CSE 373 / L-17 / 21.04.2024 /

Chapter - 15

* Fibonacci Numbers.

  - $F(n) = F(n-1) + F(n-2)$ ⇒ Recursive Call
  - $F(0) = 0$ } Base Case
  - $F(1) = 1$ }

  ⇒ Top - Down Aproach

* Function:

```
FIB(n):
    if n=0
        return 0
    if n=1
        return 1
    return FIB(n-1) + FIB(n-2)
```

Slide - 3 → illustration

* Time Complexity:

$$T(n) = F(n) = O(1.6^n)$$

⊛ Problem with this algorithm:

- each sub-problem was solved for many times.

⇒ Solutions:

- save the solution of an sub-problem in an array and avoid calculating more than once.

⇒

```
FIB(n)
    if F[n] = -1
        F[n] = FIB(n-1) + FIB(n-2)
    return F[n]
```

- Top to Down approach

⊛ Bottom to up approach:

```
FIB(n):
    F[0] = 0
    F[1] = 1
    for i = 2 to n
        F[i] = F[i-1] + F[i-2]

    return F[n]
```

Time = $O(n)$
Space = $O(n)$

Dynamic Programming:

- comes from control theory
- design for optimization problem
- use tables (array) ~~for~~ to construct solutions.
- solves problem by combining solutions to sub-problem, just like divide and conquer, but sub-problem are not independent. Sub-problem may share sub-problems.

- initially it saves all possible optimal solutions in a table. and then table is used for finding best optimal solutions for larger problem.

- reduce time but increase the space.

- bottom-up approach

$\boxed{\text{Slide-13 - Problem}}$

⇒ For a rod of length n

total possible cut $= 2^{n-1}$ ⇒ exponential function

if n-large
it will be exhaustive approach