2025-2-22

上次的

1. 1 二进制倒数第二个 P1157

原因是题目要求对所有序列，按字典序排序输出

```
#include <bits/stdc++.h>
#include <iomanip>
using namespace std;
int n,r;
vector<vector<int>> res;
int mycmp(vector<int> x, vector<int> y){
  for(int i=0;i<x.size();i++){
    if(x[i]!=y[i]) return x[i] < y[i];
  }
}
int cal(int x){
    int res = 0;
    for(int i=0;i<21;i++){
        if(x & (1LL<<i)) res++;
    }
    return res;
}
signed main(void)
{
    cin >> n >> r;
    for(int i=0;i<(1LL<<n);i++){
        int state = i;
        if(__builtin_popcount(state)==r){
      vector<int> temp;
            for(int j=0;j<n;j++){
                if(state&(1LL<<j)){
          temp.push_back(j+1);
                }
            }
            res.push_back(temp);
        }
    }
  sort(res.begin(), res.end(), mycmp);
  for(int i=0;i<res.size();i++){
    for(int j=0;j<res[i].size();j++){
      cout << setw(3) << res[i][j];
    }
    cout << '\n';
  }
    return 0;
}
int n,r;
signed main(void)
{
    cin >> n >> r;
```

```
    for(int i=0;i<(1LL<<n);i++){
        int state = i;
        if(__builtin_popcount(state)==r){
            for(int j=0;j<n;j++){
                if(state&(1LL<<j)){
                    cout << setw(3) << j+1;
                }
            }
            cout << '\n';
        }
    }
    return 0;
}
```

2. 8 双指针比较基础的例子 P1102

首先TLE问题出在双指针声明的位置，这也是双指针两重循环，但线性复杂度的保障

然后再回来看一下问题的性质，如果想反过来从后往前扫，不能只修改两个指针移动的方向

最后，如果从后往前扫，在"避免三个指针有重叠"问题上处理有变化

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
int n,C;
int a[200020] = {0};
signed main(void)
{
    cin >> n >> C;
    for(int i=1;i<=n;i++) cin >> a[i];
    sort(a+1,a+1+n);
    // 对于每个B 找满足A = B+C 的A
    // 由于数组递增 从后往前遍历B, C不变 则需要寻找的A也是递减的
    int res = 0;
    int pos1 = n, pos2 = n; // 处理重复元素问题
    for(int i=n;i>=1;i--){
        int B = a[i];
        while(pos1 >=1 && B+C < a[pos1]) pos1--; // 直到 a[pos1] == B+C
        while(pos2 >=1 && B+C <= a[pos2]) pos2--; // 直到 a[pos2] < B+C
        // pos1 是最右合法的A pos2+1是最左一个合法的A [pos2+1, pos1]
        if(B+C==a[pos1] && B+C==a[pos2+1]){
            res += (pos1-pos2);
            if(pos2+1 <= i && i <= pos1) res--;
        }
    }
    cout << res;
    return 0;
}
```

3. 9 经典的P1638

（我也是两个指针都从左往右的？）

4. 13 的离散化

用区间合并的思路做也是可以的，并且对于这道题很简洁

讲一下这三个思路

```
// 模拟计数思想
#include <bits/stdc++.h>
using namespace std;
#define int long long
int l,n;
int a[10010] = {0}; // a[i]
signed main(void)
{
  cin >> l >> n;
  for(int i=0;i<=l;i++) a[i] = 1;

  while(n--){
    int lt,rt;
    cin >> lt >> rt;
    for(int i=lt;i<=rt;i++) a[i] = 0;
  }
  int res = 0;
  for(int i=0;i<=l;i++) res += a[i];
  cout << res << '\n';
  return 0;
}

// 差分
#include <bits/stdc++.h>
using namespace std;
#define int long long
int l,n;
int a[10010] = {0}; //
int pre[10010] = {0};
signed main(void)
{
  cin >> l >> n;

  while(n--){
    int lt,rt;
    cin >> lt >> rt;
    // for(int i=lt;i<=rt;i++) a[i] = 1;
    a[lt]++;
    a[rt+1]--;
  }
  for(int i=0;i<=l;i++){
    if(i==0) pre[i] = a[i];
    else pre[i] = pre[i-1] + a[i];
  }
  int res = 0;
```

```
  for(int i=0;i<=l;i++){
    if(pre[i]==0) res++;
  }
  cout << res << '\n';
  return 0;
}

// 离散化
#include <bits/stdc++.h>
using namespace std;
#define int long long

signed main(){
    int n,l;
    cin >> l >> n;

    // 输入区间[L,R+1)，并同时记录离散化坐标
    vector<pair<int,int>> intervals(n);
    vector<int> coords;
    for (int i = 0; i < n; i++){
        int L, R;
        cin >> L >> R;
        intervals[i] = {L, R};
        coords.push_back(L);
        coords.push_back(R + 1);
    }
    // 1 3 5 7 --> 1 2 3 4
    // 加入 0 和 l+1 这两个坐标也需要离散化
    coords.push_back(0);
    coords.push_back(l + 1);

    // 离散化 排序去重
    sort(coords.begin(), coords.end());
    coords.erase(unique(coords.begin(), coords.end()), coords.end());
    int m = coords.size();

    // 差分数组
    vector<int> diff(m+10, 0);
    for(auto &seg : intervals){
        int L = seg.first, R = seg.second;
        // 对于闭区间 [L, R]，在差分数组中更新 [L, R+1)
        // 1 3 5 7 --> 1 2 3 4 对于区间[3,5] 需要得到idxL = 2, idxR = 3
        int idxL = lower_bound(coords.begin(), coords.end(), L) - coords.begin();
        int idxR = lower_bound(coords.begin(), coords.end(), R + 1) -
coords.begin();
        diff[idxL] += 1;
        if(idxR < diff.size()){
            diff[idxR] -= 1;
        }
    }

    // 前缀和
    vector<int> cover(m+10, 0);
    cover[0] = diff[0];
    for (int i = 1; i < m; i++){
        cover[i] = cover[i-1] + diff[i];
```

```
    }

    // 计算被移除的树的数量 和上面差分的思路一样
    int removed = 0;
    for (int i = 0; i < m - 1; i++){
        if (cover[i] > 0){
            removed += (coords[i+1] - coords[i]);
        }
    }

    // 原始马路上树的总数为 l+1
    int total = l + 1;
    int ans = total - removed;

    cout << ans << "\n";
    return 0;
}
```

新内容

二、贪心

1. 基于"性价比"的 比较基础的贪心

两个例子 直观说明下贪心的思想

1. https://www.luogu.com.cn/problem/P2240

N堆金币，每堆总重量和总价值已知，可以只拿一堆的一部分。

有一个承重为T的背包，问最多可以带走多少价值的金币。

```
#include<bits/stdc++.h>
using namespace std;
struct node{
    double w;
    double v;
    double p;
}a[105];
int n;
double sum,c;
inline bool cmp(node a,node b){
    return a.p >b.p;
}
int main(){
    cin>>n>>c;
    for(register int i=1;i<=n;++i){
        cin>>a[i].w>>a[i].v;
        a[i].p=a[i].v/a[i].w;
    }
    sort(a+1,a+n+1,cmp);
    for(register int i=1;i<=n;++i){
```

```
        if(c>=a[i].w){
            c-=a[i].w;
            sum+=a[i].v;
        }
        else{
            sum+=c*a[i].p;
            break;
        }
    }
    printf("%.2f",sum);
    return 0;
}
```

2. https://www.luogu.com.cn/problem/P1223

   n个人排队接水，每人需要Ti时间，怎么排队使得平均等待时间最短

```
#include <bits/stdc++.h>
using namespace std;
int n;
struct node
{
    int id;
    int time;
}a[1005];

int mycmp(node x, node y)
{
    if(x.time == y.time) return x.id < y.id;
    else return x.time < y.time;
}

int main(void)
{
    cin >> n;
    for(int i=1;i<=n;i++)
    {
        a[i].id = i;
        cin >> a[i].time;
    }
    sort(a+1,a+n+1,mycmp);
    double ans = 0.0;
    for(int i=1;i<=n;i++)
    {
        cout << a[i].id << ' ';
        for(int j=1;j<i;j++) ans += a[j].time;
    }
    cout << endl;
    printf("%.2f\n",ans/n);

    return 0;
}
```

3. 上一题的换皮题 https://www.luogu.com.cn/problem/P1090

```
 #include<bits/stdc++.h>
using namespace std;
priority_queue<int, vector<int>, greater<int>> q; // heap
priority_queue<node, vector<node>, cmp> q; // 大根堆

int main()
{
    int n,sum=0;
    cin>>n;
    for(int i=1;i<=n;i++)
    {
        int x;
        cin >> x;
        q.push(x); // logn
    }

    while(q.size()>=2)
    {
        int cur1 = q.top();
        q.pop();
        int cur2 = q.top();
        q.pop();
        int merge = cur1 + cur2;
        sum += merge;
        q.push(merge);
    }
    cout<<sum;
    return 0;
}
```

2. 其他一些经典贪心

1. https://www.luogu.com.cn/problem/P1803

给定 n 个区间，问最多能选择几个，使得它们都不重合

一种竞赛经常出现的直觉

```
#include <bits/stdc++.h>
using namespace std;
int n;
int ans = 1;
struct node
{
    int start;
    int end;
}a[1000005];

int mycmp(node x, node y)
{
    return x.end < y.end; // 核心: 按照右端点排序区间
}

int main(void)
```

```
{
    cin >> n;
    for(int i=1;i<=n;i++) cin >> a[i].start >> a[i].end;
    sort(a+1,a+1+n,mycmp);
    int idx = 2;
    int tail = a[1].end;
    while (idx <= n)
    {
        if(a[idx].start >= tail) // 可选的区间
        {
            ans++;
            tail = a[idx].end;
            idx++;
        }
        else // 不可选 跳过
        {
            idx++;
        }
    }
    cout << ans << endl;
    return 0;
}
```

2. 分组 https://www.luogu.com.cn/problem/P1094

n个物品分组，每组最多两个物品，并且每组的 sum 不能超过w。输出最少的分组数目

```
#include <bits/stdc++.h>
using namespace std;
int w,n;
int a[100005];//降序
int ans;

int mycmp1(int x, int y)
{
    return x > y;
}

int mycmp2(int x, int y)
{
    return x < y;
}

int main(void)
{
    cin >> w >> n;
    for(int i=1;i<=n;i++)
    {
        cin >> a[i];
    }
    sort(a+1,a+1+n,mycmp1);
    ans = n; // 合并一对少一个
    int left = 1;
    int right = n;
    while (left < right)
```

```
    {
        if(a[left] + a[right] <= w)
        {
            ans--;
            left ++;
            right --;
        }
        else
        {
            left ++; // 大的自成一组
        }
    }
    cout << ans << endl;
    return 0;
}
```

3. 贪心算法的理论验证

1. 考虑上面排队接水的问题，怎么从理论上说明这种方法的正确性？

```
考虑只有两个人i,j的顺序,假设T[i] < T[j]
先i后j，时间T[i]+T[i]+T[j]  T[i]
先j后i，时间T[j]+T[j]+T[i]  T[j]
做差，后面-前面 = T[j]-T[i] > 0
说明第一种方法耗时更短
```

2. 一个更复杂的例子 https://www.luogu.com.cn/problem/P1080

   一个国王和n个大臣，排成一排，每个人左右手各有一个整数。

   每位大臣获得的金币数分别是：排在该大臣前面的所有人的左手上的数的乘积除以他自己右手上的数，然后向下取整得到的结果。

   找到一个排列方法，使得获得奖赏最多的大臣，所获奖赏尽可能的少

   输出这个"获得奖赏最多的大臣"获得的奖赏数量

   证明过程讲一下frankchenfu题解

（这个题满分涉及高精度，如果想自己实现代码的话可以跳过这个）

三、二分

1. 基本的二分模板

整数二分

可以用这题测试 https://www.luogu.com.cn/problem/P2249

```
#include <bits/stdc++.h>
using namespace std;
#define int long long

int n,m;
```

```
int a[1000010] = {0};

signed main(void)
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n >> m;
    for(int i=1;i<=n;i++) cin >> a[i];
    while(m--){
        int num;
        cin >> num;
        int low = 1, high = n;
        while(low < high){
            int mid = (low + high) / 2;
            // int mid = low + (high-low)/2;
            if(a[mid] < num) low = mid + 1;
            else high = mid;// a[mid] >= num
        }
        if(a[high] == num) cout << high << ' ';
        else cout << "-1 ";
    }
    return 0;
}
```

了解下 lower_bound 和 upper_bound 函数

```
vector<int> a;
1 3 3 3 5
0 1 2 3 4 iterator a.end()
int pos = lower_bound(a.begin(), a.end(), 7) - a.begin();
```

浮点二分

可以用这个测试 https://www.luogu.com.cn/problem/P1163

```
#include <bits/stdc++.h>
using namespace std;
int tot, one, times;
int main(void)
{
    cin >> tot >> one >> times;
    double l = 0;
    double r = 1000;
    while(r-l >= 0.0001)
    {
        double e,f;
        double mid = (l+r) / 2;
        e = mid/100 + 1;
        f = 0;
        for(int i=1;i<=times;i++) O(nlogn)
        {
            f += one/e;
            e = e*(mid/100+1);
```

```
        }
        if(f > tot) l = mid + 0.00001; 0.1 --> log10000
        else r = mid - 0.00001;
    }
    if(l<=0) printf("0.0\n");
    else printf("%.18lf\n",r);

    return 0;
}
```

带check的形式

```
假设是 xxxxxyyyyyy 单调，需要找到最左边的y
int check(int num){ //
    if(num is y) return true;
    else return false;
}
while(low < high){
    int mid = (low+high)/2;
    if(check(mid)) high = mid;
    else low = mid+1;
}
if(check(high)) return high;


假设是 xxxxxyyyyyy 单调，需要找到最右边的x
int check(int num){ O(n) nlogn
    if(num is x) return true;
    else return false;
}
while(low < high){
    int mid = (low+high)/2;
    if(check(mid)) low = mid;
    else high = mid-1;
}
if(check(low)) return low;
else return -1;
```

比较经典的跳石子题 https://www.luogu.com.cn/problem/P2678

二分答案，check该答案在题干约束下是否合法（给定的M值）

```
#include <bits/stdc++.h>
using namespace std;
long long l,m,n,ans = -1;
long long a[100000];
long long b[100000];
// 0 2 11 14 17 21 25
int check(int mid){ // mid
    int now = 0;
    int s = 0;
```

```
    for(int i=1;i<=n+1;i++) // O(n)
    {
        if(a[i] - a[now] < mid) s++;
        else now = i;
    }
    return s;
}
// 1 2 3 4 5 .. x.. ..l

int main(void)
{
    cin >> l >> n >> m;
    for(int i=1;i<=n;i++) cin >> a[i];
    a[n+1] = l;
    long long left = 0;
    long long right = l;
    while(left <= right)
    {
        long long mid = (left + right) /2;
        if (check(mid)<=m) // 使得最短跳跃距离为mid时  需要移除的最少石头数量
        {
            ans = mid;
            left = mid + 1;
        }
        else right = mid - 1;
    }
    cout << ans << endl;
    return 0;
}
```

习题

贪心

性价比思路 https://www.luogu.com.cn/problem/P1208

局部最优的思路 https://www.luogu.com.cn/problem/P1478 和https://leetcode.cn/problems/best-time-to-buy-and-sell-stock-ii/submissions/255345889/

有思维成分的贪心1 https://www.luogu.com.cn/problem/P3817

有思维成分的贪心2 https://www.luogu.com.cn/problem/P1106

难想一点的分组贪心 https://www.luogu.com.cn/problem/P4447

二分

之前双指针的A-B 也可以用二分求解 可以用lower_bound和upper_bound简化 https://www.luogu.com.cn/problem/P1102

下面两题和跳石子类似 二分答案的思想

https://www.luogu.com.cn/problem/P1824

https://www.luogu.com.cn/problem/P2440

https://www.luogu.com.cn/problem/P1873

二分之前 需要处理数据 https://www.luogu.com.cn/problem/P1182

浮点二分的应用 https://www.luogu.com.cn/problem/P1577