Algorithms Report1 (Due date: 5PM, Sep. 30, 2022)

Problem solving manually

(Must write down the problem solving process.)

1. Using Figure 2.4 (in the text book) as a model, illustrate the operation of merge sort (ascending order) on the array A = <3, 41, 6, 26, 22, 11, 9, 4>

2. Consider sorting n numbers stored in array A by first finding the largest element of A and exchanging it with the element in A[1]. Then find the second largest element of A, and exchange it with A[2]. Continue in this manner for the first n-1 elements of A.

a. Write pseudocode for this algorithm, which is known as *selection sort.*

b. Why does it need to run for only the first $n-1$ elements, rather than for all n elements?

c. Give the best-case and worst-case running times of selection sort in Θ-notation.

d. Using Figure 2.2 as a model, illustrate the operation of the selection sort on the array A = <13, 16, 12, 21, 7, 8, 25, 32>.

3. Express the following functions in terms of Θ-notation.

a. $2n^3 + n^2 + 1$

b. $n^2 + 2n + \lg n$

4. Draw the recursion tree for $T(n) = 2T(n/2) + cn^2$ where, c is constant. Provide a good asymptotic upper bound (O-notation). Also, verify your bound by the substitution method.

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

a) $T(n) = 9T(n/3) + n$

b) $T(n) = 9T(n/3) + n^2$

c) $T(n) = 9T(n/3) + n^3$

Programming (C language)

1. **Write the SELECTION-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, that is, avoid identical values when randomly generating values.)
     2) int A[100] : already sorted (Write a function for filling in A[]).
     3) int A[100] : reversely sorted (Write a function for filling in A[]).

   (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 the above inputs.
   - Print the number of comparisons for each case of the above inputs.

2. **Write the MERGE-SORT function to sort into ascending 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, that is, avoid identical values when randomly generating values.)
     2) int A[100] : already sorted (Write a function for filling in A[].)

3) int A[100] : reversely sorted (Write a function for filling in A[].

(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 the above inputs.
- Print the number of comparisons for each case of the 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 3rd last integer x in the linked list.
  e.g.) delete(lst)

c) print

- prints the content of a linked list in three lines as described below
  1st line : 1st third of the list
  2nd line : 2nd third of the list
  3rd line : 3rd third of the list
  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 60 and should be filled by
   rand()%1000 (execute srand(time(NULL)) first).
   (Avoid same values when generating the values randomly.)
2) Then execute the delete function in b).
3) Print the content of the linked list using print function in c).
4) Repeat 2) and 3) two more times.

4. Program the matrix multiplication using

1) standard algorithm (class note, page 18)
2) divide-and-conquer algorithm (class note, page 20)
3) strassen algorithm (class note, page 28)

• For the above cases 1), 2), 3)

a) Compare the number of computations (multiplication, subtraction, addition) among 1), 2), 3) cases.
In the matrix computation of C = A×B, matrices A and B are filled with rand()%1000, execute srand(time(NULL)) first.
(Note that identical values are allowed.)

• For the case 2) and 3)

b) Print whenever a partial matrix (except 1×1) of C is constructed, that is, whenever a return value from a recursion is determined, until the completion of the matrix multiplication.

▶ Execute with the 4x4 matrix multiplication and the 8x8 matrix multiplication.
Print matrices, A, B, and C for 4x4 and 8x8 matrices.

How to submit the report.

▶ Need to upload the report1 in a zip file in the i-campus.
Refer to the manual file for uploading in the i-campus.

▶ The zip file should contain the following three files.

1) Document file (.hwp, photo, or scan): Problem solving manually part.
2) C program file(s): Programming part.
3) Test result file(s): Contains all the screen copy of the test results.

▶ The zip file should be named as shown below,

report1_id_name.zip
example) report1_2020123456_HongGilDong.zip
or report1_2020123456_홍길동.zip

The zip file contains above 1), 2), and 3).

▶ Use windows OS and visual studio program.