Istanbul Technical University **November 3, 2006**
Electrical and Electronics Faculty
Dept. of Computer Engineering Department
Advanced Data Structures - BLG341E
Midterm Examination ANSWERS
Asst.Prof.Dr.D Turgay ALTILAR
Asst.Prof.Dr. Zehra ÇATALTEPE

Duration: 120 minutes. Books and notes are closed. Good Luck!

Questions

Q1) [30 points] Solve the given recurrences and determine good asymptotic bounds. You may use any of the methods we covered in the class.

$$T(n) = 4T(n/4) + n$$
$$T(n) = T(n-1) + \frac{1}{n}$$

Answer 1) Begin----------

$$\begin{aligned} T(n) &= 4T(n/4) + n \\ 4T(n/4) &= 4^2 T(n/16) + 4n/4 \\ 4^2 T(n/16) &= 4^3 T(n/64) + 4^2 n/16 \\ .... &= ... \\ 4^{\lg_4 n-1} T(4) &= 4^{\lg_4 n} T(4/4) + 4^{\lg_4 n-1} n/(4^{\lg_4 n-1}) \end{aligned}$$

There are $\lg_4 n$ rows above. Summing up all rows:

$$\begin{aligned} T(n) &= 4^{\lg_4 n} T(1) + n \lg_4 n \\ &= cn + n \lg_4 n \\ &= \Theta(n \lg_4 n) \end{aligned}$$

Using master theorem 2nd case would also give the same bound.

$$\begin{aligned} T(n) &= T(n-1) + \frac{1}{n} \\ T(n-1) &= T(n-2) + \frac{1}{n-1} \\ ... &= ... \\ T(2) &= T(1) + \frac{1}{2} \end{aligned}$$

There are n-1 rows above. Summing up all rows:

$$
\begin{aligned}
T(n) &= T(1) + \sum_{i=1}^{n} \frac{1}{n} - 1 \\
&= T(1) - 1 + H_n \\
&= O(\ln n)
\end{aligned}
$$

Answer 1) End----------

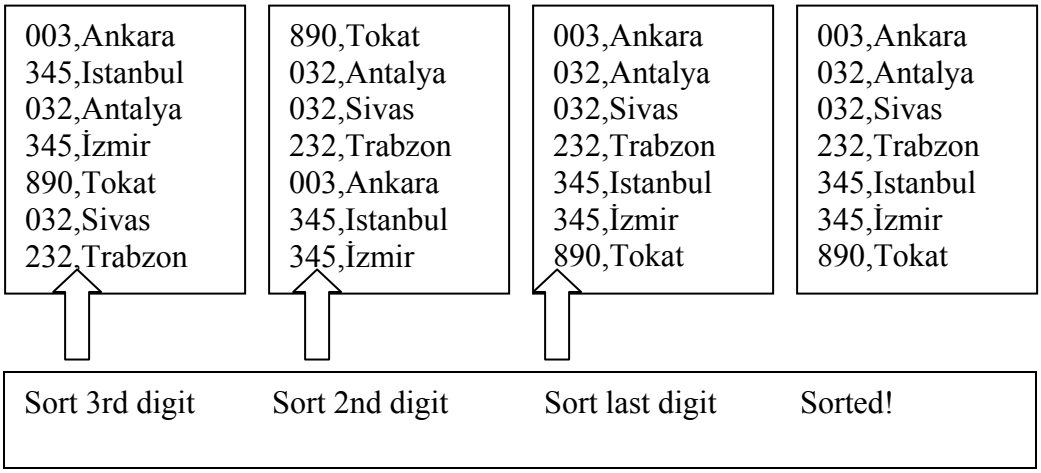
Q2) [20 points] Consider the problem of sorting the following (key, satellite data) pairs: C={(3,Ankara), (345,Istanbul), (32,Antalya), (345,İzmir), (890,Tokat), (32,Sivas), (232,Trabzon)} according to increasing key values.

a) [10 points] Illustrate sorting of C using radix sort.

b) [10 points] Illustrate sorting of C using bucket sort?

Answer 2) Begin----------

Answer 2a)



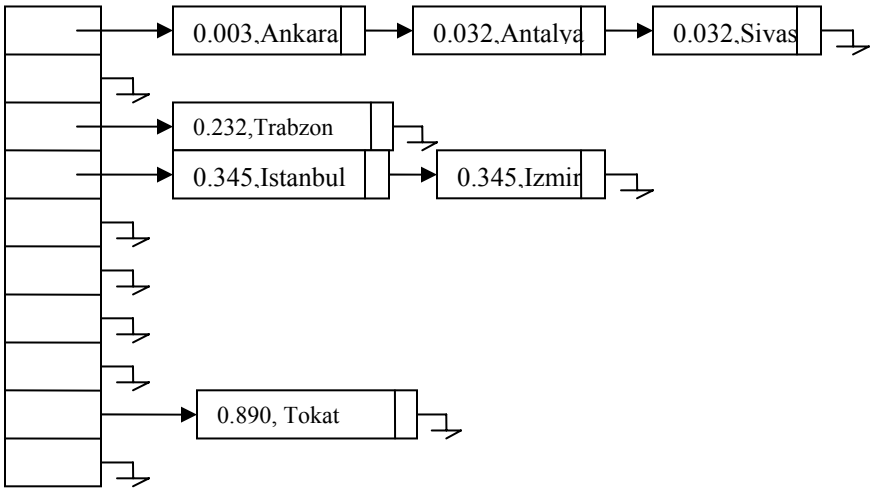
Answer 2b)

Transform keys to [0-1] range.

10 buckets,

For keys in [0:0.1), [0.1:0.2), [0.2:0.3), [0.3:0.4), [0.4:0.5), [0.5:0.6), [0.6:0.7), [0.7:0.8), [0.8:0.9), [0.9:1.0]

Insert key,data pairs into the bucket it belongs. Insert using insertion sort, so the linked lists contain sorted keys all the time.



Answer 2) End----------

Q3) [20 points] Chaining is a method for collision resolution in hash tables. Assume that the chaining algorithm is modified so that the lists are kept in sorted order with respect to the key values. Noting that chaining algorithm in the conventional sense crates the chain by simply inserting the new key to the end of the relevant chain, how does such a change affects the running time for successful and unsuccessful search, as well as for deletion and insertion operations?

Answer 3) Begin----------

Successful search: Since the list is sorted searching for a key will stop when it is found. In average case analysis the running time of a random key search over an unsorted or a sorted list would not be different.

Unsuccessful search: Quite different from the previous approach since accessing the first greatest key value in the sorted array terminates the search and declare the search as unsuccessful. As all the keys in the list has to be visited once in the original one the running time will be shorter.

Deletion: Deleting a key requires a successful search. Therefore the analysis for successful search applies.

Deletemin and Deletemax: (When considered) these operations would run as a single operation. Note that the min and max keys are the first and the last keys in the list. In the original approach a search would be required to find in the original algorithm. Therefore deletemin and deletemax operations are dramatically faster in the new approach.

Insertion: Insert operation in the original requires a single operation as opposed to the new approach in which a search operation has to be performed to provide the location of the insertion. That search operation increases the running time of the insert operation.

Answer 3) End----------

Q4) [10 points] 1. Discuss whether the operation of deletion is commutative in the sense that deleting x and then y from a binary search tree leaves the same tree as deleting y and then x?

Answer 4) Begin----------

There are three cases that have to be considered in binary search tree delete algorithm. Assume that z is the node to be deleted.

Case a) if z has no children
Case b) if z has one child
Case c) if z has two children

These three cases have to be kept in mind for the further scenarios.

Assume that *y* is the other node to be deleted. (Note that the above cases also applies to *y*)

Considering both *z* and *y* nodes

Case i) z and y are nodes in separate sub-trees
Case ii) z and y are nodes in the same sub-tree having a parent-child or a grandparent-child relation.

Discussing the combination of the above mentioned cases (in total six) over abstract binary search trees would conclude that it is commutative.

Answer 4) End----------

Q5) [20 points] Counting sort assumes that each of the n input elements is an integer in the range 0 to k, for some integer k . When k=O(n) the sort runs in Θ(n) time. If we relax the assumptions as follows:

a) the range of integers is only limited by the number of bits to represent a signed integer, and
b) the minimum and the maximum values of the given set of n inputs is unknown

is it possible to modify counting sort algorithm so that it still runs in Θ(n) time? Discuss your answer.

Answer 5) Begin----------

a) The counting sort assumes integers to be positive. If signed integers, i.e. positive and negative integers, are in use the algorithm has to be modified to change these numbers into positive. The simplest approach would be adding a threshold value to all values. This threshold value is the smallest term in the list. Once the smallest term is found the addition of that term to all (*n*) integers in the list individually would have a time complexity of Θ(n). The only question left is the time complexity of finding the smallest term in the list. If it is proved to be Θ(n) or less the overall time complexity would be Θ(n). (Note that this is proved in the second part of the question.)
b) In order to find the min and max values the list has to be revisited once that brings the complexity of Θ(n).

Since it is explained that both assumption can be covered with modifications that have the time complexity of Θ(n) the modified version would also run in Θ(n).

Pseudocodes

```
min_val=max_val=array_s[1];

for i=2 to n
    do
            if (array_s[i] < min_val ) min_val=array_s[i];
            if (array_s[i] > max_val ) max_val=array_s[i];
    done


for i=1 to n
    do
            array_us[i]= =array_s[i]-min_val;
    done
```

Answer 5) End----------