Homework #4

(Due: Dec 19)

```
SELECT-MOVIES( $n$, $m$, $k$, $r[1..n][1..m]$ )
(Return a subset $M \subseteq \{1, 2, \ldots, m\}$ with $|M| \leq k$ that maximizes
$\sum_{i=1}^{n} \max_{j \in M}\{r[i][j]\}$. Here $n > 0$, $m \geq k > 0$, and $r_{i,j} = r[i][j] \geq 0$
for $i \in [1, n]$ and $j \in [1, m]$.)

1. Let $R(S) = \sum_{i=1}^{n} \max_{j \in S}\{r[i][j]\}$ for any given $S \subseteq \{1, 2, \ldots, m\}$.
2. $M \leftarrow \emptyset$
3. while $|M| < k$ do
4.     Let $i \in \{1, 2, \ldots, n\} \setminus M$ maximizes $R(M \cup \{i\}) - R(M)$
5.     $M \leftarrow M \cup \{i\}$
6. return $M$
```

Figure 1: [Task 1] An approximation algorithm for selecting movies.

Task 1. [90 Points] SBUCS Movie Club

Some students of the SBU Computer Science Department are toying with the idea of forming a movie club. The plan is to screen one movie per week during regular semesters. The approach they will be taking for selecting the movies is as follows. Suppose the club has $n > 0$ members and they need to select at most $k > 0$ movies for a semester from a pool of $m \geq k$ movies. Each member $i \in [1, n]$ will give a rating $r_{i,j} \geq 0$ to each movie $j \in [1, m]$. Then they will find a subset $M \subseteq \{1, 2, \ldots, m\}$ with $|M| \leq k$ that maximizes $\sum_{i=1}^{n} \max_{j \in M}\{r_{i,j}\}$. The movies in M will be screened that semester. An approximation algorithm for selecting the movies based on the criteria above is given in Figure 1.

This task asks you to find the approximation bound of SELECT-MOVIES given in Figure 1.

(*a*) [**20 Points**] Let $R(S) = \sum_{i=1}^{n} \max_{j \in S}\{r_{i,j}\}$ for any given $S \subseteq \{1, 2, \ldots, m\}$. Now let $S' \subset \{1, 2, \ldots, m\}$ and $S'' \subseteq S'$. Then prove that for any $i \in [1, m]$ such that $i \notin S'$, the following holds: $R(S' \cup \{i\}) - R(S') \leq R(S'' \cup \{i\}) - R(S'')$.

(*b*) [**30 Points**] Let $M_{opt} \subseteq \{1, 2, \ldots, m\}$ with $|M_{opt}| \leq k$ be a set that maximizes the objective function, i.e., ensures $R(M_{opt}) \geq R(S)$ for every $S \subseteq \{1, 2, \ldots, m\}$. Let M be the set of movies chosen before some iteration of the ***while*** loop of lines 3–5 of SELECT-MOVIES and let i be the movie chosen in that iteration. Then use part (*a*) to show that $R(M \cup \{i\}) - R(M) \geq \frac{1}{k}(R(M_{opt}) - R(M))$.

(*c*) [**40 Points**] Use part (*b*) to prove that SELECT-MOVIES is a $1 + \frac{1}{e-1}$ approximation algorithm for the movie selection problem.

Task 2. [90 Points] Parallel Sorting

This task asks you to sort n distinct numbers in $\mathcal{O}\left(\log^2 n\right)$ span. Recall that the parallel mergesort algorithm we saw in the class (lecture 12) runs in $\Theta\left(\log^3 n\right)$ span in the worst case, and the randomized quicksort algorithm runs in $\Theta\left(\log^3 n\right)$ span w.h.p. in n. We assume that a parallel ***for*** loop executing n iterations has span $\mathcal{O}\left(\log n + k\right)$, where k is the maximum span (among all iterations) of the loop body.

(a) [**30 Points**] Give a simple algorithm that sorts n distinct numbers in $\Theta\left(n^2\right)$ work and $\Theta\left(\log n\right)$ span in the worst case.

(b) [**30 Points**] Show that the parallel randomized quicksort algorithm we saw in the class can be run in expected $\mathcal{O}\left(n \log n\right)$ work and expected $\mathcal{O}\left(\log^2 n\right)$ span by modifying the parallel partition algorithm not to use prefix sums.

(c) [**30 Points**] Show that the parallel randomized quicksort algorithm we saw in the class can be run in $\mathcal{O}\left(\log^2 n\right)$ span w.h.p. in n by using a parallel partition algorithm that does not use prefix sums. How much work does it perform?