DS6_24.py

```
001|
002| # ========================
003| # Corrigé DS6 Graphes
004| # ========================
005|
006|
007| # --------- DS6 Graphes ---------
008| # Q1
009|
010| # Q2
011| LA = [
012|       [1,3,4,6,7], # 0
013|       [0, 2, 3], # 1
014|       [1,3],      # 2
015|       [0, 1,2,4],# 3
016|       [0, 3,5,6,7],  # 4
017|       [ 4, 6, 7], #5
018|       [0,4,6,7],   # 6
019|       [0,4,5, 6]     # 7
020| ]
021|
022| # Q3 liste d'adjacence
023| ''' avantage : cout mémoire plus faible
024| inconvénient : plus difficile accès aux
successeurs/prédécesseurs ?
025| '''
026|
027| # Q4
028| ''' degré : nombre d'arcs pour un sommet
029| s   | 0 | 1 | 2| 3| 4 |5 |6 | 7
030| deg | 5 | 3 | 2| 4| 5 |3 |4 | 4
031| '''
032|
033| # Q5
034| def voisins_ds6(i, j, LA):
035|     return j in LA[i]
036|
037| # Q6
```

```
038| def coloration_valide(LA, C):
039|     n = len(LA)
040|     for i in range(n):
041|         for j in LA[i]:
042|             if i < j and C[i] == C[j]:
043|                 return False
044|     return True
045| ''' la condition i<j n'est pas nécéssaire
mais diminue par 2 la complexité'''
046|
047|
048| # Q7
049| '''
050| Dans le pire des cas, chaque sommet  est
relié à n-1 sommets. LA contient donc n listes de
n-1 élements. La complexité est quadratique
051| '''
052|
053| # Q8
054| n=8
055| C = n*[-1]
056|
057| # Q9
058| def colore_sommet(C, s, LA):
059|     coul_vois = [C[v] for v in LA[s] if C[v]
!= -1]
060|     num_coul = 0
061|     while num_coul in coul_vois:
062|         num_coul += 1
063|     C[s] = num_coul
064|
065| # Q10
066| def colorer1(LA):
067|     n = len(LA)
068|     C = [-1] * n
069|     for s in range(n):
070|         colore_sommet(C, s, LA)
071|     return C
072|
073| # Q11
```

```
074| def colorer2(ordre, LA):
075|     n = len(LA)
076|     C = [-1] * n
077|     for s in ordre:
078|         colore_sommet(C, s, LA)
079|     return C
080|
081| ## Variante de Welsh POWELL
082|
083| # Q 12
084| def degre(LA):
085|     return [len(LA[i]) for i in
range(len(LA))]
086|
087| # Q13
088| def degre_max(LA):
089|     degres = degre(LA)
090|     dmax=0
091|     for d  in degrees :
092|         if d> dmax: dmax =d
093|     return dmax
094| #ou ...
095| def degre_max(LA):
096|     return max(degre(LA))
097|
098| # Q14
099| def ranger(LA):
100|     max_deg = degre_max(LA)
101|     R = [ [] for _ in range(max_deg + 1) ]
102|     d= degre(LA)
103|     for i in range(len(LA)):
104|         R[ d[i] ].append(i)
105|     return R
106|
107| # Q15
108| def trier_sommets(LA):
109|     R = ranger(LA)
110|     L  = []
111|     for x in  R[::-1] :
112|         L+=x
```

```
113|     return L
114| # test
115| LA = [[1,2],[0,2,3],[0,1,3],[1,2,4],[3]]
116| print(trier_sommets(LA))
117|
118| # Q16
119| ''' Dans le pire des cas, le degré max est
n-1 -> o(n)'''
120|
121| # Q17
122| def colorer3(LA):
123|     ordre = trier_sommets(LA)
124|     C = [-1] * len(LA)
125|     for s in ordre:
126|         colore_sommet(C, s, LA)
127|     return C
128| '''
129| trier : o(n)
130| for + colore_sommet -> o(n²)
131| '''
132|
133| # Q18
134| '''[0, 1, 0, 2, 1, 0, 2, 3]'''
135|
136|
```