CSE 202: Algorithm Design and Analysis Spring 2015

Problem Set 4

Instructor: Fan Chung Graham **Due on:** Wed, June 3, 2015

Instructions

- For your proofs, you may use any lower bound, algorithm or data structure from the text or in class, and their correctness and analysis, but please cite the result that you use.
- If you do not prove that your algorithm is correct, we will assume that it is incorrect. If you do not provide an analysis of the running time, we will assume you do not know what the running time is.

Problem 1

Consider the following modification to the Load Balancing problem with access to machines with different processing power. Suppose you have a system that consists of m slow machines and k fast machines. The fast machines can perform twice as much work per unit time as the slow machines. Now you?re given a set of n jobs; job i takes time t_i to process on a slow machine and time $\frac{1}{2}t_i$ to process on a fast machine. You want to assign each job to a machine; as before, the goal is to minimize the makespan–that is the maximum, over all machines, of the total processing time of jobs assigned to that machine.

Give a polynomial-time algorithm that produces an assignment of jobs to machines with a makespan that is at most three times the optimum.

Problem 2

Suppose you are given a set of positive integers $A = \{a_1, a_2, \ldots, a_n\}$ and a positive integer B. A subset $S \subseteq A$ is called *feasible* if the sum of the numbers in S does not exceed B:

$$\sum_{a_i \in S} a_i \leq B.$$

The sum of the numbers in S will be called the *total sum* of S.

You would like to select a feasible subset S of A whose total sum is as large as possible.

Example. If $A = \{8, 2, 4\}$, and $B = 11$, then the optimal solution is $S = \{8, 2\}$.

(a) Here is an algorithm for this problem:

```
Initially S is empty
Define T = 0
for i = 1, 2, ..., n
    if T+a[i] <= B:
        add a[i] to S
        T = T + a[i]
```

Give an instance in which the total sum of the set S returned by this algorithm is less than half the total some of some other feasible subset of A.

(b) Give a polynomial-time approximation algorithm for this problem with the following guarantee: It returns a feasible set $S \subseteq A$ whose total sum is at least half as large as the maximum total sum of any feasible set $S' \subseteq A$. Your algorithm should have a running time at most $O(n \log n)$.