ISM MP, Informatique
Année 2024/2025

Corrigé du TP Informatique évalué 01

Exercice 1

1. On saisit :

```
def sommets(config):
    S=[]
    noir=1
    L=len(config)
    C=len(config[0])
    for i in range(L):
        for j in range(C):
            if config[i][j]==noir:
                S.append((i,j))
    return S
```

2. On saisit :

```
def aretes(config):
    A={}
    noir=1
    S=sommets(config)
    L=len(config)
    C=len(config[0])
    for s in S:
        i,j=s
        A[s]=[]
        if i>0 and config[i-1][j]==noir:
            A[s].append((i-1,j))
        if i<L-1 and config[i+1][j]==noir:
            A[s].append((i+1,j))
        if j>0 and config[i][j-1]==noir:
            A[s].append((i,j-1))
        if j<C-1 and config[i][j+1]==noir:
            A[s].append((i,j+1))
    return A
```

Exercice 2

1. On saisit :

```
def dfs_rec(S,A,s0,etat):
    non_visite,visite=0,1
    etat[s0]=visite
    for s in A[s0]:
        if etat[s]==non_visite:
            dfs_rec(S,A,s,etat)
```

2. On saisit :

```
def nb_taches_dfs(S,A):
    etat={}
    non_visite,visite=0,1
    for s in S:
        etat[s]=non_visite
    nb=0
    for s in S:
        if etat[s]==non_visite:
            nb+=1
            dfs_rec(S,A,s,etat)
    return nb
```

Exercice 3

1. On saisit :

```
def bfs(S,A,s0,etat):
    non_visite,visite=0,1
    etat[s0]=visite
    F=deque()
    F.append(s0)
    while len(F)!=0:
        sorg=F.popleft()
        for s in A[sorg]:
            if etat[s]==non_visite:
                F.append(s)
                etat[s]=visite
```

2. On saisit :

```
def nb_taches_bfs(S,A):
    etat={}
    non_visite,visite=0,1
    for s in S:
        etat[s]=non_visite
    nb=0
    for s0 in S:
        if etat[s0]==non_visite:
            nb+=1
            bfs(S,A,s0,etat)
    return nb
```

Exercice 4

1. On saisit :

```
def dijkstra_deb_fin(S,A,deb,fin):
    cout,predec,djvu={},{},{}
    for s in S:
        cout[s]=np.inf
        djvu[s]=False
    cout[deb]=0
    while not djvu[fin]:
        smin,cmin=deb,np.inf
        for s in S:
            if not djvu[s] and cout[s]<cmin:
                smin,cmin=s,cout[s]
        djvu[smin]=True
        for s,nu in A[smin]:
            if not djvu[s] and cout[s]>cout[smin]+nu:
                cout[s]=cout[smin]+nu
                predec[s]=smin
    return predec
```

2. On saisit :

```
def aretes_dijkstra(config):
    A={}
    noir=1
    S=sommets(config)
    L=len(config)
    C=len(config[0])
    for s in S:
        i,j=s
        A[s]=[]
        i,j=s
        if i>0 and config[i-1][j]==noir:
            A[s].append(((i-1,j),1))
        if i<L-1 and config[i+1][j]==noir:
            A[s].append(((i+1,j),1))
        if j>0 and config[i][j-1]==noir:
            A[s].append(((i,j-1),1))
        if j<C-1 and config[i][j+1]==noir:
            A[s].append(((i,j+1),1))
    return A
```

3. On saisit :

```
def dijkstra_all_deb_fin(S,A,deb,fin):
    cout,predec,djvu={},{},{}
    for s in S:
        cout[s]=np.inf
        djvu[s]=False
    cout[deb]=0
    F_config=deque()
    L_sol=[]
    F_config.append([deb,cout,predec,djvu])
    while len(F_config)!=0:
        config=F_config.popleft()
        smin,cout,predec,djvu=config
        djvu[smin]=True
        # si pas fini
        if not djvu[fin]:
            # relachement depuis smin
            for s,nu in A[smin]:
                if not djvu[s] and cout[s]>cout[smin]+nu:
                    cout[s]=cout[smin]+nu
                    predec[s]=smin
            # recherche des prochains sommets à dist min
            smin,cmin=deb,np.inf
            L_smin=[]
            for s in S:
                if not djvu[s]:
                    if cout[s]<cmin:
                        L_smin=[s]
                        cmin=cout[s]
                    elif cout[s]==cmin:
                        L_smin.append(s)
            for smin in L_smin:
                F_config.append([smin,dict(cout),dict(predec),dict(djvu)])
        else:
            cout_ppc=cout[fin]
            L_sol.append(predec)
    return L_sol
```