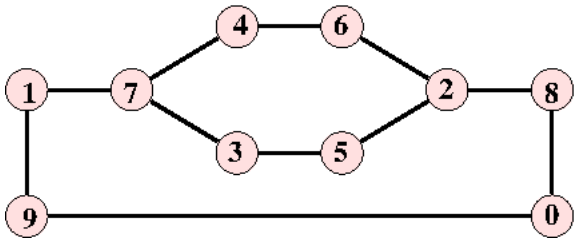
Algorithmique des Graphes
L3 informatique

mercredi 8 janvier de l'an $2^3 + 3^3 4^3 + 5^3 + 6^3 + 7^3 + 8^3 + 9^3$

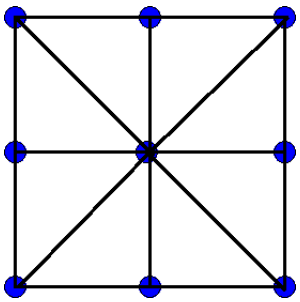
Vous êtes invités à remettre une copie claire, concise, sans rature ni surcharge en répondant aux questions dans l'ordre de l'énoncé... La note tiendra compte de la présentation générale de la copie.

Q1. Quel langage doit-on à N. Wirth ? T. Hoare est célèbre pour l'analyse d'un algorithme. Lequel? Quel algorithme du cours est associé à E. Dijkstra?



Q2. Observer le graphe d'ordre 10 ci-dessus. Quel est son nombre chromatique?

Q3. On pondère l'arête ij du graphe par le coût $i + j$. Quel est le coût d'un ACM pour cette pondération?



Q4. Combien de 3-coloration possède le graphe ci-dessus?

Q5. Préciser la distribution des degrés des sommets de , puis celle de son complémentaire.

Q6. Le graphe est-il semi-eulérien ?

Q7. Le complémentaire de est-il hamiltonien?

```
1  all : a b c d e f g h
2  a : b e
3      echo $@
4  b : c f
5      echo $@
6  c : d g
7      echo $@
8  d : h
9      echo $@
10 e : f
11     echo $@
12 f : g
13     echo $@
14 g : h
15     echo $@
16 h :
17     echo $@
```

Listing 1: Makefile

Q8. On rappelle que dans une règle make, la variable `$@` identifie la cible, et que l'option `-s` supprime le préaffichage des commandes.

(a) Dessiner le graphe des dépendances.

(b) Quelle est la nature de ce graphe?

(c) Faire un tri topolique. Prédire le résultat de `make -s`

Q9. Montrer que le théorème GPT est faux.

Trouver tous les graphes acycliques connexes dont le complémentaire est un arbre.

```
1 char W( ullong z ) {
2     char w = 0;
3     while ( z ) {z &= z-1; w++;}
4     return w;
5 }
```

Q10. On rappelle que `W(z)` calcule le poids binaire de l'entier z (méthode de Kernighan). Le type de retour de `W` est-il adéquat? Quel est temps de calcul moyen de `W(z)` sur les mots de 64bits. Peut-on faire mieux sur une machine moderne ?

```
1 WELSH-POWEL ( graphe g )
2  L ← ensemble des sommets de g
3  r ← 0
4  tantque non vide ( L )
5   r ← r + 1
6   VISITE( L, r );
7  ftq
8 retourner r
```

Listing 2: coloration de Welsh-Powel

L'algorithme de Welsh-Powel est une variation de l'algorithme glouton vu en cours. Il procède en plusieurs étapes. A l'étape numéro r, les sommets de L sont visités par **ordre de degré décroissant** par `VISITE(L,r)` qui colorie certains des sommets de L avant de les supprimer de la liste.

L'ensemble des sommets coloriés au cours de `visite( L, r)` sont stockés dans l'ensemble X. Un sommet visité s qui n'est pas adjacent à un sommet de X est colorié avec la couleur r avant d'être définitivement supprimé de la liste L pour être ajouter X, c'est en particulier le cas du premier sommet de L.

```
1 VISITE( L, r )
2  X ← {}
3  pour chaque s de L
4   si s non voisin ( X )
5   colorier s avec r
6   enlever  s de L
7   placer   s dans X
8  ftq
```

Listing 3: coloration de Welsh-Powel

Q11. Appliquer l'algorithme au graphe  .

Q12. Justifier l'arrêt de l'algorithme.

```
1 typedef struct {
2     int nbs;
3     char ** mat;
4     liste  *adj;
5     int    *clr ;
6 } graphe;
```

Pour une mise en oeuvre en langage C, nous utiliserons la structure de graphe du cours incluant un champ de couleur `clr`.

Q13. Donner une définition pour le type `liste`.

On choisit de représenter la liste L par un **tableau de sommets** ordonné par degré décroissant.

Q14. Coder `int degre(int s, graphe g)` qui retourne le degré du sommet s en utilisant la structure la plus efficace.

Q15. Coder une `int * lsd( graphe g )` qui calcule une liste de sommets du graphe g ordonnée par degré décroissant en utilisant un tri linéaire par répartition.

Q16. Préciser le temps de calcul.

Q17. On convient de "supprimer" l'élément s de L en affectant -1 à $L[s]$. Coder la fonction `visite( int L[], int r, graphe g)`.

Edsger Dijkstra Niklaus Wirth Tony Hoare

Théorème GPT Le complémentaire d'un graphe acyclique connexe est un arbre.