Lab01-Preliminary-Solution

VE281 - Data Structures and Algorithms, Xiaofeng Gao, TA: Qingmin Liu, Autumn 2019

∗ Please upload your assignment to website. Contact webmaster for any questions.

∗ Name:____________ Student ID:____________ Email: ________________

1. What is the time complexity of the following code?

```
1  // REQUIRES: an integer k
2  // EFFECTS: return the number of times that Line 12 is executed
3  int count(int k)
4  {
5      int count = 0;
6      int n = pow(2,k); // n=2^k
7      while (n>=1)
8      {
9          int j;
10         for (j=0;j<n;j++)
11         {
12             count += 1;
13         }
14         n /= 2;
15     }
16     return count;
17 }
```

Solution. Explanation 15′, correct answer 15′

For each $n = n_0$, the **for** loop repeats n_0 times.

In the **while** loop, $n = 2^k, 2^{k-1}, ..., 1$ respectively.

Therefore, the total number is

$$2^0 + 2^1 + ... + 2^k = 2^{k+1} - 1$$

,

Finally,

$$T(n) = O(n)$$

□

2. Given an array **nums** of n integers, are there elements a, b, c in nums such that $a+b+c = 0$? Write a program to find all unique triplets in the array which gives the sum of zero. Give your code as the answer. **Claim that the time complexity of your program should be less than or equal to $O(n^2)$.**

Examples: Input array [-1, 0, 1, 2, -1, -4], the solution is [[-1, 0, 1], [-1, -1, 2]]

Solution. Correct code 15′, explanation 15′

Please explain your design and fill in the following block:

```cpp
// REQUIRES: an integer array nums of size n
// EFFECTS: return a list of triplets, the sum of each triplet equals to 0.
#include <vector>
vector<vector<int>> findTriplet(vector<int>& nums, int n)
{
    vector<vector<int>> res;

    sort(nums.begin(),nums.end());
    for(int i = 0; i < n; ++i){
        if(i > 0 && nums[i] == nums[i-1]){
            continue;
        }

        int l = i + 1;
        int r = n - 1;

        while(l < r){
            while(l > i+1 && l < r && nums[l] == nums[l-1]){
                l++;
            }
            while(r < n-2 &&  l < r && nums[r] == nums[r+1]){
                r--;
            }
            if(l < r){
                int sum = nums[i] + nums[l] + nums[r];
                if(sum == 0){
                    vector<int> t;
                    t.push_back(nums[i]);
                    t.push_back(nums[l]);
                    t.push_back(nums[r]);
                    res.push_back(t);
                    l++;
                    r--;
                }else if(sum < 0){
                    l++;
                }else{
                    r--;
                }
            }
        }
    }

        return res;
    }
}
```

The first **for** loop repeats n times.

For each i, the first **while** loop repeats $n - i - 1$ times.

Thus the total number is

$$(n-1)+(n-2)+...+0=\frac{n(n-1)}{2}$$

Finally,

$$T(n)=O(n^2)$$

□

3. Equivalence Class

Definition 1 (o-Notation)**.** *Let $f(n)$ and $g(n)$ be functions from the set of natural numbers to the set of nonnegative real numbers. $f(n)$ is said to be $o(g(n))$, written as $f(n)=o(g(n))$, if*

$$\forall c>0.\exists n_0.\forall n\geq n_0.f(n)<cg(n).$$

An equivalence relation $\mathcal{R}$ on the set of complexity functions is defined as follows:

$$f\mathcal{R}g \text{ if and only if } f(n)=\Theta(g(n)).$$

A complexity class is an equivalence class of $\mathcal{R}$.

The equivalence classes can be ordered by $\prec$ defined as: $f\prec g$ iff $f(n)=o(g(n))$.

Example: $1\prec\log\log n\prec\log n\prec\sqrt{n}\prec n^{\frac{3}{4}}\prec n\prec n\log n\prec n^2\prec 2^n\prec n!\prec 2^{n^2}$.

Please order the following functions by $\prec$ and give your explanation:

$$(\sqrt{2})^{\log n},(n+1)!,ne^n,(\log n)!,n^3,n^{1/\log n}.$$

Solution. Correct answer $10'$, each explanation $6'$

$$n^{1/\log n}\prec(\sqrt{2})^{\log n}\prec n^3\prec(\log n)!\prec ne^n\prec(n+1)!$$

(a) $n^{1/\log n}\prec(\sqrt{2})^{\log n}$
Because $n^{1/\log n}=(2^{\log n})^{1/\log n}=2$ and $(\sqrt{2})^{\log n}=2^{1/2\log n}=\sqrt{n}$.
$2\prec\sqrt{n}$

(b) $(\sqrt{2})^{\log n}\prec n^3$
Because $\sqrt{n}\prec n^3$

(c) $n^3\prec(\log n)!$
Because by taking logs:

$$\begin{aligned}\log(\log n)! &= \log(\sqrt{2\pi\log n}(\frac{\log n}{e})^{logn}) \\ &= \Theta(\log n\log\log n) \\ \log(n^3) &= 3\log n\end{aligned}$$

And $\log\log n=\Omega(3)$

(d) $(\log n)! \prec ne^n$

Because

$$
\begin{aligned}
(\log n)! &= \sqrt{2\pi \log n}(\frac{\log n}{e})^{logn} \\
&= \Theta((\log n)^{\log n + 1/2} e^{-\log n}) \\
&= \Theta((\log n)^{\log n + 1/2} n^{-\log e})
\end{aligned}
$$

Thus, $(\log n)! \prec n^{\log \log n}$ while $n^{\log \log n} \prec 2^n \prec ne^n$

Finally we have $(\log n)! \prec ne^n$

(e) $ne^n \prec (n+1)!$

Because

$$
\begin{aligned}
n! &= \sqrt{2\pi n}(\frac{n}{e})^n \\
&= \Theta(n^{n+1/2} e^{-n}) \\
n! &\prec (n+1)!
\end{aligned}
$$

Thus $ne^n \prec n! \prec (n+1)!$

$\square$