uOttawa

CSI 2510 - Structures de données et algorithmes

Automne 2021 Section A

Professeure Dorra Riahi

Devoir 8: Graphe

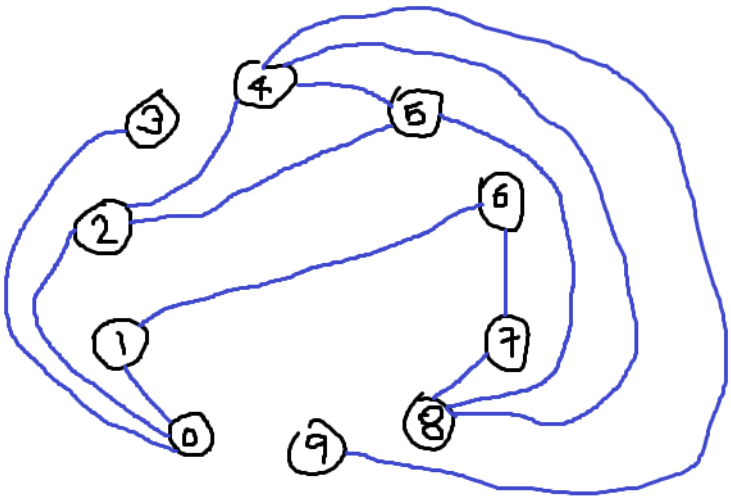
Kien Do (ID: 300163370)

Voici un graphe non-orienté représenté avec une matrice d'adjacence. Cette matrice est symétrique, nous ne montrons ici que la partie supérieure. Une valeur 1 dans la matrice correspond à une arête entre les deux sommets correspondants.

Below is a non-oriented graph represented using and adjacency matrix. This matrix is symmetrical, so we show here only the upper half of it. A value of 1 in the matrix indicates that there is an edge between the two corresponding nodes.

	0	1	2	3	4	5	6	7	8	9
0		1	1	1	0	0	0	0	0	0
1			0	0	0	0	1	0	0	0
2				0	1	1	0	0	0	0
3					0	0	0	0	0	0
4						1	0	0	1	1
5							0	0	1	0
6								1	0	0
7									1	0
8										0
9										

1. Dessiner le graphe correspondant à cette matrice.



2. Remplacer cette représentation avec matrice d'adjacence pour une représentation avec listes d'adjacence. Chacune des arêtes dans les listes d'adjacence doit être identifiée comme suit :
   0-1 pour l'arête liant le noeud 0 au noeud 1.
   Ordonner les arêtes en suivant l'ordre de gauche à droite dans les rangées de la matrice.

   0: (0-1), (0-2), (0-3)
   1: (1-0), (1-6)
   2: (2-0), (2-4), (2-5)
   3: (3-0)
   4: (4-2), (4-5), (4-8), (4-9)
   5: (5-2), (5-4), (5-8)
   6: (6-1), (6-7)
   7: (7-6), (7-8)
   8: (8-4), (8-5), (8-7)
   9: (9-4)

3. Effectuer une traversée en profondeur de ce graphe à partir du noeud 0 en utilisant une pile tel que vu en classe. Lorsque les arêtes sont empilées, toujours suivre l'ordre de gauche à droite comme pour la ques- tion 2 (par exemple pour le sommet 2, l'arête 2-4 sera empilée avant l'arête 2-5). Bien montrer l'état de la pile après chaque visite de sommet. Donner aussi, dans l'ordre, les arêtes de l'arbre T utilisées pour passer d'un noeud courant au prochain noeud non-exploré.

   Visited: {0} to visit

0-3
0-2
0-1

   POP → (0-3)
   Visited: {0,3,2}
   T={(0-3),(0-2)} to visit

0-2
0-1

   POP → (0-2)
   Visited: {0,3,2}
   T={(0-3),(0-2)} to visit

2-5
2-4
0-1

   POP → (2-5)
   Visited: {0,3,2,5}
   T={(0-3),(0-2),(2-5)} to visit

5-8
5-4
2-4
0-1

   POP → (5-8)
   Visited: {0,3,2,5,8}

T={(0-3),(0-2),(2-5),(5-8)} to visit $\begin{vmatrix} 8\text{-}7 \\ 8\text{-}4 \\ 5\text{-}4 \\ 2\text{-}4 \\ 0\text{-}1 \end{vmatrix}$

POP → (8-7)
Visited: {0,3,2,5,8,7}

T= $\left\{ \begin{array}{c} (0\text{-}3),(0\text{-}2),(2\text{-}5),(5\text{-}8), \\ (8\text{-}7) \end{array} \right\}$ to visit $\begin{vmatrix} 7\text{-}6 \\ 8\text{-}4 \\ 5\text{-}4 \\ 2\text{-}4 \\ 0\text{-}1 \end{vmatrix}$

POP → (7-6)
Visited: {0,3,2,5,8,7,6}

T= $\left\{ \begin{array}{c} (0\text{-}3),(0\text{-}2),(2\text{-}5),(5\text{-}8), \\ (8\text{-}7),(7\text{-}6) \end{array} \right\}$ to visit $\begin{vmatrix} 6\text{-}1 \\ 8\text{-}4 \\ 5\text{-}4 \\ 2\text{-}4 \\ 0\text{-}1 \end{vmatrix}$

POP → (6-1)
Visited: {0,3,2,5,8,7,6,1}

T= $\left\{ \begin{array}{c} (0\text{-}3),(0\text{-}2),(2\text{-}5),(5\text{-}8), \\ (8\text{-}7),(7\text{-}6),(6\text{-}1) \end{array} \right\}$ to visit $\begin{vmatrix} 8\text{-}4 \\ 5\text{-}4 \\ 2\text{-}4 \\ 0\text{-}1 \end{vmatrix}$

POP → (8-4)
Visited: {0,3,2,5,8,7,6,1,4}

T= $\left\{ \begin{array}{c} (0\text{-}3),(0\text{-}2),(2\text{-}5),(5\text{-}8), \\ (8\text{-}7),(7\text{-}6),(6\text{-}1),(8\text{-}4) \end{array} \right\}$ to visit $\begin{vmatrix} 4\text{-}9 \\ 5\text{-}4 \\ 2\text{-}4 \\ 0\text{-}1 \end{vmatrix}$

POP → (4-9)
Visited: {0,3,2,5,8,7,6,1,4,9}

T= $\left\{ \begin{array}{c} (0\text{-}3),(0\text{-}2),(2\text{-}5),(5\text{-}8), \\ (8\text{-}7),(7\text{-}6),(6\text{-}1),(8\text{-}4), \\ (4\text{-}9) \end{array} \right\}$ to visit $\begin{vmatrix} 5\text{-}4 \\ 2\text{-}4 \\ 0\text{-}1 \end{vmatrix}$

L'algorithme s'arrête ici car le nombre de noeuds visités est égal au nombre total de noeuds (cela veut dire que tous les noeuds ont été visités et la traversée est terminée).

4. Utilisant l'algorithme ci-dessous, effectuer une traversée en largeur du graphe en partant du noeud 0. Dessiner le graphe en identifiant les arêtes DISCOVERY et CROSS. Lister les sommets dans l'ordre de visite en encerclant les groupes correspondants aux listes: L_0, L_1, L_2,...

```
Algorithm BFS(G)

Input graph

Output labeling of the edges and
partition of the vertices of G
for all u ∈ G.vertices()
  setLabel(u, UNEXPLORED)
for all e ∈ G.edges()
  setLabel(e, UNEXPLORED)
for all v ∈ G.vertices()
  if getLabel(v) = UNEXPLORED
      BFS(G, v)
```

```
Algorithm BFS(G, s)

L_0 ← new empty sequence
L_0.insertLast(s)
setLabel(s, VISITED)
i ← 0
while ! L_i.isEmpty()
  L_(i+1) ← new empty sequence
  for all v ∈ L_i.elements()
    for all e ∈ G.incidentEdges(v)
     if getLabel(e) = UNEXPLORED
      w ← opposite(v,e)
      if getLabel(w) = UNEXPLORED
        setLabel(e, DISCOVERY)
        setLabel(w, VISITED)
        L_(i+1).insertLast(w)
      else
        setLabel(e, CROSS)
  i ← i + 1
```