第一题：*escape*

60 分

O($N^{2}M$^2)预处理出 dis[i][j]，枚举答案，判断是否可行

100 分

首先我们通过一次 BFS （多源 bfs）预处理出 dis[i][j]，表示(i,j)到最近的敌人的距离。这步的复杂度是 O(NM)

具体操作是把所有敌人全部加入队列里面，跑一遍 bfs

发现问题是最近时最远，即最小值最大问题，考虑是否可以二分，发现答案具有可二分性，我们可以直接二分答案 ans，然后把所有 dis[i][j]<= ans 的格都认为不可达，然后通过一次 BFS 来判断是否存在从起点到终点的路径即可，这样做的时间复杂度是 O(NMlog$N$ + M)。

（由于是第一题，问题主要考的是预处理，对于枚举答案的也给过了）

第二题：matrix

30分

枚举行怎么排列，然后求最大全1矩阵

70分

算法一：（一个丑陋的做法）

考虑行会因排列改变，但列不会变，我们用对每一列建立一个树状数组，

求答案时，我们先枚举矩阵的最右边那一列，再枚举矩阵在一行的长度，用树状数组求出此时全1的最多有多少行即可

复杂度 O(n*m*log(n))

算法二：

标程+sort排序

100分

我们可以在O(n*m)时间内求得right[i][j]，其中right[i][j] = (j,i)格子右边有多少个全1的数

我们将每个right[i]采用O(n)的方法计数排序，那么

```
For j in [1..n]

   If right[i][j]*(n-j+1) > answer then answer = right[i][j]*(n-j+1)
```

即先枚举矩阵左端点在哪，再对于每个右边的不同长度计算面积

总复杂度 O(n*m)

（注意标程写的是 left[i]）

第三题：*pay*

20 分

支付的黄金和钻石的值肯定等于某一条边设立的值，考虑枚举两个值，然后跑一遍 bfs 或 dfs 判断能否到 n 个点即可

另 20 分

此时 m 很大，但 g，d 值很小，考虑枚举 g，d，跑一边 bfs 或 dfs 判断即可

60 分

考虑如果 g 确定，我们要求最小的 d 使满足条件，即求一颗最小瓶颈生成树（使最大边权值尽量小的生成树）

有最小生成树就是最小瓶颈生成树(from 货车运输)，

【原因：因为我们把边排序之后，是从小到大加入边，最大边最小也

就是最后加进去的边尽量靠前，而我们设 Kruskal 加进去的最后一条边 e，e 边加进去之前用更小的那些边是构成不了一个生成树的。】

于是我们枚举 g，每次求一次最小生成树即可，只排序一次的话，复杂度 O(m^2*并查集常数)

100 分

我们观察到 m 很大，g、d 更大，但 n 很小，同时又很难得到带有 log 的算法，复杂度应该与 n 也有关，大致在 n*m 左右，考虑怎么把 n 加进来？

考虑把所有边按 g 为第一关键字，d 为第二关键字排序

我们枚举 g，试图快速求出对于给定 g，d 至少为多少，怎么做？

考虑动态维护一棵最小生成树，每次 g 增加，我们又多了一些可用边，对于每条边(u,v)，如果 u,v 未连通，则连接 u,v，否则删除 u,v 路径上 d 最大的边，连接 u，v（前提是 u,v 的 d 比最大的小），每次把所有边加入后，dfs 一遍，如果 n 个点连通，那么树边中最大的 d 即对于目前 g 所需最小的 d。

【此处利用性质：最小生成数的充分必要条件：对于任意非最小生成树上的边（u，v），它的权值一定大于等于最小生成树 u 到 v 路径上每条边的权值】

（其实也可以不这样做，我们保存哪些边是树边，每次加入 x 条新的边，我们用这 n + x 条新的边重建一颗最小生成树也可以，采用插入

排序的话可以使复杂度到 O(nm*并查集常数))