CS 141, Spring 2019 Homework 6

Problem 1. (25 points) [Graph Traversals]

Give a $O(n+m)$ time algorithm to determining whether the vertices of a connected undirected graph G can be colored by two different colors (say, red and blue) such that for every edge (u, v), u and v have different colors. When such coloring exists, your algorithm should also compute it.

Answer: Modify DFS as follows. Start running a DFS from an arbitrary node v and color v red. Traverse the rest of the graph using DFS: if you traverse a discovery edge, flip the color from the current color (i.e., red to blue or vice versa); if you consider a back edge, you check whether the color of the node at the "other" end has the same color of the source of that edge; if the two colors are the same, you have to stop and declare the graph not two-colorable. If the DFS colors all the nodes either red or blue and finds no conflicts, the graph is two-colorable (or bipartite).

Problem 2. (25 points) [Divide-and-conquer on Graphs]

Let $G = (V, E)$ be an undirected graph. A *triangle* in G is a cycle consisting of exactly three vertices (or, equivalently, three edges). Suppose that G is represented as an adjacency matrix. Give an algorithm to determine whether G contains any triangle in $O(n^{\log_2 7})$ time.

Answer: Let A be the adjacency matrix of G. Matrix A is squared. Find A^3 using Strassen's algorithm in time $O(n^{\log_2 7})$ and if any of entries $A[i, i] > 0$, then there is a 3-cycle.

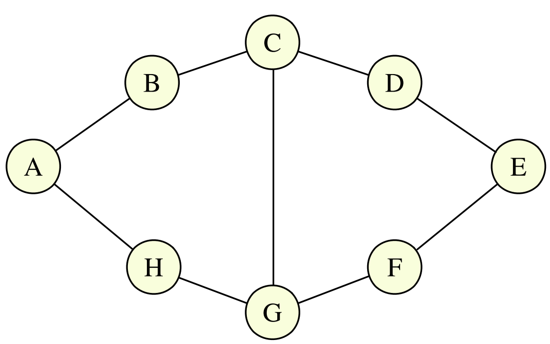
Problem 3. (25 points) [Greedy on Graphs]

Given an undirected graph $G = (V, E)$, an *independent set* in G is any set $I \subseteq V$ of vertices such that no two vertices in I are connected by an edge. In the maximum independent set problem (MIS), for a given graph G, we want to find an independent set of maximum size.

Here is our proposed greedy algorithm: (1) Set $I \leftarrow \emptyset$; (2) Repeat (3-4) until no nodes are left; (3) Choose a vertex v in G of minimum degree (breaking ties arbitrarily). (4) Add v to I and remove from G vertex v and all its neighbors.

Does this greedy algorithm always return the optimal solution? If you think it does, give a proof for the greedy choice property. If you think it does not, give a counterexample.

Answer: The algorithm is not optimal. Consider, for instance, the graph below.



The minimum degree is 2, so the algorithm could start, say, by choosing A, and remove B and H. Then it is left with a cycle of length 5, $CDEFGC$, from which it will choose two more vertices, for the total of three vertices. But the maximum independent set has size 4, namely $\{B, D, F, H\}$.

Problem 4. (25 points) [Dynamic Programming on Graphs]

Given a directed graph with non-negative integer edge weights, a pair of vertices s and t, and integers K and W, describe a dynamic-programming algorithm for deciding whether there exists a path from s to t that has total weight W and uses exactly K edges.

Your algorithm should run in time $O((n+m)WK)$, where n is the number of vertices and m is the number of edges. Analyze the time and space complexity of your solution.

Hint: You will have to define a three-dimensional table for the recurrence relation.

Answer:

Define $P[v,w,k]$ to be true if there is a path from s to v that has total weight w and uses exactly k edges. (For any vertex v and any integers w and k with $0 \le w \le W$ and $0 \le k \le K$.)

The following recurrence holds:

$P[v,0,0]$ is true for each vertex v.

$P[v,w,0]$ is false for $w > 0$ and each vertex v.

For $k > 0$, $P[v,w,k]$ is true if and only if $P[u,w-wt(u,v),k-1]$ is true for some edge (u,v).

Based on the recurrence, here is an algorithm:

1. `Set` $P[v,0,0]$ `= true for each vertex` v`.`
2. `Set` $P[v,w,0]$ `= false for each vertex` v `and` $w=1,2,...,W$`.`
3. `For` $k=1,2,...,K$ `do:`
4.   `For` $w=0,1,2,...,W$ `do:`
5.   `Set` $P[v,w,k]$ `= true if there is an edge` (u,v) `such that` $P[u,w-wt(u,v),k-1$ `is true (for each vertex` v`).`
6. `Return` $P[t,W,K]$`.`

The outer loop executes K times, the inner loop executes W times (for each iteration of the outer loop), and implementing line 5 takes linear time.