University of Toronto
ECE-345: Algorithms and Data Structures
Solutions to Final Examination (Fall 2019)

1. **Multiple Choice, 20 points.**

   Each question is worth 2 points and it has *only one* correct answer. *Write clearly!* If we cannot understand your answer, you will receive no credit.

   (a) b
   (b) a
   (c) b
   (d) a
   (e) b
   (f) a
   (g) a
   (h) b
   (i) b
   (j) a

2. **Short Answers [Amortized Analysis, Max Flow, Minimum spanning trees], 5+7+8 points.**

   (a) It is easy to show that `Insert` in a heap can happen using $O(\log n)$ amortized time since a heap is a balanced tree. We can also deposit an additional $O(\log n)$ credits on each node to use when it is deleted. Clearly, every newly inserted node can be deleted only once. Using this new credit, `DeleteMin` can run in $O(1)$ amortized time, and a sequence of k delete operations can run in $O(k)$ time.

   (b) i. Compose the residual graph on the original flow. Add a positive 1 capacity on the edge that has been increased. Using BFS, search for an augmenting path; if the path exists, we can update the flow, otherwise, the flow is unchanged. We only need to do this once, as the augmenting path, if it exists, increases the flow by 1, which is the maximum increase possible.

   ii. Again, compose the residual graph on the original flow. If the decreased edge was not at capacity (that is, it still has positive residual capacity), then we can decrease the edge capacity by one without affecting the maximum flow. If not, then we add one to the negative capacity on the edge, and look for an augmenting path in reverse (going from t to s instead of from s to t) which includes the decreased edge.

   (c) Consider an MST T. Suppose there exists a spanning tree T' such that the largest edge weight is smaller than the largest edge weight in T. Call the corresponding edges, e' and e respectively. Remove e from T. This breaks T into two connected components. There must exist an edge e'' in T' that connects these components. Clearly, $w(e'') \leq w(e') < w(e)$. Thus the tree T'' obtained from T by replacing the edge e with e'' has weight $w(T'') = w(T) + w(e'') - w(e) < w(T)$, which is a contradiction.
   We can use Prim's algorithm to solve the problem in time $O(ElogV + E)$.

3. **Greedy Algorithms, 5+5+10 points.**

   (a)

i	1	2
d_i	1	2
t_i	3	2
$\frac{d_i}{t_i}$	$\frac{1}{3}$	1

   Here, choosing the easiest assignment first (1,2) yields $1 \cdot 3 + 2 \cdot 5 = 13$.
   Choosing (2,1) yields $2 \cdot 2 + 1 \cdot 5 = 9$ Generally any instance where the ratio $\frac{d_i}{t_i}$ of the first assignment is smaller than the second.

(b)

i	1	2
d_i	1	3
t_i	2	3
$\frac{d_i}{t_i}$	$\frac{1}{2}$	1

Here, choosing the easiest assignment first (1,2) yields $1 \cdot 2 + 3 \cdot 5 = 17$.
Choosing (2,1) yields $3 \cdot 3 + 1 \cdot 5 = 14$ Generally any instance where the ratio $\frac{d_i}{t_i}$ of the first assignment is smaller than the second.

(c) First, consider what happens when we swap adjacent assignments.

$$
\begin{aligned}
T(S) &= \sum_{j=1}^{n} \sum_{k=1}^{j} f_{s_j} l_{s_k} \\
T(S') &= T(S) - \sum_{k=1}^{i} d_{s_i} t_{s_k} - \sum_{k=1}^{i+1} d_{s_{i+1}} t_{s_k} + \sum_{k=1}^{i+1} d_{s_i} t_{s_k} + \sum_{k=1}^{i} d_{s_{i+1}} t_{s_k} \\
&= T(S) + d_{s_i} t_{s_{i+1}} - d_{s_{i+1}} t_{s_i}
\end{aligned}
$$

What can we say about the new ordering S'? This depends on $\frac{d_{s_i}}{t_{s_i}}$ relative to $\frac{d_{s_{i+1}}}{t_{s_{i+1}}}$.

$$\frac{d_{s_i}}{t_{s_i}} \leq \frac{d_{s_{i+1}}}{t_{s_{i+1}}} \tag{1}$$

$$d_{s_i} t_{s_{i+1}} - d_{s_{i+1}} t_{s_i} \leq 0 \tag{2}$$

$$T(S') \leq T(S) \tag{3}$$

So if the ratio of the earlier assignment was lower than the higher assignment, then we can reduce the total effort by swapping them.

Let g_1 be the assignment with the highest ratio. To prove the greedy choice property, we need an optimal solution to contain g_1 as the first assignment. If we consider any optimal solution O where $o_1 \neq g_1$ then we can construct a solution that is just as optimal by swapping g_1 with whichever comes before it, until it is the first choice.

To prove optimal substructure. Given that we greedily pick the first assignment, we are left with $n-1$ programs that we need to order to minimize T. Say we have optimal solution $O = \{o_1, o_2, \ldots, o_n\}$, and optimal subsolution $O_s = \{o_2, \ldots, o_n\}$. For the sake of contradiction, assume that O_a is not the optimal solution to ordering the remaining $n-1$ assignments. Then there exists O_b such that $T(O_a) > T(O_b)$. Then construct $O' = \{o_1\} \cup O_b$. Then $T(O') = d_{o_1} t_{o_1} + T(O_b) < d_{o_1} t_{o_1} + T(O_a) = T(O)$, yielding a contradiction as we assumed that O was optimal. Correctness follows from optimal substructure and the greedy choice property.

4. **Shortest Paths, 20 points.**

Initialize $N[u] = 0$ for all $u \neq v$ and $N[v] = 1$. Change RELAX as follows:

RELAX(u, v)
  if $d[v] > d[u] + w(u, v)$ **then**
    $d[v] \leftarrow d[u] + w(u, v)$
    $N[v] \leftarrow N[u]$
  else if $d[v] = d[u] + w(u, v)$ **then**
    $N[v] \leftarrow N[v] + N[u]$

The idea is that, when we find a new shortest path to a node v, we set the number of shortest paths to that of the predecessor u. However, when we find a second shortest path to v that is equal to the current shortest path estimate $d[v]$, (and u is the predecessor for the new path) we add the number of shortest paths to u to the existing value of $N[v]$.

5. **NP-Completeness, 3+12+5 points.**

(a) Certificate: Boolean assignments to the n variables of Φ.

Verify: Walk the formula and check if each clause has at least one `true` literal and at least one `false` literal. Since the formula is walked once, the runtime is clearly $\mathcal{O}(mn)$, since there are m clauses and a clause cannot contain more than one copy of the same literal.

(b) Given a 3-CNF formula Φ, we will construct a 4-CNF formula Φ' where Φ is satisfiable *iff* Φ' is NAE-satisfiable. The formula Φ' has the same n variables as Φ and a new variable z. For each clause $(x_i \vee x_j \vee x_k)$ of Φ, create the clause $(x_i \vee x_j \vee x_k \vee z)$ in Φ' (that is, disjoin z to each clause).

Claim: Φ is satisfiable $\Leftrightarrow$ Φ' is NAE-satisfiable.

Proof:

$\Rightarrow$: Assume Φ is satisfiable with a truth assignment I. Therefore I assigns `1` to at least one literal in each clause of Φ. The truth assignment $I \cup \{z = \texttt{0}\}$ is therefore an NAE-satisfying assignment of Φ'.

$\Leftarrow$: Assume Φ' is NAE-satisfiable with truth assignment I. Notice from the definition of an NAE-satisfying assignment, the complement of any NAE-satisfying assignment is also an NAE-satisfying assignment. We must consider two cases. If $z = \texttt{0}$, then I assigns `1` to at least one literal of each clause of Φ, so I is a satisfying assignment for Φ and the theorem holds. If $z = \texttt{1}$, then the complement of I is also an NAE-satisfying assignment and it has $z = \texttt{0}$, so it is an NAE-satisfying assignment of Φ.

(c)

$$\Phi' = (x_1 \vee x_2 \vee x_3 \vee z) \wedge (\bar{x_2} \vee \bar{x_3} \vee x_4 \vee z)$$

One satisfying assignment of Φ assigns `1` to every variable. A corresponding NAE-satisfying assignment of Φ' extends that assignment by assigning `0` to z.