1.减法（sub）

改编自 P11062 【MX-X4-T2】「Jason-1」加法

将输入中的 b 取反，两题互相转化，后文默认在输入后立刻将 b 赋值为 $-b$。

首先考虑进行两种操作后 $a, b, |a-b|$ 分别会变化成什么。

操作类型	A	B	$\|a-b\|$
1	$a+b$	b	$\|a\|$
2	a	$a+b$	$\|b\|$

分两种情况考虑

1. a, b 符号相同。此时，执行任意操作，$|a|, |b|$ 都不会变小，无法通过执行多次操作得到更小的答案，因此只需要执行不超过一次操作。若执行操作，差的绝对值分别为 $|a|, |b|$，若不执行操作，差的绝对值为 $|a-b|$，对这三者取最小值即可。
2. a, b 符号不同。此时，可以不断将绝对值较小者加到绝对值较大者上，根据经典的辗转相减法，最终一定会有一个数变为 0，此时将这个数加上另一个数，两数相等，答案为 0。

代码如下：

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

int main(){
    int t;
    cin >> t;
    while(t--){
        int a, b;
        scanf("%d%d", &a, &b);
        b=-b;
        if(1ll*a*b<=0){
            puts("0");
        }else{
            printf("%d\n", min({abs(a), abs(b), abs(a-b)}));
        }
    }
    return 0;
}
```

2.染色（paint）

Subtask 1, 2

考虑观察一些性质，如果一次操作的操作没有对颜色造成任何影响，则该操作为无效操作。

考虑到每一次有效操作，其颜色的连续段段数至少会减 2，因此我们可以直接搜索所有的状态，转移为所有有效操作

复杂度为 $O(q \times m^n \times \text{poly}(n))$。记忆化后复杂度为 $O(q \times n + m^n \times \text{poly}(n))$，常数不大。

Subtask 3

留给一些可能的复杂度更劣做法

Subtask 4, 5

我们观察一下特殊性质，我们可以发现如果将操作视为反转一个位置的颜色，则：

- 最左边和最右边的格子是无法反转的；
- 相邻两个格子是无法同时反转的。

考虑相邻两个格子如果同时反转，先反转的那一个格子一定会与后反转的格子相同，此时两格子在之后的操作后也一定相同，矛盾。

否则，我们如果需要反转第 i 个格子，则直接选择 $[i-1, i+1]$ 进行操作即可。

因此我们可以直接使用 dp 求解，状态 $f_{i,0/1}$ 表示第 i 个格子是否反转，时间复杂度为 $O(q \times n)$，使用 2×2 的 $(\min, +)$ 矩阵与线段树可以实现单点修改和区间查询。

Subtask 6

考虑扩展性质，发现我们可以将每一个 $0/1$ 连续段缩为一个格子，权值求和，这样操作是不影响正确性的，这个时候就可以套用上方特殊性质的解法了。

Subtask 7, 8

观察出一个更强的性质：

- 当一个节点被染成不同的颜色后，一定存在最优解，该节点所在连通块不会去染其他颜色。

证明可以考虑如果节点被染色后又与另外一个端点进行染色，则我们可以直接一次操作全部染色。

因此我们发现所有染色操作都是不交的区间，考虑设置 dp 状态 $f_{i,0\sim m}$ 代表当前在第 i 个格子，并且该格子正在染色为 $0 \sim m$，0 代表当前没有操作。

转移的时候，考虑先有 $f_{i,j} + a_{i,j} \to f_{i+1,j}$，特殊的，$f_{i,0} + a_{i,c_i} \to f_{i+1,0}$， 然后考虑当前的颜色 c_i，我们可以选择在这里开始一段区间染色，也可以选择在这里结束一段区间染色，因此我们额外增加两条转移 $f_{i,c_i} + a_{i,c_i} \to f_{i+1,0}$ 和 $f_{i,0} + a_{i,c_i} \to f_{i+1,0}$。最后的答案即为 $f_{n,0} + a_{n,c_n}$。

时间复杂度为 $O(q \times n)$，使用 6×6 的 $(\min, +)$ 矩阵与线段树可以实现单点修改和区间查询。

3.新月争霸（xinyue）

注意到攻击的过程可以分为两个阶段：

1. 不断挑战攻击力比自己更高的 Boss

2. 得到攻击力最高的剑，把其他 Boss 全杀了

所以可以给答案加上 2 阶段的答案 $\sum_{i=1}^{n}\left\lfloor\frac{h_i-1}{\max_{j=1}^{n}a_j}\right\rfloor\times a_i$，然后减去 1 阶段的。

令 dp_i 表示当前最高攻击力为 i 时 1 阶段的最小血量耗费，从 $dp_{\max\ a}=0$ 推到 dp_{a_0}。

钦定下一次攻击谁，有转移：

$$dp_i\xleftarrow{a_j>i}dp_{a_j}+\left\lfloor\frac{h_j-1}{i}\right\rfloor\times a_j-\left\lfloor\frac{h_j-1}{\max\ a}\right\rfloor\times a_j$$

对 $\left\lfloor\frac{h_j-1}{i}\right\rfloor$ 的值进行钦定，总枚举量是调和级数 $O(n\ln n)$ 的（假设 n 与 $\max a,\max h$ 同阶）。

现在需要进行 h 在一个区间内的一次函数求最值。由于从大往小枚举 i，用线段树维护单调队列即可（或树状数组，因为有单调性）。

复杂度 $O(n\ln n\log n)$。

一个小问题：如果在 1 阶段时，对于每种攻击力只保留血量最低的那个，是错误的。

原因：$f(x)=\frac{x}{a}-\frac{x}{b}$ 是一个单调函数，但 $f(x)=\left\lfloor\frac{x}{a}\right\rfloor-\left\lfloor\frac{x}{b}\right\rfloor$ 不是单调函数。

4.最小值（min）

分治，令当前分治区间是 $[l,r]$，$mid=\lfloor(l+r)/2\rfloor$ 。考虑跨越两侧的区间对答案的贡献。

令：

- $C_i=min\{A_i,\cdots,A_{mid}\}$
- $D_i=min\{B_i,\cdots,B_{mid}\}$
- $E_i=min\{A_{mid+1},\cdots,A_i\}$
- $F_i=min\{B_{mid+1},\cdots,B_i\}$

则 $w(l,r)=|min\{C_l,E_r\}-min\{D_l,F_r\}|$。

因为 C,D 是递增的，E,F 是递减的，所以固定 $k=r-l+1$ 后，在一段前缀中，$min\{C_l,E_r\}=C_l$，一段后缀中，$min\{C_l,E_r\}=E_r$ 。

同理，在一段前缀中，$min\{D_l,F_r\}=D_l$ ，一段后缀中，$min\{D_l,F_r\}=F_r$ 。

分四种情况讨论：

- $w(l,r)=|C_l-D_l|$，直接求区间 min 就行。
- $w(l,r)=|E_r-F_r|$ ，还是直接求区间 min 就行。
- $w(l,r)=|C_l-F_r|$， C 是递增的 F 是递减的，你只需要二分出交点就行。
- $w(l,r)=|E_r-D_l|$，E 是递减的 D 是递增的，还是只需要二分出交点就行。

复杂度 $O(nlog^2n)$