Codeforces（Div.2） 补题 合集1

C. Meximum Array

题意

共 $t\ (1 \leq t \leq 100)$组测试，给出数组元素 $n\ (1 \leq n \leq 2 \cdot 10^5)$，随后给出 n 个数，$0 \leq a_i \leq n$

可以做如下操作：

选择前 k 个数，求出它们的 MEX，放在 b 数组的末尾，同时删去前 k 个数。

要求让 b 数组字典序上最大，输出 b 的长度和组成。

思路

贪心思想，因为是按字典序，所以越早存入 b 的应该越大，即每次找出最大的mex存入，并在次情况下要求删除最少的数字。对于求mex，可以先在输入的时候，预处理记录每个数字出现的次数，保存在另一个数组中，随后遍历该数组，找出第一个等于0的元素即可。但由于 n 的值较大，所以每次都遍历会超时。

可以发现，对于有相同部分的两段数组元素的mex值是有关系的。假如不同部分有等于mex的值，mex就应该加1……当然也可以想到有很多不同的情况。但可以先确定储存更新mex的策略。

如何更新mex值呢，可以在遍历 a 的循环里加个循环，即如果在已搜索过的 $a[i]$ 里出现了mex，那就将mex++，再看新的mex有没有……而当之后未搜索的 $a[i]$ 里没有等于mex的，该整个数组的mex就已经求出了。（由于预处理了每个数的个数，那么每搜索一个就让该数的次数减一，如果mex的剩余次数为0，就代表mex已经求出）。

总结

学习到了复杂度为 $nlog(n)$ 的复杂度的求 mex 的方法。其实这也应该是我们用人脑去考虑 mex 时的做法（至少是我的）。所以有时我们本身的思维方式也可以带来程序设计的灵感。

> 要重视预处理的作用，可以简化很多运算。

AC代码

Problem	Lang	Verdict	Time	Memory
C - Meximum Array	GNU C++17	**Accepted**	46 ms	3100 KB

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N=200010;
int a[N];
int ct[N];//该数字剩余次数
int r[N];//a中的第几个数字使用在第几轮
int b[N];
int n;
void solve()
{
    cin >> n;
```

```
    memset(ct, 0, sizeof(ct));
    memset(b, 0, sizeof(b));
    memset(r, 0, sizeof(r));
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        ct[a[i]]++;
    }
    int id = 1;
    int m = 0;
    for (int i = 1; i <= n; i++) {
        ct[a[i]]--;
        r[a[i]] = id;//代表a[i]这个数存在且在第一轮
        while (r[m] == id)
            m++;
        if (ct[m] == 0) //说明mex已经是最大的了
        {
            b[id] = m;
            id++;
            m = 0;
        }
    }
    cout << id-1 << endl;
    for (int i = 1; i <= id-1; i++)
        cout << b[i] << ' ';
    cout << endl;
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    int tt;
    cin >> tt;
    while (tt--) {
        solve();
    }
    return 0;
}
```

C. Poisoned Dagger

题意

$t\ (1 \le t \le 1000)$ 组测试，给出 $n(1 \le n \le 100)$ 和 $h(1 \le h \le 10^{18})$，分别表示数组元素个数和怪物血量，对于各数组元素，$a_i(1 \le a_i \le 10^9)$ 。数组元素的值表示每次攻击开始的时刻，持续 k 个单位时间（从该时刻开始），每个单位时间对怪物造成1点伤害，直到下一次攻击停止，更新攻击效果。

求能使得怪物被打败的最小的k 。

思路

由于 h 的数值很大，所以不能采取逐个累加的方式，会超时；所以使用二分查找 k，通过一个函数来判断是否可以，因为 n 不大，所以可以通过遍历数组 a 来判断即可。

总结

> 二分查找（Binary search）是个好东西。

AC代码

Problem	Lang	Verdict	Time	Memory
C - Poisoned Dagger	GNU C++17	**Accepted**	46 ms	0 KB

```cpp
#include<bits/stdc++.h>
using namespace std;
long long n, h;
long long a[110];

bool check(long long x)
{
    long long sum = 0;
    for (int i = 1; i < n; i++){
        sum += min(x, a[i + 1] - a[i]);
    }
    sum += x;
    return sum >= h;
}
void solve()
{
    cin >> n >> h;
    for (int i = 1; i <= n; i++)
        cin >> a[i];
    long long l = 0, r = h;
    long long  Min = 0;
    while (l <= r)
    {
        long long  mid = (l + r) / 2;
        if (check(mid))
        {
            r = mid - 1;
            Min = mid;
        }
        else
        {
            l = mid + 1;
        }
    }
    cout << Min << endl;

}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    int tt;
    cin >> tt;
    while (tt--) {
        solve();
```

```
    }
    return 0;
}
```

C. Chat Ban

题意

共 $t\ (1 \leq t \leq 10^4)$组数据，给出 k 和 $x\ (1 \leq k \leq 10^9; 1 \leq x \leq 10^{18})$，从第一行到第 k 行每行的个数为行数，从 $k+1$ 行递减至 0 ,总个数需要小于等于 x，问最多可以有几行（包括不完整的行）。

思路

因为 x 较大，所以采用二分查找，将整个图形分成上下两个三角形即可。二分的主要麻烦的地方就是端点的调整。

总结

> Binary search yyds！！！

AC代码

Problem	Lang	Verdict	Time	Memory
C - Chat Ban	GNU C++17	**Accepted**	31 ms	0 KB

```cpp
#include<bits/stdc++.h>
using namespace std;
long long k, x;
long long ch1(long long pos)
{
    if ((2 + pos) * (pos+1) / 2>=x)
        return 1;
    else return 0;
}
long long ch2(long long pos)
{
    if ((k-1+k-pos)*pos/2>=x)
        return 1;
    else return 0;
}
void solve()
{
    cin >> k >> x;
    long long sum = 0;
    if ((1 + k) * k / 2 >= x) {
        long long l = 0, r = k;
        long long ans = 0;
        while (l <= r)
        {
            long long  mid = (l + r) / 2;
```

```
            if (ch1(mid))
            {
                r = mid - 1;
                ans = mid;
            }
            else
            {
                l = mid + 1;
            }
        }
        cout << ans + 1 << endl;
        return;
    }
    else if ((1 + k) * k / 2 + (1 + k - 1) * (k - 1) / 2 <= x)
        cout << 2 * k - 1 << endl;
    else {
        x -= (1 + k) * k / 2;
        long long l = 1, r = k;
        long long ans = 0;
        while (l <= r)
        {
            long long mid = (l + r) / 2;
            if (ch2(mid))
            {
                r = mid - 1;
                ans = mid;
            }
            else
            {
                l = mid + 1;
            }
        }
        cout<<k+ans<<endl;
    }
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    int tt;
    cin >> tt;
    while (tt--) {
        solve();
    }
    return 0;
}
```

B. Special Permutation

题意

给出 n, a, b ，分别为数组元素个数，所求排列的左半最小值和右半最大值，排列由 1 到 n 组成 。如果能做到，输出排列；不然输出-1。

思路

先分类存储，均可和已固定位置的和均不可的，随后分类讨论即可。

总结

做的时候相当然了，看到 $n \leq 100$ ，就给左右的数组各开了60，没想到运算过程中越界了。

> 所以不仅看输入输出，也要考虑运算过程中是否存在越界。

AC代码

Problem	Lang	Verdict	Time	Memory
B - Special Permutation	GNU C++17	**Accepted**	15 ms	0 KB

```cpp
#include<bits/stdc++.h>
using namespace std;
int n, a, b;
int l[100];
int r[100];
int m[100];
//比a大可以放左边，比b小可以放右边
void solve()
{
    cin >> n >> a >> b;
    l[1] = a;
    r[1] = b;
    int re = 0;
    int cnt1 = 1;
    int cnt2 = 1;
    for (int i = 1; i <= n; i++) {
        if (i == a || i == b)continue;
        if (i > a && i < b) {
            re++;
            m[re] = i;
        }
        else if (i > a && i > b) {
            cnt1++;
            l[cnt1] = i;//报错：out of bounds
            //如果l，r数组没开到100，这里的cnt可能会越界
        }
        else if(i < a && i < b){
            cnt2++;
            r[cnt2] = i;
        }
        else if (i < a && i > b) {
            cout << -1 << endl;
            return;
        }
```

```
    }
    if (abs(cnt2 - cnt1) > re)cout << -1 << endl;
    else {
        while (cnt1 != cnt2||re) {
            if (cnt1 < cnt2) {
                cnt1++;
                l[cnt1] = m[re];
                re--;
            }
            else if(cnt1>cnt2){
                cnt2++;
                r[cnt2] = m[re];
                re--;
            }
            else {
                while (re) {
                    cnt1++;
                    l[cnt1] = m[re];
                    re--;
                    cnt2++;
                    r[cnt2] = m[re];
                    re--;
                }
            }
        }
        for (int i = 1; i <= cnt1; i++)
            cout << l[i] << ' ';
        for (int i = 1; i <= cnt2; i++)
            cout << r[i] << ' ';
        cout << endl;
    }
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    int tt;
    cin >> tt;
    while (tt--) {
        solve();
    }
    return 0;
}
```

D. Make Them Equal

题意

给出各 n 个元素的数组 b, c ， 以及最多操作数 k 。

初始时另一个数组 a 中每个元素都是 1 ， 每次操作可以使 $a_i = a_i + \lfloor a_i / x \rfloor$

`if(a[i]==b[i])sum+=c[i]`，求 sum 的最大值。

数据范围：

$1 \le i \le n, x > 0, 1 \le t \le 100, 1 \le n \le 10^3; 0 \le k \le 10^6, 1 \le b_i \le 10^3, 1 \le c_i \le 10^6$

思路

dp（背包）。

先打表出到达1到1000所需要的步数，再对于每次加不加该操作判断即可。

注意到 k 的数据较大，所有1变成1000，即使总共1000个数，也只需要12000步，所以开始时对 k 处理一下，使它对 12000 取 min ，这样一来dp数组大小也只需要取12010左右，同时运算量减少很多（不然本题会tle）。

总结

当发现变量范围很大时，看看能不能剪枝，以免简单大数据超时。

AC代码

Problem	Lang	Verdict	Time	Memory
D - Make Them Equal	GNU C++17	**Accepted**	31 ms	100 KB

```cpp
#include<bits/stdc++.h>
using namespace std;
int ti[2010];
int b[2010];
int c[2010];
int dp[12010];
int n, k;
void init()
{
    memset(ti, 0x3f, sizeof(ti));//这里注意，因为是取min，初始化成最大值
    ti[1] = 0;//起点要给出
    for (int i = 1; i < 1000; i++)
        for (int j = 1; j <= i; j++)
            ti[i + i / j] = min(ti[i] + 1, ti[i + i / j]);
}
void solve()
{
    cin >> n >> k;
    k = min(k, 12 * n);//剪枝，ti[1000]=12
    memset(dp, 0, sizeof(dp));
    for (int i = 1; i <= n; i++)
        cin >> b[i];
    for (int i = 1; i <= n; i++)
        cin >> c[i];
    for (int i = 1; i <= n; i++)
        for (int j = k; j >= ti[b[i]]; j--)
            dp[j] = max(dp[j], dp[j - ti[b[i]]] + c[i]);
    cout << dp[k] << endl;
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
```

```
    int tt;
    cin>>tt;
    init();
    while (tt--)solve();
    return 0;
}
```