2025-3-9

上次的

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

   注意结束状态下所有盒子中糖果数量 >= 0

```
for(int i=1;i<n;i++)
{
    if(a[i-1]+a[i]>x)
    {
        ans+=a[i-1]+a[i]-x;
        a[i]-=a[i-1]+a[i]-x;
    }
```

```
#include <bits/stdc++.h>
using namespace std;
int n,x;
int a[10000005];
long long  ans = 0;
int main(void)
{
    cin >> n >> x;
    for(int i=1;i<=n;i++) cin >> a[i];
    for(int i=1;i<n;i++)
    {
        if(a[i] + a[i+1] > x)
        {
            int add = a[i]+a[i+1]-x;
            ans += add;
            // 考虑减少哪个值 尽量减a[i+1]而不是a[i]
            if(add > a[i+1])
            {
                a[i+1] = 0;
                a[i] = a[i]-(add-a[i+1]);
            }
            else a[i+1] -= add;
        }
    }
    cout << ans << endl;

    return 0;
}
```

2. 二分等号和边界问题

   （用2.22的md说一遍）

while条件的边界问题可以分类讨论一下 如果low = high 且这个值合法会怎样（死循环），不合法又会怎样（结果不对）

3. 跳石子 https://www.luogu.com.cn/problem/P2678

```
#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
// 固定mid 求它最少需要移除多少个石头
// 1 2 3 4 5 6 7
// 0 0 1 2 2 3 3
// 其实是给上面一行求下面一行的过程
// 具体求法是贪心思想
// 如果存在一跳a[last]到a[i]的距离小于mid，我们就移除石头a[i]
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;
}

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;
    // 单调性在于 要想使得mid越大 需要移除的石头越多
    // 我们需要找到移除数量 = m时，最大的mid
    // 1 2 3 4 5 6 7
    // 0 0 1 2 2 3 3
    // 对于 m = 2 可以达到mid，即跳跃间隙达到5，最后输出的答案是5
    while(left < right)
    {
        long long mid = (left + right+1) /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
        {
            left = mid;
        }
        else right = mid - 1;
    }
    cout << left << endl;
    return 0;
}
```

4. 分组的二分 https://www.luogu.com.cn/problem/P4447

```
#include <stdio.h>
#include <iostream>
#include <algorithm>
#define inf 199999999
using namespace std;
int n,a[100001],b[100001],f[100001],s,l,r,mid;
//a[i]用来表示第i个队员的能力值，把每个组的人数放入f数组，b数组能表示某组最大的能力值[题解大佬的]
//f[i]是第i组的人数
//b[i]是第i组末尾人的能力值
inline bool check(int x)
{
    register int i,j,position(0),point(0),minx;//position表示当前是第几个组
    for(i=1;i<=n;i++) // 遍历每个人
    {
        point=0,minx=inf;
        for(j=1;j<=position;j++)//循环现在所有的组 试图找到a[i]可以接在后面的组中 人数最少的
        {
            if(f[j]<minx && b[j]+1==a[i])//如果能力值递增，也就可以组队，这个就是排序的重要性；并且队伍长度要尽量短
            {
                point=j;//记录位置
                minx=f[j];//记录人数最少的组
            }
        }
        if(point==0)//a[i]不能直接接在某个组后面
        {
            f[++position]=1;//再建立一个小组
            b[position]=a[i];//记录能力值
        }
        else//a[i]可以接在point组后面
        {
            b[point]=a[i];//point组末尾人的能力值
            f[point]++;//point组人数++
        }
    }
    for(i=1;i<=position;i++)//最后再扫一遍，position的意义就是能分成几个组，所以到position即可
    {
        if(f[i]<x)//如果第i组人数少于x return false
        {
            return 0;
        }
    }
    return 1; // 否则 人数最少的组确实>=x
}
signed main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    register int i;
```

```
    cin>>n;
    for(i=1;i<=n;i++)
    {
        cin>>a[i];
    }
    sort(a+1,a+n+1);//将每个人的能力值从小往大排序 意义在于 组成每个队的时候都是从能力值小往大添加人
    l=1,r=n;
    // 人数最少的组人数越少 答案越可行  xxxxx[x]yyyy x是可行的 y是不可行的 需要找到[]的那个x
    while(l<=r)//二分答案
    {
        int mid=(l+r)>>1;
        if(check(mid)) // check函数：人数最少的组为mid个人 是否可行
        {
            s=mid;
            l=mid+1; // 可行了 看它的右边
        }
        else
        {
            r=mid-1; // 不可行 看它的左边
        }
    }
    cout<<s<<endl;
    return 0;
}
```

5. 切绳子浮点二分的做法 https://www.luogu.com.cn/problem/P1577

   N 个绳子，长度分别为 L_i，需要切割出 K 跳相同的绳子，长度最多多少?

```
#include <bits/stdc++.h>
using namespace std;
int n,k;
double a[10005];
bool check(double mid)
{
    long long ans = 0;
    // 贪心 能切就切
    for(int i=1;i<=n;i++)
    {
        long long t = int(a[i]/mid); //a[i]最多能切t个长度为mid的
        if(ans+t>=k) return true;
        ans += t;
    }
    return false;
}
int main(void)
{
    cin >> n >> k;
    double low = 0;
    double high,mid;
    for(int i=1;i<=n;i++)
    {
        cin >> a[i];
        high = max(high,a[i]); // high <= 原本最长的绳子
```

```
    }
    // 二分答案 答案越小越可能成立 xxxx[x]yyyy 找[x]
    while(low+1e-10<high) // high - low <= 1e-10
    {
        mid = (low+high)/2;
        if(check(mid)) low = mid; // check：看能不能切出来k条长度为mid的绳子
        else high = mid;
    }
    printf("%.4f\n",low);
    return 0;
}
```

新内容

四、搜索

1. 基本DFS

之前用二进制枚举法做的组合输出 https://www.luogu.com.cn/problem/P1157

用dfs就不会有后面要手动排字典序的问题

```
#include<bits/stdc++.h>
using namespace std;
int r,a[100],n;
// 2^n n=30  n! 10 15
void dfs(int k){//搜索第k个数
    int i;
    if(k==r+1){ // k=4
        for(i=1;i<=r;i++){
            cout<<setw(3)<<a[i];//输出，场宽为三
        }
        cout<<endl;
        return ;//回到前一层
    }
    for(i=a[k-1]+1;i<=n;i++){ // 2 - n
        a[k]=i; // i
        dfs(k+1);//直接进行下一次调用
    }
}
int main()
{
    cin>>n>>r; // n=5 r=3
    dfs(1);
    return 0;
}
```

经典八皇后 https://www.luogu.com.cn/problem/P1219

给一个n*n的棋盘，放 n 个棋子，同一行只能有一个，同一列只能有一个，对角线只能有一个

```
#include <bits/stdc++.h>
using namespace std;
int a[15][15] = {0};
int n,c=0;
int yi[50],xjiany[50],xjay[50];

void out(int& c)
{
    if(c>=3) c++; // 超过3个解不用输出解的内容
    else
    {
        for(int i=1;i<=n;i++)
        {
            for(int j=1;j<=n;j++)
            {
                if(a[i][j] == 1) cout << j << ' ';
            }
        }
        cout << endl;
        c++;
    }
}

void zhanlin(int x, int y)
{
    a[x][y] = 1;
    yi[y] = 1;
    xjiany[x-y+n] = 1;
    xjay[x+y] = 1;
}

void fangqi(int x, int y)
{
    a[x][y] = 0;
    yi[y] = 0;
    xjiany[x-y+n] = 0;
    xjay[x+y] = 0;
}

int isok(int x, int y) // 检测[x][y]位置上能不能放棋子
{
    if(a[x][y]==0 && yi[y]==0 && xjiany[x-y+n]==0 && xjay[x+y]==0) return 0;
    else return 1;
}

void dfs(int x) // 第x行放哪一列
{
    if(x==n+1) out(); //输出答案
    else
    {   // 放第x行的棋子的过程
        for(int i=1;i<=n;i++)
        {
            if(isok(x,i)==0) //能放在[x][i]
            {
                zhanlin(x,i); // 在[x][i]放
                dfs(x+1);     // 搜索第i+1行
```

```
                fangqi(x,i);  // 回溯
            }
        }
    }
}

int main(void)
{
    cin >> n;
    dfs(1); // 放第1行的棋子
    cout << c << endl; //输出解的个数
    return 0;
}

// T
int main(void)
{
    // cin >> T;
    T = 1;
    while(T--){
        solve();
    }
}
```

2. 基本BFS

经典马走棋盘 https://www.luogu.com.cn/problem/P1443

n * m棋盘，马的起点 [x] [y]，计算马到棋盘任意位置要走几步

注意vis数组和复杂度分析

```
#include <bits/stdc++.h>
using namespace std;
int f[500][500]; // n*m <= 1e5 MLE 9e18
bool vis[500][500]; // O(n*m)
queue< pair<int,int> > q;
const int dx[8] = {-2,-1,1,2,2,1,-1,-2};
const int dy[8] = {1,2,2,1,-1,-2,-2,-1};
int main(void)
{
    int n,m,x,y;
    cin >> n >> m >> x >> y;
    memset(f,-1,sizeof(f));// memset(f,0,sizeof(f)); 00000
    memset(vis,false,sizeof(vis));
    q.push(make_pair(x,y));
    vis[x][y] = true;
    f[x][y] = 0; // <s,t>
    while(q.size())
    {
        int xx = q.front().first;
        int yy = q.front().second; // q.top() O(1) deque
        q.pop();
        //  break;
```

```
        for(int i=0;i<8;i++)
        {
            int u=xx+dx[i], v=yy+dy[i];
            if(u<1||u>n||v<1||v>m||vis[u][v]) continue;
            vis[u][v] = true;
            f[u][v] = f[xx][yy] + 1;
            q.push(make_pair(u,v));
        }
    }
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=m;j++)
        {
            printf("%-5d",f[i][j]);
        }
        cout << endl;
    }
    return 0;
}
```

电梯 https://www.luogu.com.cn/problem/P1135

广搜 O(n)的做法

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

typedef struct{
    int floor;
    int step;
}QE;
queue<QE> que;
int n,a,b;
int s[1000];
int t[1000];

int main(void)
{
    cin >> n >> a >> b;
    for(int i=1;i<=n;i++) cin >> s[i];
    QE e1,e2;
    e1.floor = a;
    e1.step = 0;
    t[a] = 1;
    que.push(e1);
    while(que.size())
    {
        e2 = que.front();
        que.pop();
        if(e2.floor == b) break;
        int now;
        now = e2.floor + s[e2.floor];
        if(now<=n && t[now]==0)
        {
            e1.floor = now;
```

```
            e1.step = e2.step+1;
            t[now] = 1;
            que.push(e1);
        }
        now = e2.floor - s[e2.floor];
        if(now>0 && t[now]==0)
        {
            e1.floor = now;
            e1.step = e2.step+1;
            t[now] = 1;
            que.push(e1);
        }
    }
    if(e2.floor == b) cout << e2.step << endl;
    else cout << -1 << endl;
    return 0;
}
```

深搜有O(n^2)的做法 也可以试试

```
#include<bits/stdc++.h>
using namespace std;
int n,a,b,k[201],dis[201];
void dfs(int node,int step){
    dis[node]=step;//一定可以更新
    int v=node-k[node];
    if(1<=v&&step+1<dis[v]/*可以更新在搜索*/)//下
        dfs(v,step+1);
    v=node+k[node];
    if(v<=n&&step+1<dis[v])//上
        dfs(v,step+1);
    return;
}
int main(){
    memset(dis,0x3f,sizeof(dis)); //
    cin>>n>>a>>b;
    for(int i=1;i<=n;i++)
        cin>>k[i];
    dfs(a,0);// 距离
    cout<<(dis[b]==0x3f3f3f3f?-1:dis[b]);
    return 0;
}
```

3. 一些剪枝技巧

最简单的剪枝可以参考 n个数选k个，和 = C的问题

可以先排序，搜到还剩t个数时，如果当前的sum + t*(剩余的最小数)已经 > C就可以剪枝

1 3 5 7 8

条件剪枝 https://www.luogu.com.cn/problem/P1036

n 个整数，和一个整数k。从n个数选k个相加，得到的和为质数的种类数

（指不考虑顺序的四元组的种类数）

```
#include <bits/stdc++.h>
using namespace std;
int n,k;
int a[21];

bool isprime(int n)
{
    if (n<2) return false;
    for(int i=2;i<=sqrt(n);i++)
        if (n%i==0) return false;
    return true;
}

// 还剩几个数要选
// 当前已选的数的总和
// 可以选择数的范围
int rule(int left_num,int _sum,int start,int end)
{
    if (left_num==0) return isprime(_sum);
    int sum = 0;
    for(int i=start;i<=end;i++)
    {
        // 不剪枝是 sum += rule(left_num-1, _sum+a[i], 0, n-1);
        // 现在这一层选了a[i]
        // 下一次选的数一定是 i 之后的，否则会重复
        sum += rule(left_num-1, _sum+a[i], i+1, end);
    }
    return sum;
}

int main(void)
{
    cin>> n >> k;
    for(int i=0;i<n;i++)
        cin >> a[i];
    cout << rule(k,0,0,n-1);
    return 0;
}
```

剪枝 https://www.luogu.com.cn/problem/P1120

n 个一样长的木棍，随意砍成几段，直到每段的长都不超过 50。找出原始木棍的最小可能长度。

```
#include <bits/stdc++.h>
using namespace std;
int n,tot = 0,maxn = 0,minn = 70;
int a[70];
// 还应该拼几根 目前正在拼的这根拼了多少 长度 剩余木棍的长度最大值
void dfs(int lt, int now, int target, int high)
{
    if(lt==0) // 拼完了
    {
        cout << target;
        exit(0);
    }
```

```
    if(now==target) // 拼好一根
    {
        dfs(lt-1,0,target,maxn);
        return;
    }
    for(int i=high;i>=minn;i--) // 从长到短枚举长度
    {
        if(a[i]&&i+now<=target) // 长度为i的还有，且拼上去长度不超过原本木棍的长度
        {
            a[i]--;
            dfs(lt,now+i,target,i);
            //
            a[i]++; // 回溯
            if(now==0 || now+i==target) break; // 这根拼完了
        }
    }
    return;
}
int main(void)
{
    cin >> n; // 当前木棍数量
    int temp;
    for(int i=1;i<=n;i++)
    {
        cin >> temp; // 当前木棍
        if(temp<=50)
        {
            maxn = max(maxn,temp); // 维护最大值，最小值和总长度
            minn = min(minn,temp);
            tot += temp;
            a[temp]++;               // 维护每种长度有几根
        }
    }
    temp = tot>>1; //  tot/2
    // i  tot/i
    for(int i=maxn;i<=temp;i++) // 枚举原本木棍的长度
        if(tot%i==0) dfs(tot/i,0,i,maxn); //如果是tot的因子才是可能的长度
    cout << tot; // tot
    return 0;
}
```

迭代加深搜索 https://www.luogu.com.cn/problem/P1763

给定a,b，需要用一堆形如 1/x 的数之和表示 a/b，且x不能相同。

所有可行的方案中，找加数数量最少的；个数相同时，找最小的分数最大的

这个问题涉及一些比较难的剪枝，主要想用来说明迭代加深搜索

0 pts

考虑进行 DFS。设 $\frac{a}{b}=\sum_{i=1}^{p}a_i(2\le i)$，由于 a_i 单调递增，所以可以每次调用 DFS 函数时将 $\frac{a}{b}$ 减去 $\frac{1}{i}$，直到 $\frac{a}{b}=0$，即 $a=0$。

但这样操作的时间复杂度过高，对于 $\frac{2}{3}$ 这样的小数据都会超时，因此考虑引入 IDDFS。

在 DFS 中引入一个递归深度变量 limit，并在主函数中进行多次 DFS，每次将 limit 的值加一，直到搜索到合法的答案为止（在本题中指搜索到至少一组 a_i）。下面是形式化代码：

```
void dfs(int lim, /* 你想传入的参数 */) {
    if (lim == 0) {
        // 记录答案
        return;
    }
    for (int i = preans + 1; i <= maxans; i++) {
        if (ok(i)) dfs(lim - 1, /* 更新后的答案 */);
    }
}
```

【1】本题大框架：迭代加深搜索（IDDFS）

看到 $1<a<b<1000$，可以猜测分数的个数不会很多，考虑搜索。

那么怎么搜？因为我们不能确定最少分数的个数（这是我们首要要求的），可以考虑枚举分数的个数再进行搜索。

这相当于每次限制了搜索树的深度，当确定一个深度发现了解，深度就不会再加深。

这种每次限制了深度的搜索就叫**迭代加深搜索（IDDFS）**（又称 IDS）。

我们明显感觉到，每次限制深度为 dep，那么搜索深度为 $dep+1$ 时，搜索深度 dep 对应的搜索树又会被重新搜一遍。浪费了一些时间。

所以这种搜索有什么优势呢？

如果我们不限定深度，那么我们可能花大力气搜出了一个答案但远没达到最优解（分数的个数超过正解），此时再及时回溯已经很难了（搜索树深度增加一层增加的节点数是指数级的），浪费的时间会更多。

由于问题的第一关键字是分数的个数

问题转化为，找到 dep 个x，使得它们的 1/x 之和 = a/b

如果在较小的深度已经找到解，不用再考虑更大的深度

否则可能一开始搜出来分数个数很多的可行解

```
const int N = 11,INF = 1e7;//N为层数上限，INF为分母最大值
int dep = 0;
// tmp[]//当前正在搜索的答案 x_1 x_2
// ans[]//当前最好的答案
void dfs(int a,int b,int x){//(a/b)表示剩余分数大小，x表示当前搜索深度
    if(x > dep)//超过了限定深度，退出
        return;
```

```
    if(a == 1){//找到了解（此时x一定等于限定的深度dep，否则在dep更小的时候就会找到解）最后一个加数就是 1/b
            tmp[x] = b;
            if(!flag || tmp[x] < ans[x])//找到了更优解
                for(int i = 1;i <= dep;i++)
                    ans[i] = tmp[i]; // 更新最优解
        flag = 1;//标记为已经找到了答案
        return;
    }
    int l = max((b + a - 1) / a,tmp[x - 1] + 1),r = min((dep - x + 1) * b / a,INF);//下一个分数分母的上下界
    //
    if(flag && r >= ans[dep]) // 在当前深度dep下已经有了答案 且答案的最后一个分数的分母<=r
        r = ans[dep] - 1;      // r最多和已知答案最后一个数的分母一样大
    for(int i = l;i <= r;i++){ // 搜索下一个加数 1/i [l,r]
        tmp[x] = i;
        //请自行模拟分数加减法
        int A = a * i - b,B = b * i; // a/b - 1/i
        int gcd = GCD(A,B); // __GCD(x,y)
        dfs(A / gcd,B / gcd,x + 1); // 约分(a/b - 1/i) 搜下一层
    }
}
signed main(){
    a = read();b = read(); // cin >> a >> b;
    c = GCD(a,b);
    a /= c;b /= c; // 约分
    tmp[0] = 1;
    for(dep = 1;dep <= N - 1;dep++){//枚举深度
        dfs(a,b,1);  // dep --> dep
        if(flag){找到了答案 // 如果找到了答案 一定是符合题目要求最优的
            for(int i = 1;i <= dep;i++)
                printf("%lld ",ans[i]);
            return 0;
        }
    }
    return 0;
}
```

习题

DFS https://www.luogu.com.cn/problem/P2036 https://www.luogu.com.cn/problem/P5194

BFS https://www.luogu.com.cn/problem/P2895

BFS障碍版 https://www.luogu.com.cn/problem/P1605

综合BFS https://www.luogu.com.cn/problem/P1825

这两个也是很典的BFS 类似力扣海岛那类

https://www.luogu.com.cn/problem/P1596

https://www.luogu.com.cn/problem/P1162

难一点的BFS https://www.luogu.com.cn/problem/P1126 https://www.luogu.com.cn/problem/P3956

DFS 可以剪枝 https://www.luogu.com.cn/problem/P1433