

## A) Pile ou file ...

Pour chacun des ordres de sortie suivant, la(les) structure(s) possible(s) entre pile et file est(sont) :

- A B C D E F (pile et file)
- D E C B F A (pile)
- B D E F A C (aucune des deux)
- F E D C B A (pile)

## B) Tri fusion ...

1. La fonction **partition(L)** sépare la liste **L** en deux listes de longueurs quasi identiques (à 1 près) : une moitié dans chaque liste.

```
def partition(L):  
  
    n = len(L)  
    L1 = []  
    for i in range(0, n//2):  
        L1.append(L[i])  
  
    L2 = []  
    for i in range(n//2, n):  
        L2.append(L[i])  
  
    return (L1, L2)
```

2. La fonction **merge(L1, L2)** fusionne deux listes **L1** et **L2** triées en ordre croissant en une seule liste triée.

```
def merge(L1, L2):  
  
    R = []  
    i = j = 0  
    n1 = len(L1)  
    n2 = len(L2)  
  
    while (i < n1) and (j < n2):  
        if L1[i] <= L2[j]:  
            R.append(L1[i])  
            i = i+1  
        else:  
            R.append(L2[j])  
            j = j+1  
  
    for i in range(i, n1):  
        R.append(L1[i])  
    for j in range(j, n2):  
        R.append(L2[j])  
  
    return R
```

3. La fonction **mergesort(L)** trie la liste **L** en ordre croissant (pas en place : la fonction construit une nouvelle liste qu'elle retourne).

```
def mergesort(L):  
    if len(L) <= 1:  
        return L  
    else:  
        (L1, L2) = partition(L)  
  
        return merge(mergesort(L1), mergesort(L2))
```

### C) Maximum Gap ...

La fonction **maxGapMatrix(M)** retourne le gap maximum des lignes de la matrice non vide **M**.

```
def gapList(L):  
    valMin = L[0]  
    valMax = L[0]  
    for i in range(1, len(L)):  
        valMin = min(valMin, L[i])  
        valMax = max(valMax, L[i])  
    return valMax - valMin  
  
def maxGapMatrix(M):  
    mgap = gapList(M[0])  
    for i in range(1, len(M)):  
        mgap = max(mgap, gapList(M[i]))  
    return mgap
```

En une seule fonction (gapList inlined) :

```
def maxGapMatrix2(M):  
    mgap = 0  
    (l, c) = (len(M), len(M[0]))  
    for i in range(l):  
        valMin = M[i][0]  
        valMax = M[i][0]  
        for j in range(1, c):  
            valMin = min(valMin, M[i][j])  
            valMax = max(valMax, M[i][j])  
        mgap = max(mgap, valMax - valMin)  
    return mgap
```

## D) Matrices : symétrique ...

La fonction **is\_symmetric(A)** teste si la matrice **A** non vide est symétrique.

```
def isSymmetric(A):  
    (l, c) = (len(A), len(A[0]))  
    if l != c:  
        return False  
    i = 0  
    sym = True  
    while i < l and sym:  
        j = i+1  
        while j < l and sym:  
            sym = A[i][j] == A[j][i]  
            j += 1  
        i += 1  
    return sym
```