2025-2026 秋《数据结构与算法》第三次实验题解

助教-陈孙一硕

Difficulties:

- Easy: A,B
- Medium-Easy: C,D,E
- Hard: F,G

A 修理牧场

哈夫曼树模版题。时间复杂度 $O(n \log n)$。

```
#include <iostream>
#include <queue>
#include <vector>
using namespace std;
#define int long long
int n, x, wpl = 0, tmp1, tmp2;
struct cmp {
    bool operator()(const int &x1, const int &x2) {
        return x1 > x2;
    }
};
priority_queue<int, vector<int>, cmp> q;
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    cin >> n;
    for (int i = 1; i <= n; i++)
```

```
        cin >> x, q.push(x);
    while (q.size() > 1) {
        tmp1 = q.top(), q.pop();
        tmp2 = q.top(), q.pop();
        wpl += (tmp1 + tmp2);
        q.push(tmp1 + tmp2);
    }
    cout << wpl << endl;
    return 0;
}
```

B 二叉查找树的构建及序列化

先把第一个结点作为根节点，后序的结点一次插入即可。

```
#include<iostream>
using namespace std;
#define int long long
int n,x;
struct node{
    int val;
    node* lchild=NULL,*rchild=NULL;
};
void ist(node* root,int v){
    node* tmp=root,*add=new node;
    add->val=v;
    while (tmp!=NULL){
        if (v==tmp->val) return;
        else if (v>tmp->val){
            if (tmp->rchild==NULL) tmp->rchild=add;
            else tmp=tmp->rchild;
        }else{
            if (tmp->lchild==NULL) tmp->lchild=add;
            else tmp=tmp->lchild;
        }
    }
```

```
}
void preorder(node* nd){
    if (nd==NULL) {cout<<0<<" ";return;}
    cout<<nd->val<<" ",preorder(nd->lchild),preorder(nd->rchild);
}
signed main(){
    cin>>n>>x;
    node* root=new node;
    root->val=x;
    for (int i=2;i<=n;i++) cin>>x,ist(root,x);
    preorder(root);
    return 0;
}
```

C 部落冲突

二分答案模版题。先将绿洲按左端点的 x 坐标升序排序，计算总面积. 然后二分断点，计算断点左侧的面积并与总面积的一半相比较，最后返回答案即可。

```
#include<iostream>
#include<algorithm>
using namespace std;
#define int long long
struct sq{
    int x,y,w,h;
};
bool cmp(sq l1,sq l2){
    return l1.x<l2.x;
}
int n,all=0;
sq q[101];
signed main(){
    cin>>n;
    for (int i=0;i<n;i++) cin>>q[i].x>>q[i].y>>q[i].w>>q[i].h,all+=q[i].w*q[i].h;
    sort(q,q+n,cmp);
    int left=q[0].x,right=q[n-1].x+q[n-1].w,mid,total=0;
```

```
    while (left<=right){
        mid=(left+right)/2,total=0;
        for (int i=0;i<n;i++){
            if (q[i].x+q[i].w<=mid) total+=q[i].w*q[i].h;
            else if (q[i].x<mid) total+=(mid-q[i].x)*q[i].h;
            else break;
        }
        if (total==(all+1)/2) break;
        else if (total>(all+1)/2) right=mid-1;
        else left=mid+1;
    }
    cout<<left<<endl;
    return 0;
}
```

D 奇偶平衡子段

首先构造前缀和数组，遇到奇数 +1，遇到偶数-1，那么我们就可以知道每个前缀包含的奇数和偶数数量之差，进而可以 $O(1)$ 查询每个子段的奇数和偶数数量之差。

对于每个值，查询前缀数组在这个位置之前出现的，值相同且下标最小的项，取下标差更新答案即可。维护可以使用 set。当然，线段树和树状数组也可以。

```
#include<iostream>
#include<iomanip>
#include<vector>
#include<string>
#include<cmath>
#include<algorithm>
#include<set>
#include<map>
#include<unordered_map>
#include<unordered_set>
#include<queue>
#include<stack>
#include<fstream>
#define int long long
```

```
using namespace std;
const int N=2e6+101;
int a[N];
int pre[N];
void solve(){
    int n; cin>>n;
    for (int i=0;i<=n+1;i++) pre[i]=0;
    for (int i=1;i<=n;i++) cin>>a[i];
    for (int i=1;i<=n;i++){
        pre[i]=pre[i-1];
        if (a[i]&1) pre[i]++;
        else pre[i]--;
    }
    set<pair<int,int>> st;//fi:val se:idx
    st.insert({0,0});
    int ans=-1;
    for (int i=1;i<=n;i++){
        auto it=st.lower_bound({pre[i],-1});
        if (it!=st.end()&&it->first==pre[i]) ans=max(ans,i-it->second);
        st.insert({pre[i],i});
    }
    cout<<ans<<endl;
}
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int T=1; cin>>T;
    while (T--) solve();
    return 0;
}
```

E　先輩からのムチ、愛をこめて

这是上学期弘深班期末考试的第三题。

首先用堆进行暴力维护可以获得 15 分的暴力分。但这样的时间复杂度是 $O(\sum a_i)$ 的，我们考虑优化。

- 如果 a_i 的最大值超过了剩余所有值的和，那么最优策略显然是每次都选中这个值和剩余的任意一个值。最后再将这个值归零。那么答案就是这个最大值。
- 否则，我们按照上述暴力的策略进行操作，一定不会有剩余（或者最后剩一个 1），答案为 $\frac{\text{sum}+1}{2}$。

```
#include<iostream>
using namespace std;
#define int long long
int n,lst[1000005],mx=0,sum=0;
signed main(){
    cin>>n;
    for (int i=1;i<=n;i++) cin>>lst[i],sum+=lst[i],mx=max(mx,lst[i]);
    if (sum-mx<=mx) cout<<mx<<endl;
    else cout<<(sum+1)/2<<endl;
    return 0;
}
```

F 乘 x 乘

这道题是去年计算机学院期末考试的一道编程题，整个计算机学院的学生中仅有 1 人 AC，得到 9 分以上的学生不超过 10%.

容易想到的方法是，先用题目提示中的方法封装一个结构体，然后用堆去模拟. 但由于 K 的规模可以达到 10^9，这样做显然是不可行的.

因此我们考虑更优的做法. 由于最小值一直在乘 q，也就是呈指数型增长. 那么直观上，很容易感受到，在操作次数足够多之后，最初数组中 a 的值几乎对排序的结果没有影响了. 如果某次操作的时候我们发现，堆顶的元素乘以 q 之后，超过了原来堆中最大的元素，那么接下来的每一次操作，优先队列的队头出队，乘以 q 之后再入队，一定会在队尾. 这样就构成了一个循环，直到操作次数耗尽为止.

考虑最坏的情况，n 的规模达到 2×10^4，序列中只有一个 10^9，其他数全部为 $1, q = 2$. 这种情况下，达到上面所说的那种情况，消耗的时间是最多的. 由于 $2^{30} \approx 10^9$，所以每个数的操作次数约为 30 次，总操作次数可以控制在 3×10^5 之内，可以在 $1s$ 的时间内运行通过.

达到上述情况后，问题就变得很简单了，先计算 $left = K\%n$，那么前 $left$ 个数被操作的次数为 $K/n + 1$，剩余的数被操作的次数为 K/n，此时花费 $O(n)$ 的时间计数即可.

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

```
#define int long long
int n,d,K,cnt[200005];
int ksm(int a,int b){
    int ans=1;
    while(b>0){
        if(b&1) ans*=a;
        a=a*a,b>>=1;
    }
    return ans;
}
struct node{
    int a=0,k=0,idx=0;
    bool operator < (const node& nd) const{
        if (k>=nd.k){
            if (a*ksm(d,k-nd.k)==nd.a) return idx>nd.idx;
            return a*ksm(d,k-nd.k)>nd.a;
        }else{
            if (a==nd.a*ksm(d,nd.k-k)) return idx>nd.idx;
            return a>nd.a*ksm(d,nd.k-k);
        }
    }
};
priority_queue<node,vector<node>> q;
node mx;
signed main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    cin>>n>>d>>K;
    int x;node nd;
    for (int i=1;i<=n;i++){
        cin>>x,nd.a=x,nd.k=0,nd.idx=i;
        q.push(nd),mx=min(mx,nd);
    }
    while (K--){
        nd=q.top(),q.pop();
```

```
        nd.k++,cnt[nd.idx]++,q.push(nd);
        if (nd<mx) break;
    }
    int lft=K%n;
    while (lft--) nd=q.top(),q.pop(),cnt[nd.idx]+=K/n+1;
    while (!q.empty()) nd=q.top(),q.pop(),cnt[nd.idx]+=K/n;
    for (int i=1;i<n;i++) cout<<cnt[i]<<" ";
    cout<<cnt[n]<<endl;
    return 0;
}
```

G 落叶归根

完全三叉树有个特性，下标为 x 的结点，从左到右三个子结点的编号依次为 $3x-1, 3x, 3x+1$. 我们担心的无非是删除某个结点为根的子树时，这棵子树已经残缺了，或者这个结点早就不在了. 因此，考虑用一个 map（也就是 std 里的 son）来记录以下标为 x 的结点为根的子树已经被删除了多少.

用一个 cnt 数组，去预处理深度为 k 的完全三叉树所包含的结点数量. 每当要删除一个结点时，先计算这个节点所在的层数，然后要删除的数量就是：以这个结点为根的完全子树的节点数减去 son[x]. 然后向上递归删除. 如果递归到某个结点 x 时，$son[x]$ 已经和以这个结点为根的子树的节点个数相等了，说明这个结点早就被删除了，也就是此次删除实际上是无效的，那么此时立即退出即可.

这道题的时间和空间卡的都比较严格. 空间要省着点用，因为被卡常导致超时的话可以尝试一下换成 unordered_map。

```
#include <iostream>
#include <map>
#define int long long
using namespace std;
map<int, int> son;
int cnt[30], h[30], p[30], t, n, m;
void dfs(int x, int k) {
    if (h[n - k + 1] == son[x]) return ;
    son[x] += t;
    if (k > 1) dfs((x + 1) / 3, k - 1);
}
```

```
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n >> m;
    int k = 1;
    for (int i = 1; i <= n; ++i) {
        cnt[i] = k;     // 每一层的数
        k *= 3;
    }
    for (int i = 1; i <= n; ++i) h[i] = h[i - 1] + cnt[i];
    for (int i = 1; i <= m; ++i) {
        int x, k = 1; cin >> x;
        for (int j = 1; j <= n; ++j) {
            if (h[j - 1] < x && x <= h[j]) {k = j; break;}
        }
        t = h[n - k + 1] - son[x];
        if (t != 0) dfs(x, k);
        cout << (h[n] - son[1]) << '\n';
    }
    return 0;
}
```