Programmation 2 Spring, 2026

TD 7: reductions

1. On suppose qu'on dispose d'une fonction `sommepaire` qui, étant donné une liste `A` et une cible `x`, retourne `True` s'il existe deux éléments dans `A` dont la somme égale `x`, `False` sinon.

   À l'aide de cette fonction, écrivez une fonction `sommetriple` qui reçoit une liste `A` et une cible `x`, et retourne `True` s'il existe trois éléments dans `A` dont la somme égale `x`, `False` sinon.

   Vous n'avez pas le droit d'utiliser des boucles imbriquées.

   Exemple: pour `A = [1, 2, 7, 11, 15]` et `x = 18`, la fonction renvoie `True` $(1 + 2 + 15 = 18)$.
   pour `A = [1, 2, 7, 11, 15]` et `x = 17`, la fonction renvoie `False`.

   Solution.

```python
def sommepaire(A, x):
    return any(A[i] + A[j] == x for i in range(len(A)) for j in range(i+1, len(A)))


def sommetriple(A, x):
    for i in range(0, len(A)):
        if sommepaire(A[:i] + A[i+1:], x-A[i]) == True:
            return True
    return False



print( sommetriple( [1, 2, 7, 11, 15], 18  ) )
print( sommetriple( [1, 2, 7, 11, 15], 17  ) )
```

2. On suppose qu'on dispose d'une fonction `bin(n).count('1')` qui retourne le nombre de bits valant 1 dans sa représentation binaire de `n`. À l'aide de cette fonction, écrivez une fonction `est_puissance_de_deux` qui reçoit un entier positif `n` et retourne `True` si `n` est une puissance de deux, `False` sinon.

   `Exemple:` pour `n = 8`, la fonction renvoie `True`; pour `n = 6`, elle renvoie `False`.

   Solution.

```python
def est_puissance_de_deux(n):
    if n <= 0:
        return False

    return bin(n).count('1') == 1

print( est_puissance_de_deux(8) )
print( est_puissance_de_deux(6) )
```

3. On suppose qu'on dispose d'une fonction `compterinversions` qui, étant donné une liste `A` d'entiers *distincte*, retourne le nombre de paires `(i, j)` avec $0 \leq$ `i` $<$ `j` $\leq$ `len(A)-1` et `A[i] > A[j]`. À l'aide de cette fonction, écrivez une fonction `esttriee` qui reçoit une liste `A` et retourne `True` si et seulement si `A` est triée dans l'ordre **décroissant**.

   Vous n'avez pas le droit d'utiliser des boucles.

   `Exemple:` pour `A = [4, 3, 2, 1]`, la fonction renvoie `True`.
   pour `A = [4, 3, 1, 2]`, la fonction renvoie `False`.

   Solution.

```
def computerinversions(A):
    return sum(A[i] > A[j] for i in range(len(A)) for j in range(i+1, len(A)))

def esttriee(A):
    n = len(A)
    x = computerinversions(A)
    return x == n*(n-1)/2


print( esttriee( [4,3,2,1] ) )
print( esttriee( [4,3,1,2] ) )
```

4. On suppose qu'on dispose d'une fonction `commonprefix` qui, étant donné *une liste* `A` contenant deux chaînes (donc `A = [s1, s2]`), retourne leur plus long préfixe commun. À l'aide de cette fonction, écrivez une fonction `min_chaines` qui reçoit une liste `A` de chaînes et retourne la chaîne minimale dans `A` (par ordre lexicographique).

   Vous n'avez pas le droit de comparer deux chaînes de caractères, mais vous pouvez comparer deux caractères.

   Vous n'avez pas le droit d'utiliser des boucles imbriquées.

   Exemple: pour `A = ["aeb", "abf", "abd", "ac"]`, la fonction renvoie `"abd"`.

   Solution.

```python
from os.path import commonprefix

def min_chaines(A):
    _m = A[0]

    for i in range(1, len(A)):
        t = len( commonprefix([_m, A[i]]) )

        if len(A[i]) == t:
            _m = A[i]
        elif len(_m) > t and A[i][t] < _m[t]:
            _m = A[i]
    return _m


print( min_chaines( ["aeb", "abf", "abd", "ac"] ) )
```

5. On vous donne une liste `A` contenant n nombres.

   Écrivez une fonction `findTwoContiguous` qui prend `A` et renvoie la distance entre les deux nombres les plus proches de `A`. Vous pouvez supposer que vous avez accès à une fonction `sorted(A)` qui renverra une liste triée par ordre croissant.

   Vous n'avez pas le droit d'utiliser des boucles imbriquées.

   `Exemple:` pour `A = [ 10, 78, 102, 20, 44, 69 ]`, la fonction renvoie `9`.

   Solution.

```
def findTwoContiguous(A):
    B = sorted(A)

    _min = abs(B[1]-B[0])
    for i in range(1, len(B)-1):
        _min = min( _min, abs(B[i]-B[i+1]) )

    return _min

A = [ 10, 78, 102, 20, 44, 69 ]

print( findTwoContiguous(A) )
```

6. Écrivez une fonction `findMajority` qui prend une liste d'entiers `A` et renvoie l'élément qui apparaît plus de `len(A) / 2` fois dans `A`. Vous pouvez supposer qu'il existe un tel élément dans `A`. Vous pouvez supposer que vous avez accès à une fonction `sorted(A)` qui renverra une liste triée par ordre croissant.

   Vous n'avez pas le droit d'utiliser des boucles.

   Solution.

```
def findMajority(A):
    return sorted(A)[ len(A) // 2 ]


A = [1,2,6,4,2,2,2]
print( findMajority(A) )
```

7. On vous donne une liste `A` contenant n nombres.

   Écrivez une fonction `findClosest` qui prend `A`, un nombre x et un entier k. Elle renvoie les k nombres de `A` les plus proches de x.

   Supposons que vous ayez accès à une fonction `sorted(B, key=lambda element: element[0])` qui prend une liste de listes `B` et retourne une nouvelle liste triée selon l'ordre croissant du premier élément (d'indice 0) de chaque sous-liste.

   Par exemple : `sortedB = sorted([[3, 'a'], [1, 'b'], [4, 'c']], key=lambda element: element[0])` va retourner `[[1, 'b'], [3, 'a'], [4, 'c']]`.

   Vous n'avez pas le droit d'utiliser des boucles imbriquées.

   `Exemple` : pour `A = [4, 1, 3, 8, 10]`, `x = 5` et `k = 3`, la fonction renvoie `[4, 3, 8]` (les 3 nombres les plus proches de 5 sont 4, 3 et 8, dans n'importe quel ordre).

   Solution.

```
def findClosest(A, x, k):
    if len(A) < k: return -1

    B = [ [abs(x-A[i]), A[i]] for i in range(0, len(A)) ]
    B = sorted(B, key=lambda element: element[0])

    R = []
    for i in range(0, k):
        R.append(B[i][1])

    return R

A = [4, 1, 3, 8, 10]
print( findClosest(A, 5, 3) )
```

8. (**difficile**) On suppose qu'on dispose d'une fonction `to_binary_string` qui, étant donné un entier positif x, renvoie une string représentant x en binaire.

   À l'aide de cette fonction, écrivez une fonction `allsubsets` qui reçoit une liste `A` et retourne une liste de listes correspondant à tous les sous-ensembles de `A`.

   `Example:` pour `A = ['a', 'af', 41.34]`, la fonction renvoie

   `[[], ['a'], ['af'], ['a', 'af'], [41.34], ['a', 41.34], ['af', 41.34], ['a', 'af', 41.34]]`.

   Solution. Énumérer sur les vecteurs caractéristiques des sous-ensembles.

```
def to_binary_string(x):
    s = ''
    while x != 0:
        if x%2 == 1:
            s = '1' + s
        else:
            s = '0' + s
        x = x // 2
    return s

def allsubsets(A):
    B = [ [] ]
    for i in range(1, 2**len(A)):
        s = to_binary_string(i)
        print(s)
        B.append([])
        for j in range(0, len(s)):
            k = len(s)-j-1
            if s[k] == '1':
                B[-1].append(A[j])
    return B


print( allsubsets( ['a', 'af', 41.34] ) )
```