CPSC 320 Sample Midterm 2
March, 2016

1. [20 points] Short Answers

(a) [4 points] Let T be the MST (Minimum Spanning Tree) of an undirected graph $G = (V, E)$ with cost function $c : E \to \mathbb{R}^+$ produced by the Kruskal's algorithm. Let $c' : E \to \mathbb{R}^+$ is defined as follows: $c'(e) = [c(e)]^2$ for every $e \in E$. **Decide whether the following statement is true or false (justify your answer shortly).**

Statement: *Then T is an MST of G with cost function c'.*

Solution: *True.* Since the Kruskal's algorithm doesn't care about the actual values of the costs of edges, just about their relative order, which does not change by taking the second power, it would produce the same tree when input changes to G, c', hence, T is still an MST for the graph with the new edge costs.

(b) [6 points] Modify the MERGE algorithm so that in addition to merging the sorted subarrays $A[first \dots mid]$ and $A[mid{+}1 \dots last]$, it also counts and returns the number of inversions (i, j) such that $first \le i \le mid$ and $mid + 1 \le j \le last$. You only need to insert a couple of lines.

```
procedure MERGE($A, first, mid, last$)
    $B \leftarrow$ empty list
    $i \leftarrow first$
    $j \leftarrow mid + 1$
    $c \leftarrow 0$
    while $i \le mid$ and $j \le last$ do
        if $A[i] \le A[j]$ then
            append $A[i]$ to $B$
            $i \leftarrow i + 1$
        else
            append $A[j]$ to $B$
            $c \leftarrow c + (mid - i + 1)$
            $j \leftarrow j + 1$
        end if
    end while
    if $i \le mid$ then
        append $A[i \dots mid]$ to $B$
    else
        append $A[j \dots last]$ to $B$
    end if
    $A[first \dots last] \leftarrow B$
    return $c$
end procedure
```

(c) [6 points] In the *merge* part of the algorithm to find the **closest pair of points** among a set of points in the 2D plane, we only considered the subset S_δ of the points whose distance from the line $\ell : x = x_0$ used to separate the two subproblems was smaller than δ.
What does δ represent and why is it sufficient to only consider these points?

Solution:

- The value δ is the smallest distance between two points that are both on the same side of the line $x = x_0$.

- It's sufficient to consider the points within a distance δ of the line $x = x_0$ because we are looking for a pair of points with the properties that (1) the two points are on opposite sides of the line $x = x_0$ and (2) their distance is smaller than δ. If the distance from a point p to the line $x = x_0$ is larger than δ, then p can not belong to the pair we are looking for.

(d) [4 points] Why is RANDOMIZEDQUICKSORT better than the regular QUICKSORT?

Solution: Because it performs well on average for every possible input, unlike QUICKSORT which performs well (all the time) for some inputs, and badly (all the time) for other inputs such as already sorted arrays.

2. [8 points] Consider an undirected graph $G = (V, E)$ and cost function $c : V \to \mathbb{R}^+$. Assume that all edge costs are distinct. You can assume that the following property holds (without proving it):

   Cut property. Let S be any subset of nodes that is neither empty nor equal to all of V, and let edge $e = (u, v)$ be the minimum cost edge with one end in S and the other in $V \setminus S$. Then every MST of G contains e.

   Prove that there is only one MST for G.

Solution: Consider an MST T of G. It is enough to show that any other MST has to contain each edge of T. Let e be any edge of T. Then $T - e$ has exactly two components, let one of them be S (other one is then $V \setminus S$). Let f be the cheapest edge crossing from S to $V \setminus S$. By the Cut Property, every minimum spanning tree of G has to contain f. Hence, T contains f. If $f \neq e$, then T contains a cycle, which is not possible. Hence, $f = e$ and every MST contains e. We are Done.

Alternative solution. Assume there are two distinct MST trees T and T'. There must be an edge $e \in E(T)$ such that $e \notin E(T')$. Then $T - e$ has exactly two components, let one be S, the other one is then $V \setminus S$. Let e' be the minimum cost edge between S and $V \setminus S$. By the Cut Property, every MST contains e'. If $e' \neq e$, then T cannot contain e (it would create a cycle in T), a contradiction. If $e' = e$, then T' does not contain e', a contradiction.

3. [16 points] Recurrence relations

   (a) [8 points] Write a recurrence relation that describes the running time of the following algorithm MEDIAN as a function of $n = j - i + 1$. Algorithm SORT(A, i, j) runs in time $\Theta(j - i + 1)$.

```
1:  function MEDIAN($A, i, j$)
2:      if $j \le i + 4$ then
3:          SORT($A, i, j$)
4:          return $A[\lceil (i + j)/2 \rceil]$
5:      else
6:          $B \leftarrow []$
7:          for $k \leftarrow 0$ to $\lfloor (j - i + 1)/5 \rfloor - 1$ do
8:              append MEDIAN($A, i + 5 * k, i + 5 * k + 4$) to $B$
9:          end for
10:         return MEDIAN($B, 1, \lfloor (j - i + 1)/5 \rfloor$)
11:     end if
12: end function
```

   Remark: Do not try to figure out what this algorithm does and whether it actually finds a median.

Solution: Note that sorting in line 3 takes constant time (since $j - i + 1 \leq 5$), so $T(n) = \Theta(1)$ for $n \leq 5$. Calls to MEDIAN in line 8, takes also constant time, since $n = 5$ for each of them, and so does adding an element to B. The **for** loop iterated $\lfloor n/5 \rfloor$ times. The call in line 10, works on array of size $\lfloor n/5 \rfloor$. Hence, we have the following recurrence: $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 5 \\ T(\lfloor n/5 \rfloor) + \Theta(n) & \text{otherwise} \end{cases}$

(b) [8 points] Consider the following recurrence: $T(n) = \begin{cases} 3T(n/3) + dn & \text{if } n \geq 3 \\ b & \text{if } n \leq 2 \end{cases}$

You have guessed that $T(n) \in O(n \log n)$. **Verify** that this guess is correct using the substitution method (induction).

Solution: *Induction hypothesis:* For every $m < n$, $T(m) \leq cm \log_3 m$ (the base of the log does not really matter, so for convenience, let's choose base 3).
Induction step:

$$\begin{aligned} T(n) &= 3T(n/3) + dn \\ &\leq 3c(n/3)\log_3(n/3) + dn \\ &= cn(\log_3 n - 1) + dn \\ &= cn\log_3 n + n(d - c) \end{aligned}$$

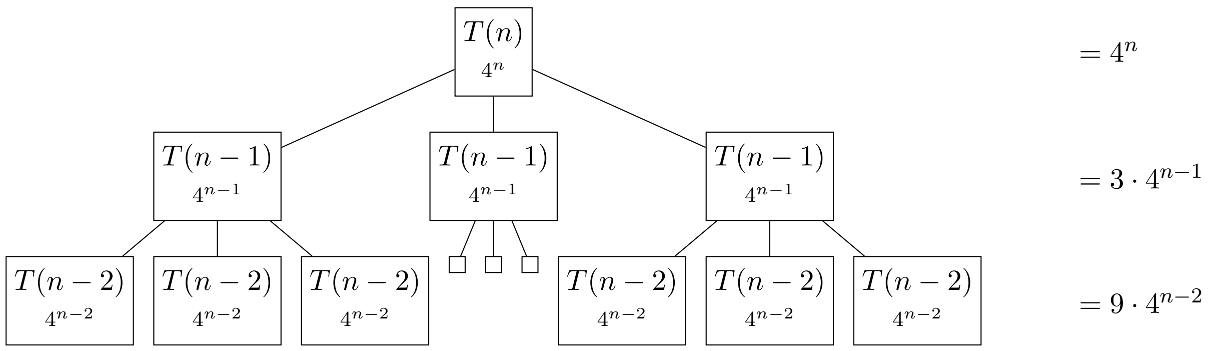
and so $T(n) \leq cn \log_3 n$ as long as $c \geq d$.

4. [16 points] *Recursion Tree method.* Prove tight upper and lower bounds on the function $T(n)$ defined by

$$T(n) = \begin{cases} 3T(n-1) + 4^n & \text{if } n \geq 2 \\ 1 & \text{if } n = 1 \end{cases}$$

Hint. Recall the formula $\sum_{i=0}^{\infty} c^i = \frac{1}{1-c}$ for any $0 < c < 1$.

Solution: Here are the first three levels of the recursion tree:



The root node does 4^n work. The nodes on the second level do $3 \cdot 4^{n-1} = (3/4) \cdot 4^n$ work. The nodes on the third level do $9 \cdot 4^{n-2} = (3/4)^2 \cdot 4^n$ work. The depth of the tree is n. The total amount of work done is

$$\sum_{i=0}^{n-1} (3/4)^i 4^n = 4^n \sum_{i=0}^{n-1} (3/4)^i < 4^n \sum_{i=0}^{\infty} (3/4)^i = 4^n \cdot \frac{1}{1 - 3/4} = 4 \cdot 4^n$$

and so $T(n) \in O(4^n)$. Also, since the root of the tree itself does work 4^n, we have $T(n) \in \Omega(4^n)$. Hence, $T(n) \in \Theta(4^n)$.

5. [14 points] Consider the following implementation of the randomized QUICKSORT that waits for a "perfect" pivot. A pivot is **perfect** if the size of the left partition is $\lfloor (n-1)/2 \rfloor$ and the size of the right partition is $\lceil (n-1)/2 \rceil$.

```
procedure PERFECTPIVOTQUICKSORT($A, first, last$)
    if $first < last$ then
        $n \leftarrow last - first + 1$
        repeat
            $mid \leftarrow$ RANDOMIZEDPARTITION($A, first, last$)
                                                ▷ chooses pivot uniformly at random
        until $mid - first = \lfloor (n-1)/2 \rfloor$
        PERFECTPIVOTQUICKSORT($A, first, mid - 1$)
        PERFECTPIVOTQUICKSORT($A, mid + 1, last$)
    end if
end procedure
```

(a) [2 points] What's the probability that a **perfect** pivot is selected in one run of RANDOMIZEDPARTITION? Justify shortly your answer.

Solution: Out of n possible pivots only one of them is perfect, hence, the probability that the perfect pivot is selected is $1/n$.

(b) [6 points] Write down the recurrence for the expected running time $S(n)$ of the PERFECTPIVOTQUICKSORT algorithm. Assume that each call to RANDOMIZEDPARTITION takes time cn. Let $g(n)$ be the number iteration of the **repeat**-**until** loop. If the probability of the perfect pivot is p, then the expected value of $g(n)$, $E[g(n)]$ is $1/p$.

Solution:

$$S(n) = \begin{cases} cnE[g(n)] + S(\lfloor (n-1)/2 \rfloor) + S(\lceil (n-1)/2 \rceil) & \text{if } n > 1 \\ \Theta(1) & \text{if } n = 1 \end{cases}$$
$$= \begin{cases} cn^2 + S(\lfloor (n-1)/2 \rfloor) + S(\lceil (n-1)/2 \rceil) & \text{if } n > 1 \\ \Theta(1) & \text{if } n = 1 \end{cases}$$

(c) [6 points] Solve the recurrence from part (c).

Hint. If you use the Master Theorem, you can ignore additive constants in the recurrence (in addition to ignoring floor and ceiling parts). For example, you can replace $S(\lfloor (n-2)/3 \rfloor)$ with $S(n/3)$.

Solution: After simplifying the recurrence, we have $S(n) = 2S(n/2) + cn^2$ for sufficiently large n's. We have $f(n) = cn^2$ and $s(n) = n^{\log_b a} = n^{\log_2 2} = n$, so $f(n) \in \Omega(s(n).n)$. Hence, this could be the case 3. Regularity condition: $2f(n/2) = 2c(n/2)^2 = cn^2/2 < \delta cn^2 = \delta f(n)$ if $\delta = 0.51$. Hence, $S(n) \in O(f(n)) = O(n^2)$.
Comment: So, waiting for the perfect pivot does not lead to an efficient algorithm.

Points total: 74

End of exam.