cpt_s 350

Homework 2

11641327 Yu-Chieh Wang

2020/2/7

1. Psuedo-code for partition(A, p, q)

```
partition(A, p, q){
        // r (Element to be placed at right position)
        r = A[q]
        i = (p-1) // index of smaller element
        for (j = p; j < q; j++){
                // if current element is smaller than the r
                if (A[j] < r){
                        i++; // increment index of smaller element
                        swap A[i] and A[j]
                }
        }
        swap A[i+1] and A[q]
        return (i+1)

}
```

2. The average-case complexity of insertsort is $\theta(n^2)$

$$T_{AVG}(n) = 1\% \times \theta(n^2) + 99\% \times O(n^2) = \theta(n^2)$$

3. The best-case, worst-case, and average-case complexities of iqsort.

	Best-case	**Worst-case**	**Average-case**
Insertsort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Quicksort	$O(nlogn)$	$O(n^2)$	$O(nlogn)$

First of all, we separate the process of iqsort to three steps:

(1) do partition

low part **(r-1)**	**r**	**high part** **(n-r)**

(2) run quicksort on the low part

(3) run insertsort on the high part

Then, we count the complexities by these steps:

(a) The best-case complexity:

(1) $T_{BEST}(n) = O(n)$

(2) $T_{BEST}(n) = \min_{1 \le r \le n} \{T_{BEST}(r-1)\} = O(rlogr)$

(3) $T_{BEST}(n) = \min_{1 \le r \le n} \{T_{BEST}(n-r)\} = O((n-r)^2)$

$$T_{BEST}(n) = O(n) + O(rlogr) + O((n-r)^2) = O(n^2)$$

(b) The worst-case complexity:

(1) $T_{WORST}(n) = O(n)$

(2) $T_{WORST}(n) = \max_{1 \le r \le n} \{T_{WORST}(r-1)\} = O(r^2)$

(3) $T_{WORSt}(n) = \max_{1 \le r \le n} \{T_{WORST}(n-r)\} = O((n-r)^2)$

$$T_{WORST}(n) = O(n) + O(r^2) + O((n-r)^2) = O(n^2)$$

(c) The average-case complexity:

(1) $T_{AVG}(n) = O(n)$

(2) $T_{AVG}(n) = \frac{1}{n}\sum_{r=1}^{n} \{T_{AVG}(r-1)\} = O(rlogr)$

(3) $T_{AVG}(n) = \frac{1}{n}\sum_{r=1}^{n}\{T_{AVG}(n-r)\} = O((n-r)^2)$

$T_{AVG}(n) = O(n) + O(rlogr) + O((n-r)^2) = O(n^2)$

4. The best-case, worst-case, and average-case complexities of mixsort.

First of all, we separate the process of iqsort to three steps:

(1) do partition

low part (r-1)	**r**	**high part (n-r)**

(2) run mixsort on the low part

(3) run insertsort on the high part

Then, we count the complexities by these steps:

(a) The best-case complexity:

(1) $T_{BEST}(n) = O(n)$

(2) $T_{BEST}(n) = \min_{1\le r\le n}\{T_{BEST}(r-1)\} = O(r^2)$

(3) $T_{BEST}(n) = \min_{1\le r\le n}\{T_{BEST}(n-r)\} = O((n-r)^2)$

$T_{BEST}(n) = O(n) + O(r^2) + O((n-r)^2) = O(n^2)$

(b) The worst-case complexity:

(1) $T_{WORST}(n) = O(n)$

(2) $T_{WORST}(n) = \max_{1\le r\le n}\{T_{WORST}(r-1)\}$

(3) $T_{WORSt}(n) = \max_{1 \le r \le n} \{T_{WORST}(n-r)\} = O((n-r)^2)$

Then, we get the worst-case time complexity is:

$$T_{WORST}(n) = \max_{1 \le r \le n} \{O(n) + T_{WORST}(r-1) + O((n-r)^2)\} = O((n-r)^2)$$

Therefore, we guess the result of $T_{WORST}(n) = O(n^2)$.

That is $\exists c > 0$ which $T_{WORST}(n) \le c \times n^2$ for almost all n.

(I.H. $\forall i < n, T_{WORST}(i) \le c \times i^2$)

Finally, we test the result to see if we guess right.

$$T_{WORST}(n) = \max_{1 \le r \le n} \{a \cdot n + T_{WORST}((r-1)) + a \cdot ((n-r)^2)\}$$

According to the I.H, we can modify the formula as follow:

$$T_W(n) = \max_{1 \le r \le n} \{a \cdot n + T_{WORST}((r-1)) + C \cdot ((n-r)^2)\} \le \max_{1 \le r \le n} \{a \times n + C \cdot (r-1)^2 + C \cdot ((n-r)^2)\}$$

Let$a \times n + C \cdot (r-1)^2 + C \cdot ((n-r)^2)$ as a formula call $F(r)$.

To solve the previous equation, we need to solve $F(r)$ first. Therefore, we made a differential action on this formula to get the value of r which has the biggest or smallest value of $F(r)$. The process is as follow:

$F'(r) = 2c \cdot (r-1) - 2a \cdot (n-r) = 0$ where $r = \ldots$

However, there is a quick way to check if the $F(r)$ is the biggest or smallest value by checking the graph of the formula is concave-up or concave-down. For the reason that we do the differential twice as follow:

$F''(r) = 2 \cdot C + 2 \cdot a > 0$ which shows the formula is concave-down.

Finally, we get that $F(r) = max\{F(1), F(n)\}$ because the graph is concave-down.

Then, we solve the formula as follow:

$F(r) = max\{a \cdot (r-1)^2 + a \cdot n, C \cdot (n-1)^2 + a \cdot n\} = C \cdot (n-1)^2 + a \cdot n$

because $c >> a$.

As a result, we get $T_W(n) \leq C \cdot (n-1)^2 + a \cdot n$ which means that $T_W(n) = O(n^2)$.

(c) The average-case complexity:

(1) $T_{AVG}(n) = O(n)$

(2) $T_{AVG}(n) = \frac{1}{n} \sum_{r=1}^{n} \{T_{AVG}(r-1)\}$

(3) $T_{AVG}(n) = \frac{1}{n} \sum_{r=1}^{n} \{T_{AVG}(n-r)\} = O((n-r)^2)$

$$T_{WORST}(n) = \frac{1}{n} \sum_{r=1}^{n} \{O(n) + T_{AVG}((r-1)) + O((n-r)^2)\} = O(n^2)$$

Since I know the worst-case time complexity is $O(n^2)$, I can just guess the average-case time complexity is $O(n^2)$, too, because the big O means upper bounder.