AED1 - Aula 02 - Recursão, fatorial e torres de Hanoi

"Ao tentar resolver o problema, encontrei obstáculos dentro de obstáculos. Por isso, adotei uma solução recursiva" - citação atribuída a um estudante no livro do Paulo Feofiloff.

Recursão é uma técnica de projeto de algoritmos que nos permite resolver um problema a partir da solução de instâncias menores do mesmo problema.

Fatorial

Definição recursiva de fatorial:
n! = { 1, se n = 0,
n (n - 1)!, se n > 0. }

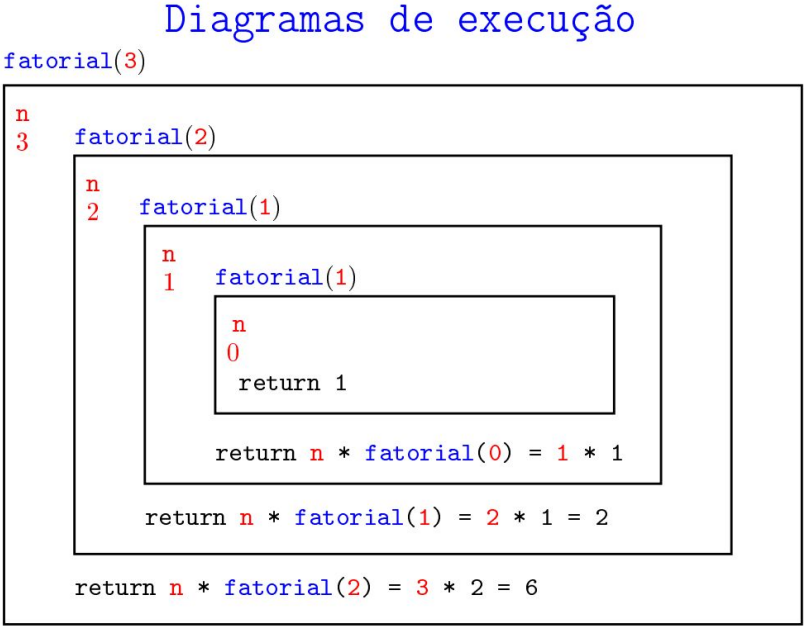
Código para fatorial recursivo:

```
long fatorialR(long n)
{
   if (n == 0)
       return 1;
   return n * fatorialR(n - 1);
}
```

Qual o número de chamadas recursivas em função de n?

- Seja T(n) o número que desejamos descobrir.
- Temos, T(n) = T(n-1) + 1 e T(0) = 0

Diagrama de execução do fatorial recursivo:



Código para fatorial iterativo:

```
long fatorialI(long n)
{
   long i, fat = 1;
   for (i = 1; i <= n; i++)
       fat *= i;
   return fat;
}
```

Verificar que o código faz o que promete usando o seguinte invariante:

- no início de cada iteração temos que fat vale (i-1)!

Qual o número de iterações em função de n?

Estrutura geral de um programa recursivo

se a instância em questão é pequena,
    resolva-a diretamente;
senão,
    reduza-a a uma instância menor do mesmo problema,
    aplique o método à instância menor
    e volte à instância original.

Torres de Hanoi

Lenda:

- Num templo Hindu, situado no centro do universo, Brama criou uma torre com 64 discos de ouro e mais duas estacas equilibradas sobre uma plataforma. Então ordenou aos monges do templo movessem todos os discos de uma estaca para outra respeitando as seguintes regras: apenas um disco poderia ser movido por vez e nunca um disco maior deveria ficar por cima de um disco menor. Segundo a lenda, quando todos os discos fossem transferidos de uma estaca para a outra, o templo iria desmoronar e o mundo desapareceria.

Supondo que a lenda seja verdadeira, será que devemos ficar preocupados com o iminente fim do mundo?

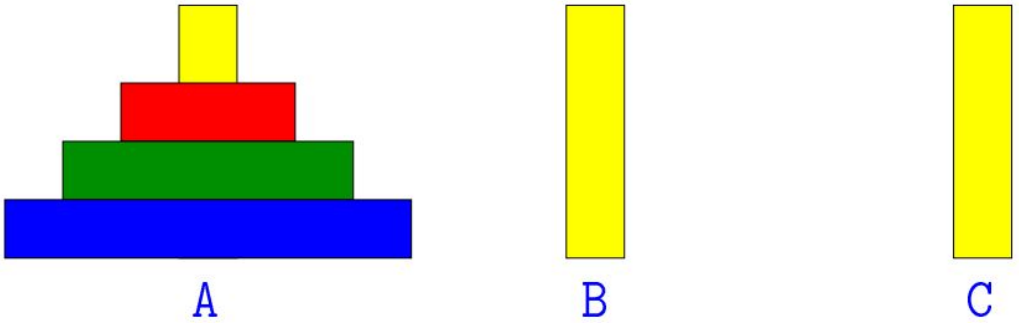
Definição do problema:

Temos n discos, cada um com um diâmetro diferente, empilhados em ordem decrescente de diâmetro na coluna origem (A). Queremos mover todos os discos de A até a coluna destino (C) usando a coluna B como auxiliar e respeitando as regras:

- podemos mover apenas um disco por vez.
- nunca um disco de diâmetro maior pode ficar sobre um disco de diâmetro menor.

Estratégia para atacar o problema:

Embora não seja óbvio qual é o primeiro movimento, o movimento do meio nós sabemos qual é, por exemplo, na seguinte instância do problema



o movimento do meio é mover o disco azul de A para C, pois sendo o maior disco ele deve estar na base da torre de discos no destino.

Para conseguirmos mover o disco azul, primeiro precisamos liberá-lo removendo os demais discos de cima dele, ou seja, tudo que está sobre ele deve ser movido para B. E, depois que o azul estiver em C, tudo que está em B deve ser movido para C.

Para descrever o problema, e nossa solução, de modo mais claro e preciso vamos chamar de Hanoi(n, A, B, C) o problema de mover:

- n discos
- de A para C
- usando B como auxiliar

Assim, nossa ideia de solução pode ser descrita como:

- Hanoi(n-1, A, C, B)
- mover disco restante (n) de A para C
- Hanoi(n-1, B, A, C)

Note que estamos reduzindo o problema de mover n discos para 2 problemas de mover n-1 discos.

Adendo importante, quando soubermos resolver o problema diretamente paramos de reduzir,

- como acontece quando n = 0

Código recursivo para Hanoi:

```
void Hanoi(int n, char origem, char auxiliar, char destino)
{
```

```
    if (n == 0)
        return;
    Hanoi(n - 1, origem, destino, auxiliar);
    printf("mova o disco %d de %c para %c.\n", n, origem, destino);
    Hanoi(n - 1, auxiliar, origem, destino);
}
```

Qual o número de movimentos realizados em função de n?

- Seja T(n) o número que desejamos descobrir.
- Temos que T(n) = 2 T(n - 1) + 1 e T(0) = 0

Resolvendo a recorrência:
T(n) = 2 T(n - 1) + 1
T(n - 1) = 2 T(n - 2) + 1
T(n - 2) = 2 T(n - 3) + 1
T(n - 3) = 2 T(n - 4) + 1

T(n) = 2 T(n - 1) + 1
= 2 (2 T(n - 2) + 1) + 1 = 4 T(n - 2) + 3
= 4 (2 T(n - 3) + 1) + 3 = 8 T(n - 3) + 7
= 8 (2 T(n - 4) + 1) + 7 = 16 T(n - 4) + 15

Observe que:
T(n) = 2 T(n - 1) + 1 = 2^1 T(n - 1) + 2^1 - 1
= 4 T(n - 2) + 3 = 2^2 T(n - 2) + 2^2 - 1
= 8 T(n - 3) + 7 = 2^3 T(n - 3) + 2^3 - 1
= 16 T(n - 4) + 15 = 2^4 T(n - 4) + 2^4 - 1
…
= 2^k T(n - k) + 2^k - 1

Para k = n temos:
T(n) = 2^n T(n - n) + 2^n - 1
= 2^n - 1
pois T(0) = 0.

Ou seja, o número de movimentos cresce exponencialmente (mais especificamente, uma exponencial de base 2) em função do número de discos n.

Vale observar que nossa solução não realiza movimentos desnecessários. Por isso, não é possível resolver este problema com menos movimentos.

Pra finalizar, voltemos à nossa preocupação inicial com os monges e o fim do mundo. Quantos movimentos eles tem que fazer pra mover 64 discos?

- 2^64 - 1 ~= 1,84 * 10^19

Supondo que levem um segundo para realizar cada movimento, eles precisarão de aproximadamente:

- 3,07 * 10^17 minutos,
- 5,11 * 10^15 horas,
- 2,13 * 10^14 dias,
- 5,83 * 10^11 anos ~= 583 bilhões de anos.

Podemos dormir tranquilos!