SE220 Lecture1&2 Notes

I. Elementary Data Structure Revision(omitted)

linked list, stack, queue, matrix, rooted trees

II. **Insertion Sort**(in increasing order)

i. Analysis of time complexity

Give the pseudocode of the algorithm

```
INSERTION-SORT(A, n)
1  for i = 2 to n
2      key = A[i]
3      // Insert A[i] into the sorted subarray A[1 : i − 1].
4      j = i − 1
5      while j > 0 and A[j] > key
6          A[j + 1] = A[j]
7          j = j − 1
8      A[j + 1] = key
```

To compute the total running time(denoted by $T(n)$) of the algorithm, two parameters that determine the time should be defined – cost(which means time needed when a line is executed) and times. Therefore, if we define the cost from c_1 to c_8 for each line(including the annotation line), we can obtain:

INSERTION-SORT(A, n)	*cost*	*times*
1 **for** $i = 2$ **to** n	c_1	n
2 $\quad$ *key* $= A[i]$	c_2	$n-1$
3 $\quad$ // Insert $A[i]$ into the sorted subarray $A[1:i-1]$.	0	$n-1$
4 $\quad$ $j = i - 1$	c_4	$n-1$
5 $\quad$ **while** $j > 0$ and $A[j] >$ *key*	c_5	$\sum_{i=2}^{n} t_i$
6 $\qquad$ $A[j+1] = A[j]$	c_6	$\sum_{i=2}^{n} (t_i - 1)$
7 $\qquad$ $j = j - 1$	c_7	$\sum_{i=2}^{n} (t_i - 1)$
8 $\quad$ $A[j+1] =$ *key*	c_8	$n-1$

Consider the best and worst case respectively, we can obtain that

Best case: All elements in the array are sorted so that command in line 5 is executed 1 time($t_i = 1$).

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{i=2}^{n} 1 + c_8(n-1)$$

$$= (c_1 + c_2 + c_4 + c_8)n + c_5(n-1) - (c_2 + c_4 + c_8)$$

$$= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8)$$

Thus, $T(n) = \Theta(n)$

Worst case: All elements in the array are reversed so that command in line 5 is executed i times($t_i = i$).

$$
\begin{aligned}
T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{i=2}^{n} i + c_6 \sum_{i=2}^{n}(i-1) + c_7 \sum_{i=2}^{n}(i-1) + c_8(n-1) \\
&= (c_1 + c_2 + c_4 + c_8)n + (c_5 + c_6 + c_7) \sum_{i=2}^{n} i - (c_6 + c_7) \sum_{i=2}^{n} 1 - (c_2 + c_4 + c_8) \\
&= (c_1 + c_2 + c_4 + c_8)n + (c_5 + c_6 + c_7)\left[\frac{1}{2}n(n+1) - 1\right] - (c_6 + c_7)(n-1) - (c_2 + c_4 + c_8) \\
&= (c_1 + c_2 + c_4 + c_8)n + (c_5 + c_6 + c_7)\left(\frac{1}{2}n^2 + \frac{1}{2}n - 1\right) - (c_6 + c_7)n + (c_6 + c_7 - c_2 - c_4 - c_8) \\
&= \frac{1}{2}(c_5 + c_6 + c_7)n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2}\right)n - (c_2 + c_4 + c_5 + c_8)
\end{aligned}
$$

Thus, $T(n) = \Theta(n^2)$

III. **Divide-and-Conquer** in Merge Sort

The core of Merge sort is **Divide-and-Conquer**, which can be illustrated in following steps:

1. Divide the original array into **two** subarrays $A[p{:}q]$ and $B[q+1{:}r]$, where $q = \left\lfloor \frac{p+r}{2} \right\rfloor$.
2. Conquer by recursively sorting the two subarrays.
3. Combine the sorted subarrays into the new-sorted original one.

According to the procedure described above, we can obtain the recurrence:

$$
T(n) = \begin{cases} \Theta(1) & n = 1 \\ 2T\left(\frac{n}{2}\right) + D(n) + C(n) & n > 1 \end{cases}
$$

where $D(n)$ represents the time needed to divide the original array, which is $\Theta(1)$ since this operation only includes computing q; $C(n)$ represents the time needed to combine the subarrays, which is $\Theta(n)$. Thus, $D(n) + C(n) = \Theta(n)$.

Suppose using c_1 and $c_2 n$ to represent the function of $\Theta(1)$ and $\Theta(n)$. We can obtain the recurrence as:

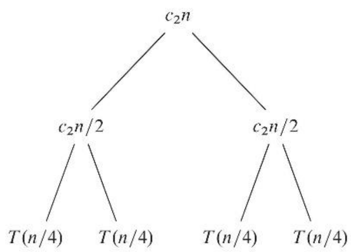
$$T(n) = \begin{cases} c_1 & n = 1 \\ 2T\left(\dfrac{n}{2}\right) + c_2 n & n > 1 \end{cases}$$

To solve this recurrence equation, one of the methods we can use is to draw the recursion tree to compute $T(n)$. Before drawing the tree, we can use iteration to determine what does the tree look like.

Assume that $n > 1$

$$\begin{aligned} T(n) &= 2T\left(\frac{n}{2}\right) + c_2 n \\ &= 2\left(2T\left(\frac{n}{4}\right) + c_2 \cdot \frac{n}{2}\right) + c_2 n \\ &= 2 \cdot 2T\left(\frac{n}{4}\right) + 2 \cdot c_2 \cdot \frac{n}{2} + c_2 n \\ &= \ldots \end{aligned}$$

According to the formula we can obtain the tree(incomplete).



$$\begin{aligned} T(n) &= \sum_{i=0}^{\log_2 n} 2^i \cdot \left(\frac{1}{2}\right)^i c_2 n + 2^{\log_2 n} c_1 \\ &= c_2 n \log_2 n + c_1 n \\ &= \Theta(n \log_2 n) \end{aligned}$$