CS 341: Algorithms @ Waterloo **Summer 2017**
Final Exam **Date: Aug 1**

There are totally 115 marks. The full mark is 100. This exam paper has two pages and seven questions.

---

1. **Printing Neatly** (20 marks)

   Consider the problem of neatly printing a paragraph with a monospaced font (all characters having the same width) on a printer. The input text is a sequence of n words of lengths $l_1, l_2, \ldots, l_n$, measured in characters. We want to print this paragraph neatly on a number of lines that hold a maximum of M characters each. Our criterion of "neatness" is as follows. If a given line contains words i through j, where $i \leq j$, and we leave exactly one space between words, the number of extra space characters at the end of the line is $M - j + i - \sum_{k=i}^{j} l_k$, which must be nonnegative so that the words fit on the line. We wish to minimize the sum, over all lines except the last, of the squares of the numbers of extra space characters at the ends of lines.

   Design a dynamic programming algorithm to print a paragraph of n words neatly on a printer. Describe your algorithm clearly (including the subproblems and the recurrence relation), briefly justify its correctness and analyze its time complexity. Note that your algorithm needs to return an optimal solution (i.e. where to put the line breaks). You will get full marks if your algorithm is correct and your explanation is good and the time complexity is $O(nM)$.

2. **Program Committee** (15 marks)

   In a computer science conference, there are a number of research paper submissions, and there is a set of program committee members to review these submissions. Each program committee member has different expertise, and each will tell the program committee chair his/her preferences, by choosing a subset of the submissions that he/she is willing to review. The program committee chair needs to assign each submission to three program committee members, and ideally each program committee member is assigned the same number of submissions (i.e. balance workload) and is only assigned the submissions that he/she prefers. Suppose n is the number of program committee members and m is the number of submissions and $3m$ is divisible by n. Design an efficient algorithm to help the program committee chair to find an ideal assignment if it exists (and report none if it does not exist). Describe your algorithm clearly, briefly justify its correctness, and analyze its time complexity. You will get full marks if your algorithm is correct and your explanation is good and the time complexity is polynomial in n and m.

3. **Simple Shortest Paths** (15 marks)

   (a) (7 marks) Prove that the following Hamiltonian s-t path problem is NP-complete.
   Input: A directed graph $G = (V, E)$ and two vertices $s, t \in V$.
   Output: Does there exist a Hamiltonian path from s to t in G?

   (b) (8 marks) Use (a) or otherwise, prove that the shortest simple path problem is NP-complete.
   Input: A directed graph $G = (V, E)$ in which each edge $e \in E$ has a length l_e, two vertices $s, t \in V$ and an integer L. Note that the numbers l_e and L could be negative.
   Output: Does there exist a *simple* path P from s to t with total length at most L, i.e. $\sum_{e \in P} l_e \leq L$? A path is simple if it visits every vertex at most once.

4. **Vertex Cover on Trees** (15 marks)

   Input: An undirected tree $T = (V, E)$ and a positive weight w_v on each vertex $v \in V$.

   Output: The minimum total weight of a vertex cover. Recall that a subset of vertices $S \subseteq V$ is a vertex cover if $S \cap \{u, v\} \neq \emptyset$ for every $uv \in E$.

   Give an efficient algorithm to solve this problem. Describe your algorithm clearly, briefly justify its correctness and analyze its time complexity. Note that your algorithm only needs to return the optimal value (not required to output an optimal vertex cover solution). You will get full marks if your algorithm is correct and your explanation is good and the time complexity is $O(|V|)$.

5. **SAT Solver** (10 marks)

   Suppose there is a black box algorithm B that given any 3SAT formula with n variables and m clauses it will determine whether the formula is satisfiable or not (just return YES/NO) in time complexity polynomial in n and m. Show how to use B (possibly multiple times) to find a satisfying assignment (a truth assignment to the variables that satisfies all the clauses) when it exists. Describe your algorithm clearly, briefly justify its correctness, and analyze its time complexity. You will get full marks if your algorithm is correct and your explanation is good and the time complexity is polynomial in n and m.

6. **Hitting Set** (10 marks)

   Prove that the following hitting set problem is NP-complete.

   Input: m sets $S_1, S_2, \ldots, S_m$ where each $S_i \subseteq \{1, \ldots, n\}$, and an positive integer k.

   Output: Does there exist a subset $T \subseteq \{1, \ldots, n\}$ with $|T| \leq k$ such that $T \cap S_i \neq \emptyset$ for $1 \leq i \leq m$? In words, does there exists a subset T with at most k elements that intersects every set S_i?

7. **Diet Hard** (30 marks)

   Your friend is on a diet. He/she went to see a doctor and the doctor recommended your friend to have enough vitamins A, B, and C in his/her daily diet. The doctor sets out a target intake for each vitamin and suggests a list of healthy food items for your friend to choose from. Help your friend to construct a diet that fulfills the target intakes of vitamins and minimizes the total calories.

   Input: We are given a set of n food items, each item i has a_i units of vitamin A, b_i units of vitamin B, c_i units of vitamin C, and k_i units of calories. We are also given the target intake units of vitamin A,B, and C, denoted by $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$ respectively. You can assume that all a_i, b_i, c_i and $\mathcal{A}, \mathcal{B}, \mathcal{C}$ are nonnegative integers.

   Output: A subset $S \subseteq \{1, 2, \ldots, n\}$ of food items with minimum total calories that fulfills the target intakes of each vitamin. More formally, your algorithm should return a subset S that minimizes $\sum_{i \in S} k_i$ satisfying the constraints $\sum_{i \in S} a_i \geq \mathcal{A}$, $\sum_{i \in S} b_i \geq \mathcal{B}$, and $\sum_{i \in S} c_i \geq \mathcal{C}$. You can assume that such a set always exists.

   (a) (20 marks) Design an algorithm to solve this problem. Describe your algorithm clearly, briefly justify its correctness, and analyze its time complexity. Note that your algorithm should return an optimal subset, not just the optimal value. The time complexity may depend on $n, \mathcal{A}, \mathcal{B}, \mathcal{C}$.

   (b) (10 marks) Prove that the problem is NP-complete, i.e. the decision problem of determining whether there is a subset with total calories K fulfilling the target values is NP-complete. (This justifies the dependency on $\mathcal{A}, \mathcal{B}, \mathcal{C}$ in the time complexity in the previous part.)

"I don't count calories or carbs.
Dieting is hard enough without math!"