

2025-2-15

上次的

1.

这题我试了下，选 pivot 用 $a[l]$ 和 $a[r]$ 都会超时，用 $a[(l+r)/2]$ 可以通过

传统快排超时是对于某些数据，算法退化到了 $O(n^2)$ 的时间复杂度（举个例子）

例如原本是完全逆序排序的，递归深度就从 $O(\log n) \rightarrow O(n)$

关于递归算法复杂度的求法可以参考 **主定理 (Master Theorem)**

主定理: $T[n] = aT[n/b] + f(n)$

其中 $a \geq 1$ and $b > 1$ 是常量 并且 $f(n)$ 是一个渐近正函数，为了使用这个主定理，您需要考虑下列三种情况：

Case1: 如果 $f(n) = O(n^{\log_b a - \epsilon})$ 对于某个常量 $\epsilon > 0$ 成立, 那么 $T[n] = O(n^{\log_b a})$

Case2: 如果 $f(n) = O(n^{\log_b a})$, 那么 $T[n] = O(n^{\log_b a} \log n)$

Case3: 如果 $f(n) = O(n^{\log_b a + \epsilon})$ 对于某个常量 $\epsilon > 0$ 成立, 并且 $af(n/b) \leq cf(n)$ 对于某个常量 $c < 1$ (n 足够大) 成立, 那么 $T[n] = O(f(n))$

例如快排，形如 $T(n) = 2T(n/2) + O(n)$

$a = 2, b = 2 \rightarrow \log_b(a) = 1 \Rightarrow d \rightarrow O(n \log n)$ case 2

比较常用的改进方法是用 `rand()` 函数取 $[l, r]$ 区间内随机值，或取 $a[l], a[r], a[(l+r)/2]$ 三数的中值

算法竞赛中一般不会卡这个，用三数取中的方法就可以

↩

2. 排序模板 ↩

一开始选最左端。 ↩

R202255214 记录详情

编译语言	代码长度	用时	内存
C++14 (GCC 9) O2	610B	3.63s	1.62MB

测试点信息

测试点	状态	用时	内存
#1	AC	4ms/556.50KB	
#2	AC	25ms/628.00KB	
#3	TLE	1.30s/916.00KB	
#4	TLE	1.30s/1.62MB	
#5	TLE	1.30s/708.00KB	

测试数据下载

测试点 #3: 下载数据

洛谷免费提供该记录中一个非AC的输入输出数据下载；部分题目因为版权等原因，不开放数据下载。

所属题目: P1177 【模板】排序

评测状态: Unaccepted

评测分数: 40

提交时间: 2025-02-09 19:04:28

后来我把这道题改成了归并排序，能否说一下为什么咱们讲的传统快排超时，如果首次选的中枢恰好是最小最大值，有什么改进办法不。 ↩

2.

3. 快排思想的应用 <https://www.luogu.com.cn/problem/P1923>
4. 这题我有小问题，我一开始用的快排和归并都超时（两个案例，原因不明）

洛谷 / 评测记录 / 评测详情

R202449494 记录详情

测试点信息 源代码

测试点信息

#1	AC	12ms/748.00KB	#2	AC	12ms/768.00KB	#3	AC	14ms/680.00KB	#4	AC	715ms/18.32MB	#5	AC	841ms/19.40MB
----	----	---------------	----	----	---------------	----	----	---------------	----	----	---------------	----	----	---------------

n 在 $5e6$ 范围 标准快排（或者直接用sort）时间复杂度 $O(n\log n)$ 比较危险（虽然更多会用 $1e7$ 的数据量卡掉 $n\log n$ 算法）

讲一个快排优化的 $O(n)$ 思路 分块

```
// 0-indexed
int x[5000005], k;
void qsort(int l, int r) // 对于区间[l, r]
{
    int i = l, j = r, mid = x[(l+r)/2];
    do
    {
        while(x[j] > mid)
            j--;
        // 此时j指向从右到左第一个<=mid的值
        while(x[i] < mid)
            i++;
        // 此时i指向从左到右第一个>=mid的值
        // 交换i j指向的值
        if(i <= j)
        {
            swap(x[i], x[j]);
            i++;
            j--;
        }
    } while(i <= j);
    //快排后数组被划分为三块: l<=j<=i<=r
    // 此时 [l, j] 段和 [i, r] 段待排序, [j, i] 段已经排序好
    if(k <= j) qsort(l, j); //在左区间只需要搜左区间
    else if(i <= k) qsort(i, r); //在右区间只需要搜右区间
}
```

```

else //如果在中间区间直接输出
{
    printf("%d",x[j+1]);
    exit(0);
}
}

```

主定理: $T[n] = aT[n/b] + f(n)$

其中 $a \geq 1$ and $b > 1$ 是常量 并且 $f(n)$ 是一个渐近正函数, 为了使用这个主定理, 您需要考虑下列三种情况:

Case1: 如果 $f(n) = O(n^{\log_b a - \epsilon})$ 对于某个常量 $\epsilon > 0$ 成立, 那么 $T[n] = O(n^{\log_b a})$

Case2: 如果 $f(n) = O(n^{\log_b a})$, 那么 $T[n] = O(n^{\log_b a} \log n)$

Case3: 如果 $f(n) = O(n^{\log_b a + \epsilon})$ 对于某个常量 $\epsilon > 0$ 成立, 并且 $af(n/b) \leq cf(n)$ 对于某个常量 $c < 1$ (n 足够大) 成立, 那么 $T[n] = O(f(n))$

用上面主定理的方法证明就是 $T(n) = 2T(n/2) + O(n)$

形如 $T(n) = T(n/2) + O(n)$

$a = 1, b = 2 \rightarrow \log_b(a) = 0$; case 3

这个题直接用 nth_element 函数也可以, 底层也是 $O(n)$ 的

3.

8. 高精度比较 <https://www.luogu.com.cn/problem/P1781>
 字符串怎么使用 sort(有点不会) 尤其是涉及到数字的这种
 后来自己写的另一种做法
 字符串怎么使用 sort(有点不会) 尤其是涉及到数字的这种
 后来自己写的另一种做法

通过 / 评测记录 / 评测详情

R202699218 记录详情

测试点信息 源代码

测试点信息

#1	#2	#3	#4	#5
AC	AC	AC	AC	AC
4ms/564.00KB	3ms/600.00KB	4ms/680.00KB	3ms/564.00KB	3ms/532.00KB

关于字符串 sort 的算法

```
#include<iostream>
```

```

#include<cstring>
#include<algorithm>
using namespace std;
struct node
{
    string x; //装票数
    int num; //装号数
    int lenx; //装票数的位数 s.size();
}s[25];

// 核心是cmp函数 其他就是直接调用sort
// 需要按票数降序排序
bool cmp(node a,node b)
{
    // 首先位数不同一定是位数多的在前
    if(a.lenx>b.lenx) return 1; //前一个比后一个位数多，不交换
    // 如果位数相同，字典序大的在前
    // 字典序是这样：两个string从前往后比，第一个不同的字符
    // 例如 "543" 和 "534" 因为 '4'>'3' 所以 "543" > "534"
    if(a.lenx==b.lenx&& a.x>b.x) return 1; //位数相同，但前一个按字典序排列比后一个大，也不交换。
    return 0; //剩下情况均要交换。
}

int main()
{
    int n;
    cin>>n;
    for(int i=1;i<=n;i++)
    {
        cin>>s[i].x;
        s[i].num=i; //存号数
        s[i].lenx=s[i].x.size(); //存票数的位数
    }
    sort(s+1,s+n+1,cmp); //排序
    cout<<s[1].num<<endl; //输出首位答案即可，注意先输出号数
    cout<<s[1].x<<endl; //再输出票数
    return 0;
}

```

4.

P1271 【深基9.例1】选举学生会

[提交答案](#)[加入题单](#)[复制题目](#)

题目描述

[复制 Markdown](#) [展开](#) [进入 IDE 模式](#)

学校正在选举学生会成员，有 n ($n \leq 999$) 名候选人，每名候选人编号分别从 1 到 n ，现在收集到了 m ($m \leq 2000000$) 张选票，每张选票都写了一个候选人编号。现在想把这些堆积如山的选票按照投票数字从小到大排序。

输入格式

输入 n 和 m 以及 m 个选票上的数字。

输出格式

求出排序后的选票编号。

输入输出样例

输入 #1

[复制](#)

输出 #1

[复制](#)

```
5 10
2 5 2 2 5 2 2 2 1 2
```

```
1 2 2 2 2 2 2 2 5 5
```

计数思想比较常见，提一下

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

int n,m,x;
int cnt[1000] = {0}; //
int main(void)
{
    cin >> n >> m;
    for(int i=1;i<=m;i++)
    {
        cin >> x;
        a[x]++;
    }
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=a[i];j++) cout << i << ' ';
    }
    return 0;
}
```

今天新内容

三、枚举/模拟/基本算法

4 前缀和、差分

一维前缀和的理论和代码

可以 $O(1)$ 获取区间和

```
S[i] = a[1] + a[2] + ... a[i] sum  
a[l] + ... + a[r] = S[r] - S[l - 1] [l,r]
```

最基本的模板 <https://www.luogu.com.cn/problem/P8218>

```
#include <bits/stdc++.h>  
using namespace std;  
#define int long long  
int n,m;  
int a[100010] = {0};  
int pre[100010] = {0};  
int back[100010] = {0}; // back[i]  
signed main(void)  
{  
    cin >> n;  
    for(int i=1;i<=n;i++) cin >> a[i];  
    for(int i=1;i<=n;i++) pre[i] = pre[i-1]+a[i];  
    cin >> m;  
    while(m--){  
        int lt,rt;  
        cin >> lt >> rt; // a[lt] - a[rt] [3,4]  
        // x y z t u  
        int res = pre[rt] - pre[lt-1];  
        cout << res << '\n';  
    }  
    return 0;  
}
```

一维差分的理论和代码

可以 $O(1)$ 实现一次区间加

```
int a[100010] = {0};  
int diff[100010] = {0}; // diff[i] = a[i]-a[i-1];  
// 0, 1, 3, 2  
// 0, 1, 1  
// 0, 1, 2  
给区间 [l, r] 中的每个数加上 c: B[l] += c, B[r + 1] -= c
```

模板题 <https://www.luogu.com.cn/problem/P2367>

```
#include <bits/stdc++.h>
```

```

using namespace std;
#define int long long
int n,p;
int a[500010] = {0};
int diff[500010] = {0};
int pre[500010] = {0};
signed main(void)
{
    cin >> n >> p;
    for(int i=1;i<=n;i++) cin >> a[i];
    // diff[1] = a[1];
    for(int i=1;i<=n;i++) diff[i] = a[i]-a[i-1];

    while(p--){
        int x,y,z;
        cin >> x >> y >> z; // [x,y] +z
        diff[x] += z;
        diff[y+1] -= z;
    }

    for(int i=1;i<=n;i++) pre[i] = pre[i-1] + diff[i]; // pre[]

    int mmin = 1e12;
    for(int i=1;i<=n;i++) mmin = min(mmin, pre[i]);
    cout << mmin << '\n';
    return 0;
}

```

二维前缀和的理论和代码

可以 $O(1)$ 获取二维区间和，一个矩形内的 sum

$S[i, j]$ = 第 i 行 j 列格子左上部分所有元素的和
 以 $(x1, y1)$ 为左上角， $(x2, y2)$ 为右下角的子矩阵的和为：
 $S[x2, y2] - S[x1 - 1, y2] - S[x2, y1 - 1] + S[x1 - 1, y1 - 1]$

比较模板的 <https://www.luogu.com.cn/problem/P1719>

$O(n^2 * n^2) \rightarrow O(n^4)$

```

#include <bits/stdc++.h>
using namespace std;
#define int long long
int n,m;
int a[130][130] = {0};
int pre[130][130] = {0};
signed main(void)
{
    cin >> n;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            cin >> a[i][j];
    for(int i=1;i<=n;i++){

```

```

        for(int j=1;j<=n;j++){
            pre[i][j] = pre[i-1][j] + pre[i][j-1] - pre[i-1][j-1] + a[i][j];
        }
    }
    // o(n^2)
    int mmax = -1e18;
    for(int x1=1;x1<=n;x1++)
        for(int y1=1;y1<=n;y1++)
            for(int x2=x1;x2<=n;x2++)
                for(int y2=y1;y2<=n;y2++)
                    mmax=max(mmax,pre[x2][y2]-pre[x2][y1-1]-pre[x1-1][y2]+pre[x1-1][y1-1]);
    cout << mmax;
    return 0;
}

```

二维差分的理论和代码

给以(x1, y1)为左上角, (x2, y2)为右下角的子矩阵中的所有元素加上c:

$S[x1, y1] += c$, $S[x2 + 1, y1] -= c$, $S[x1, y2 + 1] -= c$, $S[x2 + 1, y2 + 1] += c$

模板题 <https://www.luogu.com.cn/record/62934676>

```

#include <bits/stdc++.h>
using namespace std;
#define int long long
int n,m;
int a[1005][1005] = {0};
int pre[1005][1005] = {0};
signed main(void)
{
    cin >> n >> m;
    for(int t=1;t<=m;t++)
    {
        int x1,x2,y1,y2;
        cin >> x1 >> y1 >> x2 >> y2;
        // 给以(x1, y1)为左上角, (x2, y2)为右下角的子矩阵中的所有元素加上1
        a[x1][y1] += 1;
        a[x2 + 1][y1] -= 1;
        a[x1][y2 + 1] -= 1;
        a[x2 + 1][y2 + 1] += 1;
        // for(int i=x1;i<=x2;i++)
        //     for(int j=y1;j<=y2;j++)
        //         a[i][j]++;
    }
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            pre[i][j] = a[i][j] + pre[i-1][j] + pre[i][j-1] - pre[i-1][j-1];
        }
    }
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=n;j++) cout << pre[i][j] << ' ';
    }
}

```



```

        cout << endl;
    }
    return 0;
}

```

一个思维量更大的题:

<https://www.luogu.com.cn/problem/P4552>

5 离散化

适用的情况

处理范围大, 但数据量不大的整数数组

在 $1e9$ 范围里面, 有 $1e5$ 个数字, 我们需要把这些数字映射到数组上的 $1e5$ 个位置

代码怎么写

```

#include <bits/stdc++.h>
using namespace std;
#define int long long
int l,m;
int a[222][2] = {0};
int b[222][2] = {0};
vector<int> v;
map<int,int> mp; // <key,value> <key,value> logn
map<int,pair<int,int>> mp;
map<int,string> mp;

// 1 100 100 10000 1000000000
// 1 2 2 3 4
// <1,1> <2,100> <3,10000>,
// <1,1> <100,2> <10000,3>
signed main(void)
{
    cin >> l >> m;
    for(int i=1;i<=m;i++){
        int x;
        cin >> x;
        a[i] = x;
        v.push_back(x);
    }
    sort(v.begin(), v.end());
    for(int i=0;i<v.size();i++){
        mp[a[i]] = i+1;
        inv_mp[i+1] = a[i];
    }
    for(int i=1;i<=n;i++) b[i] = mp[a[i]];

    return 0;
}

```

可以练习的题

<https://www.luogu.com.cn/problem/P1496>

6 双指针

利用题目的一些性质，把看上去需要 $O(n^2)$ 复杂度的问题转化成 $O(n)$ 线性复杂度

比较基础的例子 <https://www.luogu.com.cn/problem/P1102>

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
int n,C;
int a[200020] = {0};
// B+C = A BB aaaaAAAAAZ
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 从前往后遍历B C不变 则需要寻找的A也是递增的
    int res = 0;
    int pos1 = 1, pos2 = 1; // 处理重复元素问题
    for(int i=1;i<=n;i++){
        int B = a[i]; // B+C=A
        while(pos1 <= n && B+C > a[pos1]) pos1++;
        // B+C=A n+1
        while(pos2 <= n && B+C >= a[pos2]) pos2++;
        // pos1 是第一个合法的A pos2-1是最后一个合法的A
        if(B+C==a[pos1] && B+C==a[pos2-1] && pos1 > 1) res += (pos2-pos1);
    }
    cout << res;
    return 0;
}
```

很经典的 <https://www.luogu.com.cn/problem/P1638>

选择一个区间 $[a,b]$ 使得区间内包含所有在这个数组中出现过的数字 并且使区间 $[a,b]$ 长度最短

l, r

```
#include <bits/stdc++.h>
using namespace std;
int n,m,cnt;
int a[1000005] = {0};
int vis[2005] = {0};
int l,r,ll,rr,ans;
int main(void)
{
    cin >> n >> m;
    for(int i=1;i<=n;i++) cin >> a[i];
```

```

l = r = cnt = 1;
// [l,r] 为当前选择的区间 cnt为当前区间内画的种类数
// vis[i] 为第i个画家的画在区间中出现的次数
vis[a[l]] = 1;
ans = n+1; // 用于维护最小区间长度 先设成类似mmax的值
while(l<=r && r<=n)
{
    if(cnt == m) // 如果区间中包含了所有种类的画 试图移动左边的指针
    { //[l,r]
        if(ans>r-l+1) // 出现更优答案（更短区间）时更新答案
        {
            ans = r-l+1;
            // [ll,rr] 表示目前最优的答案
            ll = l;
            rr = r;
        } // l --> l+1
        vis[a[l]]--;
        if(vis[a[l]]==0) cnt--;
        l++;
    }
    else // 如果区间中没有包含所有种类的画，右边指针继续向后移动
    {
        r++;// a[r];
        vis[a[r]]++;
        if(vis[a[r]]==1) cnt++;
    }
}
cout << ll << ' ' << rr << endl;
return 0;
}

```

习题

前缀和/差分

二位前缀和应用 <https://www.luogu.com.cn/problem/P2004>

从加法拓展到乘法 并考虑后缀和 <https://leetcode.cn/problems/product-of-array-except-self/description/?envType=problem-list-v2&envId=prefix-sum>

不限于用来输出区间和的一维前缀和应用 <https://leetcode.cn/problems/removing-minimum-number-of-magic-beans/description/?envType=problem-list-v2&envId=prefix-sum>

思维量比较大的差分 <https://www.luogu.com.cn/problem/P4552>

离散化

<https://ac.nowcoder.com/acm/contest/20960/1010>

涉及区间合并处理 <https://www.luogu.com.cn/problem/P1496>

双指针

比较模板的 <https://ac.nowcoder.com/acm/contest/20960/1014>

A-B 换皮题 并且比那个简单一点 (<https://leetcode.cn/problems/two-sum-ii-input-array-is-sorted/description/?envType=problem-list-v2&envId=two-pointers>)

画展换皮题 <https://ac.nowcoder.com/acm/contest/20960/1015>

可以用map可以用单调队列（还没说这个）也可以用双指针的 <https://www.luogu.com.cn/problem/P3029>

关于子串的 <https://leetcode.cn/problems/permutation-in-string/description/?envType=problem-list-v2&envId=two-pointers>

“性质”比较难发掘的 <https://leetcode.cn/problems/container-with-most-water/description/?envType=problem-list-v2&envId=two-pointers>