CF #643 (div2)

cy

A. Sequence with Digits

题意：$a_{n+1} = a_n+MinDig(a_n)*MaxDig(a_n)$,Dig表示十进制表示中的最大的一个数字和最小的一个数字。给出a_1和K求a_K两个数字都$\Leftarrow1e18$

题解：刚开始的时候没想出什么好办法，想到了如果对于其中某一项MinDig为0，则后面都是0,但这一项是否一定会出现呢？是否有一种构造使其永远不会出现0？不存在的，我们发现，每一项比前一项最多大$9*9=81$，现在考虑$[1000*(a_i/1000+1),1000*(a_i/1000+1)+99]$这一段区间，显然这段区间所有的数字都含有0这个数位，并且这段区间长度大于81，也就是说a_n只要存在大于这个区间的值，就一定会有一项落到这个区间内，其必含有0，所以可以直接暴力计算a_1,a_2...

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <stdio.h>
using namespace std;
typedef unsigned long long ull;
int main()
{
    int t;
    scanf("%d", &t);
    while(t--)
    {
        ull x, k;
        cin >> x >> k;
        for (int i = 2; i <= k;i++)
        {
            ull tem = x;
            ull mmin = 9, mmax = 0;
            while (tem)
            {
                ull jkl = tem % 10;
                mmin = min(mmin, jkl);
                mmax = max(mmax, jkl);
                tem /= 10;
            }
            x += mmin * mmax;
            if(mmin == 0 || mmax == 0)
                break;
        }
        cout << x << endl;
    }
    return 0;
```

```
}
```

B. Young Explorers

题解：直接贪心，没啥好说的

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <stdio.h>
using namespace std;
const int MAX = 2e5 + 100;
int pic[MAX];
int n;
int main() {
    int T;
    scanf("%d", &T);
    while(T--)
    {
        scanf("%d", &n);
        for (int i = 1; i <= n;i++)
            scanf("%d", &pic[i]);
        sort(pic + 1, pic + 1 + n);
        int ans = 0;
        int tem = 0;
        for (int i = 1; i <= n;i++)
        {
            tem++;
            if (pic[i] == tem)
            {
                ans++;
                tem = 0;
            }
        }
        printf("%d\n", ans);
    }
    return 0;
}
```

C. Count Triangles

题意：给定ABCD□□$\Leftarrow5e5$□,求出有多少个x,y,z可以组成一个三角形($A\Leftarrowx\LeftarrowB\Leftarrowy\LeftarrowC\Leftarrowz\LeftarrowD$)

题解：做的时候脑子不清醒，想错了，分了8类讨论。。。这题官方题解的方法非常好。用一个大数组A_i来表示$x+y=i$的个数，计算数组A时，可以枚举x的值，然后用前缀和的方式来计算区间加法，然后对数组A再来一次前缀和，就能求出$x+y\Leftarrowi$的个数，最后累加C到D之间的答案即可。

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <stdio.h>
using namespace std;
typedef long long ll;
const int MAX = 1e6 + 20;
ll pic[MAX];
int main()
{
    int a, b, c, d;
    cin >> a >> b >> c >> d;
    for (int i = a; i <= b;i++)
    {
        pic[i + b]++;
        pic[i + c + 1]--;
    }
    for (int i = 1; i < MAX; i++)
        pic[i] += pic[i - 1];
    for (int i = 1; i < MAX; i++)
        pic[i] += pic[i - 1];
    ll ans = 0;
    for (int i = c; i <= d;i++)
        ans += pic[MAX-1] - pic[i];
    cout << ans;
}
```

D. Game With Array

题解：这题很容易构造出来，条件放的太宽了。

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <stdio.h>
using namespace std;
typedef long long ll;
int main()
{
    int n, s;
    cin >> n >> s;
    int mean = s / n;
    int lef = s - mean * n + mean;
    if (mean > 1)
    {
        cout << "YES" << endl;
        for (int i = 1; i < n;i++)
            cout << mean << ' ';
```

```
        cout << lef << endl;
        cout << 1;
    }
    else
    {
        cout << "NO" << endl;
    }
    return 0;
}
```

E. Restorer Distance

题意：初始有 n 根柱子，每根柱子有一个高度 h[i]现在有三种操作：花费 A 使得某根柱子高度+1、花费 R 使得某根柱子高度-1、花费 M 使得某根柱子高度-1而另一根+1。问最少的花费，使得最终所有柱子高度一样。

题解：如果有一个给定的h那么最终花费很好求，只要求一个前缀和然后二分查找即可。问题在于这个h如何去确定，官方题解给出的答案是，先假设一个h再给h增加1，观察最终花费的变化，最终发现，最优答案的h一定和某个柱子的砖块数量一样，或者是在均值附近，于是可以nlogn枚举答案，还有一种思路，发现最终答案和h的大小呈二次函数关系，于是三分。

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <stdio.h>
using namespace std;
typedef long long ll;
const int MAX = 1e5 + 20;
ll pic[MAX];
ll sum[MAX];
ll n, a, r, m;
ll cal(ll h)
{
    int pos = lower_bound(pic + 1, pic + 1 + n, h) - pic - 1;
    ll res = 0;
    ll k1 = h * pos - sum[pos];
    ll k2 = sum[n] - sum[pos] - h * (n - pos);
    res = min(k1, k2);
    k1 -= res;
    k2 -= res;
    res *= m;
    res += k1 * a;
    res += k2 * r;
    return res;
}
int main()
{
```

```
    cin >> n >> a >> r >> m;
    for (int i = 1; i <= n;i++)
    {
        scanf("%lld", &pic[i]);
    }
    sort(pic + 1, pic + 1 + n);
    m = min(m, a + r);
    for (int i = 1; i <= n;i++)
        sum[i] = sum[i - 1] + pic[i];
    ll ans = 0x3f3f3f3f3f3f3f3f;
    ans = min(ans, cal(sum[n] / n));
    ans = min(ans, cal(sum[n] / n + 1));
    for (int i = 1; i <= n; i++)
        ans = min(ans, cal(pic[i]));
    cout << ans;
}
```

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <stdio.h>
using namespace std;
typedef long long ll;
const int MAX = 1e5 + 20;
ll pic[MAX];
ll sum[MAX];
ll n, a, r, m;
ll cal(ll h)
{
    int pos = lower_bound(pic + 1, pic + 1 + n, h) - pic - 1;
    ll res = 0;
    ll k1 = h * pos - sum[pos];
    ll k2 = sum[n] - sum[pos] - h * (n - pos);
    res = min(k1, k2);
    k1 -= res;
    k2 -= res;
    res *= m;
    res += k1 * a;
    res += k2 * r;
    return res;
}
int main()
{

    cin >> n >> a >> r >> m;
    for (int i = 1; i <= n;i++)
    {
        scanf("%lld", &pic[i]);
    }
    sort(pic + 1, pic + 1 + n);
```

```
    m = min(m, a + r);
    for (int i = 1; i <= n;i++)
        sum[i] = sum[i - 1] + pic[i];
    ll ans = 0x3f3f3f3f3f3f3f3f;
    int L = 0, R = 1e9;
    while (L < R)
    {
        int mid = (L + R) >> 1;
        int mmid = (mid + R) >> 1;
        if (mid == mmid)
            break;
        if (cal((ll)mid) < cal((ll)mmid))
            R = mmid;
        else
            L=mid;
    }
    for (int i = L; i <= R;i++)
        ans = min(ans, cal((ll)i));
    cout << ans;
}
```