Weekplan: Introduction to Graphs

Philip Bille

Reading

Introduction to Algorithms, Cormen, Rivest, Leisersons and Stein (CLRS): Introduction to Part VI + Chapter 22.1-22.4 + Appendix B.4-B.5.

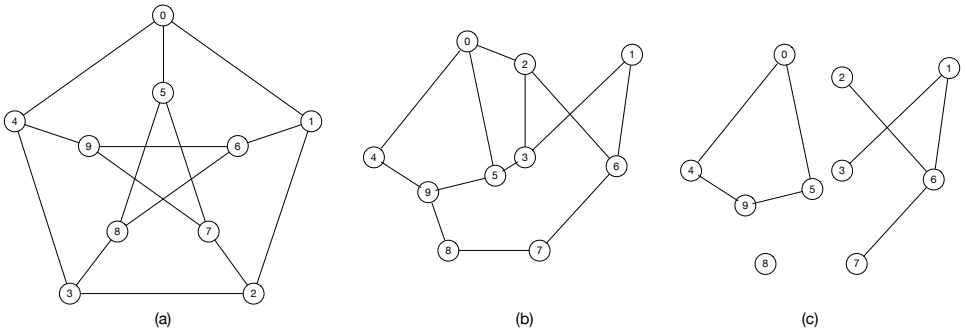


Figure 1: Graphs for the exercises. (a) is the *Petersen graph*.

Exercises

1 Representation, Properties and Algorithms Look at the graphs in Figure 1. Solve the following exercises.

1.1 [w] Show adjacency lists and adjacency matrices for (a) and (c).

1.2 [w] Simulate DFS on (a) starting in node 0. Assume the adjacency lists are sorted. Specify the DFS-tree, and start and end times.

1.3 [w] Simulate BFS on (a) starting in node 0. Assume the adjacency lists are sorted. Specify the BFS-tree, and the distance for each node.

1.4 Specify the connected components of (a), (b), and (c).

1.5 Which of (a), (b), and (c) are bipartite?

2 Depth First Search using a Stack Explain how to implement DFS without using recursion. *Hint:* use an (explicit) stack.

3 Find a Cycle Give an algorithm that determines if a graph i *cyclic*, ie. contains a cycle. How fast is your algorithm?

4 Labyrinths Solve exercise 3 in the exam set from 2010 (this exercise is the same in 02326 and 02105).

5 Number of Shortest Paths Give an algorithm that given two nodes s and t in G returns the *number* of shortest paths between s and t in G.

6 Implementation of Graphs We want to support the following operations on a dynamic graph G.

- ADDEDGE(u, v): add an edge between the nodes u and v.
- ADJACENT(u, v): return if u and v are adjacent in G.
- NEIGHBOURS(v): prints all neighbors of node v.

Solve the following exercises.

6.1 [†] Implement the operations on an adjacency matrix.

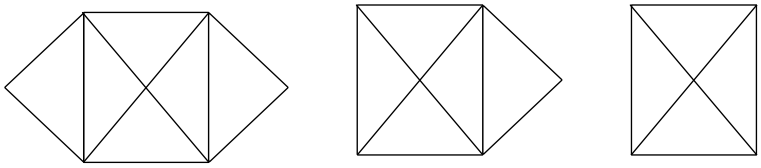
6.2 [†] Implement the operations on an adjacency list.

7 Euler Tours and Euler Paths Let G be a connected graph with n nodes and m edges. An *Euler tour* in G is a cycle that contains all edges in G exactly once. An *Euler path* in G is a path that contains all edges in G exactly once. Solve the following exercises.

7.1 [$*$] Show that G has an Euler tour if and only if all nodes have even degree.

7.2 [$*$] Show that G has an Euler path if and only if at most two nodes have have odd degree.

7.3 Which of the drawings below can you draw without lifting the pencil? Can you start and end at the same place?



7.4 Give an $O(n + m)$ time algorithm that determines if G has an Euler tour.

7.5 [$*$] Give an $O(n + m)$ algorithm that finds an Euler tour in G if it exists.

8 Diameter of Trees Let T be a tree with n nodes. The *diameter* of T is the longest shortest path between a pair of nodes in T. Solve the following exercises.

8.1 Give algorithm to compute the diameter of T in $O(n^2)$ time.

8.2 [$**$] Give algorithm to compute the diameter of T in $O(n)$ time.

M Mandatory Exercise: X-mas Decorations You're living on a long road with n houses placed side-by-side (all on the same side of the road). There is a proud tradition of decorating for Christmas and people come from all over the world to visit your road. Also there is a strong competition among the families on the road to have the coolest decorations. A jury of 3 judges lead by Blachman gives each house a score based on their Christmas factor – in short the X-factor. You have been given the task to keep track of the X-factor of each house. Obviously you want to impress Blachman, so you want to design an efficient data structure to keep track of this information. Solve the following exercises (remember to analyze the running time of the operations).

M.1 Initially, each house H_i has an X-factor X_i of 0 until the Christmas season begins. Whenever a family installs new decorations Blachman notifies you of their new X-factor. Also often people call you and asks for the current X-factor of a given house. Thus your data structure must support UPDATEXFACTOR(H_i, X_i) and GETXFACTOR(H_i). Describe a data structure to efficiently support these two operations.

M.2 Everyone on the road is interested in knowing which house is currently the best decorated, since the best decorated will win a prize in the end. Suggest a data structure to support HIGHESTXFACTOR() efficiently while still supporting UPDATEXFACTOR and GETXFACTOR (the latter two might become slightly less efficient).

M.3 From time to time Blachman calls you to get a list of the best decorated houses, so he can show his friends the most amazing houses. He is only interested in houses with a X-factor of at least X_{min}. Explain how to also support XFACTORATLEAST(X_{min}) efficiently (the running time should depend on the number of reported houses).

M.4 Last year the winning house had some badly decorated neighbors making the view from the road bad which made Blachman furious. Therefore the judge committee has decided to pick the winning house based on the X-factor of the house plus the X-factor of the two neighboring houses. Describe how to extend your data structure to also support HIGHESTXFACTORINCLUDINGNEIGHBOURS() efficiently.