DSA Assignment

20 февраля 2022 г. 20:00

3.1.

1. $10n\log n + 500n + n^2 + 123$

By definition of big-O notation, we can find such constants $c > 0$ and $n_0 > 0$ such that for all $n \geq n_0$ $10n\log n + 500n + n^2 + 123 \leq c \cdot n^2$

Let $c = 5$: $10n(\log n + 50) + 123 \leq 4n^2$

Then, for $n \geq 140 = n_0$

$73004 \leq 78400$

Since $f(n) = 10n(\log n + 50) + 123$ is monotonically increasing function, $f(n) \leq 4n^2$ will be always true for all $n \geq 140$.

$$10n\log n + 500n + n^2 + 123 = \underline{\underline{O(n^2)}}$$

2. $n^{\frac{9}{2}} + 7n^4\log n + n^2$

By definition of big-O notation, we can find such constants $c > 0$ and $n > 0$ such that for all $n \geq n_0$ $n^{\frac{9}{2}} + 7n^4\log n + n^2 \leq c \cdot n^{\frac{9}{2}}$ $\quad \Big| : n^{\frac{9}{2}}$

$$1 + \frac{7\log n}{\sqrt{n}} + \frac{1}{\sqrt{n^5}} \leq c$$

Let us consider $f(n) = \frac{7\log n}{\sqrt{n}}$

$$\left.\begin{aligned} &f_{max} = \frac{14}{e} \text{ at } n = e^2; \\ &\frac{1}{\sqrt{n^5}} \leq 1 \quad \text{for } n \geq 1 \end{aligned}\right\} \Rightarrow$$

$$\Rightarrow 1 + \frac{7\log n}{\sqrt{n}} + \frac{1}{\sqrt{n^5}} \leq 2 + \frac{7\log n}{\sqrt{n}} \leq 2 + \frac{14}{e}$$

Then, for any $n \geq e^2 = n_0$; $n \in \mathbb{N}$

$$1 + \frac{7\log n}{\sqrt{n}} + \frac{1}{\sqrt{n^5}} \leq 2 + \frac{14}{e} \leq c \Rightarrow c = 8$$

$$n^{\frac{9}{2}} + 7n^4 \log n + n^2 = \underline{O(n^{\frac{9}{2}})}$$

3\. $6^{n+1} + 6(n+1)! + 24n^{42}$

By definition of big-O notation, we can find such constants $c > 0$ and $n > 0$ such that ~~for~~ all $n \geq n_0$ $6^{n+1} + 6(n+1)! + 24n^{42} \leq c \cdot n!$

Since $f(n) = n!$ is growing much faster than $g(n) = 6^{n+1}$ and $h(n) = 24n^{42}$, we can set $c = (n+2)(n+1) \Rightarrow c n! = (n+2)!$

Hence, for a very large $n > 0$, $n \in \mathbb{N}$, inequality $6^{n+1} + 6(n+1)! + 24n^{42} \leq c \cdot n! = (n+2)!$ will remain correct

$$6^{n+1} + 6(n+1)! + 24n^{42} = \underline{O(n!)}$$

3.2.1

```
function search(value) {
    for array in arrays {
        int l = 0, r = array.length;
        while (l < r) {
            int mid = (l + r) / 2;
            if (array[mid] < value) { l = mid + 1; }
            else if (array[mid] > value) { r = mid; }
            else { return true; }
        }
    }
    return false;
}
```

	cost	times
for array in arrays	c_1	n
int l = 0, r = array.length;	c_2	n
while (l < r)	c_3	$\frac{n}{2}$
int mid = (l + r) / 2;	c_4	1
if (array[mid] < value)	c_5	1
else if (array[mid] > value)	c_6	1
else { return true; }	c_7	1
return false;	c_8	1

Asymptotic complexity: $O(\log N)$

Worst-case: $O(N)$
(when the element is in the first or last index)

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