A empresa *SmartTrack* trabalha com o fornecimento de informações de partidas de futebol para clubes e empresas do segmento jornalístico. Um dos serviços fornecidos é baseado na obtenção do caminho percorrido pelo jogador a partir de imagens realizadas de pontos estratégicos do estádio. Por exemplo, a figura a seguir ilustra o posicionamento de um jogador no campo de futebol. Cada pixel da imagem corresponde a uma área real, em metros.



A imagem é binária. O pixel na cor branca representa o gramado ou qualquer outro elemento que seja considerado como fundo. O pixel na cor preta representa um local onde há um jogador. A partir do início da partida (T=0h0m0s), a cada segundo, o sistema fotografa o campo e gera uma nova imagem como o mesmo formato da imagem anterior.

Para simplificar o problema, considere que o sistema esteja monitorando apenas um jogador (haverá apenas um pixel na cor preta na imagem). O sistema identifica o jogador por um número inteiro maior ou igual a 1, cria uma pasta (diretório) para armazenar as fotografias relacionadas ao jogador e cria uma fotografia para cada instante de tempo, nomeando o arquivo com o tempo em segundos associado ao momento em que a fotografia foi criada. Por exemplo, considere três instantes de tempo diferentes e suas imagens relacionadas:

|  |  |  |
| --- | --- | --- |
| 0.png | 10.png | 50.png |
|  |  |  |

Considerando o canto superior esquerdo da imagem como a origem (0,0) de um sistema de coordenadas, teria-se o seguinte:

* Na imagem 0.png o jogador está na posição (4, 3)
* Na imagem 10.png o jogador está na posição (7, 2)
* Na imagem 50.png o jogador está na posição (16, 11)

Com base na posição do jogador em cada imagem, a empresa criou um método que calcula a distância percorrida pelo jogador durante o tempo do jogo. Por exemplo, se fossem consideradas apenas as três imagens anteriores, a distância percorrida seria obtida a partir do procedimento:

* Calcular a distância entre a posição do jogador na imagem 0.png e a imagem 10.png: (4,3) a (7,2) = 3,2
* Calcular a distância entre a posição do jogador na imagem 10.png e a imagem 50.png: (7,2) a (16,11) = 12,7
* Distância total: 3,2 + 12,7 = 15,9

O método usado para calcular a distância é a ***Distância Euclidiana***.

A relação entre o pixel e a área real à qual ele se refere é: 1 pixel = 1 metro. Desta forma, o jogador do exemplo anterior teria percorrido a distância de 15,9 metros.

O algoritmo distanciaPercorrida() calcula e retorna a distância percorrida por um jogador entre dois instantes de tempo, inicial e final.

**distanciaPercorrida(j, start, end)**

total = 0

intervalo = end - start

posições {}

para i de start até end faça

foto = Imagem(“/smarttrack/jogo0/fotografias/” + j + “/” + i + “.png”)

para x de 0 até Largura(foto) – 1 faça

para y de 0 até Altura(foto) – 1 faça

se Pixel(foto, x, y) == 0 então

posições (x, y)

fim-se

fim-para

fim-para

fim-para

para i de 1 até Tamanho(posições) - 1 faça

t1 = (posições[i][0] – posições[i + 1][0])^2

t2 = (posições[i][1] – posições[i + 1][1])^2

d = Raiz(t1 + t2)

total = total + d

fim-para

retorne total

Determine a ordem de complexidade do algoritmo distanciaPercorrida() encontrando a ordem de complexidade parcial, para cada estrutura do algoritmo.

**Observações:**

* O caminho /smarttrack/jogo0/fotografias é onde estão as fotografias citadas no enunciado
* A sintaxe posições {} indica que a lista chamada “posições” foi iniciada de forma vazia. Tem ordem de complexidade
* A sintaxe posições (x, y) indica que o valor à direita é inserido na lista posições. Tem ordem de complexidade
* Os elementos de uma lista podem ser acessados de forma indexada
* Uma lista pode ser definida usando a sintaxe (valor-1, valor-2, ..., valor-n)
* A lista é indexada a partir de 1
* Considere que há as funções
  + **Tamanho(lista)**: retorna a quantidade de elementos em lista. Tem ordem de complexidade
  + Imagem(caminho): abre a imagem que está em caminho e retorna uma matriz de pixels correspondente ao seu conteúdo. Tem ordem de complexidade
  + Largura(A): retorna a largura da imagem A (em pixels). Tem ordem de complexidade
  + Altura(A): retorna a altura da imagem A (em pixels). Tem ordem de complexidade
  + Pixel(A, i, j): retorna o valor [numérico] do pixel na coordenada na imagem A. Tem ordem de complexidade
  + Funções matemáticas, como Raiz(a,b), Quadrado(a), Seno(a), Log(a) etc têm ordem de complexidade

*Boa prova!*

*“Porque eu estou bem certo de que nem a morte, nem a vida, nem os anjos, nem os principados, nem as coisas do presente, nem do porvir, nem os poderes, nem a altura, nem a profundidade, nem qualquer outra criatura poderá separar-nos do amor de Deus, que está em Cristo Jesus, nosso Senhor” (Romanos 8.38-39).*