ماموریت نهایی

- **نوع تمرین:گروهی-امتیازی**

سفینه به یک شبکه‌ی متحرک وارد شده که **هر سلول در هر ثانیه ممکنه تغییر انرژی بده!** دانشمندها موفق شدن انرژی کل سلول‌ها رو برای ثانیه‌های آینده پیش‌بینی کنن، و حالا باید **در زمان مشخصی** مسیر حرکت مشخص بشه.

🚧 قوانین مسیر:

- یک شبکه‌ی سه‌بعدی از انرژی‌ها: `grid[t][i][j]` ← انرژی خانه‌ی `(i,j)` در لحظه‌ی `t` مثلاً `grid[0]` ماتریس لحظه‌ی شروع، `grid[1]` انرژی لحظه‌ی بعد و ...
- مسیر باید از `(0,0)` در زمان `t=0` شروع شود، و در زمان `t=steps` به `(n-1,n-1)` برسد.
- **در هر لحظه فقط می‌توان به راست یا پایین یا توقف حرکت کرد.**
- **در هر ثانیه فقط یک حرکت انجام می‌شود.**
- **حداکثر `k` سلول منفی مجاز است.**
- مسیر باید **دقیقاً مجموع انرژی برابر `T`** داشته باشد.

فضای حالت مساله سه‌بعدی است پس ما در یک فضای سه‌بعدی پیمایش می‌کنیم و هر حالت را با سه متغیر i, j, t توصیف می‌کنیم.

در هر لحظه‌ی t فقط یک حرکت از سه حرکت مجاز را انجام می‌دهیم و انرژی را از grid[t][i][j] می‌گیریم. در هر قدم در فضای حالت انرژی مسیر جمع زده می‌شود، اگر سلول منفی باشد شمارنده‌ی منفی‌ها را زیاد می‌کنیم و اگر شرط‌ها نقض شوند مسیر را کنار می‌گذاریم.

در نهایت وقتی به برگ‌های درخت حالت رسیدیم، فقط مسیری قبول می‌کنیم که در زمان دقیقاً steps به خانه‌ی (n-1, n-1) برسد، مجموع انرژی دقیقاً T باشد و تعداد سلول‌های منفی از k بیشتر نباشد.

از آنجایی که در هر قدم سه انتخاب داریم، پس پیچیدگی زمانی از O(3^step) می‌باشد.

```
// n, grid, T, k, step are input which are explained in question
valid_cnt = 0
func BT(t, i, j, sum, neg_cnt):
   if i >= n or j >= n or t > steps:
      return
```

```
    energy = grid[t][i][j]
    sum += value
    if value < 0:
        neg_cnt += 1

    if neg_cnt > k:
        return

    if t == step and i == n-1 and j == n-1:
        if sum == T:
            cnt += 1
        return

    // 3 move choices
    backtrack(t + 1, i, j, sum, neg_cnt)      // stop
    backtrack(t + 1, i + 1, j, sum, neg_cnt)  // down
    backtrack(t + 1, i, j + 1, sum, neg_cnt)  // right
```

4023613060 - 4023613068