Name _____

CS200 - Midterm Review Questions

1. All of the following questions are regarding the closest points in 2D algorithm.

   (a) What if we changed the Closest Points algorithm to, in the combine step, use a region within 2δ of the midline. Would the algorithm still be correct? How would our analysis change?

   (b) If we used a region within $\delta/2$ of the midline, would our algorithm be correct?

   (c) What if, in our analysis of points in the region within δ of the midline, we created imaginary squares that are $\delta \times \delta$ large. How would our analysis change?

   (d) What if we imagined squares that are $\delta/3 \times \delta/3$ large?

   (e) Why is it important to presort the arrays?

   (f) Why do we need to maintain separate arrays sorted by X and Y coordinates?

Solution

   (a) The algorithm would still be correct, because any two closest points with one point in left half and one point in right half would be in this region (otherwise, the two points would have distance greater than δ, a contradiction). However, when we divide this region into $\delta/2 \times \delta/2$ boxes, there would now be 8 boxes in a row, so there would be 16 boxes in the two relevant rows. Therefore, in our loop, we would need to check the next 15 points.

   (b) The algorithm would not be correct, because we would miss points that could be split amongst the left and right half, but have distance less than δ.

   (c) Now there could be at most 4 points in the box (put 4 points at each corner, and 5th point would have distance to one of the existing points of less than $\delta/\sqrt{2}$, a contradiction). But we would only need to consider one row of boxes, because other rows would contain points more that distance δ away if our initial point is on the top edge of the box. Thus there are two boxes where relevant points are, each with at most 4 points, so we would need to check the next 7 points (same as before!)

   (d) If we did the analysis with $\delta/3 \times \delta/3$ boxes, we would still only be able to fit one point in each box (why? what is the largest distance between two points in the box?) but now there are 18 relevant boxes (3 rows, with 6 boxes in each row - draw a picture!) so with this analysis, we should check the next 17 points.

   (e) Presorting the arrays allows us to divide into left/right halfs, and create the y-sorted array of points around the midline efficiently.

   (f) Ditto

2. Prove the following algorithm is correct.

Algorithm 1: `Maximum`(A, s, f, x)

Input : Array A of unique integers. Start index s and final index f.
Output: Maximum value in array.

```
1 if f − s == 0 then
2 |  return A[s];
3 end
4 g = ⌊(s + f)/2⌋;
5 m_1 =Maximum(A, s, g);
6 m_2 =Maximum(A, g + 1, f);
7 return max{m_1, m_2};
```

Solution We will use a proof by strong induction. Let $P(n)$ be the predicate that `Maximum` is correct for an array of size $f - s$. We will prove $P(n)$ is true for all $n \geq 0$, $n \in \mathbb{Z}$.

Base case: when $n = 0$, we have $f = s$, so there is only one element of interest, so it must be the maximum. Indeed, we see that the algorithm returns $A[s]$ when $f = s$.

Inductive step: assume for strong induction that $P(r)$ is true for all $k \geq r \geq 0$. We will prove $P(k + 1)$ is true. Since $k + 1 \geq 1$, we go to the inductive step. Since $g = \lfloor (s + f)/2 \rfloor$, we have $0 \leq g - s < k + 1$, and $0 \leq f - (g + 1) < k + 1$. Thus, by our inductive assumption, the recursive calls return the maximum elements in the left and right halves of the array respectively. Since the maximum value of the whole array is the largest value in either half, we should return the maximum of those two values, which the algorithm does. (Could be more mathematical and use the definition of maximum, but I think that is not necessary here. It is pretty obvious.)

Therefore, using strong induction, $P(n)$ is true for all $n \geq 0$, $n \in \mathbb{Z}$.

3. Probability Questions

   (a) Review Quiz

   (b) If you have a coin that has 1/4 probability of heads and 3/4 probability of tails, what is the sample space if you flip it n times? What is the expected number of heads? (Use indicator random variables).

   (c) Problem 2b from the homework, but what if there are two elements with value x in the array?

   (d) Explain why the probability of comparing z_i and z_j in Randomized QuickSort is $2/(|j - i| + 1)$

Solution

(a) The sample space is equivalent to all possible strings of length n made up of the letters H and T, corresponding to all possible sequences of heads and tails that might result. Let X_i be the indicator random variable that takes value 1 if the ith flip is heads, and zero otherwise. Then if X is the total number of heads $X = \sum_{i=1}^{n} X_i$. Using linearity of expectation, we have

$$\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i] = \sum_{i=1}^{n} Pr(i\text{th toss is heads}) = \sum_{i=1}^{n} \frac{1}{4} = \frac{n}{4}. \tag{1}$$

(b) Let X be the random variable that is the number of rounds. Let X_k be the indicator random variable that takes value 1 if round k occurs. Then

$$\mathbb{E}[X] = \sum_{k=1}^{\infty} \mathbb{E}[X_k] = \sum_{k=1}^{\infty} Pr(k\text{th round occurs}) \tag{2}$$

Now the probability that the kth round occurs is the probability that we have not found the item we are searching for *before* the kth round, which is

$$\left(\frac{n-2}{n}\right)^{k-1}. \tag{3}$$

Thus

$$\mathbb{E}[X] = \sum_{k=1}^{\infty} \left(\frac{n-2}{n}\right)^{k-1} = \sum_{k=0}^{\infty} \left(\frac{n-2}{n}\right)^{k} = \frac{1}{1-\frac{n-2}{n}} = \frac{n}{2}. \tag{4}$$

(c) Items only get compared if one is the pivot, and the other is in the section of the array in that recursive call. As long as the pivot is chosen to be z_k where $k < i$ or $k > j$, then z_i and z_j move together in a recursive call. So something interesting only happens when the pivot is either chosen to be z_i or z_j, or is chosen to be z_k with $i < k < j$. In the later case, z_i and z_j are split into different recursive calls and so are never compared. In the former, they do get compared. Because there is equal probability of any of the points from z_i to z_j being chosen, there is a $2/(|j-i|+1)$ probability that z_i or z_j is chosen from among this set, in which case, they are compared.