Universidade Federal de Minas Gerais

Documentação do Trabalho Prático - Matemática Discreta

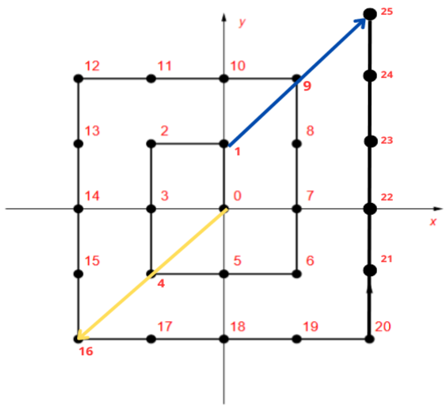
Wesley Marques Daniel Chaves

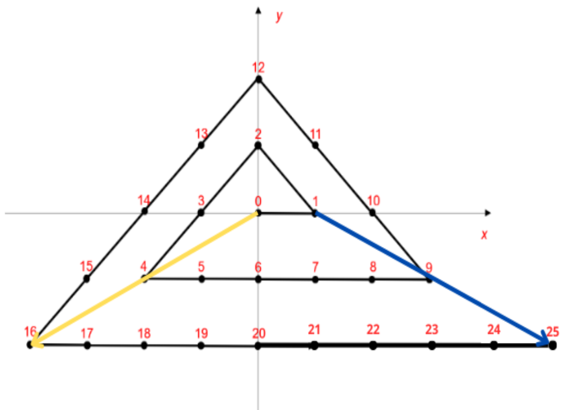
Introdução

O trabalho prático foi dividido em duas partes, uma com o problema da espiral quadrada e outra com o problema da espiral triangular. Como veremos, ambos os problemas podem ser abordados de maneiras semelhantes, com a mesma lógica subjacente e o mesmo custo computacional.

Fase de Projeto e Implementação

A solução adotada foi criada com base na identificação de padrões dos pontos na espiral quadrada e triangular, em particular, os pontos onde seu valor é um quadrado perfeito. Abaixo temos duas imagens para tornar a identificação mais clara:





Como é possível observar, os quadrados perfeitos dos inteiros não negativos estão ou sobre a reta direcionada amarela, ou sobre a reta direcionada azul. Portanto, iremos dividir o problema em dois casos distintos, com o primeiro caso em que o ponto é um quadrado perfeito, e o segundo caso em que o ponto não é um quadrado perfeito. Para ambos os problemas, temos a seguinte estrutura de código:

```c
int main(){
  int point, x, y;

  scanf("%d", &point);

  int root = (int)sqrt((double)point);

  if(root*root == point){} //quadrado perfeito
  else{} //não é quadrado perfeito

  printf("(%d,%d)\n", x, y);

  return 0;
}
```

A variável *root* é igual a raiz quadrada de *point,* caso *point* seja um quadrado perfeito, ou é igual a raiz quadrada do maior quadrado perfeito menor que *point,* devido ao truncamento ocorrido ao fazer a conversão de tipos após o cálculo da raiz quadrada não inteira.

CASO 1: O ponto é um quadrado perfeito.

Dado que o ponto é um quadrado perfeito, temos duas situações a considerar. Ou o quadrado perfeito é par, pertencendo a reta direcionada amarela, ou o quadrado perfeito é ímpar, pertencendo a reta direcionada azul. Para ambos os programas, temos:

```c
if(root*root == point){
  if(root%2 == 0){} //quadrado perfeito par
  else{} //quadrado perfeito impar
}
```

<u>*situação 1:*</u> O ponto é um quadrado perfeito par.

Espiral Quadrada

Para a espiral quadrada, a coordenada do ponto é encontrada rapidamente, bastando observar que tanto a coordenada *x* quanto a coordenada *y* são dadas pela fórmula:

$$x = y = -\frac{sqrt(ponto)}{2}$$

onde *sqrt(ponto)* é a raiz quadrada inteira do ponto.

```
if(root%2 == 0){ //quadrado perfeito par
  x = -root/2;
  y = -root/2;
}
```

Espiral Triangular

Para a espiral triangular, a coordenada do ponto também é encontrada rapidamente, porém, diferentemente da espiral quadrada, seus valores para *x* e *y* são distintos:

$$x = -sqrt(ponto) \qquad\qquad y = -\frac{sqrt(ponto)}{2}$$

onde *sqrt(ponto)* é a raiz quadrada inteira do ponto.

```
if(root%2 == 0){ //quadrado perfeito par
  x = -root;
  y = -root/2;
}
```

<u>*situação 2:*</u> O ponto é um quadrado perfeito ímpar.

Espiral Quadrada

Para este caso, as coordenadas devem ser analisadas cuidadosamente. Como a raiz quadrada de um número ímpar é um número ímpar, iremos realizar uma divisão entre inteiros no qual o numerador não é um múltiplo do denominador. Desse modo, como é obtido uma aritmética de inteiros, apenas o resultado inteiro é considerado. Temos:

$$x = \frac{sqrt(ponto)}{2} \qquad\qquad y = \frac{sqrt(ponto)}{2} + 1$$

onde *sqrt(ponto)* é a raiz quadrada inteira do ponto.

Observação: Se temos *3/2*, o resultado será *1*, e se temos *3/2* + *1*, o resultado será *2*.

```
else{ //quadrado perfeito impar
  x = root/2;
  y = root/2 + 1;
}
```

Espiral Triangular

De maneira parecida com a espiral quadrada, temos as fórmulas para as coordenadas x e y do ponto, porém, como obtemos novamente uma divisão entre números ímpares e pares, a divisão irá fornecer o resultado inteiro, desconsiderando o resto. Temos:

$$x = sqrt(ponto) \qquad\qquad y = -\frac{sqrt(ponto)}{2}$$

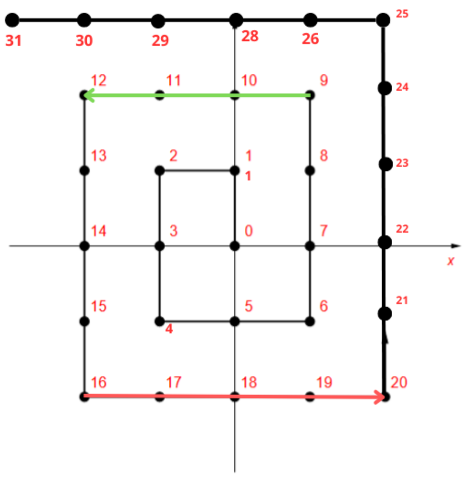
onde *sqrt(ponto)* é a raiz quadrada inteira do ponto.

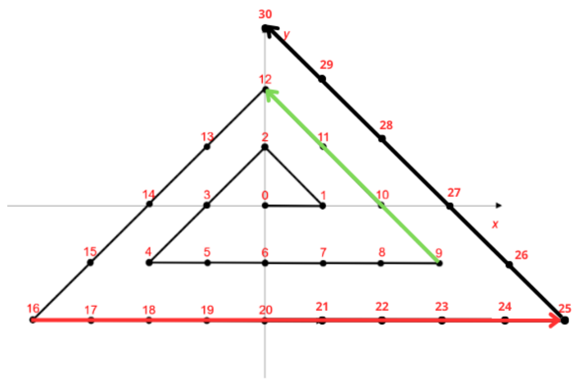
```
else{ //quadrado perfeito impar
  x = root;
  y = -root/2;
}
```

CASO 2: O ponto não é um quadrado perfeito.

A estratégia adotada neste caso para ambas as espirais foi determinar a raiz quadrada do maior quadrado perfeito menor que o ponto, e assim caminhar na espiral até obter a coordenada correta. Novamente, temos duas situações, uma em que o maior quadrado perfeito menor que o ponto é par, e uma em que o menor quadrado perfeito menor que o ponto é ímpar.

Antes de fazer a análise, observe as seguintes imagens:





Espiral quadrada

Pela imagem da espiral quadrada, é possível observar dois padrões. O primeiro nos diz que, dado um ponto quadrado perfeito ímpar, você caminha *sqrt(quadrado_perfeito)* pontos na direção e sentido de *-x* para depois caminhar na direção e sentido de *-y* (observe a reta direcionada verde). O segundo padrão nos diz que, dado um ponto quadrado perfeito par, você caminha *sqrt(quadrado_perfeito)* pontos na direção e sentido de *x* para depois caminhar na direção e sentido de *y* (observe a reta direcionada vermelha). Sabendo disso, iremos analisar cada situação.

<u>*situação 1:*</u> O maior quadrado perfeito menor que o ponto é par.

Ao determinar o maior quadrado perfeito menor que o ponto e obtermos suas coordenadas, vemos se a diferença entre o ponto e o quadrado perfeito é maior ou igual a *sqrt(quadrado_perfeito)*, e caso seja, adicionamos *sqrt(quadrado_perfeito)* na coordenada *x,* e adicionamos o restante da diferença na coordenada *y.* Caso contrário, adicionamos a diferença na coordenada x. Desse modo, caminhamos tanto na horizontal, quanto na vertical, percorrendo a espiral até a coordenada do ponto requerido.

```
else{
  int dig = root*root; //maior quadrado perfeito menor que point
  int aux = point - dig; //quanto point é maior que dig (diferença)

  if(dig%2 == 0){ //dig é par
    x = -root/2;
    y = -root/2;
    if(aux >= root){
      x += root;
      aux -= root;
      y += aux;
    }else{
      x += aux;
    }
  }
```

situação 2: O maior quadrado perfeito menor que o ponto é ímpar.

Seguimos um procedimento similar ao anterior, obtendo as coordenadas do maior quadrado perfeito menor que o ponto e vendo se a diferença entre o ponto e o quadrado perfeito é maior ou igual a *sqrt(quadrado_perfeito)*. Caso a diferença seja maior ou igual, subtraímos *sqrt(quadrado_perfeito)* da coordenada *x* e subtraímos o restante da diferença da coordenada y, e caso contrário, subtraímos da coordenada *x* a diferença. Novamente, percorremos a espiral tanto horizontalmente quanto verticalmente para obtermos a coordenada do ponto desejado.

```
  else{ //dig é impar
    x = root/2;
    y = root/2 + 1;
    if(aux >= root){
      x -= root;
      aux -= root;
      y -= aux;
    }else{
      x -= aux;
    }
  }
```

Espiral Triangular

situação 1: O maior quadrado perfeito menor que o ponto é par.

Ao determinar o maior quadrado perfeito menor que o ponto e obtermos suas coordenadas, basta adicionarmos a diferença entre o ponto e o quadrado perfeito na coordenada *x,* e manter *y* inalterado.

```
else{
  int dig = root*root; //maior quadrado perfeito menor que point
  int aux = point - dig; //quanto point é maior que dig

  if(dig%2 == 0){ //dig é par
    x = -root + aux;
    y = -root/2;
  }
```

situação 2: O maior quadrado perfeito menor que o ponto é ímpar.

Ao determinar o maior quadrado perfeito menor que o ponto e obtermos suas coordenadas, subtraímos a diferença entre o ponto e o quadrado perfeito da coordenada *x*. Para a coordenada *y,* vemos se essa diferença é maior ou igual a *sqrt(quadrado_perfeito)*, e caso seja, adicionamos *sqrt(quadrado_perfeito)* na coordenada *y* e subtraímos o restante da diferença da coordenada *y.*

```
else{ //dig é impar
  x = root;
  y = -root/2;
  if(aux >= root){
    x -= aux;
    y += root;
    aux -= root;
    y -= aux;
  }else{
    x -= aux;
    y += aux;
  }
}
```

Análise de Complexidade

Para ambos os programas, temos apenas operações simples de atribuição, soma e divisão que independem do tamanho da entrada. Considerando f(n) como o custo para se executar cada um dos algoritmos, temos:

$$f(n) = \Theta(1)$$

Ou seja, ambos os algoritmos possuem custo constante.

Observação: Para o cálculo da raiz quadrada, foi considerado que a chamada da função sqrt() da biblioteca math.h possui custo constante, como combinado com o professor da disciplina.

Discussão e Conclusão

Os problemas, mesmo sendo diferentes, foram resolvidos utilizando a mesma lógica, tendo que se alterar apenas as coordenadas dos pontos e o modo como se caminha na espiral. Como consequência, os custos exatos dos dois algoritmos são praticamente idênticos e os custos assintóticos são iguais e constantes, demonstrando um algoritmo que não altera o custo conforme aumenta o valor numérico do ponto requerido.

Por fim, deixamos abaixo duas imagens obtidas ao se plotar as coordenadas obtidas executando os programas repetidamente para os 100 primeiros pontos. Como pode-se ver, o padrão observado era o requerido inicialmente.