EECE 7205: Introduction of Computer Engineering

Assignment 1

Jiayun Xin

NUID: 001563582

College of Engineering

Northeastern University Boston, Massachusetts

Fall, 2021

Q1

Codes:

```
#include <time.h>
#include <iostream>
using namespace std;

void merge_array(int arr[], int l, int m, int r) {
   int i, j, k;
   int n1 = m - l + 1;
   int n2 =  r - m;
   int L[n1], R[n2];
   for (i=0;i<n1;i++) {
      L[i]=arr[l+i];
   }
   for (j=0;j<n2;j++) {
      R[j]=arr[m+1+j];
   }
   i=0; j=0; k=l;
   while(i<n1&&j<n2) {
      if(L[i]<=R[j]) {
         arr[k]=L[i];
         i++;
      }
      else {
         arr[k]=R[j];
         j++;
      }
      k++;
   }
   while(i<n1) {
      arr[k]=L[i];
      i++;
      k++;
   }
   while(j<n2) {
      arr[k]=R[j];
      j++;
      k++;
   }
   return ;
}

void merge_sort(int arr[], int l, int r) {
```

```
    if (l < r) {
        int m = l+(r-l)/2;
        merge_sort(arr, l, m);
        merge_sort(arr, m+1, r);
        merge_array(arr, l, m, r);
    }
    return ;
}

void insertionSort(int arr[], int length) {
      int i, j, tmp;
      for (i = 1; i < length; i++) {
            j = i;
            while (j > 0 && arr[j - 1] > arr[j]) {
                  tmp = arr[j];
                  arr[j] = arr[j - 1];
                  arr[j - 1] = tmp;
                  j--;
            }
      }
}

int main() {
   int n = 10000;
   int tmp = n;
   int arr[n];
   for(int i=0;i<tmp;i++){
       arr[i] = tmp;
       tmp--;
   }

    clock_t insertion_time_start = clock();
    insertionSort(arr, n);
    clock_t insertion_time_end = clock();

    tmp = n;
    for(int i=0;i<tmp;i++){
        arr[i] = tmp;
        tmp--;
    }

    clock_t merge_time_start = clock();
    merge_sort(arr, 0, n-1);
    clock_t merge_time_end = clock();
```

```
   cout << "Input size(n) : " << n << endl;
   cout << "Processing time of insertion sort : " << (float)(insertion_time_end -
insertion_time_start)/CLOCKS_PER_SEC << " seconds" << endl;
   cout << "Processing time of merge sort : " << (float)(merge_time_end -
merge_time_start)/CLOCKS_PER_SEC << " seconds" << endl;
   return 0;
}
```

Results:

```
Input size(n) : 1
Processing time of insertion sort : 6e-06 seconds
Processing time of merge sort : 0 seconds

Input size(n) : 10
Processing time of insertion sort : 1e-06 seconds
Processing time of merge sort : 1e-06 seconds

Input size(n) : 100
Processing time of insertion sort : 2.2e-05 seconds
Processing time of merge sort : 1.2e-05 seconds

Input size(n) : 1000
Processing time of insertion sort : 0.001676 seconds
Processing time of merge sort : 9.8e-05 seconds

Input size(n) : 10000
Processing time of insertion sort : 0.138791 seconds
Processing time of merge sort : 0.000767 seconds

Input size(n) : 100000
Processing time of insertion sort : 11.2006 seconds
Processing time of merge sort : 0.008783 seconds
```

Q2

insertion sort:

10, 5, 7, 9, 8, 3

5, 10, 7, 9, 8, 3

5, 7, 10, 9, 8, 3

5, 7, 9, 10, 8, 3

5, 7, 8, 9, 10, 3

3, 5, 7, 8, 9, 10 done

Quick sort: (method 1)

{10, 5, 7, 9, 8, 3} partition around 9

{5, 7, 8, 3} partition around 7

{10}

{5, 3} partition around 3

{8}

{}

5

3, 5, 7, 8, 9, 10. done

Quick sort (method 2)

10 5 7 9 8 3 $x \leftarrow 10$ pivot

i j

10 5 7 9 8 3

i j

10 5 7 9 8 3

i j

10 5 7 9 8 3

i j

10 5 7 9 8 3

i j loop ends

$\underbrace{5\ \ 7\ \ 9\ \ 8\ \ 3}_{\le \text{pivot}}\ \ 10$

$x \leftarrow 5$ pivot

i j

3 5 7 9 8 $\boxed{10}$

i j $7 \leftarrow$ pivot

3 $\boxed{5}$ $\boxed{7}$ 9 8 $\boxed{10}$

i j $9 \leftarrow$ pivot

3 5 7 8 9 10

Q3

1. $n+3 \in \Omega(n)$ True

$0 \leq cn \leq n$

$n_0 = 0$

2. $n+3 \in O(n^2)$ True

$0 \leq n \leq cn^2$

$cn \geq 3 \qquad n \geq \frac{3}{c}$

$n_0 = \frac{3}{c}$

3. $n+3 \in \Theta(n^2)$ False

$0 \leq c_1 n^2 \leq n \leq c_2 n^2$

$0 \leq c_1 n \leq 1 \leq c_2 n$

$\frac{1}{c_2} \leq n \leq \frac{1}{c_1}$

4. $2^{n+1} \in O(n+1)$ False

$2^{n+1} \leq c(n+1)$

$2^{n+1} \leq cn$

$2^n \leq c$

5. $2^{n+1} \in \Theta(2^n)$ True

$0 \leq c_1 2^n \leq 2^{n+1} \leq c_2 2^n$

$c_1 \leq 2 \leq c_2$

$n_0 = 2$

Q4

1. $T(n) = 8T(\frac{n}{2}) + n$

$a=8$, $b=2$

$n^{\log_b a} = n^{\log_2 8} = n^3$

$f(n) = n = O(n^{3-\varepsilon})$, for some $\varepsilon > 0$

$\Rightarrow$ case 1: $T(n) = \Theta(n^3)$

2. $T(n) = 8T(\frac{n}{2}) + n^2$

$a=8$, $b=2$, $f(n) = n^2$ $n^{\log_b a} = n^3$ $\Rightarrow$ case 1:

$T(n) = \Theta(n^3)$

3. $T(n) = 8T(\frac{n}{2}) + n^3$

$a=8$. $b=2$, $f(n) = n^3$ $n^{\log_b a} = n^3 \Rightarrow$ case 2:

$T(n) = \Theta(n^3 \cdot \lg n)$

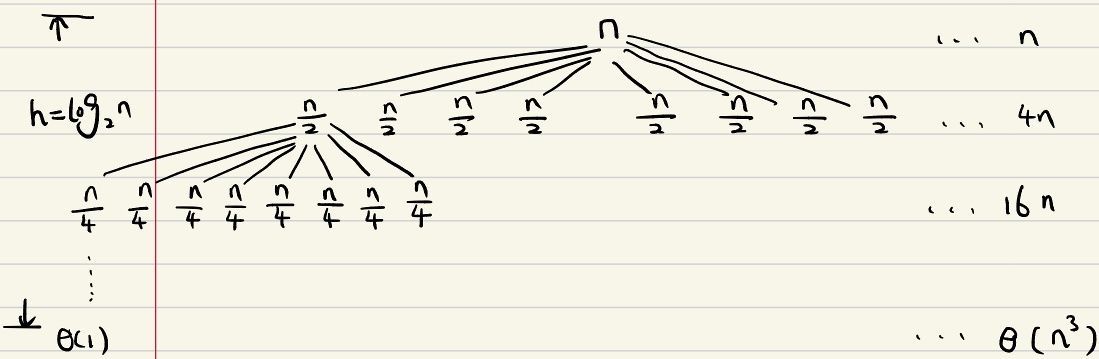
4. $T(n) = 8T(\frac{n}{2}) + n^4$

$a=8$, $b=2$, $f(n) = n^4$ $n^{\log_b a} = n^3 \Rightarrow$ case 3:

$T(n) = \Theta(n^4)$

Q5

$T(n) = 8T(\frac{n}{2}) + n$



\# leaves $a^{\log_b n} = n^{\log_b a} = n^3$

$$T_n = \underbrace{(1 + 4 + 16 + \cdots + 4^{k-1})}_{\log_2 n} n + \Theta(n^3)$$

$$= O(n^3)$$

Substitution method :

- guess $T(n) = O(n^3)$
- Assume $T(k) \le ck^3$ for $k<n$

$T(n) = 8T(\frac{n}{2}) + n \le 8 \cdot c(\frac{n}{2})^3 + n = cn^3 + n$

$= \underbrace{cn^3}_{\text{desired}} - \underbrace{(-n)}_{\text{residual}}$

- assume $T(k) \le C_1 k^3 - C_2 k$ for $k<n$

$T(n) = 8T(\frac{n}{2}) + n \le 8[C_1(\frac{n}{2})^3 - C_2(\frac{n}{2})] + n = C_1 n^3 - 4C_2 n + n$

$= \underbrace{(C_1 n^3 - C_2 n)}_{\text{desired}} - \underbrace{(3C_2 - 1)n}_{\text{residual}}$

base case $T(1) = \Theta(1) \le C_1 - C_2$ if c is chosen sufficiently large with respect to C_2