Dinamik Programlama (DP)

Genel, güçlü bir alg. tasarım tekniğidir.

DP: alt problem + tekrar kullanma

Örnek: Fibonacci Sayıları

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

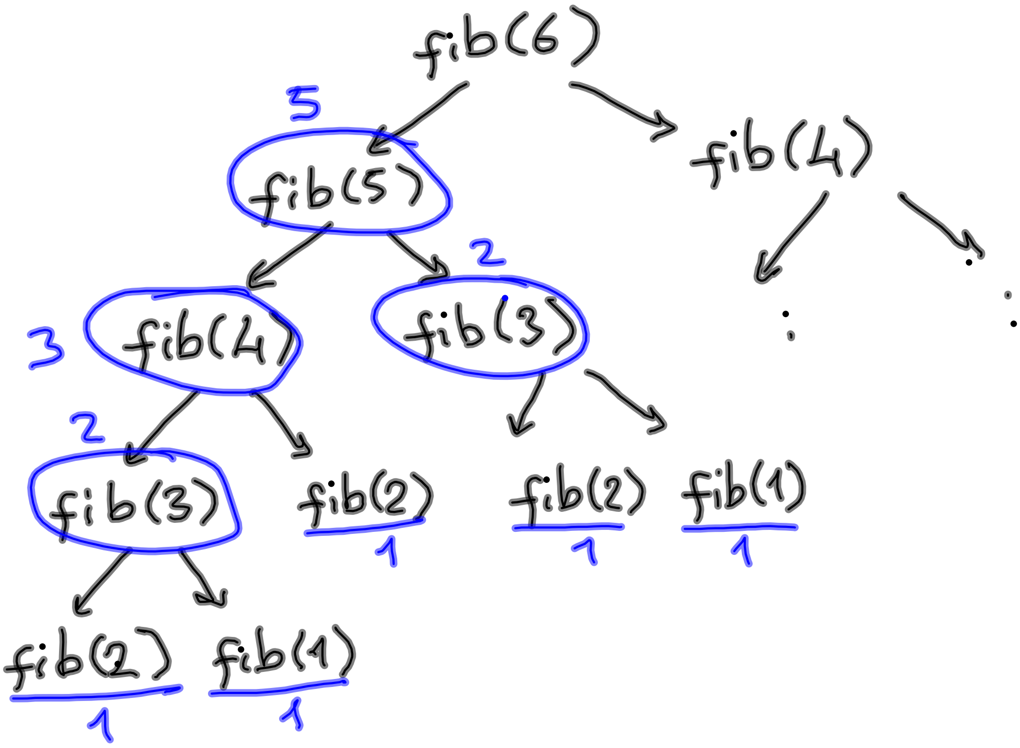
fib(1) fib(2)

$$F_n = F_{n-1} + F_{n-2}$$

Saf Özyinelemeli Alg.

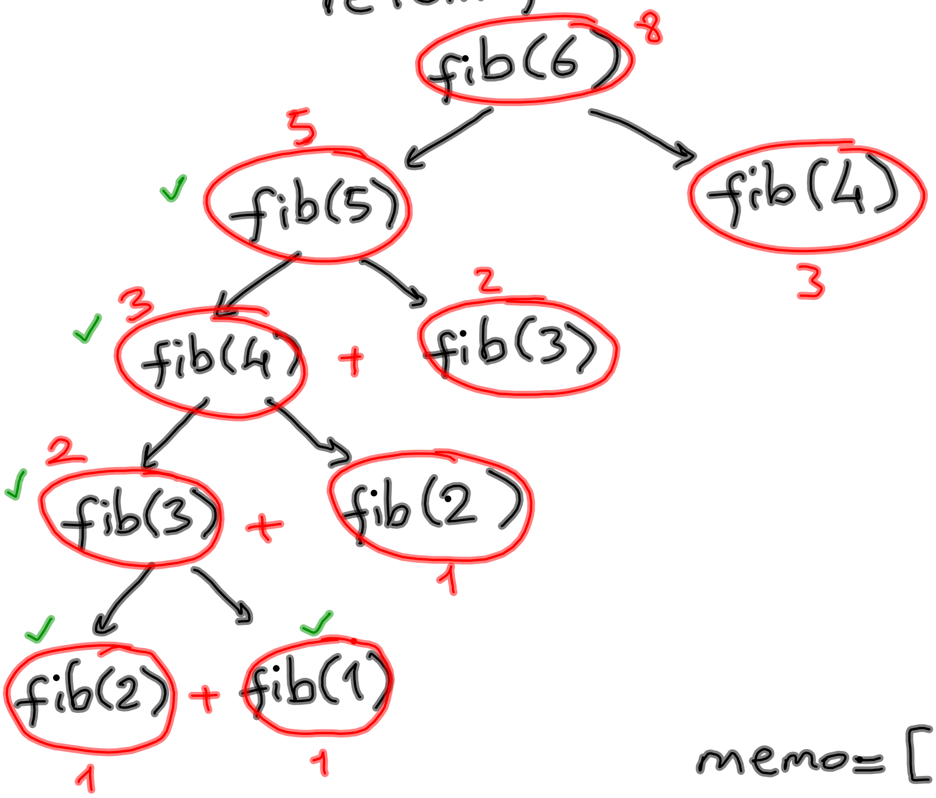
```
fib(n)
    if n≤2
        f=1
    else
        f= fib(n-1)+fib(n-2)
    return f
```

$$T(n) = T(n-1) + T(n-2) + \Theta(1)$$
$$= \Theta(2^n) \quad \text{üstel, kötü}$$



Hafızalı (Memoized) DP algoritması

```
memo = { }
fib(n)
    if n ∈ memo
        return memo[n]
    if n ≤ 2
        f = 1
    else
        f = fib(n-1) + fib(n-2)
    memo[n] = f
    return f
```



$$\begin{array}{cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \text{memo} = [& 1 & 1 & 2 & 3 & 5 & 8 &] \end{array}$$

Hafızalı (Memoized) DP algoritması Analizi

Her k için, fib(k) ilk çağırıldığında özyinelemeli çalışır.

Hafızadaki çözümlere (değerlere) erişim zamanı $\Theta(1)$.

Hafızada olmayan çağırımlar n adettir.
fib(1), fib(2), ..., fib(n)

$$T(n) = n \cdot \Theta(1) = \Theta(n)$$

Dinamik Programlama Detay

- Böl ve Fethet gibi bir alg. tasarım tekniğidir.
- B&F de alt problemler bağımsız olmalı.
- Alt problemler bağımlı ise DP uygulanır.
- DP her alt problemi bir defa çözer ve sonucu bir tabloda saklar
- DP genellikle optimizasyon problemlerine uygulanır.

DP geliştirmek için şu dört adım izlenir.

1. Optimal çözümün yapısının karakteristiği ortaya çıkarılmalı.
2. Özyinelemeli olarak çözümün değerini tanımlamalı
3. Alttan-üste (bottom-up) mantığı ile bir optimal çözümün değerini hesaplamalı
4. Hesaplanan bilgilerden optimal çözüm elde edilir.

min. $f(x) = x^2 + 2$ $\quad$ $f(0) = 2$

optimal çözüm $\quad$ optimal değer

Matris Zinciri Çarpımı

$\langle A_1, A_2, \dots, A_n \rangle$ n adet matrisden oluşan bir matris zinciri olsun.

$A_1 \cdot A_2 \cdot A_3 \cdot \dots \cdot A_n$ çarpımı isteniyor.

Matris çarpımının birleşme özelliği var ve her parantezleme aynı sonucu verir.

Örn: $\langle A_1, A_2, A_3, A_4 \rangle$

$((A_1 \cdot A_2)(A_3 \cdot A_4))$ $\quad$ $(A_1 \cdot ((A_2 \cdot A_3) A_4))$

$(((A_1 \cdot A_2) A_3) A_4)$ $\quad$ $((A_1 \cdot (A_2 \cdot A_3)) A_4)$

$(A_1 \cdot (A_2 \cdot (A_3 \cdot A_4)))$

Fakat her birinin farklı maliyetleri var.

İki matrisin çarpım maliyeti: Toplam skalar çarpım sayı

$A_{m \times n} \cdot B_{n \times k}$ için toplam $m \cdot n \cdot k$ skalar çarpma gereki

$$\underset{m \times n}{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}} \cdot \underset{n \times k}{\begin{bmatrix} b_{11} & \dots & b_{1k} \\ \vdots & & \\ b_{n1} & \dots & b_{nk} \end{bmatrix}} = \underset{m \times k}{\begin{bmatrix} n & n & \dots & n \\ n & n & \dots & n \\ & & & \\ n & \dots & & n \end{bmatrix}} \begin{array}{l} \rightarrow nk \text{ çarp} \\ \rightarrow nk \\ \\ \rightarrow nk \end{array}$$

$+$

Örnek: $\langle A_1, A_2, A_3 \rangle$ matris zinciri olsun.

Boyutları sırası ile 10×100, 100×5, 5×50 olsun.

Amaç $A_1A_2A_3$ çarpımını bulmak

1. alternatif: $(A_1 \cdot A_2)A_3 =$

$A_1 \cdot A_2$ için $10 \cdot 100 \cdot 5 = 5000$ adet skalar çarpma

$B \cdot A_3$ (10×5, 5×50) $10 \cdot 5 \cdot 50 = 2500$ adet " "

\+

Toplam 7500 adet çarpma var

2. alternatif: $A_1 \cdot (A_2 \cdot A_3)$

$A_2 \cdot A_3$ için $100 \cdot 5 \cdot 50 = 25000$ adet çarpma

$A_1 \cdot C$ (10×100, 100×50) $10 \cdot 100 \cdot 50 = 50000$ adet çarpma

\+

Toplam 75000 adet çarpma var

n elemanlı matris zinciri için $O(2^n)$ parantezleme vardır. Yani üstel (exponential), o zaman brute-force çalışmaz.

Problem: n adet matristen oluşan bir $\langle A_1, A_2, \ldots, A_n \rangle$ zinciri verilsin. A_i matrisinin boyutu $p_{i-1} \times p_i$ olsun. $A_1 \cdot A_2 \cdot \ldots \cdot A_n$ çarpımı için hangi parentezleme optimaldir yani minimum adet çarpma gerektirir.