Dia 2 - Paradigmas de resolução de problemas

Emanuel Huber - emanuel.tesv@gmail.com

Recursão

Definir um problema em termos de si mesmo.

Função fatorial: n! = n * (n - 1)!

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

Desafio - recursão

Dado dois números inteiros B (1 <= 1000) e E (0 <= 100), calcule de forma recursiva a função exponencial, utilizando B como base e E como expoente.

Entrada:

2 4

Saída:

16

Recursão - exercícios extras

Criar as seguintes funções recursivas:

1. Recebe dois valores e retorna a multiplicação entre eles; não pode utilizar o operador de multiplicação, apenas adição e subtração.

2. Recebe o valor do primeiro termo, da razão e quantidade de termos de uma PA; calcular a soma dessa PA (não pode utilizar a fórmula).

3. Recebe um vetor e retorna a soma dos elementos

4. Recebe um vetor e um valor; deve retornar o índice do valor no vetor (-1 se não existir)

Dividir para conquistar

vector<int> v = {1, 4, 6, 8, 9, 10, 13, 15, 17, 20, 22, 26, 30, 40, 49, 70, 90, 100}

bool achou = binary_search(v.begin(), v.end(), 17);

Sequência de Fibonacci

Dado um número n, descobrir o n-ésimo número da sequência de Fibonacci

0, 1, 1, 2, 3, 5, 8, 13, ...

Fib(2) = 1

Fib(7) = 13n

Sequência de Fibonacci - Definição matemática

$$Fib(n) = \begin{cases} 0, \text{ se } n = 0 \\ 1, \text{ se } n = 1 \\ Fib(n-1) + Fib(n-2) \end{cases}$$

Sequência de Fibonacci - Recursivo (Complete Search)

```
int fib(int n)
{
	if(n <= 1)
		return n;
	return fib(n-1) + fib(n-2)
}
```

Complexidade: O(φ^n) **HORRÍVEL**

Sequência de Fibonacci - Memoization

Utilizar tabela para memorizar valores já calculados

```
int r[1000000]
int fib(int n)
{
        if(r[n]!=0)
                return r[n];
        if(n <= 1)
                return n;
        return r[n] = fib(n-1) + fib(n-2)
}
```

Complexidade: O(n) **BEM MELHOR**

Sequência de Fibonacci - Divide and Conquer

$$Fib(n) = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n}$$

Pesquisar: exponenciação rápida

Complexidade: O(logn) **EXCELENTE**

Programação Dinâmica

“Programação dinâmica é um nome bonito para [recursão] com tabela...”

Exemplo clássico:

Problema da mochila: dado uma mochila com capacidade S e um conjunto de n elementos, cada um com um peso e um valor, escolher um subconjunto de elementos que maximize a soma dos valores e a soma dos pesos seja menor ou igual à capacidade da mochila.

Programação Dinâmica - Problema da mochila

V = {100, 70, 50, 10}
W = {10, 4, 6, 12}
S = 12

De quantas maneiras é possível escolher os itens?

Programação Dinâmica - Problema da mochila

V = {100, 70, 50, 10}
W = {10, 4, 6, 12}
S = 12

De quantas maneiras é possível escolher os itens?

2^n (onde n é a quantidade de elementos)

Programação Dinâmica - Problema da mochila

V = {100, 70, 50, 10}
W = {10, 4, 6, 12}
S = 12

Programação Dinâmica - Problema da mochila

V = {100, 70, 50, 10}
W = {10, 4, 6, 12}
S = 12

pm(id, 0)= 0
pm(n, Sres) = 0
pm(id, Sres) = pm(id+1, Sres), se W[id] > Sres
max (pm(id+1, Sres), V[id] + pm(id+1, Sres-W[id]))

Programação Dinâmica - Outros problemas

Max 1D Range Sum: sequência de soma máxima
UVa 507

Longest Increasing Subsequence: maior subsequência crescente
A = {-7, 10, 9, 2, 3, 8, 8, 1}

Coin change: troco de moedas
V = 7, moedas = {1, 3, 4, 5}

Exercícios

https://www.urionlinejudge.com.br/judge/en/tournaments/rank/761