2025-2-8

一、蓝桥杯大纲梳理

蓝桥杯的赛制

OI赛制，有部分分，无赛中实时评测

n m 1e3 O(n^2)

1e6 O(n) O(nlogn)

15 20% 60%

相关的应试技巧：暴力骗分、对拍

O(nlogn)

A.cpp O(n^2)

B.cpp O(nlogn)

rand.cpp rand()

duipai.cpp

目前先介绍下思想，具体代码后面专门说

以及一些OI赛制下常见的坑 long long int

官方给的大纲

1. **枚举/模拟/基本算法**
2. **排序**

   归并、快排、堆排序重要一点，其他了解思想就好

   STL 的 sort 函数的运用
3. **搜索**

   bfs、dfs，以及对应的记忆化搜索，考得非常灵活

   其他双向BFS、启发式一类的在算法竞赛基本没出现过
4. **贪心**

   说几个理论可推的例子，不过比赛时大多数靠直觉
5. **二分**

   算法本身就是二分查找/二分搜索，难点其实是在单调性的发现和check函数的编写里

   1111111 11111

   O(n)

   ```
   while(lt <= rt){
      int mid = lt + (rt-lt)/2;
      if(check(mid))  lt = mid;
      else rt = mid-1;
   ```

}

nlogn

有时间就说下浮点数二分和三分

6. **动态规划**

线性DP

背包DP 0/1 多重背包

树形DP dp[i] = 1 + dp[son]

状压DP 3 000 001 010 011 state -->

数位DP（挺难的）

n --> O(1)

常见优化（最常用的是滚动数组DP，以及通过数学推式子优化。其他斜率优化、平行四边形优化之类的基本不考）

7. **数据结构**

STL 里数据结构的灵活运用 stack queue priority_queue set map vector等

常用的，单调栈、优先队列

链表在算法竞赛里一般不会写指针形式，简单说下链表思想的数组实现

int head = 2;

a[2] --> a[1] --> a[3] --> nullptr

a[1] = 3

a[2] = 1

a[3] = -1

并查集

trie树（这个也可以算字符串的内容）

树状数组、线段树

8. **数学**

质数判断、质数埃式筛、线性筛

n [1-n] O(nlogn) O(n)

取模、取余、因子一类的概念，对分数取模（模意义下的逆元）

1/2 1e9+7

快速幂 n^m O(m) --> O(logm)

排列组合，基本的数学概念 C(i, n) for(int i=1;i<=1e6);

组合数DP计算方法、取模的快速计算方法

简单的容斥有空也可以说下，结合二进制枚举

9. **字符串**

（如果想努力一下字符串的话把哈希和模板kmp搞懂就好，其他的很难很灵活）

哈希（以及双哈希、自然溢出一类的东西）

kmp n 母串 1e5 1e6　m 模式串 子串 1e5 O(n+m)

10. **图论（包括树上的算法）**

    树上的：

    基本的树遍历：前序中序后序层序遍历

    以及经典的根据前序中序还原一棵树之类的问题 比较 leetcode 风格

    DFS序，以及时间戳

    最近公共祖先（LCA）（倍增方法能会最好，树链剖分比较难）

    图上的：

    图的存储方式：邻接矩阵、邻接表（这个用得比较多） O(n^2) 1e5 1e5

    最短路径算法：bellman-ford/SPFA 负权边可以 但负权环不行、迪杰斯特拉n^2（以及它的堆优化nlogm）负权边、floyd n^3

    最小生成树算法：课内的 prim 和 kruskal，kruskal 会涉及到并查集

    拓扑排序 AOV

    差分约束 s u -w-> v　--> dist[v] <= dist[u]+w

    二分图匹配（匈牙利算法）

11. **杂项**

    计算几何：练练代码基本功就好，常见的，判断两线段平行垂直，点是不是在圆内之类的

    博弈：说下nim博弈 n 1 4

    高精度

二、算法竞赛的注意事项

1. 时间空间复杂度

1s 2s --> 1e5 --> O(nlogn) 1e7

1s 1e8 2e8

O(logn) 1e18

O(n) 1e7

O(n根号)

O(nlogn) 1e5 1e6

O(n^2) 1e3

O(n^3) 1e2

知道概念，知道怎么计算，了解常用stl容器和算法的复杂度

vector v

v.push_back()

string s;

s.substr(1,3) O(1)

sort() --> nlogn

知道怎么估计一道题允许的复杂度

卡常一类的概念

2. 提交题目后的反馈

（虽然蓝桥杯赛制不是赛时评测，可能刷题或者机考的时候有用）

AC WA TLE MLE RE CE

runtime error --> /0 nullptr

分别什么意思，以及一般怎么去处理

3. assert的技巧

对于 OI 赛制用得不多，可以用来 debug

举个代码的例子 断言

WA

assert(result<=n); --> RE

三、枚举/模拟/基本算法

1 排序

归并、快排、堆排序重要一点，其他了解思想就好

STL 的 sort 函数的运用

```
// 基本的int类型
int a[100100] = {0};
int n;
vector<int> v;

int mycmp(int x,int y){ // 降序
    return x > y;
}

int main(void)
{
    cin >> n;
    // 1.
    for(int i=1;i<=n;i++) cin >> a[i];
    sort(a+1,a+1+n,mycmp); // 不写mycmp默认升序

    // 2.
```

```
    for(int i=1;i<=n;i++){
        int x;
        cin >> x;
        v.push_back(x);
    }
    sort(v.begin(), v.end(), mycmp);
}

// 无默认排序的自定义类型
struct node{
    int fir,sec;
}a[100010];
int n;
int mycmp(node x, node y){
    if(x.fir == y.fir) return x.sec > y.sec;
    return x.fir > y.fir;
}
int main(void){
    cin >> n;
    for(int i=1;i<=n;i++){
        cin >> a[i].x >> a[i].y;
    }
    sort(a+1,a+1+n,mycmp);
}
```

讲下归并和快排的模板

归并:

```
#include <bits/stdc++.h>
using namespace std;
long long n;
int a[1000010] = {0};
int b[1000010] = {0};
// 假设有两个排序好的数列[l,mid] 和 [mid+1, r] 如何做合并
// a[] [l,r] --> b[] [l,r]
// 逆序对 O(n^2) --> O(nlogn)
void hb(int l,int mid,int r)
// xxxx yyyyy
// y
{
    int p = l,q = mid+1;
    int i = l;
    while(p <= mid && q <= r){
        if(a[p] <= a[q]){
            b[i] = a[p];
            p++;
        }
        else{
            b[i] = a[q];
            q ++;
        }
        i++;
    }
    while(p<= mid) {
```

```
        b[i] = a[p];
        i++, p++;
    }
    while(q <= r){
        b[i] = a[q];
        i++, q++;
    }
    for(int i=l;i<=r;i++) a[i] = b[i];
}

void mer(int l,int r) // []
{
    if(l==r) return;
    int mid = (l+r)/2;
    mer(l,mid);
    mer(mid+1,r);
    hb(l,mid,r);
}
int main(void)
{
    cin >> n;
    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
    mer(1,n);
    return 0;
}
```

快排：

```
void quicksort(int left, int right) { // [left, right]
    if (left >= right) return;

    int pivot = a[left];
    int i = left, j = right;

    while (i < j) {
        while (i < j && a[j] >= pivot) j--; // 从右向左找第一个小于 pivot 的数
        while (i < j && a[i] <= pivot) i++; // 从左向右找第一个大于 pivot 的数
        if (i < j) std::swap(a[i], a[j]); // 交换
    }
    std::swap(a[left], a[i]); // 交换基准值到最终位置

    quicksort(left, i - 1); // 递归排序左半部分
    quicksort(i + 1, right); // 递归排序右半部分
}
int main(void)
{
    cin >> n;
    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
    quick_sort(1,n);
    return 0;
}
```

讲下归并求逆序对的题

https://ac.nowcoder.com/acm/contest/21763/L

2 高精度

其实用到很少，理解一下思想，然后自己实现一遍，体会一下代码调通的过程就好

大数可以用vector存，用size()表示位数。也可以用数组，把位数放在 a[0], a[1]存最低位

对着模板代码过一遍，会**高精度的加减乘和大小比较**就可以了

sign

```
// C = A + B, A >= 0, B >= 0

vector<int> add(vector<int> &A, vector<int> &B)
{
    if (A.size() < B.size()) return add(B, A);

    vector<int> C;
    int t = 0;
    for (int i = 0; i < A.size(); i ++ )
    {
        t += A[i];
        if (i < B.size()) t += B[i];
        C.push_back(t % 10);
        t /= 10;
    }

    if (t) C.push_back(t);
    return C;
}

// C = A - B, 满足A >= B, A >= 0, B >= 0
vector<int> sub(vector<int> &A, vector<int> &B)
{
    vector<int> C;
    for (int i = 0, t = 0; i < A.size(); i ++ )
    {
        t = A[i] - t;
        if (i < B.size()) t -= B[i];
        C.push_back((t + 10) % 10); // 处理负数 t<0
        if (t < 0) t = 1;
        else t = 0;
    }

    while (C.size() > 1 && C.back() == 0) C.pop_back();
    return C;
}

// sub A B

// C = A * b, A >= 0, b >= 0
int a[20200]
vector<int> mul(vector<int> &A, int b)
{
    vector<int> C;
```

```
    int t = 0;
    for (int i = 0; i < A.size() || t; i ++ )
    {
        if (i < A.size()) t += A[i] * b;
        C.push_back(t % 10);
        t /= 10;
    }

    while (C.size() > 1 && C.back() == 0) C.pop_back();

    return C;
}
```

3 二进制枚举以及相关的位运算技巧

```
for(int i=0;i<(1LL<<n);i++){ // 20 30
    int state = i;
    for(int j=0;j<n;j++){ // j bit位
        if(state & (1LL << j)) // 说明是1
        else // 说明是0
    }
}
// & | ~ ^  << >>
n --> 0001101
1LL<<3 = 1000
1LL<<n = 10000 n个0  00000 --> 11111 n个1 2^n
1LL << j = 1 j个0
000000 0 000
```

排序

纯排序的模板，可以用来测试你写的排序算法 https://www.luogu.com.cn/problem/P1177

快排思想的应用 https://www.luogu.com.cn/problem/P1923

桶排思想应用 https://www.luogu.com.cn/problem/P1271

冒泡思想 https://www.luogu.com.cn/problem/P1116

排序函数需要思考的 https://www.luogu.com.cn/problem/P1012

高精度

高精度比较 https://www.luogu.com.cn/problem/P1781

高精度加法 https://www.luogu.com.cn/problem/P1601

高精度乘法 https://www.luogu.com.cn/problem/P1303

综合的高精度 https://www.luogu.com.cn/problem/P1009

二进制枚举

基本是模板的二进制枚举 https://leetcode.cn/problems/subsets/description/?envType=problem-list-v2&envId=bit-manipulation

和 https://leetcode.cn/problems/number-of-1-bits/description/?envType=problem-list-v2&envId=bit-manipulation

位运算练习 https://leetcode.cn/problems/add-binary/?envType=problem-list-v2&envId=bit-manipulation

可以用dfs 但这里也可以用二进制方法做 https://www.luogu.com.cn/problem/P1157

难一点的 https://leetcode.cn/problems/bitwise-and-of-numbers-range/description/?envType=problem-list-v2&envId=bit-manipulation