CS217 Homework-3

HONGLI YE 12311501

September 26^{th} 2024

Contents

1 Question 1

Consider the following input for **MergeSort**:

12	10	4	2	9	6	5	25	8

Table 1: Question 1

Illustrate the operation of the algorithm (follow how it was done in the lecture notes).

Answer:

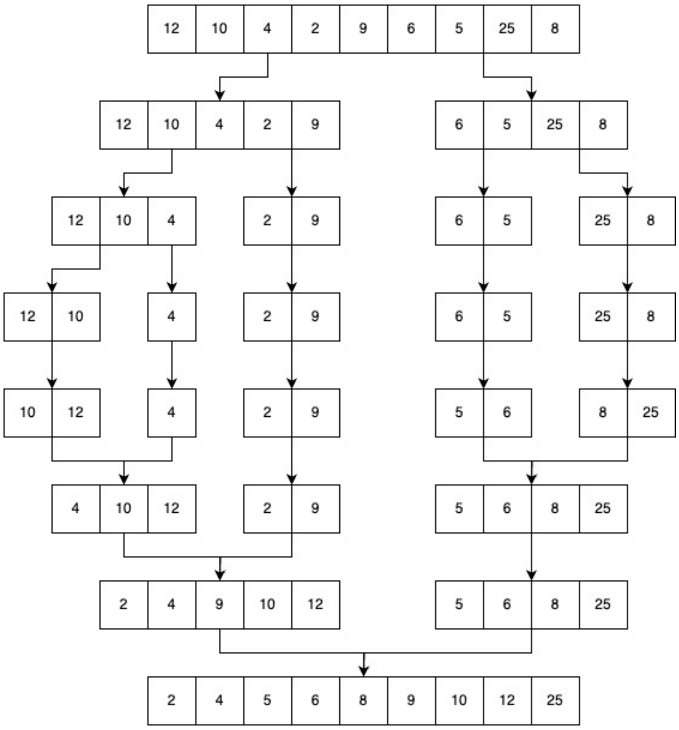

Figure 1: Question 1

2 Question 2

Prove using the substitution method the runtime of the Merge-Sort Algorithm on an input of length n, as follows. Let n be an exact power of 2, $n = 2k$ to avoid using floors and ceilings. Use mathematical induction over k to show that the solution of the recurrence involving positive constants $c, d > 0$

$$T(n) = \begin{cases} d & \text{if } n = 2^0 = 1 \\ 2T(n/2) + cn & \text{if } n = 2^k \text{ and } k \geq 1 \end{cases}$$

is $T(n) = dn + cn\log n$ (we always use log to denote the logarithm of base 2, so log $= \log_2$).

Answer:

Let $a_k = T(2^k)$, then we have

$$a_k = \begin{cases} d & \text{if } k = 0 \\ 2a_{k-1} + c2^k & \text{if } k \geq 1 \end{cases}$$

Suppose $k \geq 1$, then we have:

$$\frac{a_k}{2^k} = \frac{a_{k-1}}{2^{k-1}} + c$$

So:

$$\frac{a^k}{2^k} \text{ is an arithmetic sequence whose common difference is } c \text{ and first number } = \frac{a_0}{2^0} = d$$

So:

$$\frac{a_k}{2^k} = d + kc, T(2^k) = (d + kc) * 2^k$$

Substitute k with $\log n$. We have the general representation of $T(n)$.

$$T(n) = (d + c\log n) * n$$

3 Question 3

Use the Master Theorem to give asymptotic tight bounds for the following recurrences. Justify your answers.

1. $T(n) = 2T(\frac{n}{4}) + 1$
2. $T(n) = 2T(\frac{n}{4}) + \sqrt{n}$
3. $T(n) = 2T(\frac{n}{4}) + \sqrt{n}\log^2(n)$
4. $T(n) = 2T(\frac{n}{4}) + n$

Answer:

The general form of the four questions are $T(n) = 2T(\frac{n}{4}) + f(n)$. Using **Master Theorem**, we need to compare $f(n)$ with $n^{log_b(a)} = \sqrt{n}$ first, and then compare with $\sqrt{n}log^k(n)$.

1. $f(n) = 1 = O(n^{\frac{1}{2}-\frac{1}{4}})$, so $T(n) = \Theta(\sqrt{n})$
2. $f(n) = \sqrt{n} = \Theta(\sqrt{n}log^0(n))$, so $T(n) = \Theta(\sqrt{n}log(n))$
3. $f(n) = \sqrt{n} = \Theta(\sqrt{n}log^2(n))$, so $T(n) = \Theta(\sqrt{n}log^3(n))$
4. $f(n) = n = \Omega(n^{\frac{1}{2}+\frac{1}{4}})$, so $T(n) = \Theta(n)$

4 Question 4

Write the pseudo-code of the recursive Binary-Search$(A, x, low, high)$ algorithm discussed during the lecture to find whether a number x is present in an increasingly sorted array of length n. Write down its recurrence equation and prove that its runtime is $\Theta(logn)$ using the Master Theorem.

Answer:

Algorithm 1 Binary-Search$(A, x, low, high)$

$mid = \frac{low+high}{2}$
if $low > high$ **then**
    return No element founded.
end if

if $A[mid] = x$ **then**
    return mid
else if $A[mid] > x$ **then**
    return Binary-Search$(A, x, low, mid - 1)$
else
    return Binary-Search$(A, x, mid + 1, high)$
end if

Now we analyse its time complexity.
Let line 1 to 7's time cost be c_1, by its recursive relationship and structure, we can get a formula:

$$T(n) = T(n/2) + c_1,\ \text{where } c_1 \text{ is a constant}$$

Since:

$$c_1 = \Theta(n^{log_2(1)} \times log^0(n))$$

By **Master Theorem**, we can get the conlusion that:

$$T(n) = \Theta(log(n))$$

5 Question 5

Implement the MergeSort(A, p, r) algorithm and the BinarySearch$(A, x, low, high)$ algorithm designed in the previous question. Solve programming problems "Find x", "Inversion Pair" and "Key Letter K in Keyboard" provided on Judge.

Answer:

I Already finished on OJ.