CSE 3101 Design and Analysis of Algorithms
Practice Test for Unit 3
Graph Search and Network Flow

Jeff Edmonds

First learn the steps. Then try them on your own. If you get stuck only look at a little of the answer and then try to continue on your own.

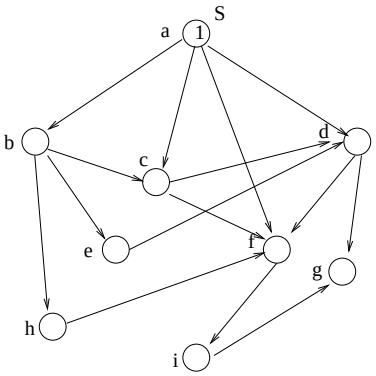
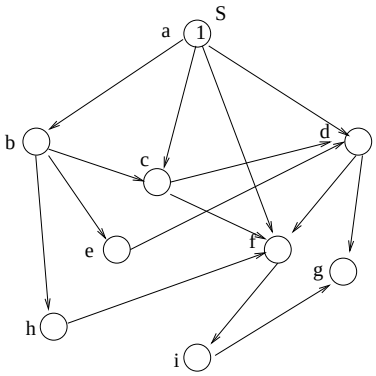
1. Trace BFS and DFS.

   (a) What are the generic loop invariants used for searching a graph starting from some node s?

   (b) What are the extra loop invariants used for Breadth First Search.

   (c) What are the extra loop invariants used for Depth First Search?

   (d) For each do the following:

Breadth First Search (BFS)	Depth First Search (DFS)
	
Data Structure?	Data Structure?

   i. Start at node s and when there is a choice follow edges from left to right. Number the nodes $1, 2, 3, \ldots$ in the order that they are found, starting with Node $s = 1$.

   ii. Darken the edges of the Tree specified by the predecessor array π.

   iii. What is the data structure used by BFS/DFS to store nodes that are found but not yet handled?

   iv. Circle the nodes that are in this data structure when Node 8 is first found.

2. DFS

   (a) What are the pre and post conditions for the Recursive Depth First Search algorithm?

   (b) Trace out the iterative and the recursive DFS algorithms on the same graph and see how they compare. Do they have the same running time?

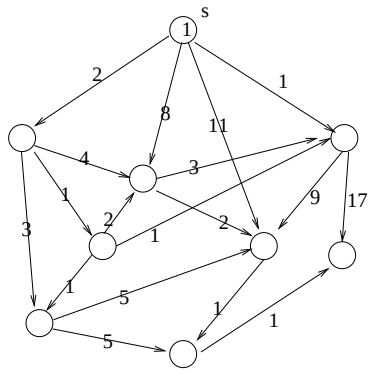
3. What are the extra loop invariants used for Depth First Search? How is the extra loop invariant for Depth First Search used to look for cycles? If the graph has a cycle, is the cycle guaranteed to be found in this way?

4. Give a Total Order (Topological Sort) of the DAG in the previous question using the letters a,b,....

5. Dijkstra's Shortest-Weighted Paths Algorithm

   (a) What are the extra loop invariants used for this algorithm?

(b) On the following example, do the following.



i. Start at node s and when there is a choice follow edges from left to right. Number the nodes $1, 2, 3, \ldots$ in the order that they are handled, starting with Node $s = 1$.

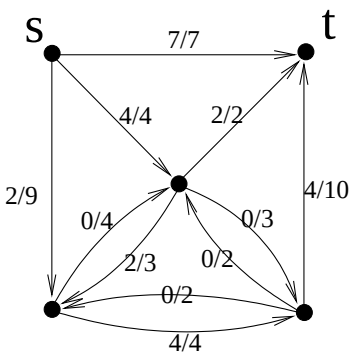
ii. Darken the edges of the Tree specified by the predecessor array π.

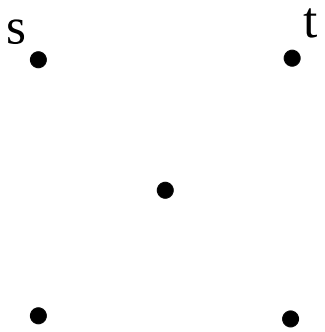
iii. What is the data structure used to store nodes that are found but not yet handled?

iv. For each node, give the list of every d value that it has at various points during the computation.

6. Given a graph where each edge weight is one, compare and contrast the computation of the BFS shortest paths algorithm from Section **??** and that of this Dijkstra shortest-weighted paths algorithm. How do their choices of the next node to handle and their loop invariants compare?

7. Starting with the flow given below, complete the network flow algorithm and prove the resulting flow is optimum.





8. In hill climbing algorithms there are steps that make lots of progress and those that make very little progress. For example, the first iteration on the input given in Figure 1 might find add a flow of c along the path s, a, t and then during the second iteration add a flow of c along the path s, b, t. It might, however, find the path s, b, a, t through which only a flow of 1 can be added. How bad might the running time be when the computation is unlucky enough to always take the worst legal step allowed by the algorithm? Start by taking the step that increases the flow by 2 through the input given in Figure 1. Then continue to take the worst possible step. You could draw out each and every step, but it is better to use this opportunity to use loop invariants. What does the flow look like after $2i$ iterations?

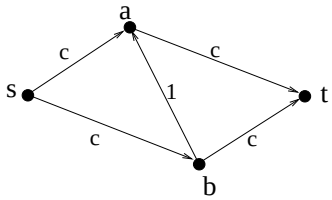


Figure 1: A network with a capacity of c on four of the edges and one on the cross edge.

(a) What is the worst case number of iteration of this network flow algorithm as a function of the capacity c?

(b) What is the worst case number of iteration of this network flow algorithm as a function of the number of edges m in the input network?

(c) What is the official "size" of a network?

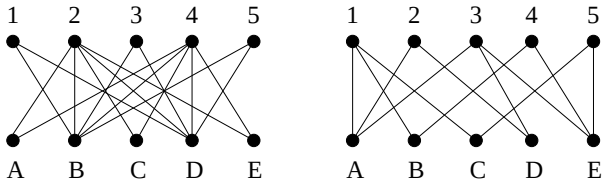
(d) What is the worst case number of iteration of this network flow algorithm as a function of the "size" of the input network.

9. Give an algorithm for solving the *Min Cut Problem.* Given a network $\langle G, s, t\rangle$ with capacities on the edges, find a minimum cut $C = \langle U, V\rangle$ where $s \in U$ and $t \in V$. The cost of cut its capacity $cap(C) = \sum_{u\in U} \sum_{v\in V} c_{\langle u,v\rangle}$. Prove that it gives the correct answer. Hint, you have already been told how to do this.

10. Express the network flow instance in Figure 1 as a linear program.

11. (Too hard for a test.) Let $G = (L \cup R, E)$ be a bipartite graph with nodes L on the left and R on the right. A matching is a subset of the edges so that each node appears at most once. For any $A \subseteq L$, let $N(A)$ be the neighborhood set of A, namely $N(A) = \{v \in R \mid \exists u \in A \text{ such that } (u, v) \in E\}$. Prove Hall's Theorem which states that there exists a matching in which every node in L is matched if and only iff $\forall A \subseteq L,\ |A| \leq |N(A)|$.

(a) For each of the following two bipartite graphs, either give a short witness to the fact that it has a perfect matching or to the fact that it does not. Use Hall's Theorem in your explanation as to why a graph does not have a matching. No need to mention flows or cuts.



(b) $\Rightarrow$: Suppose there exists a matching in which every node in L is matched. For $u \in L$, let $M(u) \in R$ specify one such matching. Prove that $\forall A \subseteq L,\ |A| \leq |N(A)|$.

(c) Look at both the slides and section 19.5 of the notes. It describes a network with nodes $\{s\} \cup L \cup R \cup \{t\}$ with a directed edge from s to each node in L, the edges E from L to R in the bipartite graph directed from L to R, and a directed edge from each node in R to t. The notes gives each edge capacity 1. However, The edges $\langle u, v\rangle$ across the bipartite graph could just as well be given capacity ∞.

Consider some cut (U, V) in this network. Note U contains s, some nodes of L, and some nodes of R, while V contains the remaining nodes of L, the remaining nodes of R, and t. Assume that $\forall A \subseteq L,\ |A| \leq |N(A)|$. Prove that the capacity of this cut, i.e. $cap(U, V) = \sum_{u\in U} \sum_{v\in V} c_{\langle u,v\rangle}$, is at least $|L|$.

(d) $\Leftarrow$: Assume that $\forall A \subseteq L,\ |A| \leq |N(A)|$ is true. Prove that there exists a matching in which every node in L is matched. Hint: Use everything you know about Network Flows.

(e) Suppose that there is some integer $k \geq 1$ such that every node in L has degree at least k and every node in R has degree at most k. Prove that there exists a matching in which every node in L is matched.