# CS 141 homework6

Celyna Su

TOTAL POINTS

**95 / 100**

QUESTION 1

## 1 Problem1 **25 / 25**

✓ **- 0 pts** Correct

**- 25 pts** No solution

**- 2 pts** Small mistake in coloring of vertices

**- 20 pts** Incorrect algorithm

**- 15 pts** Just returned true/false. Did not complete entire algorithm to color.

**- 10 pts** No coloring algorithm

**- 5 pts** Does not return if the graph is 2 colorable

**- 10 pts** No/Incorrect condition to check if neighbor is of same color

QUESTION 2

## 2 Problem2 **20 / 25**

**+ 25 pts** Correct

✓ **+ 20 pts** Correct algorithm but didn't indicate why it works

**+ 17 pts** A^2 and apply strassen

**+ 5 pts** Use strassen

**+ 0 pts** Incorrect answer

**- 5 pts** Incorrect/Missing time-complexity analyze

QUESTION 3

## 3 Problem3 **25 / 25**

✓ **- 0 pts** Correct

**- 15 pts** Showed that the algorithm is correct which is not.

**- 7.5 pts** Counter-example not given/Incorrect

**- 25 pts** No attempt

QUESTION 4

## 4 Problem4 **25 / 25**

✓ **- 0 pts** Correct

**- 2 pts** No/Incorrect proof of time complexity

**- 2 pts** No/Incorrect space complexity.

**- 25 pts** No solution

**- 10 pts** No/Incorrect algorithm

**CS 141**, Spring 2019 Homework 6

Posted: May 23rd, 2019 Due: May 30th, 2019

Name: Celyna Su

Student ID #: 862037643

- You are expected to work on this assignment on your own
- Use pseudocode, Python-like or English to describe your algorithms. Absolutely no C++/C/Java
- When designing an algorithm, you are allowed to use any algorithm or data structure we explained in class, without giving its details, unless the question specifically requires that you give such details
- Always remember to analyze the time complexity of your algorithms
- Homework has to be submitted electronically on Gradescope by the deadline. No late assignments will be accepted
- Unless otherwise noted, for all questions about graphs, $n = |V|$ is the number of vertices/nodes, and $m = |E|$ is the number of edges/links

**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**:
The runtime is $O(n+m)$, so the graph is represented by an adjacency list.
Let $n = vertices$ and $m = edges$. WTS the graph is bipartite, so we use Breadth First Search (BFS).

1. Pick a source node and color it red.
2. Color all its neighboring nodes blue.
3. For each blue node, color all is neighbor nodes red. Continue until no blue node is left.
4. Repeat steps 2 and 3 until there are no nodes left to travel.
5. Traverse graph:
    a. IF: run into node which is same color as its neighbor, return FALSE
    b. ELSE: return true and the new colored graph

**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**: (Credit to University of Toronto for help)
Let $A$ be the adjacency matrix of graph $G$. We will want to use Strassen's algorithm to compute the $B = A^3$ matrix in order to have a runtime of $O(n^{\log_2 7})$.

Looking into the entries of the $B = A^3$ matrix:

$$b_{ij} = \textstyle\sum_{k\in[n]} 2a_{ik}a_{ij} \longleftarrow \text{\# of common neighbors of } i \text{ and } j$$
$$b_{ij} > 0 \text{ if and only if there is a path of length exactly 2 between } i \text{ and } j.$$

When there is an edge between $i$ and $j$, then there also is a length 2 path between $i$ and $j$ through a vertex $k$ which constitutes a triangle.

1. Let $A$ and $B$ be square matrices of size $N$ x $N$.
2. Divide the matrices into submatrices of size $N/2$ x $N/2$.
3. We then use Strassen's matrix multiplication to multiply both matrices together.
4. If $b_{ij} > 0$ and $a_{ij} > 0$:
   a. Ret TRUE
   b. Ret FALSE

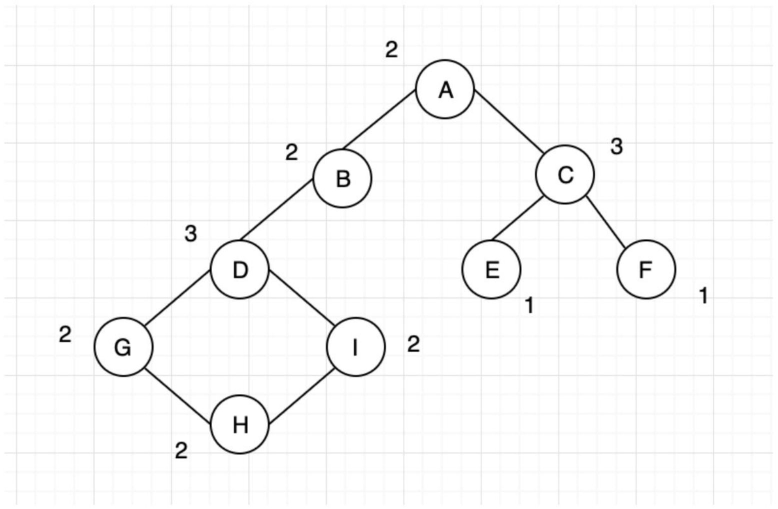
**Problem 3.** (25 points) [Greedy on Graphs]

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 above algorithm does not result in a complete solution every time.
Counterexample:



The minimum degree of the graph above is 1. Set $I$ now contains $E$, and remove $C$. Continuing through the algorithm, set $I$ now has $E, F$. Now for the left sub-tree the smallest degree is 2, however you are able to choose from $A, B, G, H, I$. If the algorithm chooses to add $H$ and then $B$, set $I$ can, at maximum, have a size of 4. $I = \{E, F, H, B\}$ whereas the maximum set is of size 5. One possible combination of size 5 is $I = \{E, F, A, I, G\}$. Since $I$ is not the maximum size it can be, this greedy algorithm is not the optimal solution.

**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 $O((n+m)WK)$. Analyze the time- and space-complexity of your solution. **Hint:** You will have to define a three-dimensional table for the recurrence relation.

**Answer**: (Credit to M.I.T for help)
Define $P[v,w,k]$ to be true if thee 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 $0 \leq w \leq W$ and $0 \leq k \leq K$).

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 $P[u, w - w(u,v), k-1]$ is TRUE for some edge $(u,v)$.

Thus, the algorithm would be:

```
1. set $P[v,0,0] \longleftarrow$ TRUE for ea. vertex $v$
2. set $P[v,w,0] \longleftarrow$ FALSE for ea. vertex $v$ and $w = 1,2,...,W$
3. for $k \longleftarrow 1,2,...,K$ do
    a. for $w \longleftarrow 0,1,2,...,W$ do
        b. set $P[v,w,k] \longleftarrow$ TRUE if there is an edge $(u,v)$ s.t $P[u, w - w(u,v), k-1] =$
           true (for ea. vertex $v$)
4. ret $P[t,w,k]$
```

The outer loop will execute $K$ times, the inner loop executes $W$ times for each iteration of the outer loop, and the innermost "set" runs in linear time. Therefore, the complexity is $O((n+m)WK)$.