CSE548, AMS542: Analysis of Algorithms, Fall 2015 **Date:** Oct 21

Homework #2

(Due: Nov 4)

Task 1. [**110 Points**] **BoT-PivoT-QuicksorT**

The *BoT-PivoT-QuicksorT* ('Better of Two' Pivot Quicksort) algorithm works as follows. Each recursive call chooses the first and the last numbers in the input subarray as potential pivots, and partitions around the one that gives a more balanced partition. We assume for simplicity that all numbers in the input array are distinct. Pseudocode of the algorithm is given in Figure 1.

```
Input: An array $A[1:n]$ of $n$ distinct numbers.
Output: Numbers of $A[1:n]$ rearranged in increasing order of value.
Steps:
    1. Pivot Selection: Let $x = A[1]$ and $y = A[n]$. Let $z$ be the one between $x$
       and $y$ that results in a more balanced partitioning of $A[1:n]$.
    2. Partition: Rearrange the numbers of $A[1:n]$ such that $A[k] = z$ for some
       $k \in [1, n]$, each number in $A[1:k-1]$ is smaller than $z$, and each in
       $A[k+1:n]$ is larger than $z$.
    3. Recursion: Recursively sort $A[1:k-1]$ and $A[k+1:n]$.
    4. Output: Output $A[1:n]$.
```

Figure 1: The BoT-PivoT-QuicksorT algorithm.

Given an input of size n, in this task we will analyze the average number of element comparisons (i.e., comparisons between two numbers of the input array) performed by this algorithm over all $n!$ possible permutations of the input numbers. We will assume that the partionting algorithm is *stable*, i.e., if two numbers p and q end up in the same partition and p appears before q in the input, then p must also appear before q in the resulting partition.

(a) [**10 Points**] Show how to implement steps 1 and 2 of Figure 1 to get a stable partitioning of $A[1:n]$ using only $2n - \text{rank}(\min(x, y)) - 2$ element comparisons, where $\text{rank}(u)$ gives you the number of entries in $A[1:n]$ with value not larger than u.

(b) [**50 Points**] Let t_n be the average number of element comparisons performed by BoT-PivoT-QuicksorT to sort $A[1:n]$, where $n \geq 1$ and the average is taken over all $n!$ possible permutations of the numbers in A. Let $m = \left\lfloor \frac{n}{2} \right\rfloor$. Show that

$$t_n = \begin{cases} 0 & \text{if } n < 2, \\ \frac{5n-7}{3} + \frac{2}{n(n-1)}\left[2\left(1-(-1)^n\right)mt_m + \sum_{k=1}^{m}(4k-3)\left(t_{k-1}+t_{n-k}\right)\right] & \text{otherwise.} \end{cases}$$

(*c*) [**50 Points**] Give an algorithm that can compute all t_i values for $1 \le i \le n$ in $\mathcal{O}(n)$ total time and $\mathcal{O}(n)$ total space, where $n \ge 1$ is an integer.

(*d*) [**extra credit**] Can you compute all t_i values, $1 \le i \le n$, in $\mathcal{O}(n)$ time and $\mathcal{O}(1)$ space?

(*e*) [**extra credit**] Given an integer $n \ge 1$, can you compute t_n in $\mathcal{O}(1)$ time and $\mathcal{O}(1)$ space?

Task 2. [90 Points] (α, β)-Quadtrees

A *quadtree* is a tree data structure that recursively and adaptively subdivides 2D space into quadrants. Each internal node in a quadtree has at most four (4) children. Quadtrees are used to store Cartesian space data. For $\beta \ge \alpha > 0$, we define an (α, β)-quadtree to be a quadtree in which each internal node contains more than α data points and no leaf contains more than β data points. Figure 2 shows a $(3, 3)$-quadtree.

In this tasks we are given a set of disks moving inside an $n \times n$ square Q_n in the Cartesian plane, where $n > 0$. Each disk has a center and a radius, and no two disks ever overlaps in space. We say that a disk is inside Q_n provided its center is on or inside Q_n, otherwise the disk does not belong to Q_n. No disk has radius less than γ, where γ is a positive constant.

This task asks you to maintain a set of moving disks (i.e., disk centers) in an (α, β)-quadtree. When a disk moves, the (α, β) properties of the quadtree may be violated, and in that case you must fix the tree so that the properties start to hold again. Figure 3 shows an example.

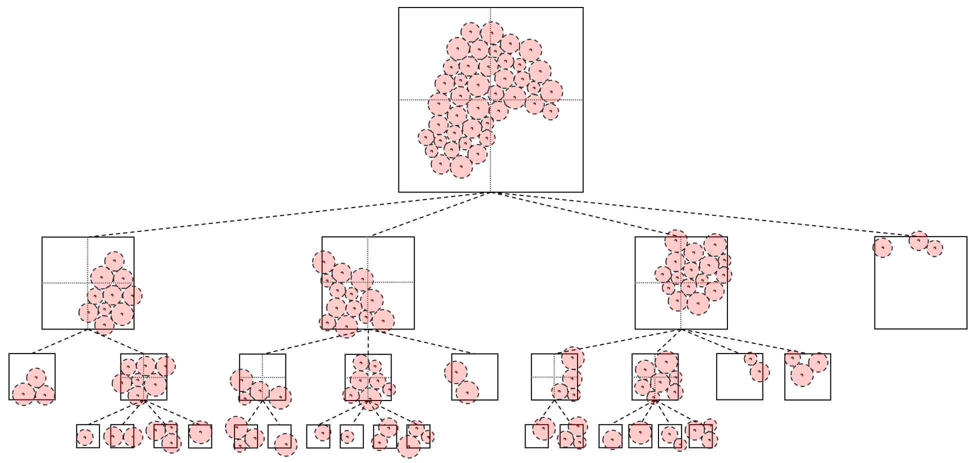

Figure 2: A $(3, 3)$-quadtree in which no leaf contains more than 3 points (i.e., disk centers) and each internal node contains more than 3 points.

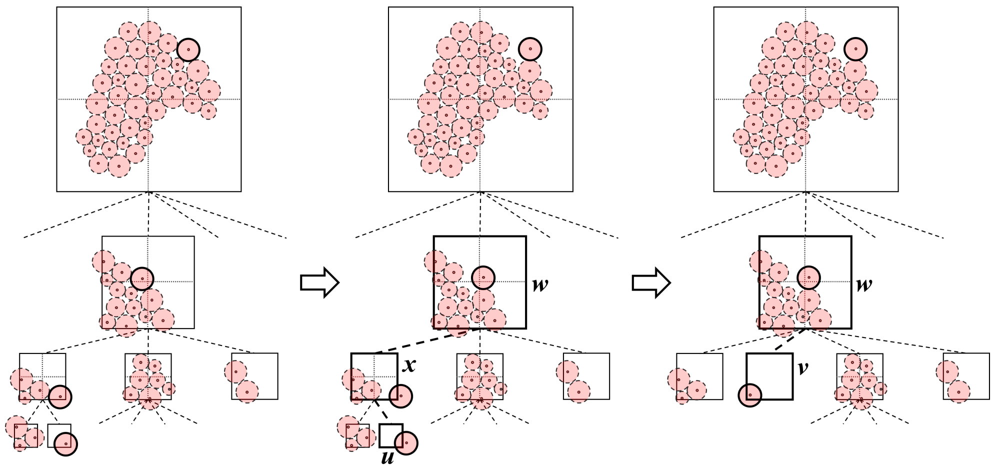

Figure 3: Updating the $(3,3)$-quadtree from Figure 2: **(a)** The disk with solid black outline changes position. **(b)** The disk moves to a new location, and so it moves out of nodes u and x, but remains inside w and all its ancestors. **(c)** Node u now becomes empty. So we remove u. Node x now includes only 3 disks. So we mark it as a leaf. The updated disk moves to the top-right quadrant of node w which was empty before, and so a new leaf v containing only the updated disk is created.

(a) [**5 Points**] Show that Q_n contains $\mathcal{O}\left(n^2\right)$ disks.

(b) [**15 Points**] Suppose the MOVE(d, p_{old}, p_{new}) operation moves (the center of) disk d from point p_{old} to p_{new} so that at least one of p_{old} and p_{new} belongs to Q_n. Now if this move destroys the properties of one or more nodes of the $(\alpha, 4\alpha)$-quadtree containing all disks of Q_n (and only those of Q_n), what is the worst-case cost of restoring those properties?

(c) [**5 Points**] Argue that every $(2\alpha, 2\alpha)$-quadtree is also an $(\alpha, 4\alpha)$-quadtree.

(d) [**15 Points**] Show that you can construct a $(2\alpha, 2\alpha)$-quadtree for the disks in Q_n in $\mathcal{O}\left(n^2 \log n\right)$ time.

(e) [**50 Points**] Show how you will implement MOVE(d, p_{old}, p_{new}) so that the properties of the $(\alpha, 4\alpha)$-quadtree can be restored in $\mathcal{O}(\log n)$ amortized time per MOVE. Use the accounting method to prove the bound. You may find results from parts (c) and (d) useful for this part.