Algorithms Report1 (Due date: 5 PM, Oct. 2)

Problem solving manually

(Consider the ascending order sorting.)

1. Using Figure 2.2 as a model, **illustrate** the operation of insertion sort on the array A = <10, 7, 1, 8, 2, 6>.

2. Using Figure 2.4 as a model, illustrate the operation of merge sort on the array A = <8, 3, 2, 1, 6, 5, 8, 10, 7>.

3. Illustrate the operation of bubble sort of n the array A = <2, 5, 3, 9, 6, 1>.

4. Write pseudocode for this algorithm, which is known as selection sort and give the best-case and worst-case running times in Θ-notation.

5. Express the following functions in terms of Θ-notation.
   (Must show intermediate steps of a solution.)
   a) $2n^2 + 2n + 5\lg n$
   b) $n^3 + 3n + 10$

6. Prove the following sum by mathematical induction.

$$\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6 \quad \text{for } n > 0$$

7. Draw the recursion tree for $T(n) = 2T(\mathrm{n}/2) + n$ and provide a asymptotic upper bound (O-notation). Also, verify your bound by the substitution method.

8. Use the master method to give tight asymptotic bounds for the following recurrences.

A. $T(n) = 2T(n/2) + n^3$.

B. $T(n) = 16T(n/4) + n^2$.

C. $T(n) = 7T(n/2) + n^2$.

Programming (C language)

1. **Write the INSERTION-SORT function to sort into descending order.**

   a. Write in pseudo-code (style as shown in the text book).
   b. The program should count the number of comparison operations.

   - Test the function with the following three types of inputs.
     1) int A[100] : filled by rand()%10000, execute srand(time(NULL)) first,
        (stdlib.h, time.h should be included)
        (Duplicate keys are ignored.)
     2) int A[100] : already sorted (Write a function for filling in A[])
     3) int A[100] : reversely sorted "
   - Print A, before and after sorting for each case of input.
   - Give the number of comparisons for each case of input.

2. **Write the MERGE-SORT function to sort into descending order.**

   The program should count the number of comparison operations.

   - Test the function with the following three types of **integer** inputs.
     1) int A[100] : filled with rand()%1000, execute srand(time(NULL)) first,
        (stdlib.h, time.h should be included)
        (Duplicate keys are ignored.)
     2) int A[100] : already sorted (Write a function for filling in A[])
     3) int A[100] : reversely sorted "
   - For the inputs of 2) and 3), A[] can be filled with the integers from 100 ~ 1 (from 100 down to 1) and 1 ~ 100 (from 1 to 100) respectively.
   - Print A[], before and after sorting for each case of above inputs.
   - Print the number of comparisons for each case of above inputs.

3. Write functions which perform according to the following descriptions.
The input to each function is a linked list of integers.

a) insert

- Inserts an integer x to the end of a linked list.
  e.g.) insert(lst, x) where lst is a pointer to a linked list and x is an integer.

b) delete

- Deletes 2nd integer x in the linked list.
  e.g.) delete(lst, x)

c) print

- prints the content of a linked list in two lines as described below
  1st line : 1st half of the list
  2nd line : 2nd half of the ist
  e.g.) print(lst)

• Test the functions as shown below.

1) Construct the linked list from a set of integers stored in an array using the insert function in a).
   Where the length of the array is 100 and should be filled by rand()%20 (execute srand(time(NULL)) first).
2) Then randomly select an integer from the array and delete this integer(s) from the linked list using delete function in b).
3) Print the content of the linked list in two lines using print function in c).
4) Repeat 2) and 3) two more times.

4. Program the divide and conquer matrix multiplication using

   1) standard algorithm
   2) recursion
   3) strassen's method.

• For above three algorithms

a) Compare the number of computations (multiplication, addition, and subtraction) among 1), 2), and 3) cases above.
   In the matrix computation of C = A×B, matrices A and B are filled with rand()%1000, execute srand(time(NULL)) first.
b) Print whenever a partial matrix is constructed, that is, a return value from a recursion is made, until the completion of a matrix. < Only for 2) and 3) >

- Test with the 4x4 matrix multiplication and the 8x8 matrix multiplication. (Print matrices, A, B, and C.)

- Submit (to the room 27319) the following in hardcopy (printout).
  a) solution of problem solving manually part
  b) the program (source code) and test results of programming part

- Mail the program (source code) only to sc4217@skku.edu (41 class)
- Mail the program (source code) only to wonjin12@skku.edu (43 class)