Tuesday, March 5, 2024 10:51 AM

Dynamic Programming (DP)

★ Why?

eg. Fibonacci nos.

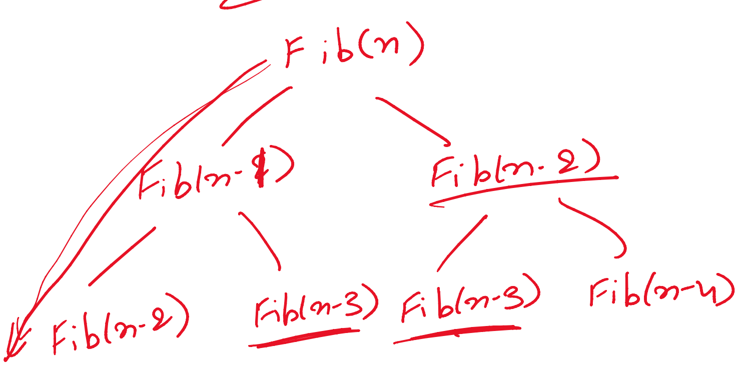
$$F_n = F_{n-1} + F_{n-2}$$

$$F_0 = 0, \quad F_1 = 1$$

```
Fib(n) {                              → T(n)
    if n=0 return 0                   } O(1)
    if n=1 return 1                   }
    return (Fib(n-1) + Fib(n-2))
               ↳ T(n-1)   ↳ T(n-2)
}
```

Time: $T(n) = T(n-1) + T(n-2) + O(1)$

$\Rightarrow \boxed{O(1.618^n)}$




```
F(n) {
    f[0] = 0;
    f[1] = 1;
    for i = 2 to n
        f[i] = f[i-1] +
               f[i-2];
    return f[n];
}
```

$O(n)$

→ Store & Reuse
|||
Memoization.

Recursion + Memoization
+ Evaluation Order
|||
DP.

DP Steps:

① Define subproblems

1. Define subproblems
2. Recursive formula to solve subprob.
3. Evaluation order.

Problem 1:

Given a string $a_1 a_2 \dots a_n$

split it in minimum # of palindroms.

eg. 0110 110 011 → size 2 sol'n opt.

0110 11 00 11 → size 3 sol'n

Observations: $a_1 a_2 \dots, a_n$

- Last pal. ends in a_n, & starts at some a_k for some $k \in \{1, \dots, n\}$
- Suppose k is given then

opt sol'n $= 1 + \left(\begin{array}{l}\text{min \# palindrom} \\ \text{split of } a_1 \dots a_{k-1}\end{array}\right)$

1. Define Subproblems: $i = 0$ to n

$PS(i) =$ min # Palindrom split of string $a_1 \dots a_i$

2. Recursive Formula: $a_1 a_2 a_j \dots a_i$ = $\boxed{1\,0\,1}\boxed{1}$ $i=4$

B.C.: $PS(0) = 0$

$PS(i) = \min\limits_{\dots} \{ 1 + PS(j-1) \}$,

$i \geq 1$: $$\boxed{PS(i) = \min_{\substack{j = 1 \text{ to } i \\ \text{s.t. } a_j \cdots a_i \\ \text{is a palindrome}}} \{ 1 + PS(j-1) \}}$$

$PS(2) = 2$

$PS(3) = 1$

③ Evaluation Order :

Increasing order of i.

$PS(4)$

$= \min_{j = 1,2,3,4} \{ 1 + PS(2), \boxed{1 + PS(3)} \}$

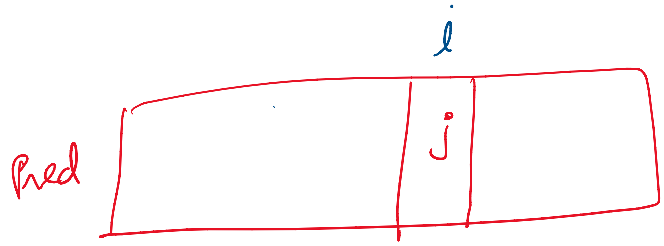
$= \min \{ 3, 2 \}$

$PS(4) = 2$, Pred $[4] = 4$

★ Psuedocode (Iterative) :

```
PS[0] = 0 , Pred[1 ... n] = undef.
for i = 1 to n                                        O(n^3)
    PS[i] = ∞
    for j = 1 to i :                                  O(n^2)
        if (a_j .. a_i) is a palindrome   O(n)        O(n)
            if PS[i] > (1 + PS[j-1]) then {
                PS[i] = 1 + PS[j-1]; Pred[i] = j
            }
Return PS[n];
```

Run Time : $\boxed{O(n^3)}$



$a_1 \quad a_{j-1} \; a_j \ldots a_i$

★ Output opt. sol'n

```
Output sol (i) {
    if i = 0 return;
    j = pred [i];
    output sol (j - 1) ;
    Print a_j ... a_i
}
```

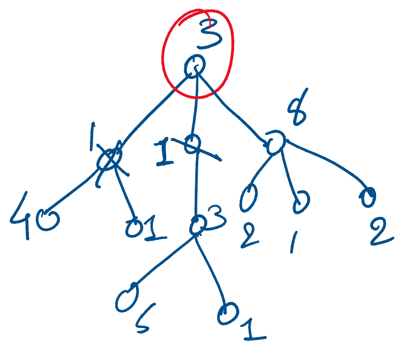
}

call : outputSol (n)

Prob 2: Max-weight Independent Set in a Tree

$T = (V, E)$

$v \in V$, weight $w(v) \geq 0$



Independent Set ↳ $S \subseteq V$ s.t. $\forall u, v \in S$ $(u,v) \notin E$.

maximizing $w(S) = \sum_{v \in S} w(v)$

$8+1+1+4+5+1 = 20$

Observation: ① If root is not in opt sol'n

opt sol'n $= \bigcup_{u \text{ is a child root}}$ opt sol'n (subtree rooted at u)

② If root is in opt sol'n

opt sol'n $= \{\text{root}\} \cup \bigcup_{u \text{ is a grandchild root}}$ opt sol'n (subtree rooted at u)