Cs445 — Homework #1. Due Tuesday 9/22/2015 midnight

Instructions.

1. Solution could **not** be submitted by students in pairs.
2. You could submit a pdf of the homework, either printed or hand-written and scanned, as long as it is **easily** readable.
3. If your solution is not clearly written, it might not be graded.
4. Prove the correctness of your answer. A correct answer without a proof might not be accepted.
5. If you have discussed the solution with other students, mentioned their names clearly on the homework. These discussions are not forbidden and are actually **encouraged**. However, you must write your whole solution yourself.
6. All questions have same weight.
7. You could refer to a data structure studied in class, and just mentioned that
8. If your answer uses one of the data structures or algorithms that were studied in class, you could refer to it without having to repeat details studied in class. If you answer requires only modifications of one of the algorithm, it is enough to mention the required modifications, and what's the effect (if any) on the running time and on other operations that the algorithm performs.
9. In general, a complete answer should contain the following parts:
   (a) High level description of the data structures (if needed). *E.g. We use a binary balanced search tree. Each node contains, a key and pointers to its children. We augment the tree so each node also contains a field...*
   (b) High level description of the main ideas of the algorithms
   (c) Pseudo code (might not be necessary if the high level description is clear).
   (d) Proof of correctness (why your algorithm provides what is required).
   (e) A claim about the running time, and a proof showing this claim.
10. When talking about tress, do not confuse the *children* of a node with the *decedents* of the node.

1. What is the running time of the following function (specified as a function of the input value n) ?

```
    Read(n)
    i=0 ; s=0 ;
    while( $s \leq n$) {
        i++ ;
        s= s+i ;
        print("*");
    }
```

   Express your answer as $\Theta(g(n))$, for a function $g(n)$ that you have to find. Prove your answer.

2. Write a recursion formula for the running time $T(n)$ of the function *NoNeed*, whose code is below. What is the running time of *NoNeed*, as a function of n ?

```
NoNeed(n) {
    if ($n \leq 1$) return;
    $i = 1$ ;
    for($i = 1$; $i < n$; $i++$) print("*");
    NoNeed( $0.8n$ ) ;
}
```

   You could use the Master theorem if you wish, but others and possibly simpler ways also exist.

3. Your friend has dropped you at some point on Speedway Blvd, Tucson. You need to walk to the nearest bus station, but you are not sure if the nearest bus station is East or West of your location, so you are not sure which direction to go.

   Let d be the distance to the nearest bus station. Suggest an algorithm that guarantees that the total distance you would walk is $\Theta(d)$. Of course, you cannot use other people, cellphones, maps etc.

   If your algorithm is iterative, specify exactly what happens in the ith iteration.

4. (a) You are given a sorted array $A[1..n]$ of keys, and another key x. The *successor* of x, denoted $succ(x)$ is the smallest element in A which is strictly larger than x. Consider a index $1 \leq t \leq n$ such that $A[t] = succ(x)$. Show how to find t in time $O(1 + \log t)$. (Note that t is not given, and can be much smaller than n).

   (b) Same, but now find t in time $O(1 + \log(\min\{t, n - t\})$.

5. When we discussed in class the stable marriage algorithm, we did not discuss how exactly a women receiving proposals from two men b_1, b_2 decides which one to reject. For the algorithm to run in $O(n^2)$, she should be able to do so in $O(1)$ time.

   Explain exactly how this part of the algorithm works. You might need some preprocessing of the data before running the algorithm itself.

6. Assume n is given to you (you do not know it ahead). Show an example where the TMA algorithm would require $\Omega(n^2)$ rejections operations. Prove your answer.

7. Assume n is given to you (you do not know it ahead). Show an example where the TMA algorithm would require no rejections operations. Prove your answer.

8. Assume n is given (not known in advance). Given an example of the preference lists of n men and n women such that $M = M'$. Here M is the result of the TMA, and M' is the results of the TMA but when the rules of the women and men are switched. Prove your answer.

9. Show preferences lists of 2 men and 2 women, so that more than one stable matching exists.

10. You are given an array $A[1..n]$ containing n keys in a sorted order. Suggest an algorithm that in time $O(n)$ construct a Skiplist, that has the following properties:

    (a) For any value of x, the operation `find`(x) would visit at most 3 elements in each level of the Skiplist, and

    (b) the total number of elements is $\leq 1.5n$.

    As usual each element has a down pointer and a right pointer.

    Note that the running time is in the worst-case, not the expected case. So do not assume any randomized procedures.

11. Suggest a data structure for a set of keys $S = \{x_1, x_2 \dots x_n\}$ where $x_1 < x_2 < \cdots < x_n$. The data structure support insert/delete/find, and also the operation Med(x, y), which is define as the median of the set of keys currently in S and are between x and y. Assume for simplicity both x and y are in S. In other words, if we define

    $$S' = \{z | x \leq z \leq y\},$$

    then Med(x, y) is a key of S', such that $\lceil\, |S'|/2\rceil - 1$ of the keys in S' are smaller than Med(x, y). Here $|S'|$ is the number of keys in S'. (If $|S'|$ is even, then there are two possible answers).

    Example: $S = \{1, 2, 11, 12, 21, 22, 24, 26, 28\}$. Then

    $$\text{Med}(11, 22) = 21, \quad \text{Med}(24, 28) = 26, \quad \text{Med}(1, 28) = 21$$

    The worst case running time for each operations is $O(\log n)$.

12. Suggest a data structure that supports the following operations on the grades that a (single) student at the university receives. Assume that the total number n of exams that the student might take is large. The data structure should support the following operations:

    Bonus 5 points: Each operation is performed in time $O(\log n)$.

    (a) `insert`$(exam\_date, exam\_grade)$. This operation informs the data structure that the student received at date $exam\_date$ the grade $exam\_grade$.

    (b) `average`$(date_1,\ date_2)$. As a response to this operation the data structure should answer what is the average of all grades that are received between the dates $date_1$ and $date_2$. Assume again that all dates are different, and for simplicity, assume that no exam took place in $date_1$ nor in $date_2$.