COS 423	Theory of Algorithms	Spring 2013

Precept 6

These problems (or a subset of them) will be solved in precept.

1. *Scheduling to minimize total flow time.* You are given a set of n jobs that each require processing on one of m processors. Job j requires $p(i,j) > 0$ units of processing time if scheduled on machine i. Your job is to find an assignments of jobs to machines *and* an order for performing the jobs assigned to each machine. The *flow time* of a job j is the time at which job j is finished. For example, if jobs 1, 2, and 3 are assigned to machine i in that order, then the flow time of job 1 is $p(i,1)$, the flow time of job 2 is $p(i,1)+p(i,2)$, and the flow time of job 3 is $p(i,1)+p(i,2)+p(i,3)$. Design a polynomial-time algorithm to find a schedule that minimizes the sum of the flow times of the n jobs.

2. *Binary counter that costs 2^k to flip the k^{th} bit.* Consider a binary counter (with no limit on the number of bits) in which it costs 2^k dollars to flip the k^{th} least-significant bit. Prove that, starting from the zero counter, any sequence of n INCREMENT operations costs $\mathcal{O}(n \log n)$ dollars.

   Prove the statement using the aggregate method.

3. *Queue with two stacks.* Prove that you can implement a *queue* with two stacks so that each queue operation takes a constant *amortized* number of stack operations. That is, starting from an empty queue, any sequence of n ENQUEUE and DEQUEUE operations takes $\mathcal{O}(n)$ stack operations. You may use only $\mathcal{O}(1)$ extra memory beyond the memory for the three stacks.

   First prove the statement using the accounting method, then use a potential function method.