CSC 373

Assignment Project Exam Help
https://powcoder.com

ALGORITHMS

Add WeChat powcoder

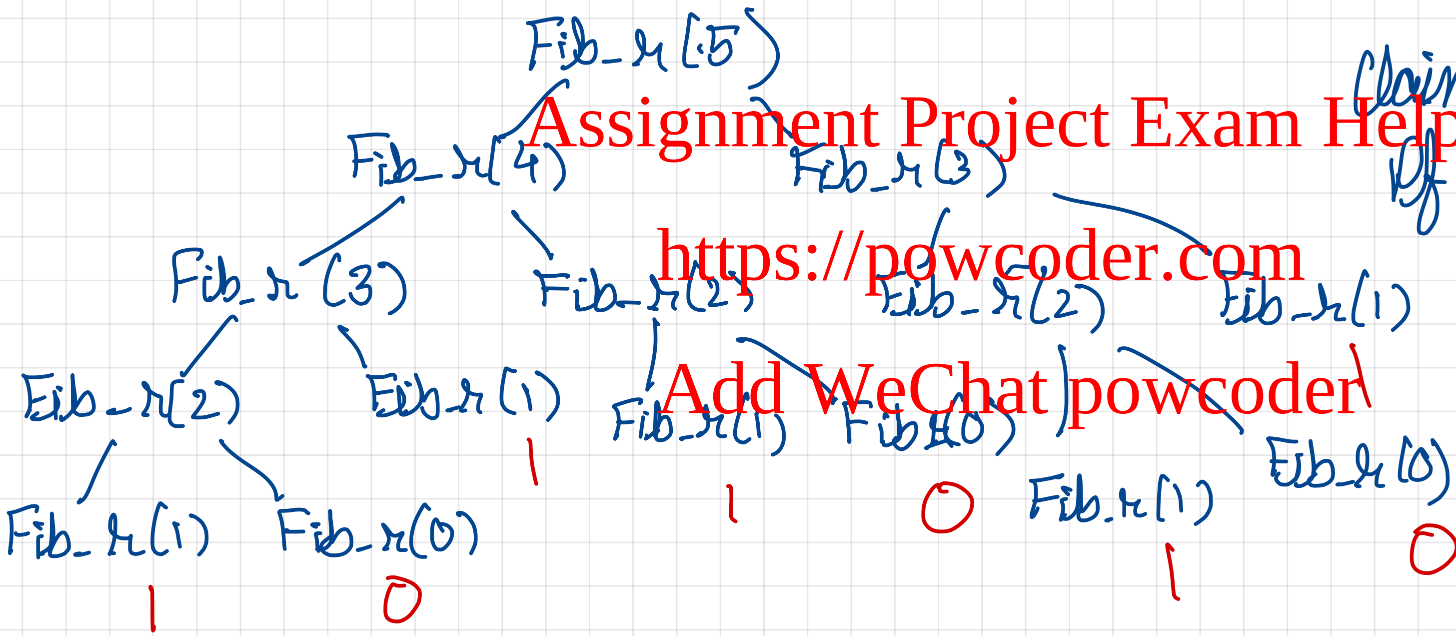
DYNAMIC PROGRAMMING

FIBONACCI NUMBERS

$Fib(0) = 0 \qquad Fib(1) = 1$

$Fib(n) = Fib(n-1) + Fib(n-2)$

```
def Fib_r(n):
    if n == 0:
        RETURN 0
    if n == 1:
        RETURN 1
    RETURN (Fib_r(n-1)
            + Fib_r(n-2))
```



Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Claim : Correct

Pf : INDUCTION

Claim : Time taken is $\geq Fib(n)$

Pf : leaves are just 0 or 1

Claim : $Fib(n) \geq 2^{n/2} - 1$

Pf : Induction

TIME TAKEN is $\Omega(2^{n/2})$

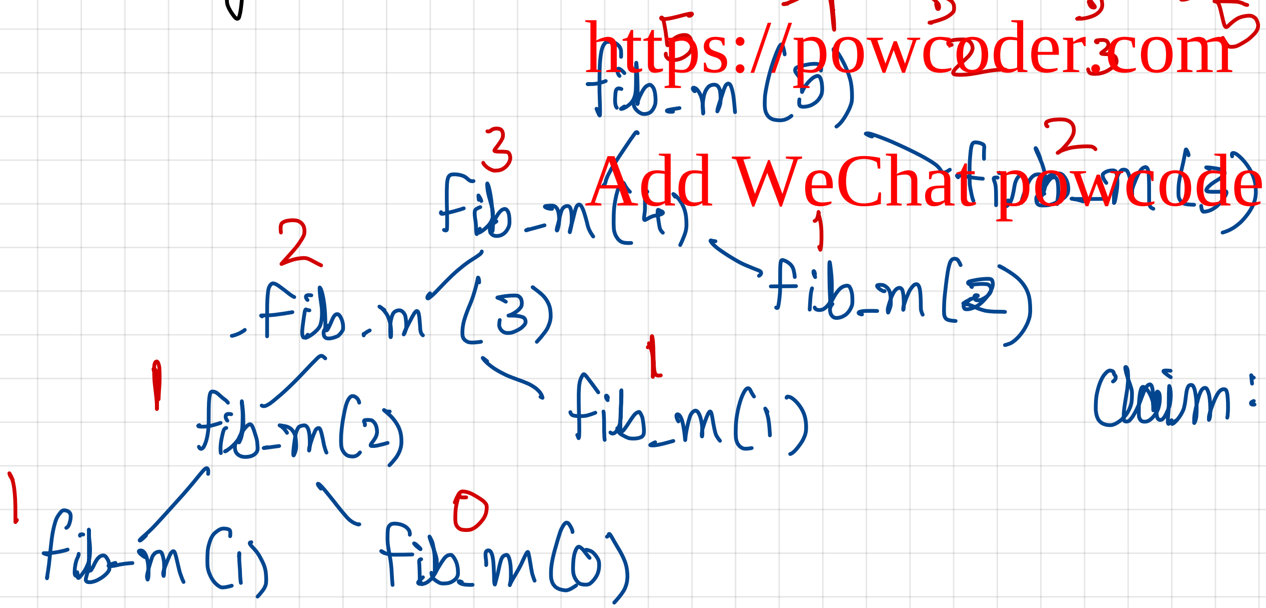
KEY IDEA 1 :

DO NOT REPEAT CALCULATIONS

STORE THEM TO BE USED LATER

MEMOIZATION

	0	1	2	3	4	5
fib_array	0	1	~~-1~~ 1	~~-1~~ 2	~~-1~~ 3	~~-1~~ 5




```
fib_array = 50 * [-1]
fib_array[0] = 0
fib_array[1] = 1

def fib_m(n):
    if fib_array[n] == -1
        fib_array[n]
            = fib_m(n-1)
              + fib_m(n-2)
    return fib_array[n]
```

Claim: Fib_m recursively computes each Fib number exactly once

$\Rightarrow$ $\boxed{\text{Running time of fib\_m is } O(n)}$

KEY IDEA 2: YOU DON'T NEED TO RECURSE
CAN COMPUTE ALL VALUES
BOTTOM-UP


```
def fib_nr(n):
  fib_array = (n+1) * [-1]
  fib_array[0] = 0
  fib_array[1] = 1
  for i in range(2:(n+1)):
      fib_array[i]
        = fib_array[i-1]
        + fib_array[i-2]
  return fib_array[n]
```

TIME O(n)

RECURSION
- REPEATED COMPUTATION
- MEMOIZATION
- BOTTOM-UP

DYNAMIC PROGRAMMING

SMART RECURSION
DONE BOTTOM-UP

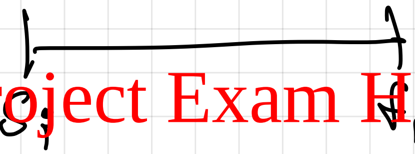
EVEN FASTER:

ONLY NEED TO STORE

Fib(n-1) & Fib(n-2)

MAX-WEIGHTED INTERVAL SCHEDULING

Given: n intervals

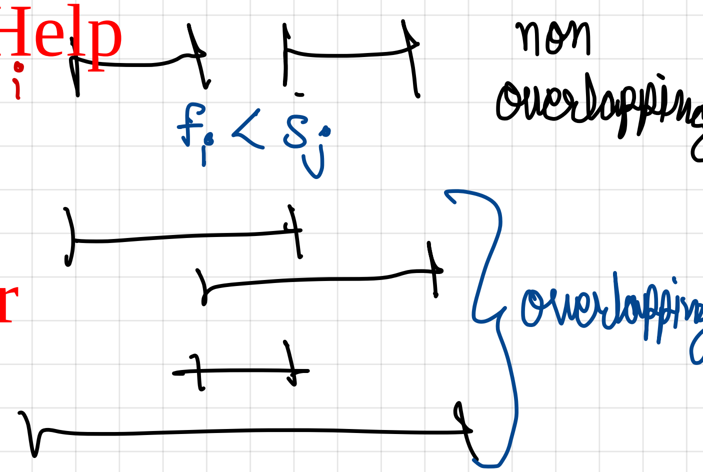


s_i start, f_i end

w_i weights

all pairs of intervals are non-overlapping

Goal: Find a subset of non-overlapping intervals with maximum total weight $\sum_{i \in I} w_i$



$\{1, 2, 3\}$ non-overlapping

$\Updownarrow$

1 & 2 don't overlap

1 & 3 " "

2 & 3 " "

non overlapping $f_i < s_j$ OR $f_j < s_i$

Question: Is there some structure to the overlap?

Given an interval j, can I identify all intervals that don't overlap with j?

$f_i < s_j$ $\qquad\qquad$ $f_j < s_i$

Idea: Sort the intervals by finish time

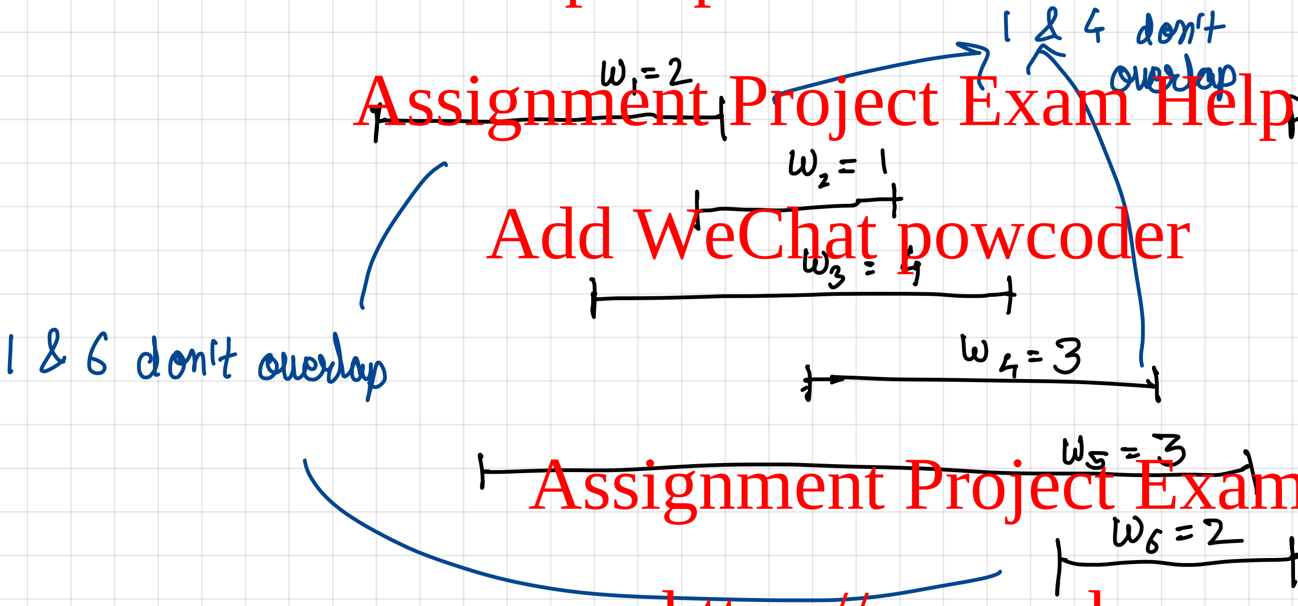
Assume: $f_1 \leq f_2 \leq f_3 \leq \dots \leq f_n$

Let $p[j]$ be the largest index i of an interval that doesn't overlap with j $\quad f_i < s_j$

j does not overlap with $\{1, 2, \dots, p[j]\}$

j overlaps with $\{p[j]+1, \dots, j-1\}$

MAXIMUM WEIGHT INTERVAL SCHEDULING



$p(1) = 0$

$p(2) = 0$

$p(3) = 0$

$p(4) = 1$

$p(5) = 0$

$p(6) = 3$

All intervals $\{1, \ldots, p[j]\}$ don't overlap with j

All " $\{p[j]+1, \ldots, j-1\}$ overlap w/ j

$OPT(\{1,2,\ldots,j\})$

j is not in OPT

j is in OPT

→ j is contained by assumption

$OPT(\{1,2,\ldots,j-1\})$

$OPT = \{j\}$

$\cup$ (some intervals that don't overlap with j) & are non-overlapping themselves

come from $\{1,2,\ldots,p[j]\}$

$$OPT(\{1,\ldots,j\}) = \begin{cases} 0 & \text{if } j=0 \\ \max\{ OPT(\{1,\ldots,j-1\}), \\ \quad w_j + OPT(\{1,2,\ldots,p[j]\}) \} \end{cases}$$

Any non-overlapping subset is okay

⇓

Claim: This subset must be $OPT(\{1,2,\ldots,p[j]\})$

Pf: O/w $\{j\} \cup OPT(\{1,2,\ldots,p[j]\})$ will have better cost