



UFOP

Universidade Federal
de Ouro Preto

Recursão

CSI030-PROGRAMAÇÃO DE COMPUTADORES I





UFOP

Universidade Federal
de Ouro Preto

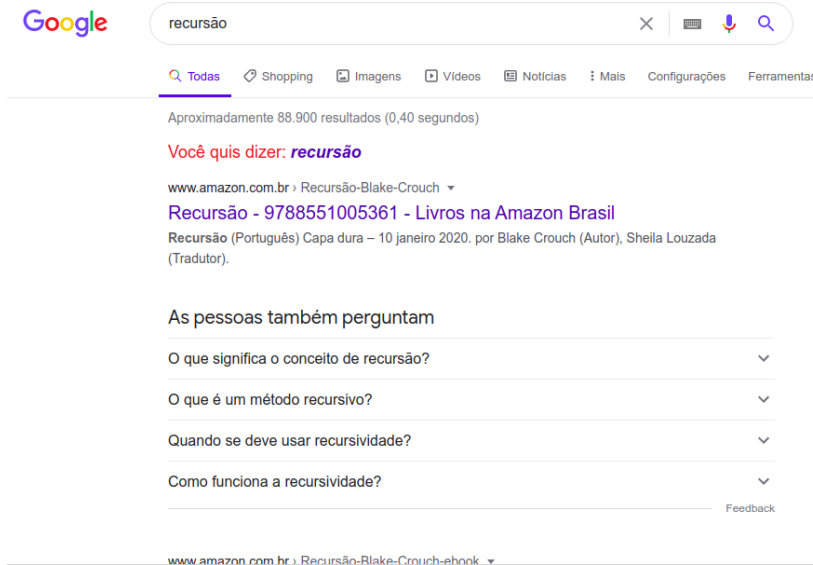
VideoAula

A videoaula do material a seguir pode ser encontrado em:

<https://www.youtube.com/playlist?list=PLKQEI0z2LK0FOWL5DQajyQ69vESiYT6BS>

Introdução

- O que é Recursão?



A screenshot of a Google search results page for the query "recursão". The Google logo is on the left. The search bar contains "recursão" with a clear button (X), a keyboard icon, a voice search icon, and a search icon. Below the search bar are tabs for "Todas" (selected), "Shopping", "Imagens", "Vídeos", "Notícias", "Mais", "Configurações", and "Ferramentas". The results show "Aproximadamente 88.900 resultados (0,40 segundos)". A suggestion "Você quis dizer: **recursão**" is shown. The first result is from "www.amazon.com.br" for the book "Recursão - 9788551005361 - Livros na Amazon Brasil" by Blake Crouch and Sheila Louzada. Below the results, a section titled "As pessoas também perguntam" contains four expandable questions: "O que significa o conceito de recursão?", "O que é um método recursivo?", "Quando se deve usar recursividade?", and "Como funciona a recursividade?". A "Feedback" link is at the bottom right of this section. At the very bottom, another result from "www.amazon.com.br" for "Recursão-Blake-Crouch-ebook" is partially visible.



UFOP
Universidade Federal
de Ouro Preto

O que é Recursão?

- É um termo usado de maneira geral para descrever o processo de repetição de um objeto de um jeito similar ao que já fora mostrado.

O que é Recursão?



UFOP

Universidade Federal
de Ouro Preto

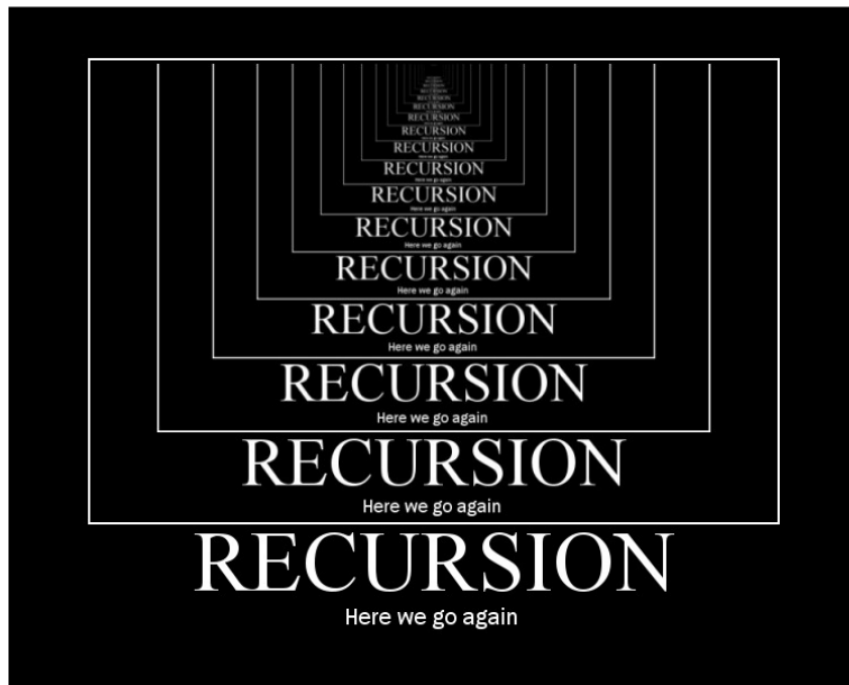


O que é Recursão?



UFOP

Universidade Federal
de Ouro Preto





O que é Recursão na Computação?

- Recursão é uma técnica de resolução de problemas em que os problemas são resolvidos através da redução desses problemas a problemas menores que possuem a mesma forma do problema original.



UFOP

Universidade Federal
de Ouro Preto

Recursão na vida real

- Podemos usar recursão para descrever de forma simples a solução para problemas complicados da vida real.



Quantas pessoas estão nessa fila?

- Imagine que você está em uma fila muito grande.
- Você gostaria de saber quantas pessoas estão na fila.
- Você não pode sair do seu lugar para contar e não é possível enxergar toda a fila.
- Você pode fazer perguntas para a pessoa à sua frente e para pessoa atrás.



Solução

- Pergunte para a pessoa à sua frente quantas pessoas têm na frente dela
 - Se ela não souber,
 - Ela deve perguntar para a pessoa à frente dela.
 - Se ela não souber,
 - Ela deve perguntar para a pessoa à frente dela.
 - Se ela não souber,
 - Ela deve perguntar para a pessoa à frente dela.
 - ...
 - Pessoa do início da fila.
- Pergunte para a pessoa atrás de você quantas pessoas têm atrás dela.
 - Se ela não souber, ela deve perguntar para a pessoa atrás dela.
 - ...

Torre de Hanói



UFOP

Universidade Federal
de Ouro Preto

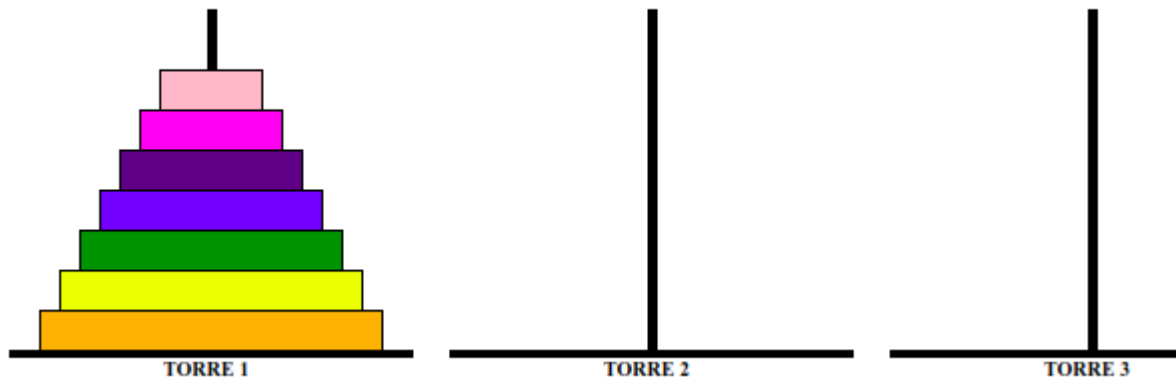
- “O famoso jogo da Torre de Hanói é um quebra-cabeça que consiste em uma base contendo três pinos, em um dos quais são dispostos alguns discos uns sobre os outros, em ordem crescente de diâmetro, de cima para baixo”

Torre de Hanói



UFOP

Universidade Federal
de Ouro Preto



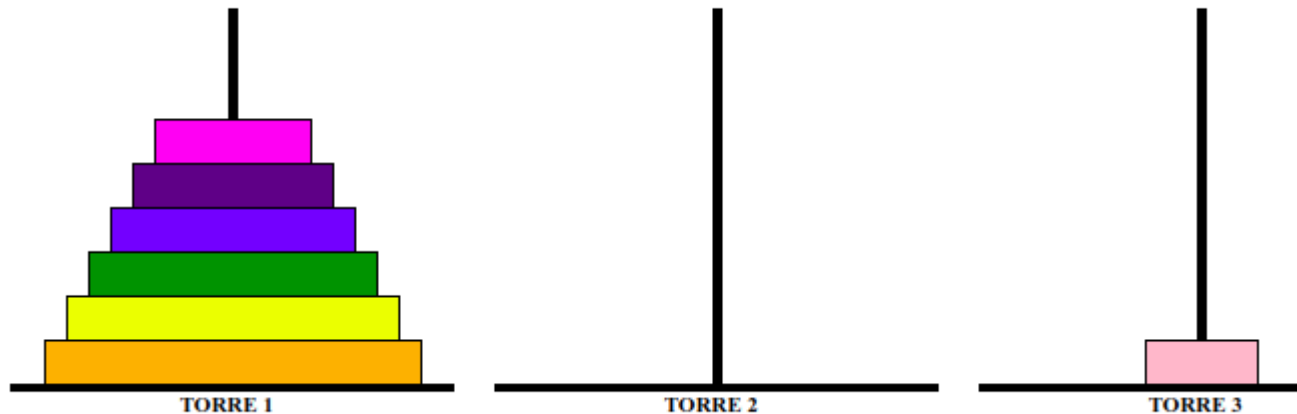


Torre de Hanói

- Objetivo: mover todos os discos para o pino da direita.
- Você deve mover um disco de cada vez, sendo que um disco maior nunca pode ficar em cima de um disco menor.
- Vocês estão convidados a jogar um pouco:
 - <https://www.somatematica.com.br/jogos/hanoi/>

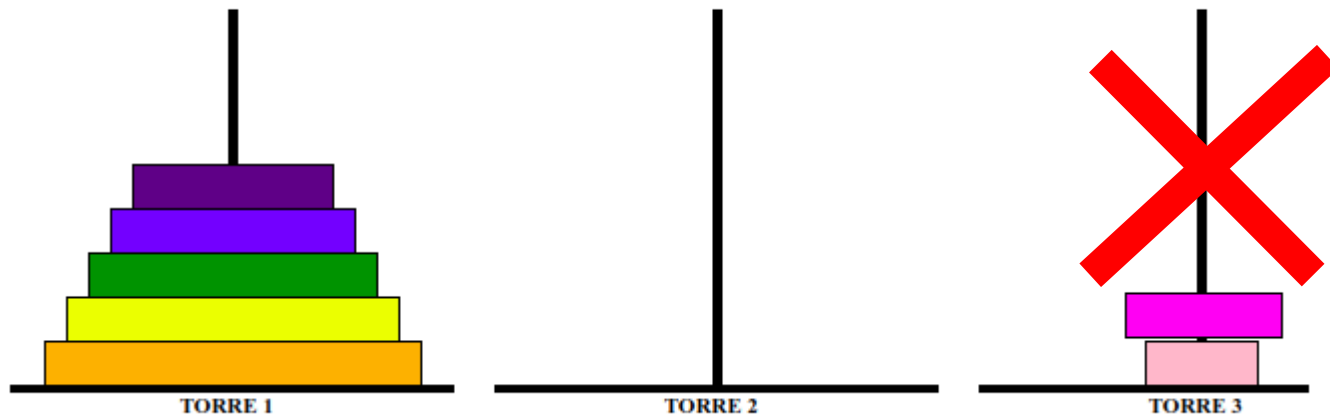
Vamos jogar!

- Temos 7 discos, mover da Torre 1 para Torre2



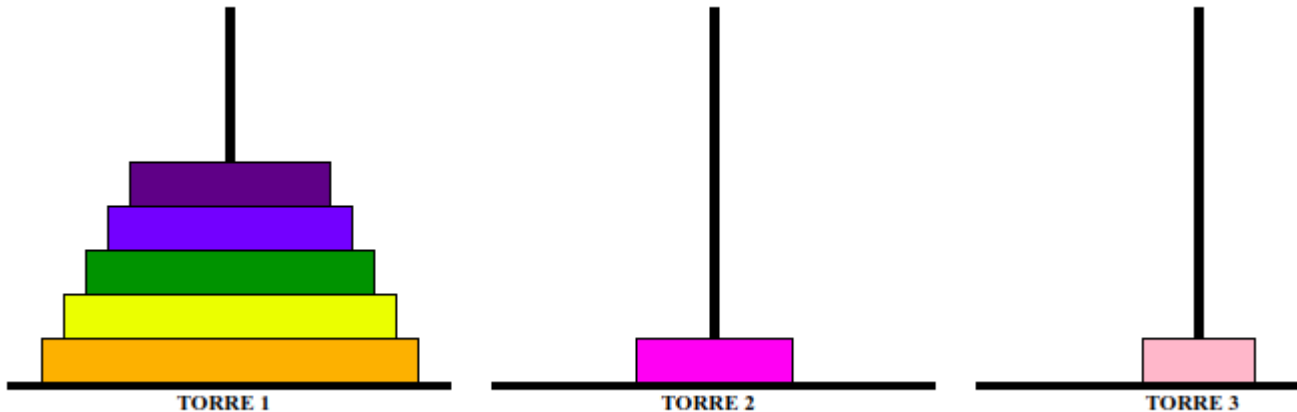
Vamos jogar!

- Não é permitido.



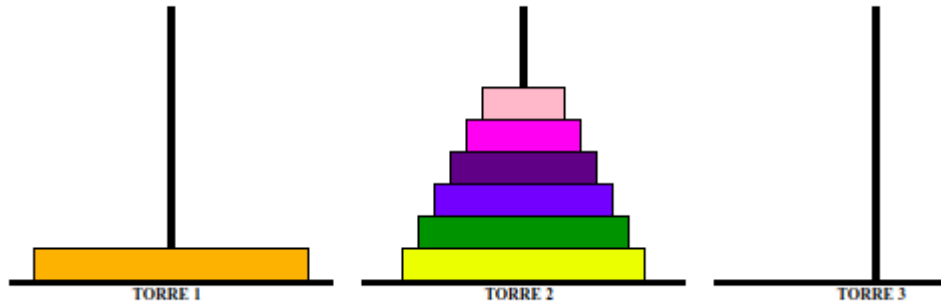
Vamos jogar!

- Movimento permitido.



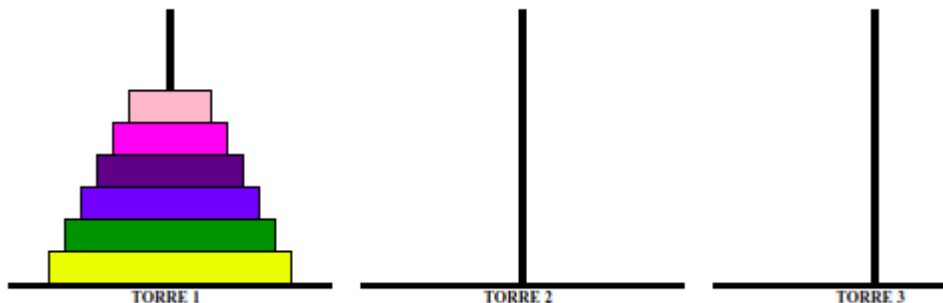
Solução

- Para conseguirmos mover o último disco para o destino, tenho que mover todos os discos menores para o pino do meio.



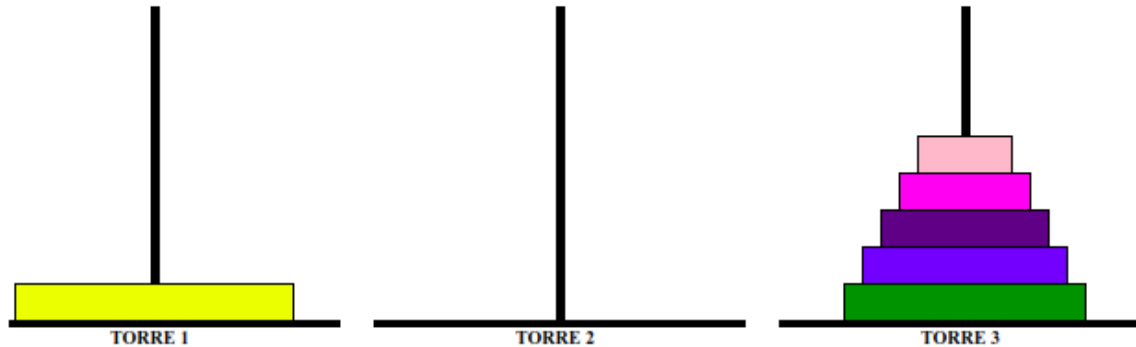
Solução

- Logo, tenho que mover uma torre de 6 disco da Torre 1 para Torre 2



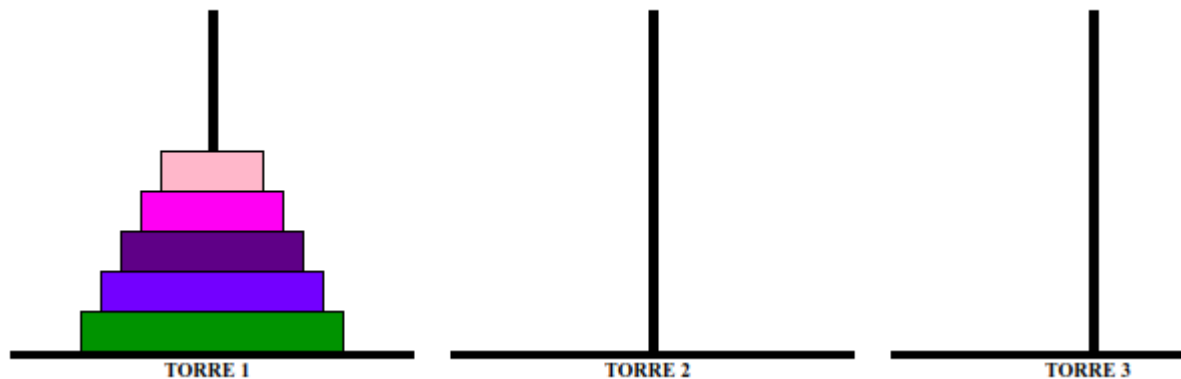
Solução

- Para conseguir mover o último disso dessa torre, temos que mover todos os discos menores para Torre 3.



Solução

- Tenho que mover uma torre de 5 disco da Torre 1 para Torre 3





UFOP

Universidade Federal
de Ouro Preto

Solução

- Tenho que mover uma torre de 5 disco da Torre 1 para Torre 3
- Tenho que mover uma torre de 4 discos da Torre 1 para Torre 2
- Tenho que mover uma torre de 3 discos da Torre 1 para Torre 3
- Tenho que mover uma torre de 2 discos da Torre 1 para Torre 2
- Tenho que mover uma torre de 1 disco da Torre 1 para Torre 3.
 - Mover o disco



Solução

- Temos que voltar agora movendo os discos e movendo as torres menores pelo mesmo processo.

Recursão em C

- Uma função é dita recursiva quando dentro do seu código existe uma chamada para si mesma.



UFOP

Universidade Federal
de Ouro Preto

Fatorial

- Podemos definir o fatorial recursivamente como:
 - $n! = n \cdot (n-1)!$

```
int Fatorial(int n){  
    if(n == 0)  
        return 1;  
    else  
        return n * Fatorial(n-1);  
}
```


Cuidado!!!!

- A main não pode ser recursiva: Segmentation fault!!!!

```
#include <stdio.h>
```

```
int main(){  
    main();
```

```
}
```



Recursão

- Todo algoritmo recursivo tem que ter pelo menos dois casos:
 - **Caso Base:** Uma ocorrência que pode ser respondida diretamente (solução trivial)
 - **Caso recursivo ou Passo recursivo:** Uma ocorrência mais complexa do problema que não pode ser respondida diretamente, mas pode ser descrita em termos de ocorrências menores do mesmo problema.



Recursão

Caso Base

```
int Fatorial(int n){
```

```
    if(n == 0)
```

```
        return 1;
```

```
    else
```

```
        return n * Fatorial(n-1);
```

Passo Recursivo

```
}
```

Recursão: Critério de parada

CSI030-PROGRAMAÇÃO DE COMPUTADORES I



UFOP
Universidade Federal
de Ouro Preto

Recursão

- Em procedimentos recursivos pode ocorrer um problema de terminação do programa, como um “looping interminável ou infinito”.



UFOP

Universidade Federal
de Ouro Preto

Três obrigações da Recursão

- Seu código deve ter um caso para todas as entradas válidas
- Você deve ter um caso base que não faça chamadas recursivas.
- Quando você faz uma chamada recursiva, deve ser para um caso mais simples e progredir em direção ao caso base.





Potenciação

- Escreva uma função recursiva que receba um número inteiro x e um expoente inteiro n e retorne a potência de x elevado a n .
 - Caso Base: $\text{pow}(x,0) = 1$
 - Passo Recursivo: $\text{pow}(x,n) = x * \text{pow}(x,n-1)$



UFOP

Universidade Federal
de Ouro Preto

Potenciação

```
int pow(int x, int n){  
    if(n == 0)  
        return 1;  
    else  
        return x*pow(x, n-1);  
}
```




Chamada de funções e a memória

- Toda vez que uma função é chamada, suas variáveis locais são armazenadas no topo da pilha.
- Quando uma função termina, suas variáveis locais são removidas da pilha.
- A execução de uma função deixa no topo da pilha o resultado da função.
- Cada chamada de uma função recursiva é uma nova chamada de função no topo da pilha
- Vamos fazer o rastreo para $\text{pow}(5,3)$!!



Potenciação

```
int pow(int x, int n){  
    if (n == 0)  
        return 1;  
    else  
        return x*pow(x, n-1);  
}
```

Pilha

x = 5	n = 3	

resposta = pow(5,3)



Potenciação

```
int pow(int x,int n){  
    if(n == 0)  
        return 1;  
    else  
        return x*pow(x,n-1);  
}
```

Pilha

x = 5	n = 3	pow(5,2)
resposta = pow(5,3)		



Potenciação

```
int pow(int x,int n){  
    int pow(int x,int n){  
        if(n == 0)  
            return 1;  
        else  
            return x*pow(x,n-1);  
    }  
}
```

Pilha

x = 5	n = 2	
x = 5	n = 3	pow(5,2)

resposta = pow(5,3)

Potenciação

```
int pow(int x,int n){  
    int pow(int x,int n){  
        if(n == 0)  
            return 1;  
        else  
            return x*pow(x,n-1);  
    }  
}
```



UFOP

Universidade Federal
de Ouro Preto

Pilha

x = 5	n = 2	pow(5,1)
x = 5	n = 3	pow(5,2)

resposta = pow(5,3)

Potenciação

```
int pow(int x,int n){  
    int pow(int x,int n){  
        int pow(int x,int n){  
            if(n == 0)  
                return 1;  
            else  
                return x*pow(x,n-1);  
        }  
    }  
}
```

Pilha

x = 5	n = 1	
x = 5	n = 2	pow(5,1)
x = 5	n = 3	pow(5,2)

resposta = pow(5,3)

Potenciação

```
int pow(int x,int n){  
    int pow(int x,int n){  
        int pow(int x,int n){  
            if(n == 0)  
                return 1;  
            else  
                return x*pow(x,n-1);  
        }  
    }  
}
```

Pilha

x = 5	n = 1	pow(5,0)
x = 5	n = 2	pow(5,1)
x = 5	n = 3	pow(5,2)

resposta = pow(5,3)



UFOP

Universidade Federal
de Ouro Preto

Potenciação

```
int pow(int x,int n){  
    int pow(int x,int n){  
        int pow(int x,int n){  
            int pow(int x,int n){  
                if(n == 0)  
                    return 1;  
                else  
                    return x*pow(x,n-1);  
            }  
        }  
    }  
}
```

Pilha

x = 5	n = 0	
x = 5	n = 1	pow(5,0)
x = 5	n = 2	pow(5,1)
x = 5	n = 3	pow(5,2)

resposta = pow(5,3)



Potenciação

```
int pow(int x,int n){  
    int pow(int x,int n){  
        int pow(int x,int n){  
            int pow(int x,int n){  
                if(n == 0)  
                    return 1;  
                else  
                    return x*pow(x,n-1);  
            }  
        }  
    }  
}
```

Pilha

x = 5	n = 0	return 1
x = 5	n = 1	pow(5,0)
x = 5	n = 2	pow(5,1)
x = 5	n = 3	pow(5,2)

resposta = pow(5,3)

Potenciação

```
int pow(int x,int n){  
    int pow(int x,int n){  
        int pow(int x,int n){  
            if(n == 0)  
                return 1;  
            else  
                return x*pow(x,n-1);  
        }  
    }  
}
```

Pilha

x = 5	n = 1	1
x = 5	n = 2	pow(5,1)
x = 5	n = 3	pow(5,2)

resposta = pow(5,3)

Potenciação

```
int pow(int x,int n){  
    int pow(int x,int n){  
        if(n == 0)  
            return 1;  
        else  
            return x*pow(x,n-1);  
    }  
}
```



UFOP

Universidade Federal
de Ouro Preto

Pilha

x = 5	n = 2	5
x = 5	n = 3	pow(5,2)

resposta = pow(5,3)



Potenciação

```
int pow(int x,int n){  
    if(n == 0)  
        return 1;  
    else  
        return x*pow(x,n-1);  
}
```

Pilha

x = 5	n = 3	25

resposta = pow(5,3)

Potenciação



UFOP

Universidade Federal
de Ouro Preto

Pilha

resposta = 125

Potenciação

- Como n está sempre reduzindo
 - É fácil perceber que em algum momento vamos atingir o 0 e essa função vai parar.



Problema de Collatz

A seguinte função recursiva vai parar?

```
void Collatz(int n) {  
    printf("n no momento: %d \n",n);  
    if(n == 1) {  
        return;  
    } else {  
        if(n % 2 == 0) {  
            Collatz(n / 2);  
        } else {  
            Collatz(3 * n + 1);  
        }  
    }  
}
```



UFOP

Universidade Federal
de Ouro Preto

Problema de Collatz

- Os matemáticos não conseguiram provar que ela para ou não.
- Ela foi testada para os valores até $5e+18$.





UFOP

Universidade Federal
de Ouro Preto

Exemplos: Recursão

CSI030-PROGRAMAÇÃO DE COMPUTADORES I

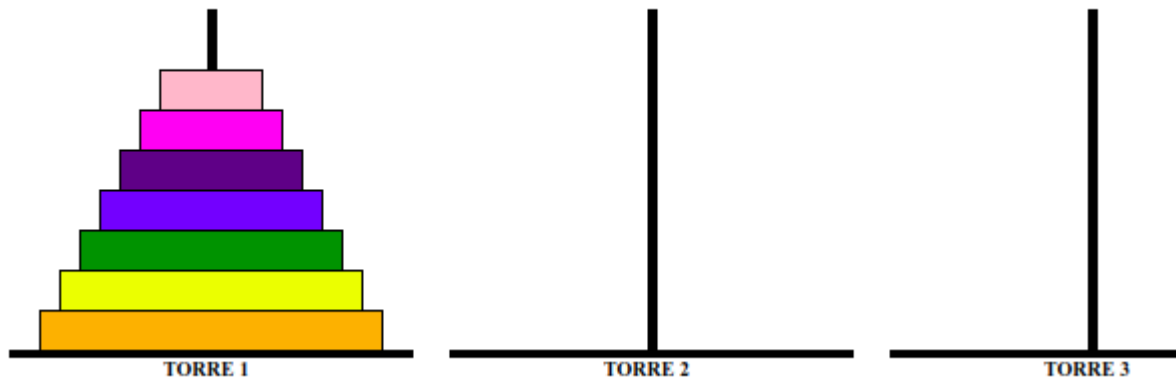


Torre de Hanói



UFOP

Universidade Federal
de Ouro Preto





Torre de Hanói

- Temos que saber quantos disco temos e para onde devemos mover eles.

```
void Hanoi(int n, int torreOrigem, int torreDestino, int torreAux) {  
  
    return;  
  
}  
  
int main(){  
    Hanoi(7, 1, 3, 2);  
    return 0;  
}
```



UFOP

Universidade Federal
de Ouro Preto

Torre de Hanói

- Caso Base:

```
void Hanoi(int n, int torreOrigem, int torreDestino, int torreAux) {  
    if(n == 1){  
        printf("Mover o disco da Torre %d para a Torre %d\n", torreOrigem, torreDestino);  
        return;  
    }  
}
```

Torre de Hanói

- Para mover o último disco da torre para o destino temos que mover todos os discos menores para a torre auxiliar.
- Nesse momento, podemos usar a torre destino como torre auxiliar.



Torre de Hanói

```
void Hanoi(int n, int torreOrigem, int torreDestino, int torreAux) {  
  
    if(n == 1){  
        printf("Mover o disco da Torre %d para a Torre %d\n", torreOrigem, torreDestino);  
        return;  
    }else{  
        Hanoi(n-1, torreOrigem, torreAux, torreDestino);  
  
    }  
  
}
```



Torre de Hanói

- Agora podemos mover o disco.

```
void Hanoi(int n, int torreOrigem, int torreDestino, int torreAux) {  
    if(n == 1){  
        printf("Mover o disco da Torre %d para a Torre %d\n", torreOrigem, torreDestino);  
        return;  
    }else{  
        Hanoi(n-1, torreOrigem, torreAux, torreDestino);  
        printf("Mover o disco da Torre %d para a Torre %d\n", torreOrigem, torreDestino);  
    }  
}
```

Torre de Hanói

- Agora só falta levar os disco menores que estão na torre auxiliar para o destino
- Podemos usar a torre original com auxiliar.

Torre de Hanói



```
void Hanoi(int n, int torreOrigem, int torreDestino, int torreAux) {  
  
    if(n == 1){  
        printf("Mover o disco da Torre %d para a Torre %d\n", torreOrigem, torreDestino);  
        return;  
    }else{  
        Hanoi(n-1, torreOrigem, torreAux, torreDestino);  
        printf("Mover o disco da Torre %d para a Torre %d\n", torreOrigem, torreDestino);  
        Hanoi(n-1, torreAux, torreDestino, torreOrigem);  
    }  
}
```



Palíndromo

- Um palíndromo é uma palavra, frase ou qualquer outra sequência de unidades que tenha a propriedade de poder ser lida tanto da direita para a esquerda como da esquerda para a direita.



Palíndromo

1. A base do teto desaba.
2. A cara rajada da jararaca.
3. Acuda cadela da Leda caduca.
4. A dama admirou o rim da amada.
5. Socorram me subi no onibus em Marrocos



UFOP

Universidade Federal
de Ouro Preto

Palíndromo

- Vamos fazer uma função que diz se uma string é um palíndromo.
- Vamos considerar que foram removidos os espaços e todas as letras são minúsculas.





UFOP

Universidade Federal
de Ouro Preto

Palíndromo

- Caso Base:
 - Qualquer string com tamanho menor que 2 é um palíndromo.
 - Se o primeiro caractere é diferente do último, não é um palíndromo.
- Passo:
 - Verifique para a frase formada sem o primeiro e último caractere.

D E C S I

Palíndromo

Caso Base

Caso Base

Passo Recursivo

```
#include <stdio.h>
#include <string.h>

void Palindromo(char frase[], int inicio, int fim) {
    if((fim - inicio) < 2){
        printf("E um palindromo\n");
    }else{
        if(frase[inicio] != frase[fim - 1]){
            printf("Nao e um palindromo\n");
        }else{
            Palindromo(frase, inicio + 1, fim - 1);
        }
    }
}

int main(){

    char frase[200] = "socorrammesubinoonibusemmarroc";

    Palindromo(frase, 0, strlen(frase));
    return 0;
}
```



UFOP
Universidade Federal
de Ouro Preto

Referências Bibliográficas

1. DEITEL, P; DEITEL, H. C How to Program. 6a Ed. Pearson, 2010.
2. Material de aula de Stanford- CS 106B- Lecture 7: Introduction to Recursion:
<https://web.stanford.edu/class/archive/cs/cs106b/cs106b.1178/lectures/7-IntroToRecursion/7-IntroToRecursion.pdf>