O Problema do Troco

Como as Empresas de Logísticas resolvem um dos problemas mais complexos da matemática sem perceber

Projeto e Análise de Algoritmos
Docente: Leonardo Nogueira Matos
Discentes:
Poliana Rafaela Moraes de Lira Lima
Thomás Silva de Araujo

O Problema do Troco e a Programação Dinâmica

O Problema do Troco é um problema clássico da ciência da computação, que serve como um excelente ponto de partida para entender algoritmos de otimização, especialmente a Programação Dinâmica.

Em essência, ele busca encontrar a maneira mais eficiente de representar um valor.

A Programação Dinâmica é uma técnica algorítmica utilizada para resolver problemas de otimização complexos, transformando-os em uma sequência de decisões simples.

Em essência, a DP resolve problemas que possuem duas características principais: **Subestrutura Ótima** e **Subproblemas Sobrepostos**.

O Problema do Troco e a Programação Dinâmica

Minimização de Unidades de Empilhamento

Em um cenário de logística de alta complexidade, uma empresa de transporte crítico deve carregar um veículo especializado para uma rota de entrega com requisitos de balanceamento extremamente rigorosos.

O veículo exige uma carga útil total de EXATAMENTE 50 kg. Desvios nessa massa comprometem a estabilidade e o consumo de combustível.

O desafio é reduzir o tempo de manuseio e maximizar a eficiência de empilhamento, o que se traduz em usar o MENOR NÚMERO DE UNIDADES (CAIXAS) POSSÍVEL.

O Conjunto de Unidades de Carga (As Moedas)

O estoque de carga consiste em três tipos de unidades (caixas) com pesos nominais fixos:

- Unidade Tipo G (Grossa): Pesa 10 kg. (A transação de maior impacto).
- Unidade Tipo M (Média): Pesa 5 kg.
- Unidade Tipo P (Pequena): Pesa 1 kg. (A unidade base).

Programação Dinâmica

- "Quem não se lembra do passado é condenado a repeti-lo" do filósofo e escritor hispano-americano George Santayana.
- Em Algoritmo que usa a estratégia da divisão e conquista é comum haver repetição de subproblemas (overlapping subproblems), como exemplificado no algoritmo para calcular termos da sequência de Fibonacci e o Fatorial de um número.
- Isso pode acaba gerando muito recálculo e consequentemente uso de mais recurso.
- A PD vem para tentar resolver esse problema.

Programação Dinâmica

▷ Sequência de Fibonacci: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$

$$F(n) = \begin{cases} 0, & \text{se } n = 0; \\ 1, & \text{se } n = 1; \\ F(n-1) + F(n-2), & \text{caso contrário.} \end{cases}$$

Por exemplo, para calcular o quinto número de Fibonacci:

$$\begin{aligned} F(0) &= 0 \\ F(1) &= 1 \\ F(2) &= F(1) + F(0) = 1 \\ F(3) &= F(2) + F(1) = 2 \\ F(4) &= F(3) + F(2) = 3 \\ F(5) &= F(4) + F(3) = 5 \end{aligned}$$

Função Recursiva com Fatorial

$$f(n) = \begin{cases} 1, & \text{se } n \leq 1; \\ n \times f(n-1), & \text{caso contrário.} \end{cases}$$

Função Recursiva

Fatorial

$$
\begin{aligned}
f(n) &= n! \\
&= n \times (n-1) \times (n-2) \times \ldots \times 1 \\
&= n \times (n-1)! \\
&= n \times f(n-1),\ n > 1
\end{aligned}
$$

$f(1) = 1,\ f(0) = 1$ (condições de contorno).

Programação Dinâmica

Recursividade com PD

$$f(n) = \begin{cases} 1, & \text{se } n \leq 1; \\ n \times f(n-1), & \text{caso contrário.} \end{cases}$$

Por exemplo, para calcular $5!$:

$$\begin{aligned} f(1) &= 1 \\ f(2) &= 2 \times f(1) = 2 \\ f(3) &= 3 \times f(2) = 6 \\ f(4) &= 4 \times f(3) = 24 \\ f(5) &= 5 \times f(4) = 120 \end{aligned}$$

Programação Dinâmica

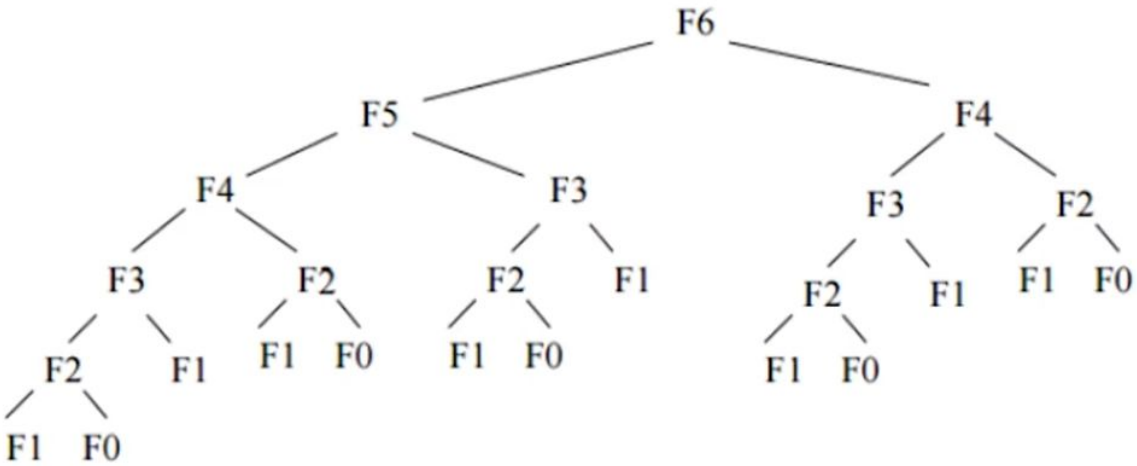
Tomemos como exemplo um algoritmo recursivo para solucionar o problema de encontrar o i-ésimo termo da sequência de Fibonacci.

```
int fib(int i) {
    if (i == 0 || i == 1)
        return 1;
    return fib(i-1) + fib(i-2);
}
```

A complexidade de tempo é **$O(2^n)$** (exponencial).

Programação Dinâmica

- F6



Programação Dinâmica

- A ideia básica da Programação Dinâmica é armazenar a solução dos subproblemas para serem utilizados no futuro.
- Isso pode ser feito por duas abordagens:
  - Top Down (Memoization)
  - Bottom Up (Tabulation)

É importante ressaltar: para que esse paradigma possa ser aplicado, é preciso que o problema tenha uma estrutura recursiva, a solução de toda instância do problema deve "conter" soluções de subinstâncias dessa instância.

Modelo Top Down

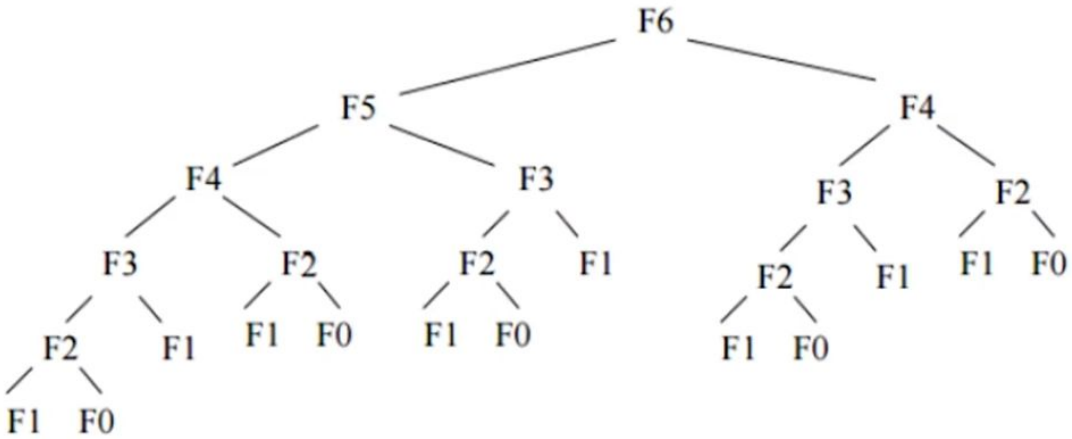
- Fibonacci

```
int memo[MAXN];

void preprocess(int n){
    memo[0] = memo[1] = 1;
    for(int i = 2; i < n; i++)
        memo[i] = memo[i-1] + memo[i-2];
}

int fib(int i){
    return memo[i];
}
```

complexidade O(n)

Programação Dinâmica

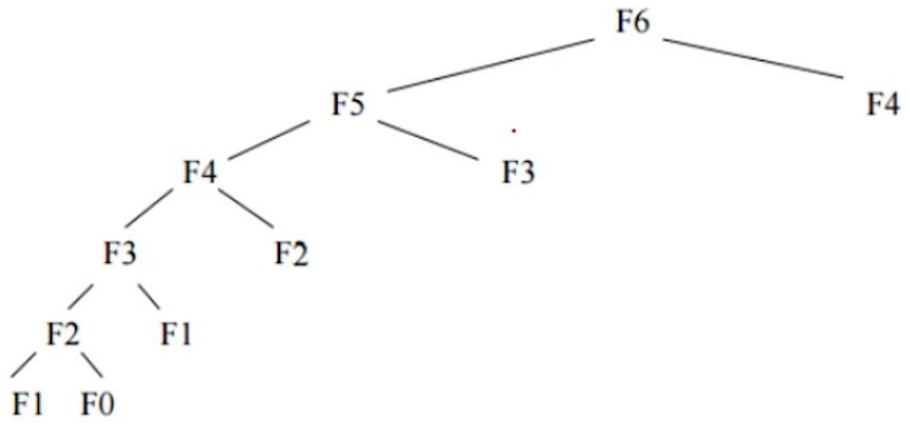
- Fibonacci



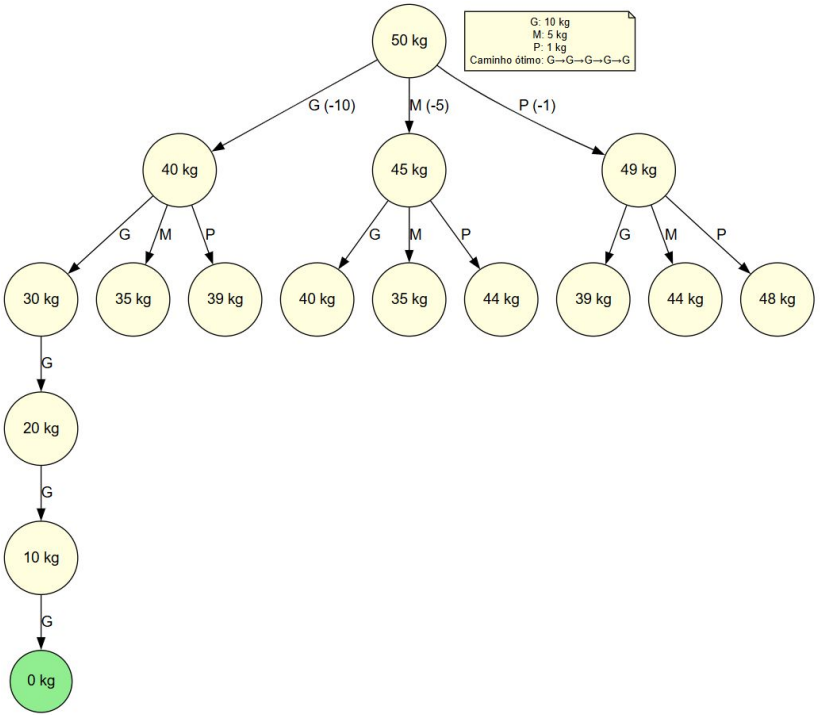
Programação Dinâmica

- F6

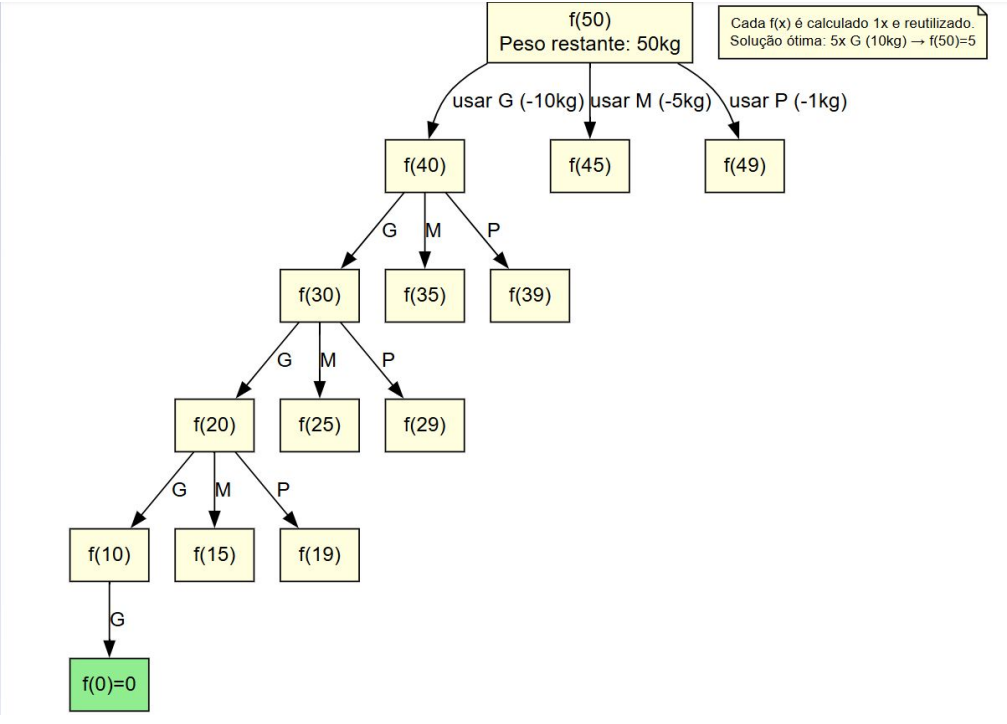
- Fibonacci



Programação Dinâmica



Programação Dinâmica



Apresentação do código

Acesse os links

GitHub:https://github.com/Poliana-llima/Semin-rio2-PAA
Colab:https://colab.research.google.com/drive/1wUaXRNDuh_nB9ilFxu2kz5iZSTQzDt8x?usp=sharing
Vídeo: https://youtu.be/gGz9A1FKwlQ

Referências Bibliográficas

- https://www.youtube.com/watch?v=pL7I1ce1FUk
- https://dreampuf.github.io/