Universidade Federal de Ouro Preto - UFOP
Departamento de Ciência da Computação - DECOM

Relatório atividade 9 - Hanoi Tower Troubles Again!

BCC402 - ALGORITMOS E PROGRAMACAO AVANCADA

Kayo Xavier Nascimento Cavalcante Leite - 21.2.4095

Professor: Rafael Alves Bonfim

Ouro Preto
31 de março de 2025

Sumário

Lista de Códigos Fonte

1 Código e enunciado.

Na Atividade 9 - o problema selecionado foi Hanoi Tower Troubles Again!. O objetivo é posicionar bolas numeradas sequencialmente (1, 2, 3, ...) em N hastes, seguindo a regra: a soma dos números de duas bolas consecutivas em uma mesma haste deve ser um quadrado perfeito. O jogo termina quando uma bola não pode ser colocada. O desafio é determinar o número máximo de bolas que podem ser posicionadas para um dado N. O código comentado e documentado, casos de teste e executável pré compilado se encontram no .zip da atividade. O código foi feito com base na referência encontrada no site:

`https://github.com/evandrix/UVa/blob/master/10276.cpp`

Caso queira, para rodar e compilar o código, é necessário ter o compiler g++ e utilizar o seguinte comando no terminal dentro do diretório da pasta da atividade específica:

Compilando e rodando o exercício

```
para compilar:
g++ Hanoi.cpp -o executavel

e para rodar basta utilizar .\executavel no cmd.

para utilizar os cenários de teste:
.\executavel < sampleinput.txt
.\executavel < testinput.txt
```

2 Problema 10276: Hanoi Tower Troubles Again!

2.1 Descrição do Problema

O objetivo é posicionar bolas numeradas sequencialmente (1, 2, 3, ...) em N hastes, seguindo a regra: a soma dos números de duas bolas consecutivas em uma mesma haste deve ser um quadrado perfeito. O jogo termina quando uma bola não pode ser colocada. O desafio é determinar o número máximo de bolas que podem ser posicionadas para um dado N.

2.2 Entrada e Saída

- **Entrada**: Um inteiro T (número de casos de teste), seguido por T linhas com um inteiro N (número de hastes).
- **Saída**: Para cada caso, o número máximo de bolas posicionáveis. Retorna -1 se infinitas bolas forem possíveis (não aplicável para $N \leq 50$).

2.3 Estratégia de Solução

O problema é resolvido utilizando uma sequência pré-calculada baseada em padrões matemáticos:

1. **Pré-cálculo de Padrões**: O código utiliza um array `pesos` para armazenar valores que seguem uma sequência específica (1, 2, 4, 4, 6, 6, ...). Cada valor representa um incremento no número de bolas adicionadas por haste.
2. **Soma Acumulada**: Um array `bolas` armazena a soma acumulada dos valores de `pesos`. O resultado para N hastes é dado por `bolas`$[N] - 1$.
3. **Otimização**: A solução aproveita um padrão observado empiricamente, onde a sequência de máximos para N hastes é conhecida e pré-computada.

2.4 Análise Matemática

A sequência de máximos é definida pela seguinte relação:

$$\texttt{bolas}[i] = \sum_{k=0}^{i-1} \texttt{pesos}[k]$$

onde $\texttt{pesos}[k]$ é definido como:

$$\texttt{pesos}[k] = \begin{cases} 1 & \text{se } k = 0 \\ 2 & \text{se } k = 1 \\ k+2 & \text{se } k \geq 2 \text{ e } k \text{ par} \\ \texttt{pesos}[k-1] & \text{se } k \text{ ímpar} \end{cases}$$

O resultado final é $\texttt{bolas}[N] - 1$, ajustando o valor acumulado para refletir a contagem correta de bolas.

```
// Objetivo: Pre-calcular uma sequencia de numeros para responder consultas
   rapidamente.

// 1. Pre-calculo da Sequencia "Pesos":
//    a. Criar um array `Pesos` de tamanho 52.
//    b. Definir `Pesos[0] = 1`, `Pesos[1] = 2`.
//    c. Para indices pares `i` de 2 ate 50:
//       - `Pesos[i]` recebe `i + 2`.
//       - `Pesos[i+1]` (indice impar seguinte) tambem recebe `i + 2`.
//    d. Ao final, `Pesos` contera [1, 2, 4, 4, 6, 6, ..., 52, 52].

// 2. Pre-calculo da Sequencia "Bolas" (Soma Acumulada):
//    a. Criar um array `Bolas` de tamanho 52.
//    b. Definir `Bolas[0] = 1`.
//    c. Para indices `i` de 1 ate 51:
//       - `Bolas[i]` recebe `Bolas[i-1] + Pesos[i-1]`.
//       - (Cada `Bolas[i]` acumula o valor anterior mais o `Peso`
   correspondente).

// 3. Processamento das Consultas:
//    a. Ler a quantidade de casos de teste (`QuantCasos`).
//    b. Para cada caso de teste (repetir `QuantCasos` vezes):
//       i. Ler o numero de entrada `n`.
//       ii. A resposta e o valor pre-calculado `Bolas[n]` subtraido de 1.
//       iii. Imprimir (`Bolas[n] - 1`).

// Programa Principal: Executa os passos 1, 2 e 3.
```

Código 1: Pseudocódigo do problema.

3 Casos teste - Input e output esperado.

Para os casos de teste do problema, foi disponibilizado junto a pasta do mesmo os seguintes arquivos :sampleinput.txt e testinput.txt, sendo o primeiro o próprio caso de teste disponibilizado pelo exercício e o segundo caso de teste encontrado na plataforma https://www.udebug.com/. Além disso, encontra-se também o arquivo com os outputs esperados para cada input. Ambos os resultados foram validados e tiveram o output esperado.

sampleinput.txt

```
2
4
25
```

output esperado

```
11
337
```

Os demais testes se encontram no diretório da atividade