## CS3230 : Design and Analysis of Algorithms (Fall 2014)

## Tutorial Set #4

[For discussion during Week 6]

**S-Problems are due: Friday, 12-Sep, before noon.**

**OUT:** 09-Sep-2014 **Tutorials:** Tue & Wed, 16, 17 Sep 2014

**IMPORTANT: Read "Remarks about Homework".**

**Submit solutions to S-Problem(s) by deadline given above.**
**Prepare your answers to all the D-Problems in every tutorial set**.

When preparing to present your answers,

- Think of a CLEAR EXPLANATION
- Illustrate with a good worked example;
- Describe the main ideas,
- Can you sketch why the solution works;
- Give analysis of running time, if appropriate
- Can you think of other (perhaps simpler) solutions?

**Helpful Hints Series:** In the haste to send out this tutorial, Prof. Leong is taking a break on this problem solving and algorithm design hint series. Will restart this series in the next tutorial.

***Routine Practice Problems** -- do not turn these in -- but make sure you know how to do them.*

**R1. (Exercise 8.2-1 on p. 196 of [CLRS]) [Exercising Counting-Sort]**
Using Figure 8.2 (or lecture note diagram) as a model, illustrate the operation of COUNTING-SORT on the array *A* = [6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2].

**R2. (Exercise 8.3-1 on p. 199 of [CLRS]) [Exercising Radix-Sort]**
[First, read up on Radix-Sort in text or from lecture notes]
Using Figure 8.3 (or lecture note diagram) as a model, illustrate the operation of RADIX-SORT on the following list of English words: COW, DOG, SEA, RUG, ROW, MOB, BOX, TAB, BAR, EAR, TAR, DIG, BIG, TEA, NOW, FOX.

**S-Problems: (To do and submit by due date given in page 1)**
Solve this S-problem(s) and submit for grading.

**IMPORTANT: Write your NAME, Matric No, Tutorial Group in your Answer Sheet.**

**S1. [Bigger Optimal Decision Trees]**
In the lecture notes and the book, we presented a decision tree for sorting *any* permutation of 3 numbers *A*[1..3]. The decision tree has 6 leaf nodes. It is an *optimal* decision tree for sorting 3 numbers, meaning that it has *minimum* height. In general, there can be many optimal decision trees.

In this problem, we want to design an *optimal* decision tree for sorting any permutation of 4 numbers *D*[1..4].

**(a)** What is the *minimum number of leaves* in the decision tree for sorting 4 numbers?
What is the *minimum height* of the decision tree for *D*[1..4].

**(b)** Give an optimal decision tree for sorting 4 numbers *D*[1..4] **where the comparison at the root node of the tree MUST be 1:3** (namely, compare *D*[1] with *D*[3] at the root node).

[*Hint on getting optimal decision trees:* When making a decision on which two items to compare at a given node in the decision tree, you should try to balance (as much as possible) the "possible outcomes" in the left and right sub-trees. Both puns intended.]

[*Hmm...* Why 1:3? Is it because Prof. Leong likes the numbers 1 and 3? While that may be true, you should also *be warned* that this could be his *built-in copy-detector* mechanism.]

**D-Problems:** Solve these D-problems and prepare to discuss them in tutorial class. You may be called upon to present your solution *or your best attempt at a solution*. Your solution presentation does NOT need to be fully correct, given your best attempt. The TA will help clarify and correct any issues or errors.

**D1. Exercise 9.1-1 (p.215)**

Find the second largest of $n$ elements in $(n + \lceil \lg n \rceil - 2)$ comparisons in the worst case.

**D2. [Alternative Decision Tree]**

In the decision tree (for sorting 3 elements) shown in the lectures (Slide L05-S5.4]), we compare 1:2 (namely, $a_1$:$a_2$) at the root of the tree.

Give an alternative decision tree that compares 1:3 (namely, $a_1$:$a_3$) at the root node of the tree and also have *height* 3 (meaning that the worst-case time to sort is still 3 comparisons).

**D3. (Exercise 8.2-4 on p. 197 of [CLRS])**

Describe an algorithm that, given $n$ integers in the range 0 to $k$, preprocesses its input and then answers any query about how many of the $n$ integers fall into a range $[a..b]$ in $O(1)$ time. Your algorithm should use $O(n + k)$ preprocessing time.

**D4. [Median of two sorted arrays] (Modified from past Mid-Term Quiz)**

**(a)** If you are given the following sorted arrays:

```
        A = [ 3,  7, 22, 29, 31, 53, 67, 78 ]
        B = [ 5, 11, 13, 17, 23, 37, 43, 97 ]
```

What is the $8^{th}$ smallest element (median) in the *union* of the two sorted arrays?

**(b) [Finding *n*th smallest in two sorted arrays , each of size *n*]**

You are given two *sorted* arrays, $A[1..n]$ and $B[1..n]$, both of size $n$. You can assume that the two arrays are sorted in *increasing order* and that the elements are *all distinct* (there are *no repeated elements*). Give an $O(\lg n)$ time algorithm for computing the median ($n^{\text{th}}$ smallest element) in the *union* of the two arrays.