CSE5311 Design and Analysis of Algorithms
Exercise Problems 1
08/27/08

General Instructions about Exercise Problems

- *Problems and solutions will be discussed in Class. Solutions will not be available online.*
- *You should attend class or meet with the Instructor/TA to discuss solutions.*
- *It is to your benefit to try and solve the problems before the next class and participate in the discussions.*

1. The list in http://www.fec.gov/pages/elecvote.htm gives the number of electoral votes for each state for the Presidential election (the candidate receiving the majority of the votes in a state collects all the electoral votes for that state.) There are altogether 538 electoral votes. Determine whether it is (mathematically) possible for the election to end up in a tie.

2. The input is a set S containing n real numbers, and a real number x.
   a. Design an algorithm to determine whether there are two elements of S whose sum is exactly x. The algorithm should run in $O(n \log n)$ time.
   b. Suppose now that the set S is given in a sorted order. Design an algorithm to solve the above problem in time $O(n)$.

3. Consider the following algorithms and answer questions **a .. d** for each.

ALGORITHM DOES_SOMETHING (n)
  Input: a nonnegative integers n
  Output : ?
  $S \leftarrow 0$
  for $i \leftarrow 1$ to n **do**
    $S \leftarrow S + i * i$
  return S

ALGORITHM MYSTERY ($A[0 .. n-1]$)
  Input: An array $A[0 .. n-1]$ of n real numbers
  $P \leftarrow A[0]$; $Q \leftarrow A[0]$
  for $i \leftarrow 1$ to $n-1$ do
    if $A[i] < P$
      $P \leftarrow A[i]$
    if $A[i] > Q$
      $Q \leftarrow A[i]$
  return $P-Q$

   a. What does the algorithm do?
   b. What is the basic operation?
   c. How many times is the basic operation executed?
   d. Is the algorithm efficient? If not, suggest a more efficient algorithm.

4. Write a recursive algorithm to find the maximum of n real numbers in an array $A\ [0 .. n-1]$.

5. Derive an expression to find the find the sum of the first n squares, where n is a positive integer. Provide a proof for the sum using induction. Write an algorithm to find the sum of the first n squares. What is the complexity of the algorithm?

Exercise Set 2

1) Answer questions 1a to 1c pertaining to the STOOGE_SORT Algorithm given below.

```
STOOGE_SORT(A,i,j)

if A[i] > A[j]
    then exchange A[i] ↔ A[j]
if i+1 ≥ j
    then return
k ← ⌊(j-i+1)/3⌋
STOOGE_SORT(A,i,j-k)
STOOGE_SORT(A,i+k,j)
STOOGE_SORT(A,i,j-k)
```

a. Argue that STOOGE_SORT (*A*,1,length[*A*]) correctly sorts the input array *A*[1 .. *n*], where *n* = length[*A*].
b. Give a recurrence for the worst-case running time of STOOGE_SORT and a tight asymptotic bound on the worst-case running time.
c. Compare the worst-case running time of STOOGE_SORT with that of insertion sort, merge sort, heapsort, and quicksort.

2) Given an array of integers $A[1..n]$, such that, for all i, $1 \le i < n$, we have $|A[i]-A[i+1]| \le 1$. Let $A[1] = x$ and $A[n] = y$, such that $x < y$. Design an efficient search algorithm to find j such that $A[j] = z$ for a given value z, $x \le z \le y$. What is the maximal number of comparisons to z that your algorithm makes?

3) What is Counting Sort? Give the Algorithm and provide analysis.

4) What are Fibonacci Heaps? Give an example of a Fibonacci Heap. Write basic algorithms for insertion and deletion in Fibonacci Heaps.

5) You are given a collection of *n* bolts of different widths and n corresponding nuts. You are allowed to try a nut and bolt together, from which you can determine whether the nut is larger than the bolt, smaller than the bolt, or matches the bolt exactly. However there is no way to compare two nuts together or two bolts together. The problem is to match each bolt to its nut. Design an algorithm for this problem with average case efficiency of $\Theta(n \log n)$.

6) You are given a list of numbers for which you need to construct a min-heap. (A Min-heap is a complete binary tree in which every key is less than or equal to the keys in its children.) Write a min-heap algorithm and analyze its complexity.

7) Suppose we are to find the k smallest elements in a list of n elements, and we are not interested in their relative order. Can a linear-time algorithm be found when k is a constant? If so give the algorithm. In either case, justify your answer.

8) Design a divide-and conquer algorithm for polynomial evaluation. How many additions and multiplications does your algorithm require? Compare your algorithm with that based on Horner's rule in terms of efficiency.