Universidade Federal de São Paulo - Campus São José dos Campos
Disciplina: Projeto e Análise de Algoritmos

Problema 1: Desafio da Cobra

Rosângela Miyeko Shigenari, 92334

28 de abril de 2018

1 Introdução

Este trabalho visa a resolução do Problema da Cobra em linguagem C/C++, com um tempo limite de 0,5s, e em seguida, realizar o estudo de sua complexidade.

2 Problema

O jogo clássico da cobra tem como objetivo controlar uma cobra de forma que a cobra não se choca com o seu próprio corpo enquanto percorre o mapa e eventualmente aumenta de tamanho. Baseado na ideia deste jogo, o professor Snape criou um novo problema. Dado um mapa de tamanho N x N, uma cobra que entra nesse mapa a partir da posição (1,1), continua entrando com o seu corpo de comprimento L da seguinte maneira: entra pelo canto superior esquerdo, percorre essa linha até o seu final, vira à direita e percorre essa coluna até o seu final e repete-se esse padrão enquanto todo seu corpo não estiver totalmente dentro deste mapa, sempre, na medida do possível, andando próximo às margens do mapa e sem se chocar com o seu corpo. A figura abaixo mostra um caso para N = 8 e L = 53, sendo que cada unidade de comprimento da cobra cabe em um dos quadrados do mapa e, logo, a cobra irá ocupar L quadrados ao final de seu ingresso no mapa. O desafio consiste em, dados os valores de N e L, determinar rapidamente qual será a posição final alcançada pela cabeça da cobra quando a cobra conseguir entrar complementamente no espaço do mapa (figura 1). No exemplo, a cobra alcançou a posição de coordenadas (4,6).

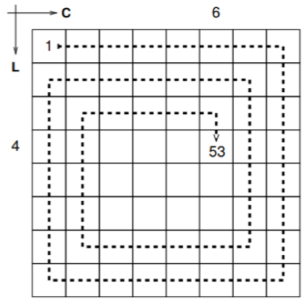

Figura 1: Tabuleiro

Entrada:

A entrada do problema consiste em uma linha contendo dois inteiros, N ($1 \leq N \leq 1.073.741.824$) e L ($1 \leq L \leq$ N2), separados por um espaço.

Saída: Seu programa deve imprimir uma linha contendo as coordenadas de linha e coluna que a cabeça da cobra irá alcançar ao final de seu ingresso no mapa.

Exemplos de entrada e saída

- **Entrada**: 8 53
  Saída: 4 6
- **Entrada**: 100000 5000000
  Saída: 99988 99363
- **Entrada**: 45008000 8000392000
  Saída: 33983877 45007956

3 Solução e Análise do Algoritmo

A solução encontrada foi baseada principalmente no reconhecimento de padrões que se repetiam, no qual relacionava o número de voltas que a cobra havia dado com a entrada N, podendo assim utilizar funções com 2 variáveis: N (tamanho do tabuleiro e voltas (número de voltas dados pela cobra no tabuleiro).

- **Calculo do número da volta**
  O N é a entrada do usuário, portanto, primeiramente, foi necessário calcular o número de voltas realizadas pela cobra, sendo implementado um laço, apresentado na figura 2

```
while (soma>0) {

  soma = soma - (4*Naux-4);
  Naux = Naux - 2;
  volta ++;
}
```

Figura 2: Laço While

  Foram utilizadas 2 variáveis do tipo *long long int i* e *j*, que são, respectivamente, o número da linha e da coluna no tabuleiro, sendo assim a coordernada representada por (i,j). Denotando a coordenada da cabeça da cobra por (linha, coluna), para o problema foi definido que sempre o início de uma volta é dado em (volta, volta) e terminado em (volta, volta+1).
  Realizando testes com entradas pequenas, foi verificado sempre um padrão, o número de quadrados percorridos a cada volta é de $4 \times N - 4$, onde o N é decrementado em 2 a cada volta (armazenado em *Naux*). Por exemplo, para o exemplo de entrada e saída 1, N = 8, portanto na primeira volta a cobra percorreu $4 \times 8 - 4 = 28$, na segunda volta, $4 \times (8-2) - 4 = 20$, e assim por diante.
  Levando essas observações em consideração, foi criada uma variável *soma* que é inicializada com o valor de L (comprimento da cobra). No laço *while* da figura 2, o valor da *soma* é decrementado com o percorrido a cada volta, até que chegue ao "rabo"da cobra, ou seja, quando a soma for menor ou igual a zero; um contador *volta* foi utilizado para contar o número de vezes que o algoritmo entra no laço, ou seja, quantas voltas há entre a "cabeça"e o "rabo"da cobra. Portanto o custo desse trecho é de **O(volta)**.

- **Quantidade de quadrados percorridos até a volta**
  Assim, sabemos a volta em que a "cabeça"da cobra se encontra, o próximo passo é encontrar a coordenada correspondente. A partir da análise de quantos quadrados foram percorridos a cada volta, podemos observar uma progressão aritmética onde o primeiro termo é $4 \times N - 4$ e o termo correspondente a volta *"volta"* é $4 \times (N - 2*\text{volta} + 2) - 4$, sendo possível então realizar a soma dos quadrados percorridos:

$$S_{volta} = \frac{a_1 + a_{volta}}{2}$$
$$\implies S_{volta} = \frac{4 \times N - 4 + [4 \times (N - 2 \times volta + 2) - 4]}{2}$$
$$\implies S_{volta} = (4\times \text{ N - } 4\times \text{ volta})\times \text{ volta}$$
$$\implies S_{volta} = 4\times\text{volta (N - volta)}$$

Foi criada também a variável *diferenca*, que recebe o valor de (L - *somaDaVolta*), ou seja, ela armazena quantos quadrados são necessários retornar do final da volta, que é a coordenada (volta, volta + 1), até a cabeça da cobra, podendo assim, calcular o valor das posições i e j correspondente à ela.

- **Percorrendo o Tabuleiro**
  Como descrito no item anterior, a variável diferença, armazena quantos quadrados são necessários andar no tabuleiro até encontrar a cabeça da cobra. Para poder andar no tabuleiro é necessário obedecer 2 condições: não é permitido chocar com o corpo da cobra e nem ultrapassar os limites do tabuleiro. Com essas premissas, foram implementados diversos *if*s, para cada um dos 4 sentidos permitidos a percorrer. Por se tratar de comparações cada um possui a complexidade constante $\theta(1)$.

  - **Situação 1: cabeça no último quadradado correspondente à volta**
    Na figura 3 é mostrado o trecho de código em que a variável *diferenca* é zero, ou seja, o tamanho da cobra é correspondente a quantidade de quadrados existentes na volta calculada.

```
i = volta+1;
j= volta;

diferenca = somaDaVolta - L;

if(diferenca == 0){

  i = volta+1;
  j= volta;
}
```

Figura 3: Situação 1

Como descrito anteriormente, o final da volta é a coordenada (volta, volta+1), portanto a cabeça da cobra está situada em i = volta e j = volta +1.

  - **Situação 2: cabeça na descida do tabuleiro** Para este caso, temos que a variável *diferenca* é maior que zero, portanto, devemos descer as posições no tabuleiro, sem chocar com o corpo. Implementada a seguinte comparação da figura 4.

```
else {
  aux = diferenca - (N-2*volta);//
  if(aux>=0){//desce
    i = i+ (N-2*volta);
    diferenca = diferenca - (N-2*volta);
    mode = 1;
  }
  else
    i = i + diferenca;
  }
```

Figura 4: Situação 2

Podemos observar pela figura acima a condição de descida no tabuleiro, como o deslocamento é dado apenas verticalmente o valor de j se manterá e i será incrementado. Primeiramente, é calculado o quanto é possível voltar as posições sem chocar com o corpo, foi observado um padrão de (N - 2×volta). Portanto as posições são percorridas, podendo resultar em mais 2 condições, se a o máximo de quadrados permitidos for percorrido e a variável *diferenca* for maior que zero, ou seja ainda há posições para serem visitadas, é necessário percorrer à direita do tabuleiro, entrando no *mode 1*, senão a posição de i será um valor na descida, onde i = i + *diferenca*.

- **Situação 3: cabeça no caminho à direita**
  Da mesma forma, para o caminho à direita no tabuleiro, é calculada a função para que não choque com o corpo ou ultrapasse o tamanho do tabuleiro, a implementação da condição é dada pela figura 5. A função encontrada a partir da análise pela pequenos valores de N é de (N + 1) - (2 × volta).

```
if(mode == 1){//direita
  if((diferenca - ((N+1)-(2*volta)))>=0){
  j = j+ ((N+1)-(2*volta));
  diferenca = diferenca - ((N+1)-(2*volta));
  mode = 2;// sobe o tabuleiro
}
else
  j = j + diferenca;
}
```

  Figura 5: Situação 3

  Como o deslocamento se dá apenas horizontalmente, o valor de i se mantém da condição anterior e o j é incrementado. Se ainda houver posições a serem percorridas, entrará para o *mode 2*, senão, a variável j é atualizado percorrendo o valor necessário, ou seja, j = j +*diferenca*.

- **Situação 4: cabeça na subida do tabuleiro**
  Para a condição de subida, foi observado um mesmo padrão que no item anterior, (N + 1) - (2 × volta), como pode ser observado na figura 6.

```
if(mode ==2){//sobe o tabuleiro
  if(diferenca - ((N+1)-(2*volta))>=0){
    i = i - ((N+1)-(2*volta));
    diferenca = diferenca - ((N+1)-(2*volta));
    mode = 3;
}
  else
    i = i - diferenca;
  }
```

  Figura 6: Situação 4

  Como será percorrido posições verticamente para cima do tabuleiro, apenas o valor de i será decrementado e o j continuará o mesmo da condição anterior. Se ainda houver posições a serem percorridas, entrará em um último modo, *mode 3*.

- **Situação 5: cabeça no caminho à esquerda** Como o valor do número de voltas foi calculado inicialmente, a cabeça da cobra certamente estará nesta volta, portanto, se a cabeça não estava em nenhum dos caminhos anteriores, ela garantidamente estará no caminho à esquerda. O *mode 3* é mostrado na figura 7.

```
      if(mode==3){esquerda
        j = j - diferenca;
      }
  }
    printf("%lld %lld\n",i, j);//imprime a posicao i e j no tabuleiro
```

  Figura 7: Situação 5

  Como serão percorridas posições horizontalmente, o valor de i continuará a mesma de *mode 2*, apenas o j será decrementado com o valor da variável *diferenca*.
  Finalmente, o valor de i e j serão impressos, que são correspondentes a posição da cabeça da cobra no tabuleiro.

- **Caso Extremo**
  Para o caso extremo de L, temos que $L = N^2$, no qual as condições anteriores não são válidas, pois o cálculo das posições é realizado sempre a partir do final da volta correspondente e em seguida, deslocamos as posições no tabuleiro até chegar à posição desejada, como o último quadrado visitado sempre será o meio do tabuleiro, o algoritmo não conseguirá encontrar o valor da coordenada do final dessa volta,

ocasionando problema de *time limit exceeded*.
Assim, foi necessário tratar essas condições, para que todos os testes fosse aceitos. No início do algoritmo foi realizado uma condição de verificação se $L = N^2$, como mostrado na figura 8.

```
if(L ==(N*N)){//verifca o caso extremo L = N^2
  if(N%2 == 0){
    i = N/2 + 1;
    j = N/2;
  }//se for impar, sera o meio do tabuleiro
  else{
    i = (N+1)/2;
    j = (N+1)/2;
  }
}
```

Figura 8: Caso extremo: $L = N^2$

Como observado acima, para valores **ímpares** de N, as posições sempre referentes a i e j serão exatamente o meio do tabuleiro $\frac{N+1}{2}$, para os valores **pares** será $\frac{N}{2} + 1$ e $\frac{N}{2}$, respectivamente.

4 Complexidade

A partir da solução acima, podemos observar que se L for o caso extremo, será realizada apenas uma comparação, retornando os valores de i e j, caso contrário, o custo será o da iteração do laço *while*, que é o número de voltas mais constantes, provenientes das comparações realizadas para andar no tabuleiro sem chocar com o corpo da cobra. Logo, a complexidade deste algoritmo pode ser descrito como:

$$Complexidade = \begin{cases} \theta(1), & \text{se} \quad L = N^2 \\ O(volta), & \text{caso contrário} \end{cases}$$