Homework #5
Introduction to Algorithms/Algorithms 1
600.363/463
Spring 2015

Due on: Tuesday, March 3th, 5pm
Late submissions: will NOT be accepted
Format: Please start each problem on a new page.
Where to submit: On blackboard, under student assessment
Please type your answers; handwritten assignments will not be accepted.
To get full credit, your answers must be explained clearly,
with enough details and rigorous proofs.

March 8, 2015

Problem 1 (20 point)

Let $G = G(V, E)$ be a directed graph represented by an adjacency list. G is a bipartite graph if it is possible to partition the vertices of G into two disjoint sets, i.e. $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$ such that there are no edges between vertices in the V_1 and there are no edges between vertices in the V_2. Design an algorithm that works in $O(|E| + |V|)$ time and checks if G is bipartite. Prove the correctness of your algorithm and analyze the running time.
Answer:
First, if we want to determine whether a graph G is a bipartite graph or not, we may try to make use of a classic result for bipartite graph

Lemma 0.1. *Every bipartite graph has chromatic number ≤ 2 (a.k.a 2-colorable). Any 2-colorable graph is bipartite.*

Proof. By definition, the chromatic number of a graph G is the smallest number of colors needed to color the vertices of G, so that there is no two adjacent vertices share the same color. From the definition of bipartite graph, V can be divided into two disjoint sets V_1 and V_2 and there are no edges between the vertices in V_1 and

between the vertices in V_2. So if there is no edge (a, b) where $a \in V_1$ and $b \in V_2$, we can just use one color to color all the vertices. If there is at least one such edge as (a, b), two colors are needs. This is because we need one color for V_1 and the other color for V_2 to avoid that two adjacent vertices share the same color.
If a graph is colored by only one color, vertex set can be divided into any two subsets that meet the definition. If a graph is colored by two colors, divide the vertices into two subsets based on color. By the coloring method mentioned above, any pair of vertices in the subsets are not adjacent.

One observation is that we can convert the directed graph G to undirected graph G^* and determine if G^* is bipartite. Let's provide a BFS coloring algorithm to decide whether G^* is bipartite.

1. Choose any uncolored vertex s and color s RED.
2. Color the neighbors of s BLACK. (use adjacency list)
3. Color the neighbors of all neighbor's RED. (use adjacency list)
4. If all vertices are colored, return TRUE, else repeat the steps starting from 1. While all the coloring steps, if a neighbor of the current vertex is already colored the same color as the current vertex, return FALSE. In this way, a proper 2-coloring is failed and graph G is not bipartite.
5. return TRUE.

Running Time:
The procedures take the same number of vertex search as BFS, which takes $O(|V^*|+|E^*|)$ time by using adjacency list. Since $|E^*| \leq |E|$, $O(|V^*|+|E^*|) = O(|V|+|E|)$.
Correctness:
By running the algorithm, we can use at most two colors to color all the vertices. If the coloring succeed finally without returning FALSE, by lemma 0.1, the graph is bipartite.

Problem 2 (20 points)

Let $G = (V, E)$ be a directed graph. $a \in V$ is a *central* vertex if for all $b \in V$ there exists a path from a to b. Provide the best algorithm that you can to test whether graph G has a central vertex. Prove the correctness of your algorithm and analyze the running time.
Answer:

First, we notice that the directed graph G can be cyclic. Is there any method we can get rid of all the cycles? Let's consider one possible algorithm described as following.

1. Use Kosaraju's algorithm to find all strongly connected components of the given graph G.
2. Build a new graph G^* by shrinking each SCC of G into one representative vertex and keeping the edges between SCCs.
3. If there is more than one vertex that has no incoming edges, return FALSE; else, return TRUE.

Running Time:
Assuming that the graph is presented by adjacency list, Kosaraju's algorithm runs in $O(|V| + |E|)$ time. Building the new graph G^* and finding outgoing-only vertices take $O(|V| + |E|)$. In total, the time complexity is $O(|V| + |E|)$.
Correctness:
I eliminated some details here. Briefly, let's consider the graph G^*. Clearly, G^* is a DAG. DAG has at least one vertex with no incoming edges (we can use proof by contradiction, eliminated here). So the only case left, one vertex with no incoming edges is actually the central vertex. This is because all other vertices have at least one incoming edge and can be reached by one of vertices. If there are more than one such vertex, e.g. two vertices, one cannot be reached by the other, which contracts the definition of central vertex.

Optional exercises

Solve the following problems and exercises from CLRS: 22.2-5, 22.2-8, 22.4-5, 22.5-4, 22-4, 22-1.