算法分析与设计

实验 01

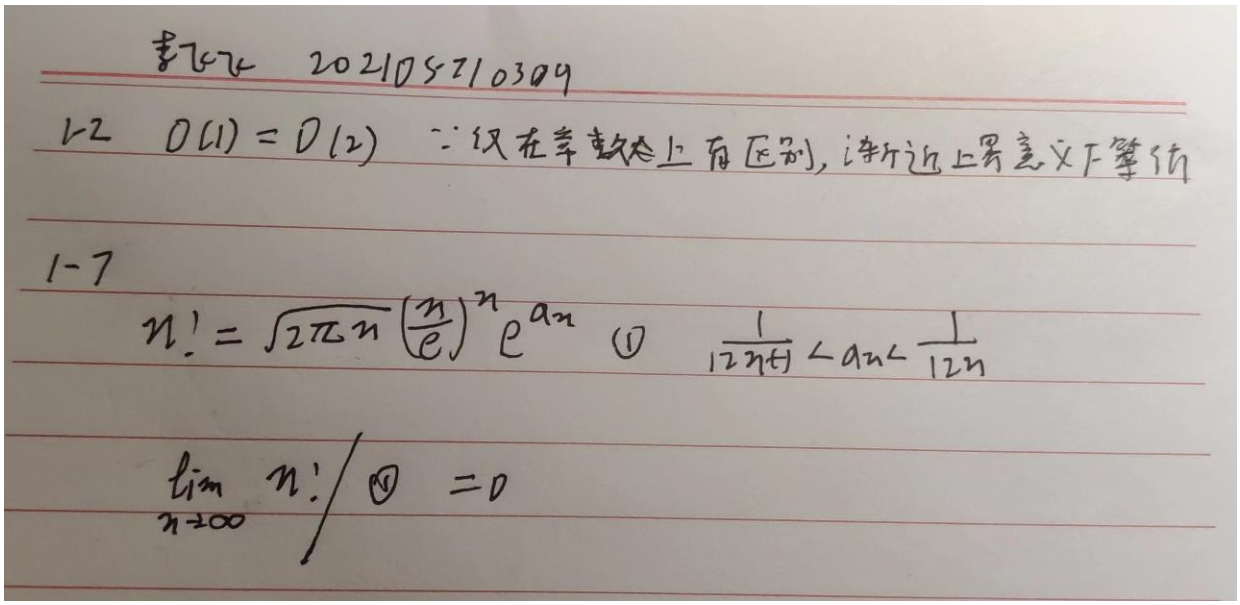
实验目的

- 掌握算法和程序的概念。
- 掌握算法复杂性的概念。
- 掌握算法复杂性的基本方法。
- 掌握递归与分治的基本思想。

实训内容

1、 算法分析题（P7）：1-2；
O（1）=O（2） 两者仅在常数上有区别，渐进上界意义下是等价的

2、 证明题（P7）：1-7
如图：



3、 算法设计题：最多约数问题 1-3(P8)。请分析自己所写代码的时间复杂度。

- 代码：（上交的程序已经将输入输出改成文件形式，这里只是为了方便展示）

Round 890 Div. 2 C > D_More_Wrong > D_More_Wrong.cpp > ...

```cpp
#include<iostream>
#include<algorithm>
#include<queue>
using namespace std;
#define ll long long
#define endl '\n'
#define pb push_back
#define fast ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
const int N=5e5+10;
ll n,m;
int ma,maCnt,serachRound; //最大约数,最大约数个数
vector<ll> ans;
int primes[9] = {2, 3, 5, 7, 11, 13, 17, 19, 23};
//当前素数编号 上个素数的个数 当前数 约束之和
void dfs(int u, int last, int p, int sum) {
    serachRound++;
    if(p>=m&&(sum>maCnt||(sum==maCnt))){
        if(sum>maCnt){
            maCnt=sum;
            ans.clear();
            ans.pb(p);
        }
        else ans.pb(p);
    }
    //可行性剪枝
    if (u==9) return;
    //最优性剪枝,不得高于上一个的频次
    for (int i = 1; i <= last; ++ i) {
        if (1ll*p * primes[u] > n) return;
        p *= primes[u];
        dfs(u + 1, i, p, sum * (i + 1));
    }
}
void solve(){
    cin>>m>>n;
    dfs(0, 30, 1, 1);
    cout<<"程序一共搜索了"<<serachRound<<"轮"<<endl;
    for(auto t:ans)cout<<t<<" ";
}
int main(){
   fast
   solve();
   return 0;
}
```

Round 890 Div. 2 C > D_More_Wrong > Tests > result.txt

```
Test 1 Passed

Input 1:
850 100000000

Expected Output :
程序一共搜索了803轮
91891800 73513440 98017920 86486400

Obtained Output :
程序一共搜索了803轮
91891800 73513440 98017920 86486400
```

样例：

```
Test 1 Passed

Input 1:
1 1000

Expected Output :
程序一共搜索了39轮
840

Obtained Output :
程序一共搜索了39轮
840
```

```
Test 1 Passed

Input 1:
841 1000

Expected Output :
程序一共搜索了39轮
960

Obtained Output :
程序一共搜索了39轮
960
```

- 做法及复杂度分析：

  首先，通过乘法原理求约数个数：

约数个数 $\sqrt{N}$

$$N = p_1^{a_1} * p_2^{a_2} * \ldots p_k^{a_k}$$

$$\text{约数个数} = (a_1 + 1) * (a_2 + 1) * \ldots (a_k + 1)$$

$$\text{约数之和} = (p_1^0 * p_1^1 + \ldots + p_1^{a_1}) * (p_2^0 * p_2^1 + \ldots + p_2^{a_2}) * \ldots * (p_k^0 * p_k^1 + \ldots + p_k^{a_k})$$

推导：乘法原理

1. 在 int 范围内最多 30 个指数（2^30）所以搜索空间十分优先
2. 在 int 范围内用到的素数最大 23（再乘更大的就爆 int 了）
3. 显然对于每个可能的质因数，为了使得约数更多，各个质因子的幂次一定单调递减，所以加剪枝

由于用的是搜索，所以不好计算复杂度，通过实际运行可知复杂度很低，开到 1e8 的数据也只搜了 803 次。