Polygon

tag 数学

题解思路

n条线段可以构成n边形的充要条件是任意一条线段都必须小于其他线段长度之和，因此判断最大的那条边是不是比其它所有边之和小就行。

参考代码

```cpp
#include <bits/stdc++.h>
using namespace std;
#define LL long long
LL n, x, mx, s;
int main(){
    cin >> n;
    for (int i = 0; i < n; i ++ ){
        cin >> x;
        s += x;
        mx = max(mx, x);
    }
    cout << (s - mx > mx ? "YES\n" : "NO\n");
    return 0;
}
```

Plus

tag 模拟

题解思路

注意数据范围，考虑是否为结论题，可以先写个暴力程序打表，如

```python
import math
n=int(input())
def isprime(x):
    if x<=1:
        return False
    if x==2:
        return True
    if x%2==0:
        return False
    for i in range(3,int(math.sqrt(x))+1,2):
        if x%i==0:
            return False
    return True
ans=0
res=[]
for i in range(1,n+1):
    for j in range(i,n+1):
        if isprime(i) and isprime(j) and isprime(i**j+j**i):
            ans+=1
            res.append({i,j})
print(ans)
for i,j in res:
    print(f'{i} {j}')
```

经过测试，发现1-1000内只有{2，3}这对解，因此大胆猜测只有一对解即可。

参考代码

```cpp
#include <bits/stdc++.h>
using namespace std;
#define LL long long
LL n;
int main(){
    cin >> n;
    if (n <= 2) cout << "0\n";
    else cout << "1\n2 3\n";
    return 0;
}
```

Generator

tag高等数学

解题思路

设E(x)为x变成1的期望次数，由样例可知$E(1)=1$，$E(2)=\frac{E(1)}{2}+\frac{E(2)+E(1)}{2}$，解出E(2)=2，同理，$E(3)=\frac{E(1)}{3}+\frac{E(2)+E(1)}{3}+\frac{E(3)+E(2)+E(1)}{3}$，解出$E(3)=\frac{5}{2}$。

历次类推，$E(x)=1+\sum_{k=1}^{x-1}\frac{E(k)}{x}=1+\sum_{k=1}^{x-1}\frac{1}{k}$，其中$\sum_{k=1}^{x-1}\frac{1}{k}$的调和级数为$\ln(x-1)+0.5772$(欧拉 - 马歇尔常数)，因此答案为$E(x)=\ln(x-1)+1.5772$，注意到精度达不到题目要求，因此考虑一种策略，对于$10^6\sim10^7$的数据，我们可以手动处理，剩余数据使用$E(x)=\ln(x-1)+1.5772$处理。

参考代码

```cpp
#include<bits/stdc++.h>
using namespace std;
#define fixd(x) fixed << setprecision(x)
int main(){
    int n;
    cin>>n;
    if(n<1e7){
        double ans=0;
        for(int i=1;i<n;i++) ans+=1.0/i;
        cout<<fixd(6)<<ans+1;
    }else cout<<fixd(6)<<double(1+0.5772+log(n-1)/log(exp(1)));
}
```

Maze

tag bfs

解题思路

容易想到 bfs。
因为沿着同一个方向走 m 步以上，所以队列中的节点要包含，坐标、方向、沿着这个方向移动的步数、以及移动的总步数。
4 2
.***
.***
.***
…

这一组数据的答案为 10。要回头走一步，然后再向前走。

考虑这一组数据可以发现，对于每一个位置不同方向不同步数的方案都要记录下来，即 bfs 的标记走过的数组要开四维。

参考代码

```
#include <bits/stdc++.h>
using namespace std;
#define LL long long
const int N = 110;
LL T, n, m, ans[N][N][4][N];
LL dx[] = {-1, 0, 1, 0}, dy[] = {0, -1, 0, 1};
char a[N][N];
struct node{
    LL x, y, d, dt, st;
};
void bfs(){
    queue <node> q;
    for (int i = 0; i < 4; i ++ )
        q.push({1, 1, i, 0, 0});
    while (q.size()){
        node t = q.front();
        q.pop();
        for (int i = 0; i < 4; i ++ ){
            LL nx = t.x + dx[i], ny = t.y + dy[i];
            if (nx < 1 || ny < 1 || nx > n || ny > n || a[nx][ny] == '*') continue;
            LL nt = t.dt + 1;    //按照当前方向移动的步数
            if (i != t.d) nt = 1;
            if (nt > m || ans[nx][ny][i][nt]) continue;
            if (nx == n && ny == n){
                cout << t.st + 1 << "\n";
                return;
            }
            ans[nx][ny][i][nt] = t.st + 1;
            q.push({nx, ny, i, nt, t.st + 1});
        }
    }
    cout << "-1\n";
}
void Clear(){
    for (int i = 1; i <= n; i ++ )
        for (int j = 1; j <= n; j ++ )
            for (int k = 0; k < 4; k ++ )
                for (int t = 1; t <= n; t ++ )
                    ans[i][j][k][t] = 0;
}
void solve(){
    cin >> n >> m;
    for (int i = 1; i <= n; i ++ )
        for (int j = 1; j <= n; j ++ )
            cin >> a[i][j];
    bfs();
    Clear();
}
int main(){
    ios::sync_with_stdio(false);cin.tie(0);
    cin >> T;
    while (T -- )
        solve();
    return 0;
}
```