Programação Dinâmica (PD)

Corte de hastes

Você recebe uma haste de aço de tamanho i e uma tabela de preços p, onde p_j indica o preço de uma haste de tamanho j. Qual a melhor forma de cortar a haste para maximizar o preço total?

Para simplificar, vamos nos concentrar em maximizar o preço total. Tome a seguinte tabela:

n	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
p_n	1	5	8	9	10	17	17	20	24

Podemos pensar na PD da seguinte forma:

- Tome c_i como o custo ótimo para a haste de tamanho i.
- Quando chegamos em i, podemos vender i inteiro ou cortar i em dois pedaços e vendê-los separadamente.
- Os pedaços são sempre menores = são subproblemas, que devemos resolver antes.
- Temos um caso base $c_0 = 0, c_1 = p_1$.
- Definimos a PD como $c_i = \max\left(p_i, \max_{j=1,\ldots,i-1}\left(c_j + c_{i-j}\right)\right)$

Agora é simples, pra evitar de repetir problemas, é só calcular c_0, depois c_1 e assim por diante. Também é interessante setar $c_i = p_i$, daí é correto usar $c_i = \max_{j=1,\ldots,i}\left(c_j + c_{i-j}\right)$ pra simplificar o código.

Eis o código:

```
// Indexando a partir do 0, isso nao e pseudo-codigo, pelo amor de deus
int cut_this_shit(vector<int> p, int n) {
    vector<int> c(p.size()); // Guardar os custos aqui
    c[0] = 0; // Caso base
    for (int i = 1; i < n; i++) c[i] = p[i];
    for (int i = 2; i < n; i++)
        for (int j = 0; j < i-1; j++)
            c[i] = max(c[i],c[j] + c[i - j]);
    return c[n-1];
}
// Precisa adaptar pra devolver os cortes, mas o grosso eh isso ai
```

Cadeias de matrizes

Considere uma sequência de de n matrizes M, onde M_i é a i-ésima matriz, que representa a multiplicação das n matrizes, na ordem da sequência. Se pede para minimizar o número de operações de multiplicação no processo.

$$M_1 M_2 \ldots M_n$$

A multiplicação de matrizes é associativa, então podemos fazer

$$(M_1 M_2) M_3 \ldots M_n$$

por exemplo.

Queremos encontrar o custo mínimo (i.e. minimizar operações) para tal.

O subproblema mais básico é o de não multiplicar nenhuma matriz, então o custo é zero. Ao multiplicar duas matrizes consecutivas ($M_i M_{i+1}$), onde M_i é $p \times q$ e M_{i+1} é $q \times r$, o custo é $p \cdot q \cdot r$, e forma uma matriz $p \times r$.

Agora precisamos quebrar o problema maior nesses subproblemas:

- Tome d como o vetor que guarda as dimensões das matrizes, daí d_{i-1} e d_i, então a matriz M_i é $d_{i-1} \times d_i$.
- Tome o custo da multiplicação das matrizes de i até j como $c_{i,j}$.
- Podemos quebrar a multiplicação $M_1 \dots M_n$ em $(M_1 \dots M_i)(M_{i+1} \dots M_n)$ com i variando de 1 até $n-1$ e calculamos o resultado, que será dado por $\max_{i=1,\dots,n-1} (c_{1,i} + c_{i+1,n})$.
- Caso base: $c_{i,i} = 0$ e $c_{i,i+1} = d_{i-1} d_i d_{i+1}$.
- PD: $c_{i,j} = \min_{k=i,\dots,j} (c_{i,k} + d_{i-1} d_k d_j + c_{k+1,j})$

TODO: Algoritmo para colocar os parênteses na expressão => Ideia: Marcar ponto de divisão `bracket[i][j]`, i.e., para o subproblema $c_{i,j}$ a solução ótima é cortar a expressão em `bracket[i][j]`.