Hand Written HW 1

1106110110 郭景淳

---

1. Since we know that:

$$n^2 \in O(n^2), n\log n \in O(n\log n), 3 \in O(1)$$

Therefore, we can derive the following result:

$$n^2 + n\log n + 3 \in O(max(n^2, n\log n, 3)) \in O(n^2)$$

2. Since we know:

$$(n+1)^{n+1} = n^n \cdot (1+\frac{1}{n})^n \cdot (n+1)$$

where $(1+\frac{1}{n})^n$ is e, which is a constant when $n \to \infty$.

Therefore:

$$O((n+1)^{n+1}) = O(n^n \cdot e \cdot (n+1))$$

$$= O(n^n) + O(n) = O(n^{n+1})$$

Since $O(n^n) \neq O(n^{n+1})$, we can prove that $f(n) \in O(n^n) \iff f(n) \in O(n^{n+1})$ is **not true**.

3. $InsertionSort$:

Original Array : $1, 4, 2, 8, 5, 7, 6, 3$

$Step1$: 2nd element compare to 1st element.

$$1, 4, 2, 8, 5, 7, 6, 3$$

$Step2$: 3rd element compare to 1st to 2nd element.

$$1, 2, 4, 8, 5, 7, 6, 3$$

$Step3$: 4th element compare to 1st to 3rd element.

$$1, 2, 4, 8, 5, 7, 6, 3$$

$$.$$

$$.$$

$$.$$

When the last elemant is compared, $InsertionSort$ is complete.

$$1, 2, 4, 3, 4, 5, 6, 7, 8$$

$SelectionSort$:

$Step1$: Find the smallest number between 1st to 8th element, then swap with 1st element.

$Step2$: Find the smallest number between 2nd to 8th element, then swap with 2nd element.

$$.$$

$$.$$

$$.$$

After repeating the same step 7 times, $SelectionSort$ is complete.

$MergeSort$:

$Step1$: Split the array into two arrays.

$Step2$: If both arrays are sorted, then continue $Step3$. If not, do $MergeSort$ in both arrays.

$Step3$: Merge two array in order.

$MergeSort$ complete.

4. $MasterTheorem$
   - (a) $T(n) = 2T(n/2) + O(1)$

     From $MasterTheorem$, we know:

     $$a = 2, b = 2, f(n) = O(1)$$

     $$n^{\log_2 2} = n > f(n)$$

     It's $Case1$, therefore :

     $$f(n) = O(n^{1-1})$$

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

   - (b) $T(n) = 2T(\frac{n}{4}) + O(n^2)$

     From $MasterTheorem$, we know:

     $$a = 2, b = 4, f(n) = O(n^2)$$

     $$n^{\log_4 2} = n^{\frac{1}{2}} < f(n)$$

     It's $Case3$, therefore :

     $$f(n) = \Omega(n^{0.5+1.5})$$

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

- (c) $T(n) = 3T(n/\sqrt{2}) + O(n^4)$

  From $MasterTheorem$, we know:

  $$a = 3, b = \sqrt{2}, f(n) = O(n^4)$$
  $$n^{\log_{\sqrt{2}} 3} = n^{2\log_2 3} \approx n^{3.17} < f(n)$$

  It's $Case3$, therefore :

  $$f(n) = \Omega(n^{3.17+0.83})$$
  $$T(n) = \Theta(n^4)$$

5. - (a).

     - Algorithm $simp$

       The outer loop is $i = 1$ to $i = n$, run n times.

       The inner loop is $j = 1$ to $j = \sqrt{i}$, therefore for every i, inner loop only at most run $\sqrt{i}$ times.

       Therefore:

       $$\sum_{i=1}^{n} \sqrt{i} \approx \int_1^n \sqrt{x}dx = \frac{2}{3}(n^{\frac{3}{2}} - 1)$$

       Time Complexity is $O(n^{\frac{3}{2}})$.

     - Algorithm har

       The outer loop is $i = 1$ to $i = n$, run n times.

       The inner loop is $j = 1$ to $j = n/i$, therefore for every i, inner loop only at most run n/i times.

       Therefore:

       $$\sum_{i=1}^{n} \frac{n}{i} \approx n\ln(n)$$

       Time Complexity is $O(n\log n)$

   - (b).

     - Let's say there is a input n, and we want to find the total number of its factors, we can do prime factorization in n to achieve that. First, we can iterate from 1 to $n^{\frac{1}{3}}$, if we find a prime factor of n : p_i, then make n be divided by p_i until it can not be divided by p_i.

       After that, every p_i has a exponent e_i, we can:

       $$temp = (e_1 + 1)(e_2 + 1)\ldots.$$

for every p_i.

And then the remaining of n only contain at most 2 prime factors.

- *Prove: Let's say the remaining of n contain 3 prime factors P_1 , P_2 , P_3 , and suppose that they are all $> n^{\frac{1}{3}}$ because of the iteration from 1 to $n^{\frac{1}{3}}$. Then :*

$$P_1 \times P_2 \times P_3 > n$$

which is a contradiction, beacause P_1 , P_2 , P_3 should all be factors of n.

Therefore, n only contain 0 or 1 or 2 different prime factors that are $> n^{\frac{1}{3}}$.

- case1: 0 prime factor Only happen when remaining of $n = 1$. $Answer = temp$
- case2: 1 prime factor $Answer = temp \times (1+1)$
- case3: 2 same prime factors $Answer = temp \times (2+1)$
- case4: 2 different prime factors $Answer = temp \times (1+1) \times (1+1)$

- Since processing the cases part's time complexity is $O(1)$ .Finally, we can conclue that total Time Complexity is

$$O(n^{\frac{1}{3}})$$

- (c). Yes, the statement is true. Because $\sum_{k=1}^{n} k^{\frac{1}{3}}$ have to iterate from 1 to n, time complexity is O(n). Since $O(n) \in O(n \log n)$

$$\sum_{k=1}^{n} k^{\frac{1}{3}} \in O(n \log n)$$

6. - (a).

```
bool COOLTWO( vector<bool> x )
{
    for( int i=0; i < x.size(); i++ )
    {
        if( x[i] == true )
        {
            return false;
        }
    }
    return true;
}
```

   - (b).

     - So we know that $y = true$ only happen when x_1 to x_{2^n} are all $false$, otherwise $y = false$.

Algorithm COOL's idea is split the x array in half, and then look one half to see if it has any $x_i = true$ in it. If it actually have one $x_i = true$ in it, then retrun $false$, otherwise look at the other half.

COOL also resurse itself to see both half arrays, therefore when subarray contain only one x ($n = 0$), then return that x's value.

In conclusion, algorithm COOL is correct.

- By $MasterMethod$,

$$T(2^n) = 3^{0.5}T(\frac{2^n}{2}) + O(1)$$

$$O(1) < O(2^{n^{\log_2 3^{0.5}}})$$

we know that its case1:

$$T(2^n) = O(2^{n^{\log_2 3^{0.5}}})$$

$$T(2^n) = O(3^{n/2})$$