Instructions.

1. You could submit a pdf of the homework, either printed or hand-written and scanned, as long as it is easily readable.

2. If your solution is not clearly written, it might not be graded.

3. Prove the correctness of your answer. A correct answer without a proof might not be accepted.

4. If you have discussed the solution with other students, mention their names clearly on the homework. These discussions are not forbidden and are actually encouraged. However, you must write your whole solution yourself.

5. All questions have same weight.

6. You could refer to a data structure studied in class, and just mention briefly their guaranties. For example “It is known that a Red-Black tree could support the insert/delete/find operations on a set of \( n \) elements in time \( O(\log n) \).”

7. 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 your 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.

8. 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 algorithms

   (c) Proof of correctness (why your algorithm provides what is required).

   (d) A claim about the running time, and a proof showing this claim.

9. When discussing trees, do not confuse the children of a node with the descendants of the node.
1. Your friend (same one from hw1) has this time dropped you at a point $p$ in the desert. You wish to walk to the nearest road, from which you could hitchhike back home. However, you have no idea in which direction you should head.

Let $q$ be the nearest road point, and let $d$ be the distance from $p$ to $q$. Explain how you could walk a distance of $c \cdot d$ and reach a road point (not necessarily $q$). Here $c$ is a constant (does not depend on $d$) that you have to compute.

For simplicity, assume that every road that crosses the desert is an infinite straight line. Each road is either horizontal (oriented East-West) or vertical (South-North).

Comments:

(a) Assume that the desert is flat, so every path is possible. Also assume that you are equipped with an accurate GPS, showing you location as well as the direction to North. You have no access to maps.
(b) For simplicity, assume that you need to be physically on the road to see it. So ignore range-of-visibility issues.
(c) Obviously, roads could cross each other, and if they do, they are orthogonal to each other.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{roads_network.png}
\caption{The roads network. Note that roads are not equally-spaced}
\end{figure}
2. Due to reasons not to be disclosed here, but related to the very same friend from the previous question, you have found yourself at a point \( p \) in a lake. Crocodiles are habituating the lake, strongly suggesting that you’d swim to the nearest point on a the lake’s shore before becoming a crocodile treat. As before, you have a GPS, but no map of the lake. Suggest your strategy, so the distance you have to swim is at most \( c \cdot d \). Here \( d \) is the distance from your starting position to the nearest point of the lake, and \( c \) is a constant (not depending on \( d \)).

Note that the lake is not convex. Show the best value of \( c \). You do not have a map, and you don’t know the crocodiles’ locations.

You could not use techniques proposed at www.youtube.com/watch?v=kmH0PP_zAKo. (Thanks to Jeremy M. for the insight).

![Figure 2: The lake. You have landed at \( p \), and wish to swim to the shore. The nearest point of the shore is \( q \).](credit: ClipPanda.com)

3. You are inserting \( n \) different keys into a SkipList \( L \). \( L \) was an empty Skiplist before the insertions took place. What is the probability that after all insertions took place, all the keys appear only on level 1, but not in level 2?

4. In this question, we are discussing a SkipList \( L \) created by inserting a set \( S \) of \( n \) keys into an (initially empty) Skiplist. As you know, in a skip list, a key that appears in level \( i \) appears also in level \( i + 1 \) with probability \( p = 0.5 \) (that is, a key has a 50% chance to be promoted to the next level).

Now assume that we re-create a Skiplist by inserting the same keys, in the same order, but this time the probability \( p \) is 0.01. Will the time to perform \( \text{find}(x) \) operation increase or decrease, compared to time for the same operation in the original SkipList?

Repeat the question, but now assume \( p = 0.9 \).

5. Prove that in the ‘perfect’ SkipList discussed in class, the worst-case time for a query is \( O(\log n) \) where \( n \) is the number of keys. You can refer to the slides at www.cs.arizona.edu/classes/cs445/spring16/SL.pdf.
6. Explain which field(s) would you add to each element in a SkipList, (that is, how would you augment the SkipList) so you could perform all operations of the dynamic order statistic on a set $S$ of $n$ keys stored in the SkipList. There operations are

(a) Insert $(x, S)$,
(b) Delete$(x, S)$.
(c) Find $(x, S)$,
(d) Succ$(x, S)$
(e) OS-SELECT$(i, S)$ and
(f) OS-RANK$(x, S)$

See the slides for exact definition of each operation.

Specify exactly what each field contains, and how you update them during insertions and deletions of keys. The expected time for every operation is $O(\log n)$. Make sure to prove carefully that this is the running time. You do not need to re-prove any claim that was already proven in class – you could just say “in class, it was proved that ...”

7. Suggest a data structure that supports the following operations on the grades that a 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: (each in $O(\log n)$ time, where $n$ is the number of exams taken).

(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 $date_2$.

Hint: Start by solving the simpler task of being able to report the number of exams that took place between $date_1$ to $date_2$.

8. Suggest a data structure that helps Tucson keep track of its population. The operations that are needed from the data structure are

(a) Joined$(x, date)$. This operation informs the data structure that a person whose Social Security Number is $x$ has moved to Tucson on date $date$. Note that this date might be in the past.
(b) Left$(x, date)$. This operation informs the data structure that a person whose Social Security Number is $x$ has left Tucson on date $date$. Note that this date might be in the past.
(c) Population$(date)$. The data structure needs to report how many people were living in Tucson on date $date$. That is, what was Tucson population at this date.

Before any operation took place, assume that Tucson is empty. Note that these operations could be given in any order (not necessarily chronologically). Each operation should be performed in $O(\log n)$ time, where $n$ is the total number of people ever lived in Tucson. Once a person has left Tucson, he/she could not move back.

9. You are given two identical glass cups, and an access to skyscraper building. You have access to each window in each floor of the skyscraper. Your goal is to experimentally find what is the the highest floor $K_{max}$ that such a cup could survive being dropped from this floor to the ground. That is, a drop from the $(K_{max} + 1)$th floor would break this cup.

Assume that only the height influences whether the cup would survive the fall. So for example if it survived a fall from the 7'th floor, it would also survive a drop from any lower floor.
Note that if the cup survived, you have to go down to the ground level to fetch it. Suggest an algorithm for finding $K_{\text{max}}$, while the number of times the surviving cup needs to be fetched is only $O(\sqrt{n})$, where $n$ is the number of floors in the skyscraper.

...and no, a broken cup could not be fixed, you could not buy new cups, etc.

**Hint:** There is some similarity between this question and some questions from hw1. However, using unbounded exponential search (e.g. check floors 1, 2, 4, 8, 16...) is an example of an immediate, brilliant and incorrect solution.

10. Same as the previous question, but now you are given 3 cups. Which strategy will you propose, if your goal again is to minimize the number of times the surviving cup needs to be fetched. What is this number?