Cs545 — Homework #1
Augmeting data structures, hash functions and some amortize analysis
Due 9/25/06

September 20, 2006

Instructions. All assignments are to be completed on separate paper. Use only one side of the paper. Assignments will be due at the beginning of class. To receive full credit, you must show all of your work.

1. Suggest a data-structures for a set of grades, such that you can support the following operations for the grades a student is obtaining during hers/his studies.

   `Insert`$(grade, date)$ — update the data structure to reflect the fact that the student got the grade *grade* at date *date*.

   `Avg`$(date_1, date_2)$ — report the average grade that the student received in all dates between dates $date_1$ and $date_2$ (strictly larger than $date_1$ and strictly smaller than $date_2$).

   Each operation should take $O(\log n)$ where n is the number of grades given. The space (memory) is $O(n)$.

2. Repeat the previous question, with the same query time, but this time you can assume that in each `Insert` operation, the date *date* is larger than the dates of all the grades inserted until *date*. Use this fact to to bound the query time, so no more than $2\log_2 n$ comparisons are needed to answer a query, and the data structure does not use any pointers. Can you bound the memory needed ?

3. Consider the interval $[0, 1]$ of all points x such that $0 \leq x \leq 1$. Assume that each point of the interval can be colored either white or black, and initially all the points of the intervals are colored white. Suggest a data structure that supports the following operations

`Reverse`(x, y) — reverse the color of each point between x and y. So a point that was white is becoming black and vice versa. The color of the other points are not effected.

`Report`(x) — What is the color of the point x.

Each operation should take $O(\log n)$ where n is the number of reverse operations. The space required is $O(n)$. Try to minimize the actual space required.

4. In class we studied how to construct a perfect hash table for an input set S of n keys.

   Does the algorithm **always** finds a perfect hash table ? Does the running time of the algorithm depends on the input ?

5. **1.** Is the family of all functions, from $U :\rightarrow \{1..n\}$ is a universal family ?

   2. Is the family of all permutation from $\{1..n\} :\rightarrow \{1..n\}$ is a universal family?

6. Consider the orthogonal grid in the plane where the size of each grid cell is $R \times R$, where R is a given fixed real value. Suggest a data structure that supports the following operations on a set $S = \{(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)\}$ of points in the plane.

   (a) `Insert`(x, y) — Insert a new point $p = (x, y)$. This operation should take expected time $O(1)$.

   (b) `Report`(x, y) — Report all the points of S that lie in the same grid cell as (x, y). This operation should take $O(k + 1)$, where k is the number of reported points.

   Write your solutions **in details**.

7. Read and understand Theorem 11.8 from the textbook

8. Let $S = \{\ell_1 \ldots \ell_n\}$ be a given set of lines, none is vertical. The *upper envelop* of S, denoted $\mathcal{U}(S)$ is define as all the points of S that lie on at least one line of S and not below any of the other lines of S. See Figure 1. A *vertex* is defined as the intersection point of two lines. The *slope* of a line is defined as the angle between the line and the x-axis. Assume that S is given to you such in an increasing order of slopes, so ℓ_1 has the smallest slope and ℓ_n as the largest slope.

   What is the complexity of the upper envelope ?

   Suggest an algorithm that computes the vertices of the boundary of the upper envelope in time $O(n)$.

   Hint — Define $S_i = \{s_1 \ldots s_i\}$. Assume by induction that you have already computed the vertices of $\mathcal{U}(S_i)$. How can you efficiently compute $\mathcal{U}(s_{i+1})$?

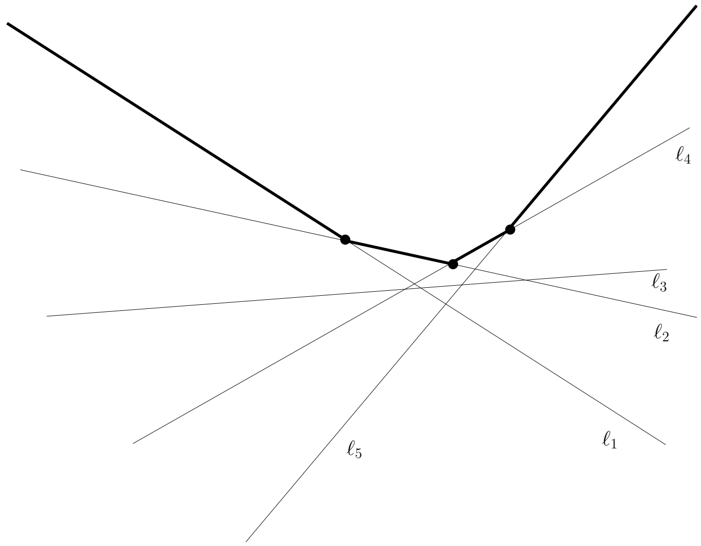

Figure 1: A set $S = \{\ell_1 \ldots \ell_5\}$. Underline is their upper envelop $\mathcal{U}(S)$. The vertices on the upper envelop are marked by tiny disks .

9. **Bonus** In class we studied a data structure called *interval tree* that accepts as input a set $S = \{(x_1, y_1), (x_2, y_2) \ldots, (x_n, y_n)\}$ of intervals, where each interval (x_i, y_i) denote all the points between the points x_i, y_i on the line. After the interval tree is constructed, it enables reporting for each query point q, all the intervals of S containing q. The reporting in time $O(\log n + k)$ where k is the number of interval reported.

   (a) Which properly of the input guarantees that the tree has height 0 ? Prove.

   (b) Suggest an $O(n \log n)$ time algorithm for constructing the tree.