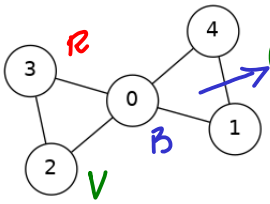
Algorithmique des Graphes
L3 informatique

13 décembre 2022

Vous êtes invités à remettre une copie claire, concise, sans rature ni surcharge en répondant aux questions dans l'ordre de l'énoncé... La note tiendra compte de la présentation générale de la copie.



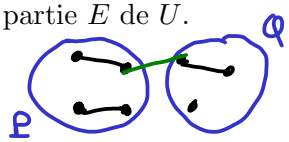
Q1. On note $\gamma(k)$ le nombre de k-coloration du graphe papillon ci-dessus. Compléter la table :

k	1	2	3
$\gamma(k)$	0	0	12

$3 \times 2 \times 2$

Q2. Soit G un graphe $G(X,U)$. Illustrer par un dessin les notions d'arête traversante et de coupure respectant une partie E de U.

Voir cours ACM



Q3. L'algorithme de Kruskal s'applique à un graphe pondéré pour déterminer un arbre couvrant de coût minimal. Peut-il être adapté pour déterminer un ACM de coût maximal?

oui, les mêmes arguments

Q4. Préciser l'objectif de l'algorithme de Nicos Christophides.

construire une $\frac{3}{2}$ approximation de TSP dans le cas complet métrique

Q5. Identifier, à isomorphisme près, les graphes à 8 arêtes ayant 2 sommets de degré 4, sans sommet isolé, ni sommet de degré 1.

Notons A, B les 2 sommets de degré 4 et S les autres sommets.

$\sum_{s \in S} deg(s) = 16 - 8 = 8$

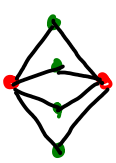
Q6. Soit G un graphe planaire à p composantes connexes, f faces, m arêtes et n sommets. Démontrer que $f - m + n - p = 1$.

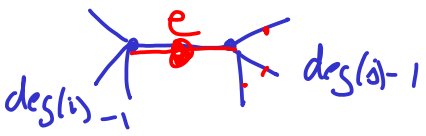
sur chaque composante connexe le nb de faces finies reste $f_i - m_i + n_i - 1 = 0$

```
1  int count, *table;
2
3  void partition( int n, int s )
4  {
5  int i, q;
6  if ( s == 0 ) {
7       count++;
8       traitement( n );
9       return;
10 }
11 if ( n > s )
12      return;
13 q = s / n;
14 for( i = 0; i <= q ; i++ ){
15      table[n] = i;
16      partition ( n + 1, s - i * n );
17 }
18 table[n] = 0;
19 }
20
21 int main( int argc, char* argv[] )
22 { int n = atoi( argv[1] );
23   table = calloc( 1+n, sizeof(int) );
24   count = 0;
25   partition ( 1, n );
26   printf ("%d %d\n", n, count );
27 return 0;
28 }
```

Listing 1: partition

	2	3	4
1	2	0	1
	1	2	0
	4	0	0





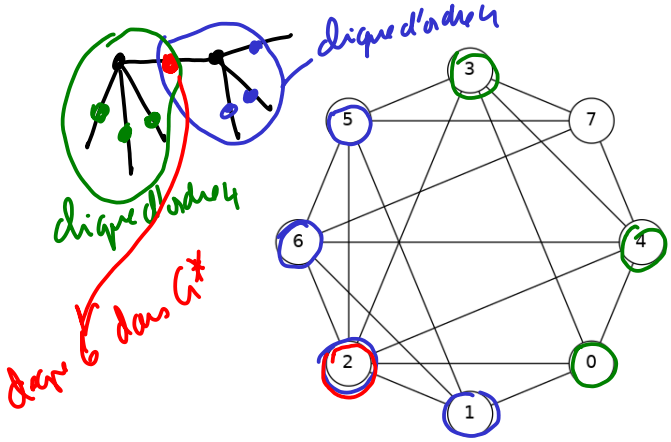
Q7. Le programme prend un entier n sur la ligne de commande pour calculer le nombre de partition de n. Donner les 3 premières partitions traitées pour $n = 5$.

00001 10010 20000 ...

```
1 maxStable( graphe G )
2   si m(G) = 0 retourner n(G)
3   s ← un sommet non isole de G
4   I ← retirerSommet( {s}, G )
5   i ← maxStable( I );
6   V ← les voisins de s dans G
7   J ← retirerSommet( V, I )
8   j ← maxStable( J )
9   retourner max( i, j+1)
```

Listing 2: calcul du stable maximum

Q8. Un ensemble stable est un ensemble de sommets deux à deux non adjacents. La taille d'un stable est égale au nombre de sommets qu'il contient. Compléter l'instruction de retour de `maxStable` pour retourner la taille maximum d'un stable.



graphe H

Q9. Une clique dans un graphe $G(S, U)$ est un ensemble de sommets deux à deux adjacents. Une clique K est dite maximale si elle maximale pour l'inclusion. Que représente une clique maximale du point de vue du graphe complémentaire? Quelle est la taille maximum d'une clique du graphe H ?

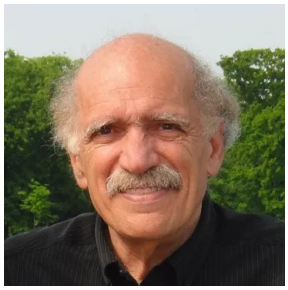
- Une clique est un stable du complémentaire
- 4

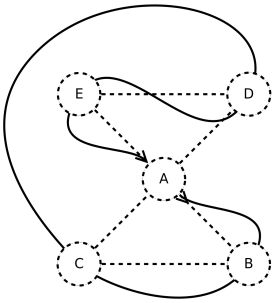
Q10. On note G^* l'adjoint du graphe $G(S, U)$. On considère un sommet e dans G^* correspondant à l'arête ij. Préciser le degré de e, puis l'ordre des cliques maximales contenant e.

- $deg(e) = deg(i) + deg(j) - 2$
- $deg(i)$ et $deg(j)$

Q11. Le graphe H est isomorphe à l'adjoint d'un graphe G bien connu. (a) Quel est le nombre d'arêtes de G? (b) Montrer que G possède 2 sommets adjacents de degré 4.

(a) 8 (b) voir dessin





Nikos Cristophides

```
1  graphe adjoint(graphe * g)
2  {   graphe r;
3      int s, t, k;
4      ullong v[g->nbs * g->nbs ];
5      k = 0;
6      for (s = 0; s < g->nbs; s++)
7      for (t = s + 1; t < g->nbs; t++)
8          if (g->mat[s][t]) {
9          v[k] = (1 << s) + (1 << t);
10         k++;
11         }
12
13     r = initGraphe( k );
14     for (s = 0; s < r.nbs; s++)
15     for (t = s + 1; t < r.nbs; t++)
16         if ( v[s] & v[t] )
17         r.mat[s][t] = r.mat[t][s] = 1;
18     return r;
19 }
```

Listing 3: line graph

Q12. La fonction `adjoint` utilise la technique de la programmation par bit pour déterminer l'adjoint du graphe g. Au delà de la ligne 12 :

1. Préciser la valeurs de k. k = nb arête de g
2. Quel est le poids binaire des $v[s]$, pour $s < k$. 2
3. Comment faut-il compléter la ligne 16?