McGill University COMP360 Winter 2011 **Instructor: Phuong Nguyen**

Assignment 6

Due February 28 before the test

The work you submit must be your own. You may discuss problems with each others; however, you should prepare written solutions alone. Copying assignments is a serious academic offense, and will be dealt with accordingly.

Question 1 (10pt) Let S be a non-empty set. Then a non-empty family $\mathcal{L}$ of subsets of S is said to be *nice* if it satisfies the following conditions:

1. **Inclusion property**: For every subsets $A, B \subseteq S$, if $B \in \mathcal{L}$ and $A \subset B$ then $A \in \mathcal{L}$ as well. (Note that the empty set $\varnothing$ is necessarily a member of $\mathcal{L}$.)
2. **Exchange property**: If $A \in \mathcal{L}$ and $B \in \mathcal{L}$ and $|A| < |B|$ (here $|A|$, $|B|$ denote the cardinalities of A and B, respectively), then there is some element $x \in (B - A)$ such that $A \cup \{x\}$ is an element of $\mathcal{L}$.

A subset A in $\mathcal{L}$ is called a *top* set if there is no other subset B in $\mathcal{L}$ such that $A \subset B$.

Here are some examples:

1. The family of all subsets of S is nice.
2. $S = \{a, b, c, d\}$ and $\mathcal{L} = \{\varnothing, \{a\}, \{b\}, \{a, b\}\}$. Then $\mathcal{L}$ is nice. But $\mathcal{L}' = \{\varnothing, \{a\}, \{b\}, \{a, b\}, \{a, b, c\}\}$ is not nice, because it violates the Inclusion property: $\{b, c\} \subset \{a, b, c\}$ and $\{a, b, c\} \in \mathcal{L}'$ but $\{b, c\} \notin \mathcal{L}'$.
3. $S = \{a, b, c, d, e\}$. $\mathcal{L}_1 = \{\varnothing, \{a\}, \{b\}, \{a, b\}\}$ and $\mathcal{L}_2 = \{\varnothing, \{c\}, \{d\}, \{e\}, \{c, d\}, \{d, e\}, \{e, c\}, \{c, d, e\}\}$ are both nice. But $\mathcal{L}_3 = \{\varnothing, \{a\}, \{b\}, \{a, b\}, \{c\}, \{d\}, \{e\}, \{c, d\}, \{d, e\}, \{e, c\}, \{c, d, e\}\}$ is not nice, because it violates the Exchange property: $\{a, b\}$ and $\{c, d, e\}$ are both in $\mathcal{L}_3$ and $|\{a, b\}| < |\{c, d, e\}|$, but there is not element x in $\{c, d, e\}$ such that $\{a, b\} \cup \{x\}$ is a member of $\mathcal{L}_3$.

Now suppose that S is a set of n elements, $n > 0$, and let $\mathcal{L}$ be a nice family of subsets of S. Suppose that each element x of S has a positive weight $w(x)$. The weight of a subset A of S is defined to the be total weight of all elements in A:

$$w(A) = \sum_{x \in A} w(x)$$

The problem here is to fine a subset in $\mathcal{L}$ that has maximum total weight. Notice that any such subset is necessarily a top set.

You are asked to solve this problem efficiently by a greedy algorithm. Note that $\mathcal{L}$ can potentially have up to 2^n members, so we don't want to go through all of them. In fact, your algorithm must be a polytime algorithm (assuming that the function $t(n)$ below is a polynomial).

a) Give a greedy algorithm that finds a subset in $\mathcal{L}$ of maximum total weight. Prove that your algorithm is correct.

b) To analyze the running time of your algorithm, we assume that checking whether a subset A is a member of $\mathcal{L}$ takes time $t(n)$. What is the running time of your algorithm in terms of n and $t(n)$? (State your answer using $\mathcal{O}$ notation.)

Question 2 (10pt) Consider the following variant of the Load Balancing problem, called Weighted Load Balancing problem here. The input is a set of k (normal) processors and m *fast* processors, that can run twice as fast as the normal processors. There are n jobs where each job i has a duration t_i and needs to be processed on one processor. On a normal processor job i takes up a contiguous duration of t_i units of time, but on a fast processor it takes up only $t_i/2$ units of time. The problem is to schedule jobs on processors in such a way that minimizes the maximum processing time (that is, the load) of the processors.

For example, suppose that there are one normal processor and one fast processor (i.e., $k = m = 1$), and there are three jobs with durations $t_1 = 15, t_2 = 2, t_3 = 6$. The the best schedule is to have job 1 on the fast processor, jobs 2 and 3 on the normal processor (maximum load is 8 here). On the other hand, if we jobs 1 and 2 on the fast processor and job 3 on the normal processor, the maximum load is 8.5.

You are asked to give a greedy approximation algorithm for this problem that achieves approximation ratio 2. That is, the output schedule must have maximum load at most twice the optimal maximum load. Prove that this is indeed the case. Your algorithm must run in time polynomial in m, k, n and $\sum_{i=1}^{n} \log t_n$. (Here all t_i are positive integers.)