Programação dinâmica

A programação dinâmica tenta resolver problemas com solução recursiva, duplicando o trabalho. Possui uma abordagem de economizar tempo e memória. Com a programação dinâmica, tentamos resolver sub-problemas por ordem, e armazenar essas soluções para quando for necessário. No entanto, nem sempre o armazenamento de soluções é necessário para a resolução do problema, o que pode economizar a memória ainda mais.

Palavras chave

Estágio - ponto de decisão do código. O problema é decomposto em estágios que são resolvidos sequencialmente.

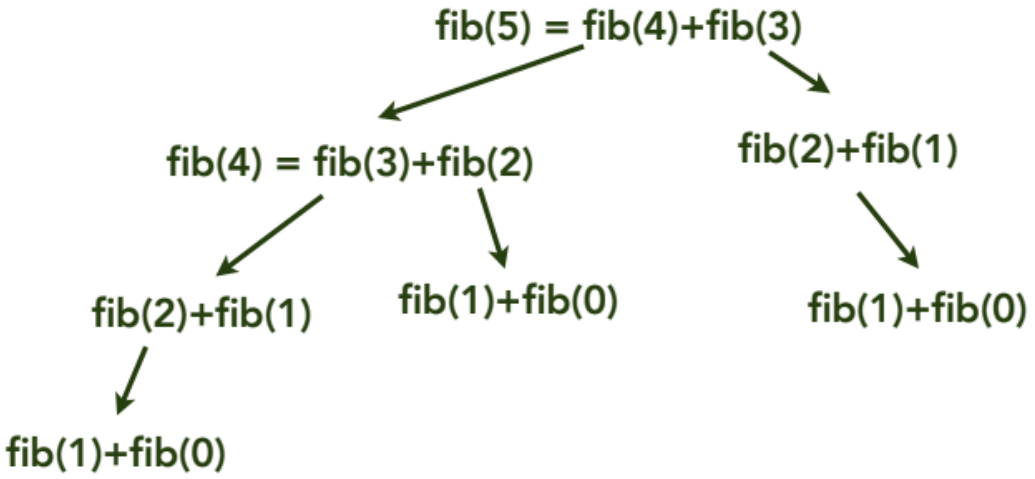
Estado - situação de um estágio. É a informação usada para tomar decisões.

Decisão - transforma o estado de um sistema. Há sempre um valor associado.

Política - sequência de decisões.

Fibonacci

Um exemplo de problema seria o Fibonacci, onde há várias operações que podem se repetir.



```
long dp[1000];
long fib(long n){
    if (dp[n] != -1) return dp[n];
    if (n < 2) dp[n] = n; // dp[0] == 0, dp[1] == 1
    else dp[n] = fib(n-1)+fib(n-2);
    //dp[2] = dp[1] + dp[0] -> dp[2] == 1
    //dp[3] = dp[2] + dp[1] -> dp[3] == 2
    //...
    return dp[n];
}
```

Esse código de Fibonacci armazena as sequências no vetor, o que faz com que mais memória seja utilizada, mas também torna o código eficiente em questão de tempo.

Sublista de soma S

Para esse problema, há o preenchimento de uma tabela. As colunas da tabela são todos os números de 0 até S. E as linhas são os índices dos elementos da lista.

No problema original de sublista de soma S utilizando recursividade, temos duas variáveis importantes para representar o estado da sublista, que é a de tamanho e a de soma.

```
int soma_sublista_bt(int a[], int tam, int soma){
    int r;
    if (soma == 0) return 1; //EXISTE SOLUÇÃO
    if (soma < 0 || tam == 0) return 0; //NÃO EXISTE SOLUÇÃO
    r = soma_sublista_bt(a, tam-1, soma - a[tam-1]);
    r = r || soma_sublista_bt(a, tam-1, soma);
    return r; //r irá indicar se existe solução ou não, testa
}
```

O exemplo acima é o código para soma de sublista utilizando backtracking. Na programação dinâmica, iremos preencher uma tabela, onde todos os estados do tamanho e da soma serão armazenados nessa tabela. Para representar a tabela, será utilizada uma matriz.

O código da sublista de soma S utilizando programação dinâmica só é eficiente até certos valores. O ideal, é que siga essas limitações:

- Número de elementos < 1000 (10^3)
- Soma < 100000 (10^5)

Dependendo dos valores, talvez utilizar a solução apenas com backtracking seja mais eficiente.

```
int tabela[1000][100000];
//Colocar todos os valores da tabela igual a -1

int soma_sublista_pd(int a[], int tam, int soma){
    int r;
    if (soma == 0) return 1;
    if (soma < 0 || tam == 0) return 0;
    if (tabela[tam-1][soma] != -1) return tabela[tam-1][soma]
    r = soma_sublista_pd(a, tam-1, soma - a[tam-1]);
    r = r || soma_sublista_pd(a, tam-1, soma);
    dp[tam-1][soma] = r;
    return dp[tam-1][soma];
}
```