Matrix

Julien BESTARD

Desclaimer : Les algorithmes du doc sont **mes** algorithmes, en aucun cas une correction. Vous avez à dispostion des corrections sur gitlab, préferez celle-la.

Contents

1 Classique

Print Algorithm

```
def printmat(M):
    c = len(M[0])
    for i in range(len(M)):
        for j in range(c):
            print(M[i][j], end =" ")
        print()
```

PrettyPrint Algorithm

```
def prettyprint(M,d):
    s = "|{:"+str(d)+"d}"
    c = len(M[0])
    for i in range(len(M)):
        print("-"*(d+2)*c)
        for j in range(c):
            print(s.format(M[i][j]), end=" ")
        print("|")
    print("-"*(d+2)*c)
```

Init & Load Algorithm

```
def init(l,c,val):
    matrix = []
    for i in range(l):
        l = []
        for j in range(c):
            l.append(val)
        matrix.append(l)
    return matrix

def __str2intlist(str):
    L = []
    nb = ""
    for c in str:
        if c != " " or c != "\n":
            nb += c
        else:
            L.append(nb)
    return L

def __str2intlist2(str):
    n = len(s)-1
    i = 0
    L = []
    while i<n:
        word = ""
        while i<n and L[i]!=" ":
            word += str[i]
            i += 1
        L.append(int(word))
        i += 1
    return L

def load(filename):
    f = open(filename)
    lines = f.readlines()
    f.close()
    M = []
    for line in lines:
        M.append(__str2intlist(line))
    return M
```

Add Matrix Algorithm

```
def add_matrices(A,B):
    l = len(A)
    c = len(A[0])
    if (l == len(B)) and (c == len(B[0])):
        M = init(l,c,0)
        for i in range(l):
            for j in range(c):
                M[i][j] = A[i][j] + B[i][j]
        return M
    else:
        raise Exception("Matrix Error")
```

Mult Matrix Algorithm

```
def mult_matrices(A,B):
    m = len(A)
    n = len(A[0])
    p = len(B[0])
    if (n != len(B)):
        raise Exception("Matrix Error")
    M = init(m,p,0)
    for i in range(m):
        for j in range(p):
            for k in range(n):
                M[i][j] += A[i][k] * B[k][j]
    return M
```

2 Recherche et test

MaxGap

```
-------------------------------------------------------------------------------

def gap(L,n):
    """
    L : a list
    n : an int representing len(L)
    return the gap of l(maxValue - minValue)
    """
    posMax = 0
    posMin = 0
    for i in range(1,n):
        if L[i] > L[posMax]:
            posMax = i
        elif L[i] < L[posMin]:
            posMin = i
    return L[posMax] - L[posMin]
def max_gap(M):
    maxgap = 0
    nbcol = len(M[0])
    for L in M:
        g = gap(L,nbcol)
        if maxgap < g:
            maxgap = g
    return maxgap

-------------------------------------------------------------------------------

def maxgap2(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
```

Symmetric

```
def isSymmetric(A):
    i = 0
    j = -1
    n = len(A)
    sym = True
    while i < n and j == i - 1:
        j = 0
        while j < i and A[i][j] == A[j][i]:
            j += 1
        i =+ 1
    return j == i-1
```

Sorted Matrix

```
def list_sorted(L,n):
    i = 1
    while i<n and L[i-1] <= L[i]:
        i= i+1
    return i == n

def matrix_sorted(M):
    c = len(M[0])
    if not listed_sorted(M[0],c):
        return false
    else:
        l = len(M)
        i = 1
        while i<n and M[i-1][c-1]<M[i][0] and list_sorted(M[i],c)
            i += 1
        return i == l
```

Magic Square

```
def magic_square_NE(n):
    M = init(n,n,0)
    (i,j) = (0,n//2)
    for k in range(1,n*n+1):
        M[i][j] = k
        if k % n == 0:
            i = (i+1)%n
        else:
            (i,j) = ((i-1)%n,(j + 1)%n)
    return M

def magic_square_SE(n):
    M = init(n,n,0)
    (i,j) = (n-1,n//2)
    for k in range(1,n*n+1):
        M[i][j] = val
        if k % n == 0:
        i = (i-1)%n
        else:
            (i,j) = ((i+1)%n,(j+1)%n)
    return M
```

Harry Potter

```
def max3(v1,v2,v3,j):
    (max,pos) = (0,0)
    if v1 >= max:
        max = v1
        pos = i-1
    if v2 >= max:
        max = v2
        pos = j
    if v3 >= max:
        max = v3
        pos = j+1
    return(max,pos)

def greedy(M):
    (l,c)=(len(M),len(M[0]))
    (i,j) = (0,0)
    sum = 0
    for k in range(c):
        if M[k] > sum:
            sum = m[k]
            j = k
    for k in range(1,l):
        (v1,v2,v3) = (0,0,0)
        if j-1 != -1:
            v1 = M[k][j-1]
        v2 = M[k][j]
        if j+1 != n:
            v3 = M[k][j+1]
        (max,nj) = max3(v1,v2,v3,j)
        sum += max
        j = nj
    return sum
```

Correction Harry Potter

```
def posmax(L):
    pos = 0
    for i in range(1,len(L)):
        if L[i]>L[pos]:
            pos = i
    return pos

def HPgreedy(T)
    c = len(T[0])
    j = posmax(T[0])
    s = T[0][j]
    for i in range(1,c):
        jmax = j
        if j>0 and T[i][j-1]>T[i][jmax]:
            jmax = j-1
        if j<c-1 and T[i][j+1]>T[i][jmax]:
            jmax = j+1
        j = jmax
        s += T[i][j]
    return s
-------------------------------------------------------------------------
def __HP_bruteforce(T):
    if i = len(T)-1:
        return T[i][j]
    else:
        m = __HP_bruteforce(T,i+1,j)
        if j>0:
            mleft = __HP_bruteforce(T,i+1,j-1)
            if m < mleft:
                m = mleft
        if j < len(T[0])-1:
            mright = __HP_bruteforce(T,i+1,j+1)
            if m < mright:
                m = mright
        return T[i][j]+m

def HP_bruteforce(T):
    maxi = 0
    for j in range(len(T[0])):
        maxi = max(maxi,__HP_bruteforce(T,0,j)
    return maxi
```