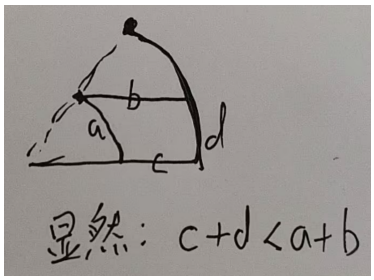
个人题解（可能有误，待大家批评指正）

QQ 交流群：1042796603

A.

最优解应该是先直线，再圆弧，就这两步。

主播画了个图：



B.

主播也不会做喵。（主播从来都是直接跳过第二个填空的）

C.

由于负数也可以，因此对于任意大于 1 的正整数一定存在：

$-x+1,-x+2,\ldots,x-1,x$ 这样的序列，因此都有解。

所以答案就是序列中大于 1 的数字个数。

D.

实在不会做的情况下，打个表就会发现，只需要不到 log 次，A,B,C 的值就不变了。

因此直接模拟这个过程，最多模拟 30 次就停了（实际上可能就十几次）。

E.

考虑这个式子描述的实际上是相邻的差距之和（类似方差），要这个东西尽可能小，那么相邻的数字一定要尽可能接近。因此答案对应的长为 M 的序肯定是排好序后连续的一段。

因此我们只需要对序列排序，滑动窗口（双指针）枚举所有长度为 M 的区间，取上式子的最小值即可。

F.

主播的做法可能比较麻烦。（但适用于 $n\times m$ 这种的网格图）

1. 首先我们跑一个并查集，把所有已经连好的连通块都连上，接着我们可以直接任取一个连通块，从它开始跑一个 $dijkstra$，每搜到一个别的连通块时，我们就把最短路加到答案，同时把新连通块的所有点都入队继续 dij，但此时注意需要把优先队列中新连通块的点的最短路都置为 0。

2. 主播还想了个 dp 的做法（不太确定对不对），把所有 $\#$ 的坐标存下来，定义 $dp_{i,0/1}$ 表示考虑到第 $0/1$ 行第 i 列，且前面全部连通的最小代价，转移时：

● 如果是 $\#$，则要么从同一行的上一个 $\#$ 转移，额外代价是 $pos_k-pos_{k-1}-1$；要么是另一行的上一个 $\#$ 转移，代价是 pos_k-pos2_{k-1}。（其中 pos_k 表示第 k 个 $\#$ 的下标。）

● 如果是 . 直接继承 dp_{i-1}。

这题煮啵可能做麻烦了，疑似是可以直接贪的。

G.

主播觉得这题题面没写清楚，没有说明白删除一个节点后，其对应的子树直接连到父亲上，还是一并删除。

但主播猜测是一并删除，那其实相当于只能删叶子。我们考虑一个背包：$dp_{u,j}$ 表示 u 节点能否供给父亲 j 的量。

然后做一个树上背包即可，枚举所有儿子转移，具体来说：

枚举所有节点当 u，再枚举 u 的所有儿子 v，再枚举 u 的 j，再枚举 v 的 j。

这样一来其实是 $1000 \times 1000 \times 1000$ 的。

这里转移这里应该是需要 $bitset$ 优化一下的，因为 dp 实际上只有 0 和 1。

所以最终正解复杂度应该是 $1000 \times 1000 \times 1000/w$ 的。

暴力转移大概是以下这个式子（伪代码），这个式子可以用 $bitset$ 优化：

```
for(auto & v : g[u]) {
    for(int j = a[u]; j >= 0; j--) {
        for(int k = a[v]; k >= 0; k--) {
            dp[u][j + k] |= dp[v][k] & dp[u][j];
        }
    }
}
```

H.

纯纯诈骗题。

我们把这个式子的最前面的符号也写上，实际上是：$+$。

也就是说：$a - b \oplus c$ 实际上是 $+a - b \oplus c$。

那我们就会发现，对于加减法来说，只有第一个数字不停地提供了 $+$ 的贡献。

也就是说，在第一个运算符不是异或的所有情况中，a_1 都提供了 $+a_1$ 的贡献，我们把这部分贡献先算上。

接着我们考虑第一个运算符是 $\oplus$ 的情况，实际上我们会发现，此时只有前缀 $\oplus$ 有贡献，因为后面的都会 $+-$ 互相抵消。

因此我们只需要枚举前缀异或的长度，对于一个长度 i，其贡献就是：$3^{n-i-1} \times 2$，因为首先第 $i+1$ 个运算符不能是 $\oplus$，就只有 2 种选择，而 $i+2$ 到最后的运算符都可以选择 $+, -, \oplus$ 三种。因此是这个贡献。

将所有的贡献求和就是答案。