**CSIE 2136: Algorithm Design and Analysis (Fall 2016)**

Final Exam

*Time: 14:20-17:20 (180 minutes) January 12, 2017*

**Instruction:** This is a close-book exam. There are 6 questions worth of 127 points, and the maximum score is 100. Please write clearly and concisely. Remember to print **your name, student ID, and page number on every answer page**.

## Problem 1: Short Answer Questions (27 points)

Answer the following questions and briefly justify your answer.

(a) True or False: A connected undirected graph has at least $|V|$ edges.

(b) True or False: A strongly connected directed graph has at least $|V|$ edges.

(c) True or False: A connected graph $G$ has a unique minimum spanning tree if all edges in $G$ have distinct weights.

(d) True or False: If a connected graph $G$ has a unique minimum spanning tree, then all edges in $G$ have distinct weights.

(e) True or False: If P = NP, every NP-hard problem can be solved in polynomial time.

(f) True or False: If P $\neq$ NP and problem A is NP-complete, then no algorithm can solve every instance of A in polynomial time.

(g) True or False: If P $\neq$ NP and problem A is NP-complete, given a problem instance $I$ of A, no algorithm can solve $I$ in polynomial time.

(h) True or False: If $A \leq_P B$ and there is an $O(n^2)$ algorithm for problem $B$, then there is an $O(n^2)$ algorithm for problem $A$.

(i) True or False: If problem A can be reduced to problem B in $O(n^2)$ time and there is an $O(n^3)$ algorithm for B, then there is an $O(n^3)$ algorithm for A.

## Problem 2: Basic Graph Problems (15 points)

No explanation is needed for this set of questions. For (b)-(e), see the figure below.

(a) (6 points) There are two common graph implementations: adjacency matrix and adjacency lists. What is the time/space complexity for each of the following in big-O notations? You can assume an direct graph, and use $V$ and $E$ to express the complexity.

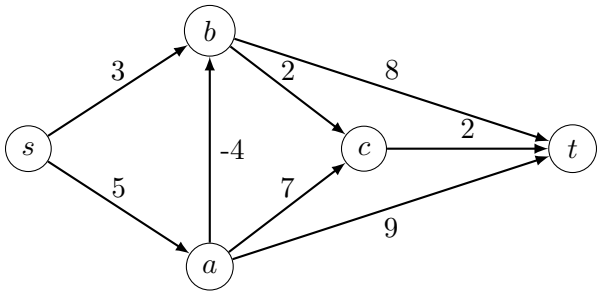
| | Adjacency lists | Adjacency Matrix |
|---|---|---|
| Storage space | | |
| Time to enumerate all edges | | |
| Time to find the **in-degree** of a vertex | | |

(b) (2 points) Find a topological order of the vertices.

(c) (2 points) What is the shortest path distance from $s$ to $t$?

(d) (2 points) What is the shortest path distance from $s$ to $t$ computed by Dijkstra's algorithm?

(e) (3 points) Find the number of possible paths from $s$ to $t$.




## Problem 3: Halting Problem (10 points)

After taking an algorithm class, your friend Ada would like to show that the Halting problem is undecidable by constructing a counterexample. Please help Ada to complete the proof: explain why 3(a) and 3(b) lead to contradiction.

1. Suppose $h(p, x)$ is an algorithm that determines whether a program $p$ halts on input $x$. That is, $h(p, x)$ returns 1 if $p$ halts on input $x$, and returns 0 otherwise.
2. Construct $g(p)$ such that it returns 0 if $h(p, p)$ is 0, and hangs forever otherwise.
3. There are two possible cases of $g(g)$, but both lead to contradiction:
   (a) If $g$ halts on $g$: ...
   (b) If $g$ does not halt on $g$: ...
4. Thus, $h(p, x)$ does not exist.

## Problem 4: Independent Set (15 points)

An independent set of a graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices such that each edge in $E$ is incident on at most one vertex in $V'$. In other words, no two vertices in $V'$ are joined by an edge. Given $G$ and $k$, the *IND-SET* problem is to determine whether $G$ has an independent set of size $\geq k$.

(a) (10 points) Suppose we know CLIQUE and VERTEX-COVER are NP-Complete. To prove IND-SET is NP-Hard, please construct a polynomial-time algorithm to reduce CLIQUE to IND-SET or VERTEX-COVER to IND-SET. (You only need to do one of them.) Please justify the correctness of this reduction.
Hint: Independent set, clique, and vertex cover are three closely related concepts. You need to show that $V'$ is an independent set of $G \iff V'$ is a clique of $G_c$. (You can also show that $V'$ is an independent set of $G \iff V - V'$ is a vertex cover of $G$.)

(b) (5 points) After taking an algorithm class, Ada learned a 2-approximation algorithm $A_{vc}$ for finding a minimum vertex cover. Now Ada would like to design a 2-approximation algorithm $A_{is}$ for finding a maximum independent set as follows:

**Algorithm 1:** Approximated Independent Set, $A_{is}$

**Input** : $G = (V, E)$
**Output**: A subset of vertex $V'$
1 $X = A_{vc}(G)$
2 return $V - X$

Is this algorithm correct? Please justify your answer.

## Problem 5: Zombie Apocalypse: 28 Days Later (20 points)

Due to a zombie virus outbreak, some cities are occupied by zombies and are no longer safe. You and your survivor team arrived at a shelter several months ago, and are now working with epidemiologists to prevent virus from spreading.

(a) (10 points) The virus propagation model can be presented as a **directed graph** $G = (V, E)$, where each vertex is a city and each edge is a road along which zombies can move and spread virus to the connected city. Each vertex is labeled with an integer $L(u)$ representing the strongest type of virus currently observed in the city; the higher the value, the stronger it is. The epidemiologists would like to predict the strongest type of virus in each city in the future when the virus spreads to all reachable cities. Please design an $O(V + E)$ algorithm to help them and briefly justify your answer. You can assume vertices are sorted based on $L(u)$.

(b) (10 points) To efficiently deliver food and medicine, the survivors decide to build secure tunnels connecting cities. However, after the initial price assessment, they found that building secure tunnels are expensive, so they will only build a tunnel if the tunnel is in at least one minimum spanning tree. Specifically, given an edge $e$ in an **undirected, weighted graph**, please design an $O(V + E)$ algorithm to determine whether the edge $e$ is in an MST and briefly justify your answer.
Hint: Recall the cycle property in MST.

## Problem 6: Redundant Ternary Counter (20 points)

We have seen that the amortized cost of an increment operation is $O(1)$ for a binary counter.

To achieve $O(1)$ amortized cost for both increment and decrement, we now consider a *redundant ternary counter*, each of its digits (called trit) can be either -1, 0, or 1. For example, $11_2 = 3$, $10(-1)_2 = 3$, and $1(-1)0_2 = 2$. Generally, the value of a k-bit redundant ternary counter is

$$\sum_{0}^{k-1} t_k 2^k,$$

where $t_k \in [-1, 0, 1]$.

The increment and decrement operations work similarly as in a binary counter. For example, the increment sequence from 0 to 7 is
$0, 1, 10, 11, 100, 101, 110, 111$.

Also, the decrement sequence from 7 to 0 is
$111, 110, 11(-1), 100, 10(-1), 1(-1)0, 1(-1)(-1), 0$.

(a) (5 points) The ternary counter is initialized to 0. What is the counter value in the ternary counter after each step of I, I, I, D, D, I, D, D? (I represents increment and D represents decrement.)

(b) (15 points) Please show that the amortized cost per operation is $O(1)$ in a redundant ternary counter. You can assume that the counter starts from 0, and the cost of changing the value of a trit is 1. In other words, you need to show that for any sequence of $n$ increment and decrement operations, the worst-case time is $O(n)$.

## Problem 7: Bonus Problem: Pokemon Master (20 points)

(a) (10 points) To become a Pokemon Master, you leave your hometown, vow to visit every city in the world exactly once before returning to your hometown. There are $n$ cities (including your hometown) in the world, and the distance from city $i$ to city $j$ is $d_{ij}$. Although you want to return home as soon as possible, you also hope to avoid seeing other competitors. Hence, please design an $O(2^n n^2)$-time algorithm to find the **number of shortest routes**.

(b) (10 points) After compeleting the world tour, your new mission is to train the $n$ Pokemons you caught. When Pokemon $i$ and Pokemon $j$ practice with each other, they both earn $p_{ij}$ experience points, where $p_{ij} \geq 0$ for all $i$ and $j$. Group practice is more effective: When Pokemons are divided into two groups $G_1$ and $G_2$, they can all earn $\sum_{i \in G_i, j \in G_2} p_{ij}$ experience points from the practice. Given a threshold $W$, the *Pokemon Training Problem* is to determine whether there is a way to divide the $n$ Pokemons into two practice groups such that they earn at least $W$ experience points. Is the Pokemon Training Problem NP-hard? If yes, please show the reduction from a known NPC problem; if not, please describe your algorithm and the running time.