**Gameshow**

Dp[i][j] là số tiền lớn nhất m có khi đến cửa I và bỏ qua j cửa.

Nếu dp[i][j] > t và i!=n-1 thì dp[i+1][j]=max(dp[i+1][j], dp[i][j]-T);

Nếu dpi[i][j]+a[i+1]>=0 f[i+1][j+1]=max(f[i+1][j+1], f[i][j]+a[i+1]);

**Zero**

Khi đọc đề, ta sẽ nghĩ ra ngay 1 thuật toán đệ quy có nhớ và có công thức cập nhập như sau

dp[i]=min(dp[i-1],dp[max(i/j,j)] (j sẽ chạy từ 2->sqrt(i) )+1;

Độ phức tạp của bài toán này là O(maxn\*sqrt(maxn)) nhưng trên thực tế sẽ nhỏ hơn hơn nhiều. Nhưng vì khi thực hiện phép tính chia thì sẽ xử lí rất chậm nên ta sẽ không thể sử dụng phép chia nếu muốn thuật toán không TLE. Ta có thể cải tiến thuật toán bằng cách sử dụng thuật toán loang từ 0( hoặc từ 1 với vai trò tương tự. Ta sẽ có tối đa 1e6 đỉnh và từ 1 đỉnh i, ta có thể đến i+1 hoặc i\*j (với j<=i và i\*j<=1e6). Dù độ phức tạp không thay đổi nhưng sẽ cải thiện được đáng kể thời gian do việc không sử dụng toán tử chia.

**Queue**

Để làm được bài này thì ta cần phải sử dụng trung vị. Đầu tiên, ta sẽ xác định chiều dài và chiều rộng của hình chữ nhật, sau đó sẽ xác định vị trí bắt đầu của hình chữ nhật. Vì ta sẽ chỉ phải tìm vị trí mở đầu nên ta có thể làm độc lập điều này cho tung độ và hoành độ. Đầu tiên ta sẽ sort lại cả tung và hoành độ, sau đó với từng kích thước hình chữ nhật, ta sẽ tìm tung và hoành độ của vị trí mở đầu và đồng thời tính chi phí của việc di chuyển. Có thể tham khảo code tại đây <https://ideone.com/PX29Pa>. Độ phức tạp của thuật toán này sẽ là O(n\*(số nghiệm của n bé hơn sqrt(n))\*log(n)).

**Map**

Duyệt 2 đỉnh liên tiếp nhau, nếu chúng cùng hoành độ thì đánh dấu những điểm nằm giữa chúng sau đó duyệt lại toàn bộ bảng, những điểm có chung hoành độ và nằm giữa những điểm đc đánh dấu thì in ra #, không thì .