Geometria Computacional

Departamento de Ciência da Computação – IME/USP

Primeiro Semestre de 2020

Lista 5

1. Seja P um polígono convexo cujos vértices estão em um vetor, como sempre no sentido anti-horário. Escreva um algoritmo que decide se um dado ponto q pertence a P ou não. O consumo de tempo do seu algoritmo deve ser $O(\lg n)$, onde n é o número de vértices de P.
2. Seja P um polígono y-monótono com n vértices. Suponha que temos em dois vetores $e[0\,..\,k-1]$ e $d[0\,..\,n-k+1]$ os índices dos vértices de P na cadeia esquerda e direita, ordenados por y-coordenadas. (Podemos obter isso em tempo linear a partir de P. Você sabe como?) Escreva um algoritmo que, dado n, P, k, e e d, decide se um dado ponto q pertence a P ou não. Seu algoritmo deve consumir tempo $O(\lg n)$.
3. Em algumas máquinas (e.g., PCs), divisão de reais pode ser cerca de 20 vezes mais lenta que multiplicação. Resolva o problema na página seguinte (Polygon, UVA, online judge, Problem 634). Aproveite o seu código para fazer alguns testes experimentais. Tenha duas implementações do algoritmo, uma com uma divisão real, no cálculo da interseção (linha 8 da segunda versão do algoritmo de ray crossing da aula sobre localização de ponto) e outra modificada, evitando a divisão. Cronometre a execução do algoritmo e verifique se há uma diferença significativa na sua máquina.
4. Argumente que, dada uma partição de uma região do plano em polígonos, a primeira versão do algoritmo de ray crossing da aula de localização de ponto classifica um ponto em no máximo um dos polígonos.
5. Torne o algoritmo de ray crossing mais rápido, evitando o cálculo da interseção quando o segmento está no lado negativo do raio.
6. Modifique o algoritmo de ray crossing para que funcione para um ponto arbitrário q e não apenas para a origem. Evite fazer a translação do polígono.
7. (a) Modifique o algoritmo de ray crossing para evitar a única operação com reais (cálculo da interseção). Use uma das primitivas vistas.
   (b) Use uma das primitivas para detectar se o ponto está numa das arestas, evitando assim a necessidade da segunda versão do algoritmo.
8. Escreva um algoritmo que determina se uma sequência de n pontos representa um polígono (ou seja, se a sequência representa uma curva poligonal fechada simples). Você consegue fazer um algoritmo $O(n \lg n)$ para isso?
9. Descreva um algoritmo $O(n \lg n)$ para decidir se dois polígonos se intersectam, onde n é a soma do número de vértices dos dois polígonos.
10. Descreva um algoritmo $O(n)$ que decide se dois polígonos convexos se intersectam, onde n é a soma do número de vértices dos dois polígonos.
11. (a) Escreva um algoritmo que usa linha de varredura para pré-processar um dado polígono P de n vértices, de forma que, com a estrutura montada, seja possível detectar se um ponto q pertence ou não a P em tempo $O(\lg n)$.
    (b) Quanto tempo consome seu algoritmo do item anterior?
    (c) Quanto espaço gasta a estrutura montada?
    (d) Escreva o algoritmo que, usando a ED montada pelo algoritmo do item (a), determine se um ponto q pertence ou não a P em tempo $O(\lg n)$.

Modern graphic computer programs can, among other, even more stunning capabilities, fill a closed region. Though not all of them can protect the user from accidentally choosing to fill the background rather than the inner part. Besides being a channel hopper at home your boss' favourite hobby is colouring the pictures, you cannot protest long about adding this magnificent protection feature to his graphic program.

This means that your job is to write a program, which determines whether a point belong to a polygon, given the array of its vertices.

To make life a bit simpler you may assume that:

- all edges of the polygon are vertical or horizontal segments
- lengths of all the edges of the polygon are even integer numbers
- co-ordinates of at least one vertex are odd integer numbers
- both co-ordinates of any vortex cannot be divisible by 7 at the same time
- the investigated point P has both co-ordinates being even integer numbers
- the polygon has at most 1000 vertices
- co-ordinates of the vertices lay in the range: -10000..10000.

Input

Input data may consist of several data sets, each beginning with a number of polygon's vertices (n). Consecutive n lines contain co-ordinates of the vertices (x followed by y). Then go the co-ordinates of investigated point P. Input data end when you find 0 the number of polygon's vertices.

Output

For each polygon and each point P you should print one character (in separate lines): '`T`' when P belongs to the polygon or '`F`' otherwise.

Sample Input

```
4
1 1
1 3
3 3
3 1
2 2
12
1 1
1 9
3 9
3 5
5 5
5 9
7 9
7 1
5 1
5 3
3 3
3 1
4 2
0
```

Sample Output

```
T
F
```