**Documentação Trabalho Prático**

**Nome:** Igor Henrique Martins de Almeida
**Matrícula:** 2023028536

**Espiral Quadrada**

**Idealização:**

Pela observação do desenho da espiral quadrada disponibilizada no PDF do TP, podemos perceber alguns padrões:

- Nos cantos superiores direito temos números que são quadrados perfeitos ímpares e nos cantos inferiores esquerdo temos números que são quadrados perfeitos pares.
- As coordenadas dos quadrados perfeitos ímpares podem ser calculadas por:

  x = piso$\left(\sqrt{n}/2\right)$

  y = teto$\left(\sqrt{n}/2\right)$
- As coordenadas dos quadrados perfeitos pares podem ser calculadas por:

  x = $-\sqrt{n}/2$

  y = $-\sqrt{n}/2$

Sendo $n$ um quadrado perfeito qualquer.

Com esses padrões, a ideia adotada para solução do TP foi:

Dado $coord$ como índice do ponto desejado, iremos calcular qual quadrado perfeito está mais próximo de $coord$;

Com o quadrado perfeito calculado, calcularemos as coordenadas desse quadrado perfeito, baseando-se no padrão descrito acima;

Por fim, basta calcular a diferença entre as coordenadas do quadrado perfeito encontrado e do ponto desejado "andando" pela espiral.

**Código:**

O código foi dividido em apenas duas partes dentro da função "main". No primeiro "if", verificamos se o ponto que desejamos encontrar é ímpar, caso seja, definimos as coordenadas de x e y como foi explicado acima.

```
if (quadradoPerfeito % 2 != 0){

    x = floor(raiz / 2);
    y = ceil(raiz / 2);
```

E, caso o ponto não seja um quadrado perfeito ímpar, o código entra em outro “if” e percorre a espiral dependendo se o ponto é maior ou menor que o quadrado perfeito.

```
if (quadradoPerfeito < coord){
    direcao = 2;
}
else{
    direcao = 3;
}
```

Por fim, caso o ponto seja par, fazemos o mesmo procedimento igual descrito anteriormente.

```
else{
    x = raiz / -2;
    y = raiz / -2;

    if (quadradoPerfeito < coord){
        direcao = 4;
    }
    else{
        direcao = 1;
    }
```

No segundo “if” do código, verificamos se o ponto desejado é diferente do quadrado perfeito e definimos uma variável “diferença” que armazena o módulo da diferença entre o quadrado perfeito e o ponto desejado. Nessa parte do código, dependendo da direção da espiral que será percorrida, que foi definida em partes anteriores do código, entramos em um dos casos do “switch”.

Para direção = 1, o y será somado com o valor da diferença. Isso ocorre apenas em casos em que o ponto desejado é menor que um quadrado perfeito par e maior ou igual que o quadrado perfeito menos a raiz quadrada desse mesmo número.

Para direção = 2, o x será subtraído do valor da diferença. Isso ocorre em casos em que o ponto desejado é maior que o quadrado perfeito ímpar e menor ou igual que o quadrado perfeito mais a raiz quadrada desse mesmo número.

Para direção = 3, o y será subtraído do valor da diferença. Isso ocorre em casos em que o ponto desejado é maior que um quadrado perfeito par e menor ou igual que o quadrado perfeito menos a raiz quadrada desse mesmo número.

Para direção = 4, o x será somado com o valor da diferença. Isso ocorre apenas em casos em que o ponto desejado é maior que um quadrado perfeito par e menor ou igual que o quadrado perfeito menos a raiz quadrada desse mesmo número.

```
if (quadradoPerfeito != coord){

    int diferenca = abs(quadradoPerfeito - coord);

    switch (direcao){
    case 1:
        y += diferenca;
        break;

    case 2:
        x -= diferenca;
        break;

    case 3:
        y -= diferenca;
        break;
    case 4:
        x += diferenca;
        break;

    default:
        break;
    }

    quadradoPerfeito += diferenca;
}
```

Por fim, imprimimos a coordenada do ponto desejado no terminal.

**Conclusão:**

Tomando a função sqrt() como custo 1, como combinado com o professor, o custo assintótico do código será igual a theta(1), uma vez que, o algoritmo não é sensível ao valor de $coord$. Logo, independente do valor de $coord$, as operações realizadas serão feitas em tempo constante.

**Espiral Triangular**

**Idealização:**

Assim como na espiral quadrada, podemos obsrevar alguns padrões:

- Nos pontos que estão mais à esquerda da base do triângulo temos números que são quadrados perfeitos pares e nos pontos que estão mais à direita da base do triângulo temos números que são quadrados perfeitos ímpares.
- As coordenadas dos quadrados perfeitos pares podem ser calculadas como:

  x = $-\sqrt{n}$

  y = $x/2$
- As coordenadas dos quadrados perfeitos ímpares podem ser calculadas como:

  x = $\sqrt{n}$

  y = $piso(-\ x/2)$

Sendo $n$ um quadrado perfeito qualquer.

Com essas ideias, a solução pensada para esse problema é parecida com a solução da espiral quadrada:

Com $coord$ sendo o ponto que desejamos descobrir a coordenada, iremos primeiramente calcular qual quadrado perfeito está mais próximo de $coord$;

Com $coord$ calculado, calculamos suas coordenadas;

Por fim, "andamos" pela espiral até o ponto que desejamos descobrir as coordenadas.

**Código:**

Novamente, o código para a solução da espiral triangular também foi dividido em apenas duas partes. Um "if" para descobrirmos se o quadrado perfeito mais próximo do ponto desejado é par ou ímpar e outro "if" para que possamos "andar" pela espiral e descobrir as coordenadas desse ponto.

```
if (quadradoPerfeito % 2 == 0){

    x = -sqrt(quadradoPerfeito);
    y = x / 2;

    if (raiz - floor(sqrt(coord)) < 0.5){
        direcao = 3;
    }
    else{
        direcao = 1;
    }
```

Caso o quadrado perfeito mais próximo do ponto de interesse seja par, entraremos no primeiro "if" onde deixaremos definido as coordenadas de x e y igual com base no padrão

observado. Caso o ponto não seja um quadrado perfeito par, o código entrará em um outro "if" que nos dirá para onde devemos "andar" na espiral. Caso contrário, o código entrará no "else", significando que o quadrado perfeito é ímpar, portanto definiremos suas coordenadas como visto no padrão, ou, novamente, o código nos dará uma direção que deverá ser percorrida na espiral.

```
else{
    x = sqrt(quadradoPerfeito);
    y = floor(x / -2);

    if (raiz - floor(sqrt(coord)) < 0.5){
        direcao = 2;
    }
    else{
        direcao = 4;
    }
```

Por fim, chegamos à segunda parte do código, que, com o mesmo raciocínio da espiral quadrada, iremos verificar se o ponto desejado é diferente do quadrado perfeito e, caso seja, definiremos uma variável "diferença" que receberá o módulo da diferença entre o quadrado perfeito e o ponto desejado. Nesse momento, dependendo da direção que será percorrida, definida em partes anteriores do código, o algoritmo entrará em um dos casos do "switch".

Para direção = 1, tanto x e y serão somados com o valor da diferença e a diferença será multiplicada por -1. Isso ocorre em casos em que a raiz menos o piso dessa mesma raiz for maior que 0,5 e o quadrado perfeito mais próximo for par.

Para direção = 2, x será subtraído da diferença enquanto y será somado. Isso ocorre em casos em que a raiz menos o piso dessa mesma raiz for menor que 0,5 e o quadrado perfeito mais próximo for ímpar.

Para direção = 3, x será somado a diferença. Isso ocorre em casos em que a raiz menos o piso dessa mesma raiz for menor que 0,5 e o quadrado perfeito mais próximo for par.

Para direção = 4, x será subtraído da diferença e a diferença será multiplicada por -1. Isso ocorre em casos em que a raiz menos o piso dessa mesma raiz for menor que 0,5 e o quadrado perfeito mais próximo for ímpar.

```
if (quadradoPerfeito != coord){

    int diferenca = abs(quadradoPerfeito - coord);

    switch (direcao){
    case 1:
        x += diferenca;
        y += diferenca;
        diferenca *= -1;
        break;

    case 2:
        x -= diferenca;
        y += diferenca;
        break;

    case 3:
        x += diferenca;
        break;

    case 4:
        x -= diferenca;
        diferenca *= -1;
        break;
    default:
        break;
    }

    quadradoPerfeito += diferenca;
}
```

Por fim, imprimimos a coordenada do ponto desejado no terminal.

**Conclusão:**

Novamente, tomando a função sqrt() como custo 1, o custo assintótico do código será igual a theta(1), uma vez que, o algoritmo não é sensível ao valor de $coord$. Logo, independente do valor de $coord$, as operações realizadas serão feitas em tempo constante.