Introduction to Algorithms (CELEN086)

Problem Sheet 5

Topics: Recursion; Linear search; Binary search; Insertion Sort; Big O notation

1. Write a recursive algorithm called **index**(x,L) that returns the position number of element x in the list L. If x is not in L, your algorithm should return -1. For example,

   index(4, [1, 3, 6, 2, 4]) = 5, index(5,[1, 3, 6, 2, 4]) = -1.

2. Write a recursive algorithm called **concat**(L1,L2) that takes two lists L1, L2 and attaches L1 to the front of L2. For example,

   concat([1,2],[4,9,7]) = [1,2,4,9,7], concat([ ], [3,5,2]) = [3,5,2]

3. Write a recursive algorithm called **insert**(x, i, L) that takes three input arguments: a number x, an index number i and a list L. It should return a list with element x added as the i-th element in the updated list. For example,

   insert(1,2,[4,5,3]) = [4,1,5,3], insert(5,4,[1,2,3]) = [1,2,3,5]

   (Note: you may call length() as a sub-algorithm here.)

4. Write a recursive algorithm called **insert**(x, sortedList) that inserts x into a sorted list.

5. Write a recursive algorithm called **subset**(L1, L2) that takes two lists L1, L2 and return True if L1 is a subset of L2. For example,

   subset([1,5, 2], [4, 5, 1, 0, 2, 9]) = True

   subset([1, 2, 3, 4, 5], [2, 3, 6, 4, 7]) = False

   subset([ ], [4, 6, 3, 2]) = True

   (Note: you may call algorithms we have learned and use them directly here.)

6. Show the process of searching element 36 in the list [2, 3, 5, 8, 13, 17, 26, 33, 36, 41] with linear search. How many operations (comparisons) are needed?

7. Show the process of searching element 36 in the list [2, 3, 5, 8, 13, 17, 26, 33, 36, 41] with binary search. You should demonstrate it by showing intermediate lists you have selected after each comparisons. How many operations (considering comparisons only) are needed?

Introduction to Algorithms (CELEN086)

Problem Sheet 5

8. Suppose we have six algorithms that can solve the same problem on a list of length n. The total operations for each algorithm are described by functions of n:

   i. $n^3 + 200n$
   ii. $15n^2 + 2n + 3$
   iii. $30n \cdot \lg(n)$
   iv. $100 + 5n + \log(n)$
   v. $5 \cdot \lg(n) + 10$
   vi. $4n^2 \log_2 n + 10n^2 + 3n$

   Use big O notation to describe the time complexity of each algorithm.

   Note: in computer science, notations such as $\lg(n)$ and $\log(n)$, are equal to $\log_2 n$.

9. If Q7, which algorithm performs best when the problem size n is relative large?

10. Show the process (or trace the algorithm in Lecture 5) of sorting the list [4,8,3,1,9] with insertion sort. You should demonstrate it by showing the left/right hand lists in each step.

11. Write a recursive algorithm called binSearch(x, sortedList) to implement the binary search method.

    (Hint: you can call functions cut(), getNth() and length() and use them directly.)

12. Write a recursive algorithm to split list L into two list L1 and L2 such that L1 have all elements less than x from L and L2 have all elements from L having value equal to or more than x.
    For example : L = [3,5,1,7,2,9,4] x = 6
    Split(6, [3,5,1,7,2,9,4]) returns L1 = [3,5,1,2,4] and L2 = [7,9]