CSE548, AMS542: Analysis of Algorithms, Fall 2017 **Date:** Dec 3

Homework #4

(Due: Dec 12)

Task 1. [80 Points] Selection in Parallel

Given an array of n distinct numbers and an integer $k \in [1, n]$, this task asks you to select and return the k-th smallest number in the array efficiently in parallel.

```
Select ( A[ q : r ], k )
    n ← r − q + 1
    if n ≤ 140 then
        sort A[ q : r ] and return A[ q + k - 1 ]
    else
        divide A[ q : r ] into blocks B_i's each containing 5 consecutive elements
                ( last block may contain fewer than 5 elements )
        for i ← 1 to ⌈ n / 5 ⌉ do
            M[ i ] ← median of B_i using sorting
        x ← Select ( M[ 1 : ⌈ n / 5 ⌉ ], ⌊( ⌈ n / 5 ⌉ + 1 ) / 2 ⌋ )  { median of medians }
        t ← Partition ( A[ q : r ], x )    { partition around x which ends up at A[ t ] }
        if k = t − q + 1 then return A[ t ]
        else if k < t − q + 1 then return Select ( A[ q : t − 1 ], k )
            else return Select ( A[ t + 1 : r ], k − t + q − 1 )
```

Figure 1: The deterministic selection algorithm from lecture 6 (slide 2) which finds the k-th smallest number in an array $A[q : r]$ of $n = r - q + 1$ distinct numbers, where $1 \le k \le n$.

(*a*) [**20 Points**] Parallelize the deterministic selection algorithm shown in Figure 1 which is taken from lecture 6 (slide 2). Write down the pseudocode of the parallel version. Analyze its work, span and parallelism.

(*b*) [**20 Points**] Consider the parallel randomized quicksort algorithm shown in Figure 2 which is taken from lecture 12 (slide 86). It sorts an array of n distinct numbers in increasing order of value. How will you modify it so that it returns the k-th smallest number in the array instead of sorting them, where $k \in [1, n]$. Write down the pseudocode of the parallel selection algorithm you obtain. Give high-probability bounds on its work, span and parallelism.

(*c*) [**40 Points**] Design a parallel selection algorithm that can return the smallest k numbers in an array of n distinct numbers in sorted order using $\Theta(nk)$ extra space, $\Theta(nk)$ work and $\Theta(\log^2 n)$ span. Write down the pseudocode and show details of your analyses of work, span and space usage.

```
Par-Randomized-QuickSort ( A[ q : r ] )
n ← r − q + 1
if n ≤ 30 then
    sort A[ q : r ] using any sorting algorithm
else
    select a random element x from A[ q : r ]
    k ← Par-Partition ( A[ q : r ], x )
    spawn Par-Randomized-QuickSort ( A[ q : k − 1 ] )
    Par-Randomized-QuickSort ( A[ k + 1 : r ] )
    sync
```

Figure 2: The parallel randomized quicksort algorithm from lecture 12 (slide 86) which sorts an array of distinct numbers in increasing order of value.

SELECT-CHOCOLATE-BOXES-TO-BUY($\mathcal{B} = \langle B_1, B_2, \ldots, B_n \rangle$, $\langle p_1, p_2, \ldots, p_n \rangle$, $\mathcal{C} = \langle C_1, C_2, \ldots, C_m \rangle$)

Input: A sequence of n chocolate boxes $\mathcal{B} = \langle B_1, B_2, \ldots, B_n \rangle$ with p_i giving the price of box B_i for $1 \le i \le n$, and a sequence of m chocolate types $\mathcal{C} = \langle C_1, C_2, \ldots, C_m \rangle$ given in non-increasing order of my likeness for them.

Output: A subset of boxes to buy from $\mathcal{B}$ such that together they contain at least one chocolate of each type.

array $f[1:n]$
for $i \leftarrow 1$ ***to*** n ***do***
    $f[i] \leftarrow p_i$
for $i \leftarrow 1$ ***to*** m ***do***
    let exactly $k' \in [1, k]$ boxes from $\mathcal{B}$ contain C_i, and ***let*** $B_{i_1}, B_{i_2}, \ldots, B_{i_{k'}}$ be those k' boxes
    let i_l be an index from $\{i_1, i_2, \ldots, i_{k'}\}$ such that $f[i_l]$ is the minimum among $f[i_1], f[i_2], \ldots, f[i_{k'}]$
    $r \leftarrow f[i_l]$
    for $j \leftarrow 1$ ***to*** k' ***do***
        $f[i_j] \leftarrow f[i_j] - r$
$S \leftarrow \emptyset$
for $i \leftarrow 1$ ***to*** n ***do***
    if $f[i] = 0$ ***then***
        $S \leftarrow S \cup \{B_i\}$
return S

Figure 3: The algorithm I used to buy boxes of chocolates so that together they contain at least one choclate of each type.

Task 2. [50 Points] Chocolates

The *Life is a Box of Chocolates* chocolatier sells m different types of chocolates in n different boxes (i.e., assortments) $B_1, B_2, \ldots, B_n$. Each box contains at most one chocolate of each type, and each type of chocolate is included in at least one box and at most $k > 0$ different boxes. If $|B_i|$ denotes the number of chocolates in box B_i then $1 \leq |B_i| \leq m$. However, $|B_i| = |B_j|$ is not necessarily true when $i \neq j$. The price of box B_i is $p_i (> 0)$, where $1 \leq i \leq n$.

Though I like some types of chocolates much more than some other types, I still want to buy enough boxes so that together they contain at least one chocolate of each type. But I want to select the boxes in a way that minimizes the money I spend in buying them. Since I do not know how to do that efficiently, I used the approximation algorithm shown in Figure 3 instead.

(a) [**50 Points**] Prove that the algorithm shown in Figure 3 is a k-approximation algorithm for selecting boxes of the minimum total cost that include at least one chocolate of each type. In other words, prove that I will not have to spend more than a factor of k more money than someone using an optimal algorithm for selecting the boxes.

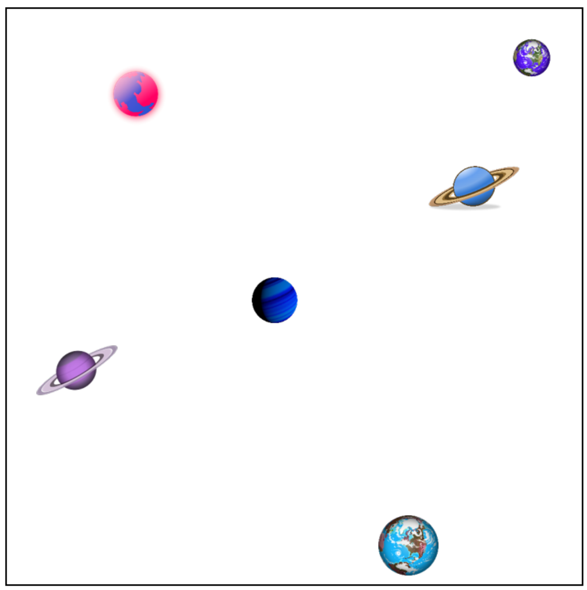

Figure 4: Celestial discs in a 2D universe.

Task 3. [**50 Points**] **Approximating Pairwise Interactions**

Suppose you are given n celestial objects in a two dimensional universe as in Task 2 of HW1. However, each of them is of circular shape, i.e., a disc, as shown in Figure 4. Let m_i and r_i be the mass and radius, respectively, of the i-th such disc, where $1 \leq i \leq n$. Let d_{ij} denote the distance between the i-th and the j-th object, where $1 \leq i, j \leq n$. We assume that the discs are nonoverlapping.

Suppose we want to compute the following interaction potential among the objects, where c_1 and

c_2 are nonnegative constants:

$$P = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{m_i m_j}{\sqrt{c_1 d_{ij}^2 + c_2 r_i r_j}}.$$

Clearly, P can be computed in $\Theta\left(n^2\right)$ time. However, in this task, we want to approximate P in asymptotically faster than quadratic time.

(*a*) [**15 Points**] Suppose $m_i = m$ for all $i \in [1, n]$, and $c_1 = 0$, but $c_2 > 0$. Give an algorithm for approximating P within a factor $1+\epsilon$ of the exact value in $\mathcal{O}\left(\log_{1+\epsilon}^2 n\right)$ time, where $\epsilon > 0$. You are allowed to spend up to $\Theta(n)$ time for preprocessing the input before you compute the approximate value of P. Give pseudocode and show your analyses of approximation ratio and running time.

(*b*) [**15 Points**] Suppose $c_1 = 0$ and $c_2 > 0$ as in part 3(a), but not all objects have the same mass. Give an $(1+\epsilon)$-approximation algorithm for this case. Give pseudocode and show your analyses of approximation ratio and running time.

(*c*) [**20 Points**] Repeat part 3(b) assuming c_1 is also positive (i.e., $c_1 > 0$).