sigma(j)} - s_{\sigma(i)}\right| + s_{\sigma(j)} - s_{\sigma(i)} - \left|t_i - s_{\sigma(j)}\right| && (11)\\ &\geqslant 0 && (12) \end{aligned}$$

Dans les deux cas, on montre donc qu'il existe un meilleur appariement global en permutant les deux skieurs i et j, c'est-à-dire en les triant par ordre croissant de taille. Nous avons donc montré que, si les paires de ski sont triées par ordre croissant de taille, l'appariement optimal est nécessaire un appariement pour lequel les skieurs sont aussi triés par ordre croissant. Lors de la recherche de cet appariement optimal, on peut se restreindre à ces cas de figure.

Exercice 5

$$p(n,m) = \min\left\{p(n-1,m-1) + |t_n - s_m|\,, p(n,m-1)\right\}$$

Lorsqu'on considère le meilleur appariement des paires $1..m$ et des skieurs $1..n$, il n'y a que deux choix possibles pour la paire m :

1. soit elle n'est associée à aucun skieur et dans ce cas : $p(n,m) = p(n,m-1)$,
2. soit elle est associée au skieur n (et à aucun autre) : $p(n,m) = p(n-1,m-1) + |t_n - s_m|$.

Quatrième demi-heure : La répartition des skis (suite et fin)

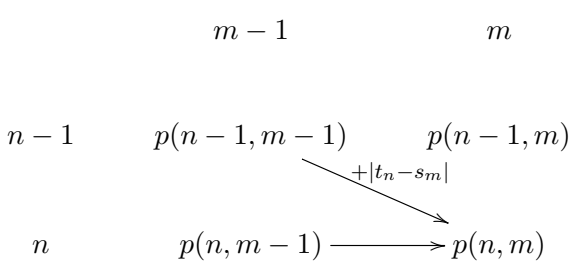
Exercice 6

```
p = { }
p [-1,-1] = 0
for n,taille in enumerate(skieurs) : p[n,-1] = p[n-1,-1] + taille
for m,paire  in enumerate(paires ) : p[-1,m] = 0
for n,taille in enumerate(skieurs) :
    for m,paire in enumerate(paires) :
        p1 = p.get ( (n  ,m-1), 1e10 )
        p2 = p.get ( (n-1,m-1), 1e10 ) + abs(taille - paire)
        p[n,m] = min(p1,p2)

print (p[len(skieurs)-1,len(paires)-1])
```

Exercice 7

Il faut imaginer que p peut être représenté sous forme de matrice et qu'à chaque fois, on prend le meilleur chemin parmi 2 :



– Chemin horizontal : on ne choisit pas la paire m.
– Chemin diagonal : on choisit la paire m pour le skieur n.

Pour retrouver la distribution, il suffit de conserver à chaque étape le meilleur des deux chemins.

```
p = { }
p [-1,-1] = 0
best = { }
for n,taille in enumerate(skieurs) : p[n,-1] = p[n-1,-1] + taille
for m,paire  in enumerate(paires ) : p[-1,m] = 0
for n,taille in enumerate(skieurs) :
    for m,paire in enumerate(paires) :
        p1 = p.get ( (n  ,m-1), 1e10 )
        p2 = p.get ( (n-1,m-1), 1e10 ) + abs(taille - paire)
        p[n,m] = min(p1,p2)

        if p[n,m] == p1 : best [n,m] = n,m-1
        else : best [n,m] = n-1,m-1

print (p[len(skieurs)-1,len(paires)-1])

chemin = [ ]
pos = len(skieurs)-1,len(paires)-1
while pos in best :
    print (pos)
    chemin.append(pos)
    pos = best[pos]
chemin.reverse()
print (chemin)
```

Pour aller plus loin ou pour ceux qui ont fini plus tôt .

Les deux algorithmes ont un coût quadratique.

fin correction TD ?? □