CSE321 - Introduction to Algorithms - HW4

Fatih Kaan Salgır - 171044009

1

The minimum number of cut is ensured by performing the cut from the middle of the steel. At any step, in case the length of the steel is an odd number, the middle is determined by floor division. The part with shorter length can be divided along with the longer part. Therefore, the algorithm called recursively, with length of longer part.

Backward substitution:

$$T(n) = 1 + T\left(\frac{n}{2}\right)$$
$$T(n) = 2 + T\left(\frac{n}{4}\right)$$
$$\vdots$$
$$T(n) = k + T\left(\frac{n}{2^k}\right)$$

Assume $\frac{n}{2^k} = 1 \therefore n = 2^k$ and $k = \log(n)$

$$T(n) = T(1) + T(\log(n))$$
$$T(n) \in \Theta(\log(n))$$

2

The worst and the best results are calculated by dividing the array into 2 parts, until there are 1 or 2 elements left in the sub-arrays. The minimum and maximum values are calculated in base case and returned. The time complexity of algorithm is $\Theta(n)$

Backward substitution:

$$
\begin{aligned}
T(n) &= T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) \\
T(n) &= 2T\left(\frac{n}{2}\right) \\
T(n) &= 2^2T\left(\frac{n}{2^2}\right) \\
&\vdots \\
T(n) &= 2^kT\left(\frac{n}{2^k}\right)
\end{aligned}
$$

Assume $\frac{n}{2^k} = 1 \therefore n = 2^k$

$$
\begin{aligned}
T(n) &= nT(1) \\
T(n) &\in \Theta(n)
\end{aligned}
$$

The space complexity of algorithm is $\Theta(n)$ because of the recursive call stack.

3

To obtain the first meaningful k^{th} experiment, minimum k^{th} element in the sorted can be returned. However sorting is not necessary to find minimum k^{th} element. The quick select algorithm can be used in order to find out the result without sorting the array. The partition algorithm will return the sorted index of the last element. Also it will ensure that all the element less than the pivot are moved to the left side of the array. Depending on the pivot index the left or right side will be valuated until the desired k is the index of pivot.

In the best case the algorithm will return the desired k^{th} element after running single partition algorithm. One of the worst cases is when the array is already sorted and we are looking for the first meaningful experiment. In this case the partition algorithm runs as the size of the array as much times.

$$
\begin{aligned}
T_{\text{worst}}(n) &= \sum_{i=0}^{n} T_{partition} \in \Theta(n^2) \\
T_{\text{best}}(n) &= T_{partition} \in \Theta(n)
\end{aligned}
$$

$$T(n) \in O(n)$$

4

In order to obtain the number of reverse-ordered pairs with divide & conquer algorithm; we can use the merge sort itself and count the number of reverse-order pairs. In merge part of the algorithm, where we compare the left and right side, we make use of the number of unordered pairs. The complexity of the algorithm is identical with merge sort which is Θ(n log(n))

$$T(n) = aT(\frac{n}{b}) + f(n) \text{ where}$$
$$a: \text{ cost of the sub problem}$$
$$\frac{n}{b}: \text{ size of the sub problem}$$

The recurrence relation is,

$$T(n) = 2T(\frac{n}{2}) + T_{merge}(n)$$
$$T(n) = 2T(\frac{n}{2}) + \Theta(n)$$

The complexity of the recurrence relation can be determined by Master theorem.

$$a = 2$$
$$b = 2$$
$$f(n) = \Theta(n^1) \implies c = 1$$
$$\log_b a = \log_2 2 = 1$$
$$\log_b a = c \implies T(n) = \Theta(n^c \log(n)) = \Theta(n \log(n))$$

Therefore the time complexity is $\Theta(n \log(n))$

5

Brute-force algorithm is a straight-forward approach which requires multiplication of a, n times. The time complexity is $\Theta(n)$.

$$T(n) = \sum_{i=0}^{n} 1 \in \Theta(n)$$

Divide & conquer algorithm is solving problems by dividing into 2 sub-problems, with half of the size, and combines them in constant time.

$$T(n) = aT(\frac{n}{b}) + f(n) \text{ where}$$
$$a: \text{ cost of the sub problem}$$
$$\frac{n}{b}: \text{ size of the sub problem}$$

The recurrence relation is,

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

The complexity of the recurrence relation can be determined by Master theorem.

$$a = 2$$
$$b = 2$$
$$f(n) = \Theta(n^0) \implies c = 0$$
$$\log_b a = \log_2 2 = 1$$
$$\log_b a > c \implies T(n) = \Theta(n^{\log_b a}) = \Theta(n)$$

Therefore the complexity of divide & conquer algorithm is $\Theta(n)$