MONASH University

MONASH
INFORMATION
TECHNOLOGY

FIT2004
Algorithms and Data Structures

Ian Wern Han Lim
lim.wern.han@monash.edu

Referencing materials by
Nathan Companez, Aamir Cheema, Arun Konagurthu and Lloyd Allison

GROUP OF EIGHT AUSTRALIA

Faculty of Information Technology, Monash University

Ready?

Agenda

- Dynamic Programming

Agenda

- Dynamic Programming
  - Brute force (aka the starting point)

Agenda

- Dynamic Programming
  - Brute force (aka the starting point)
  - Overlapping sub-problems sub-problems must be overlapped
  - Backtracking

Agenda

- Dynamic Programming
  - Brute force (aka the starting point)
  - Overlapping sub-problems
  - Backtracking

  - Fibonacci
  - Coin change
  - Knapsack
    - Unbounded
    - 0/1
  - Edit distance

Agenda

- Dynamic Programming
  - Brute force (aka the starting point)
  - Overlapping sub-problems
  - Backtracking for solution reconstruction

  - Fibonacci
  - Coin change
  - Knapsack
    - Unbounded
    - 0/1
  - Edit distance

Monash University

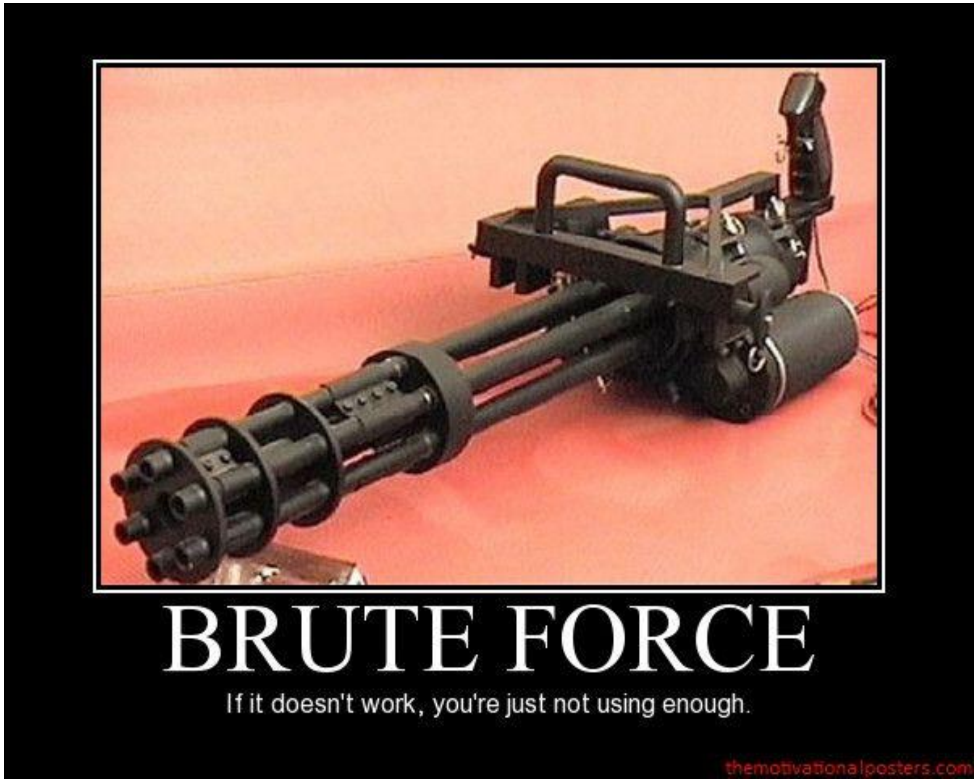
Let us begin…

Dynamic Programming

… a smarter way to brute force

- Can we brute force everything?

- Can we brute force everything?



MONASH University

- Can we brute force everything?
  - So if we can brute force anything, why not just use it?

MONASH University

- Can we brute force everything?
  - So if we can brute force anything, why not just use it?
  - Because brute forcing needs too much effort…

- Can we brute force everything?
  - So if we can brute force anything, why not just use it?
  - Because brute forcing needs too much effort…

- Consider the Fibonacci problem
  - Find the n-th Fibonacci number

Questions?

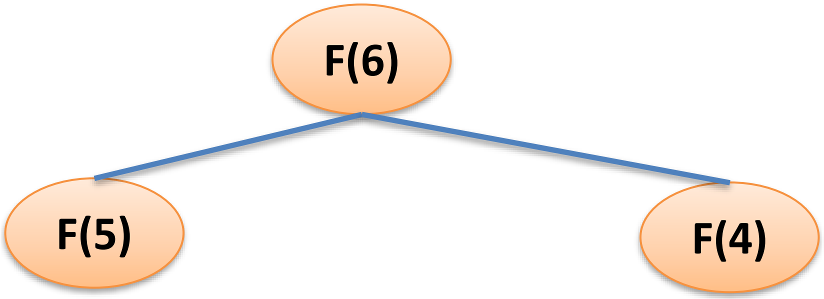
Fibonacci
n-th number in the series

- Now if we want the 6th Fibonacci number…
  - How would we get it?

F(6)

Fibonacci
n-th number in the series

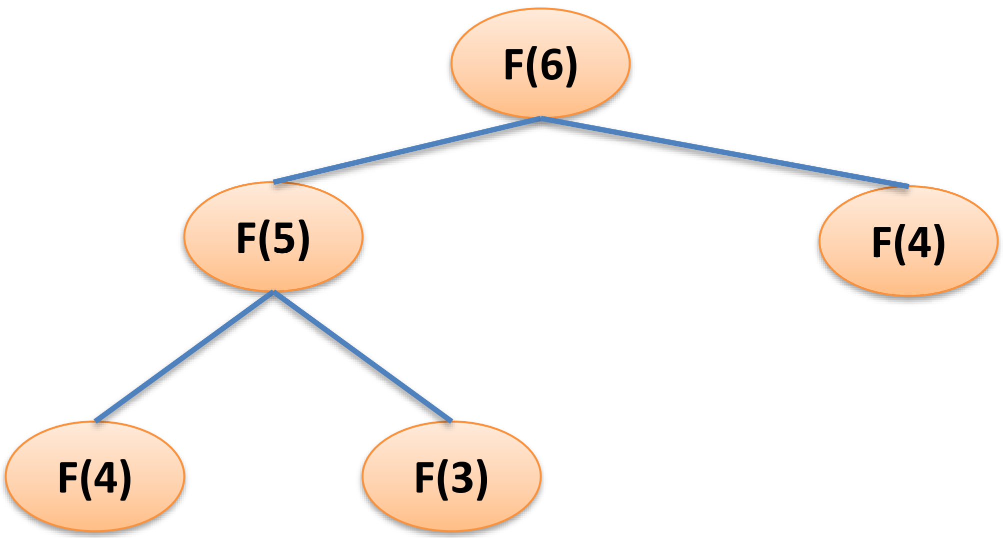
- Now if we want the 6th Fibonacci number…
  - How would we get it?



Fibonacci

n-th number in the series

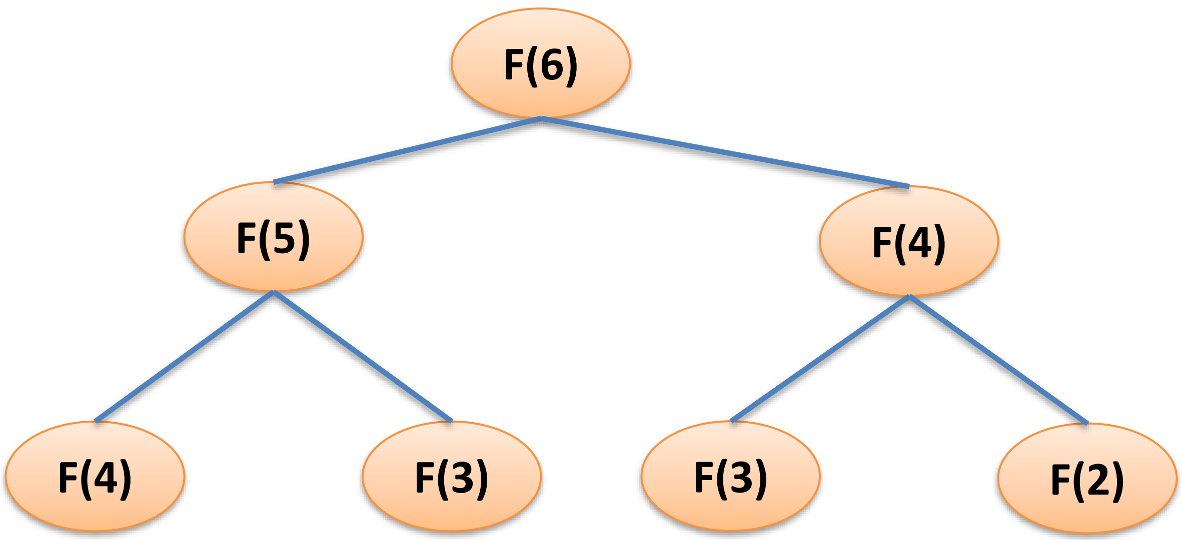
- Now if we want the 6th Fibonacci number…
  - How would we get it?



Fibonacci

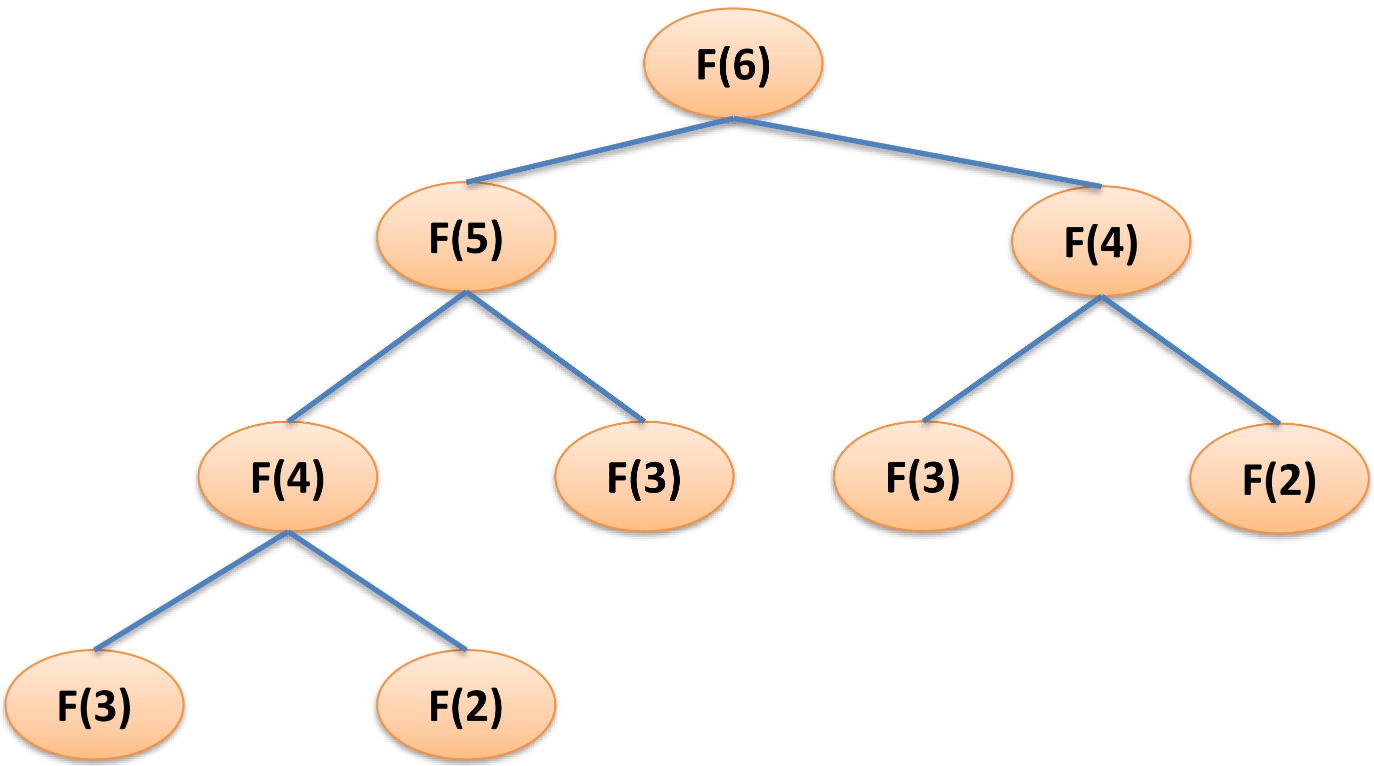
n-th number in the series

- Now if we want the 6th Fibonacci number…
  - How would we get it?

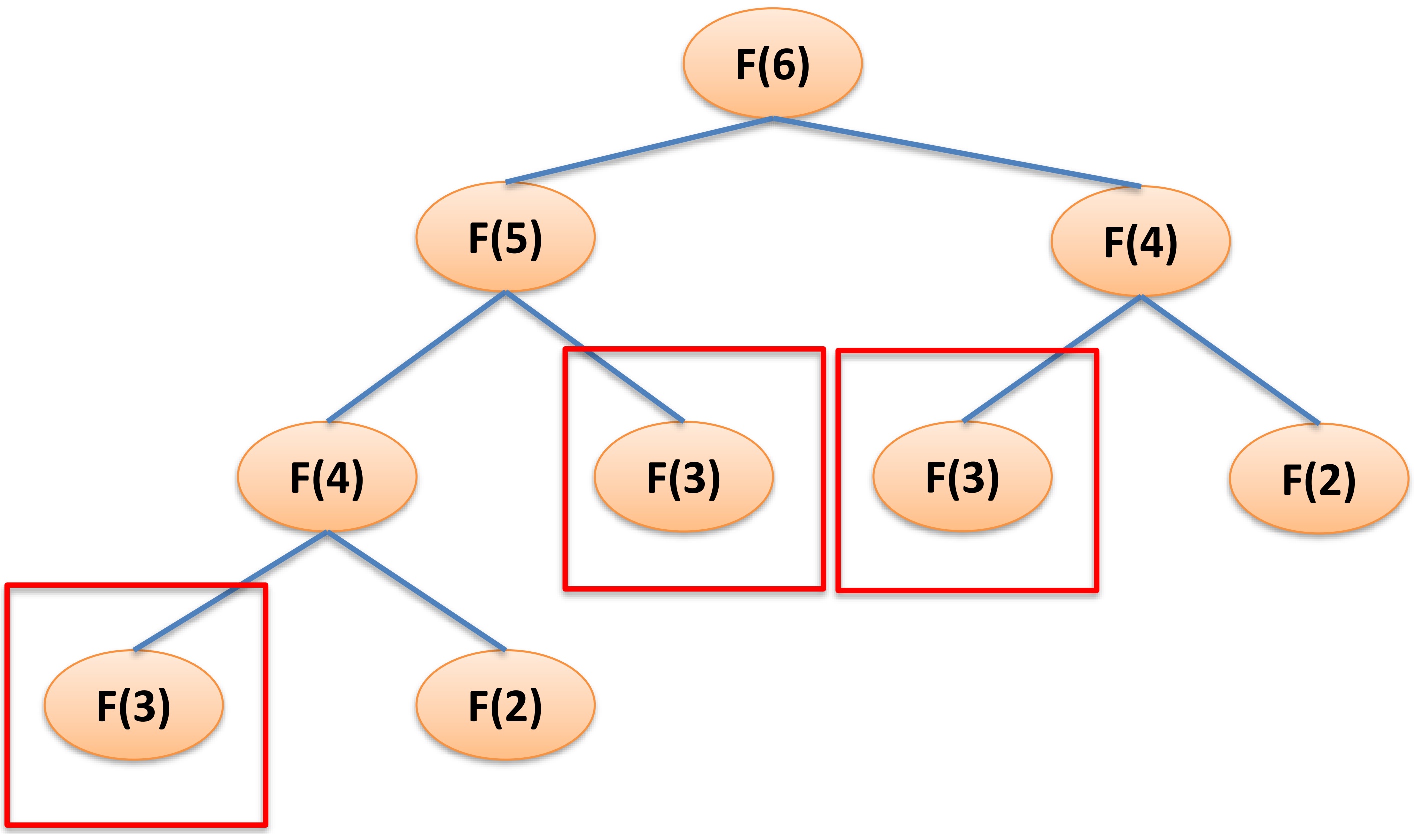


Fibonacci
n-th number in the series

- Now if we want the 6th Fibonacci number…
  - How would we get it?

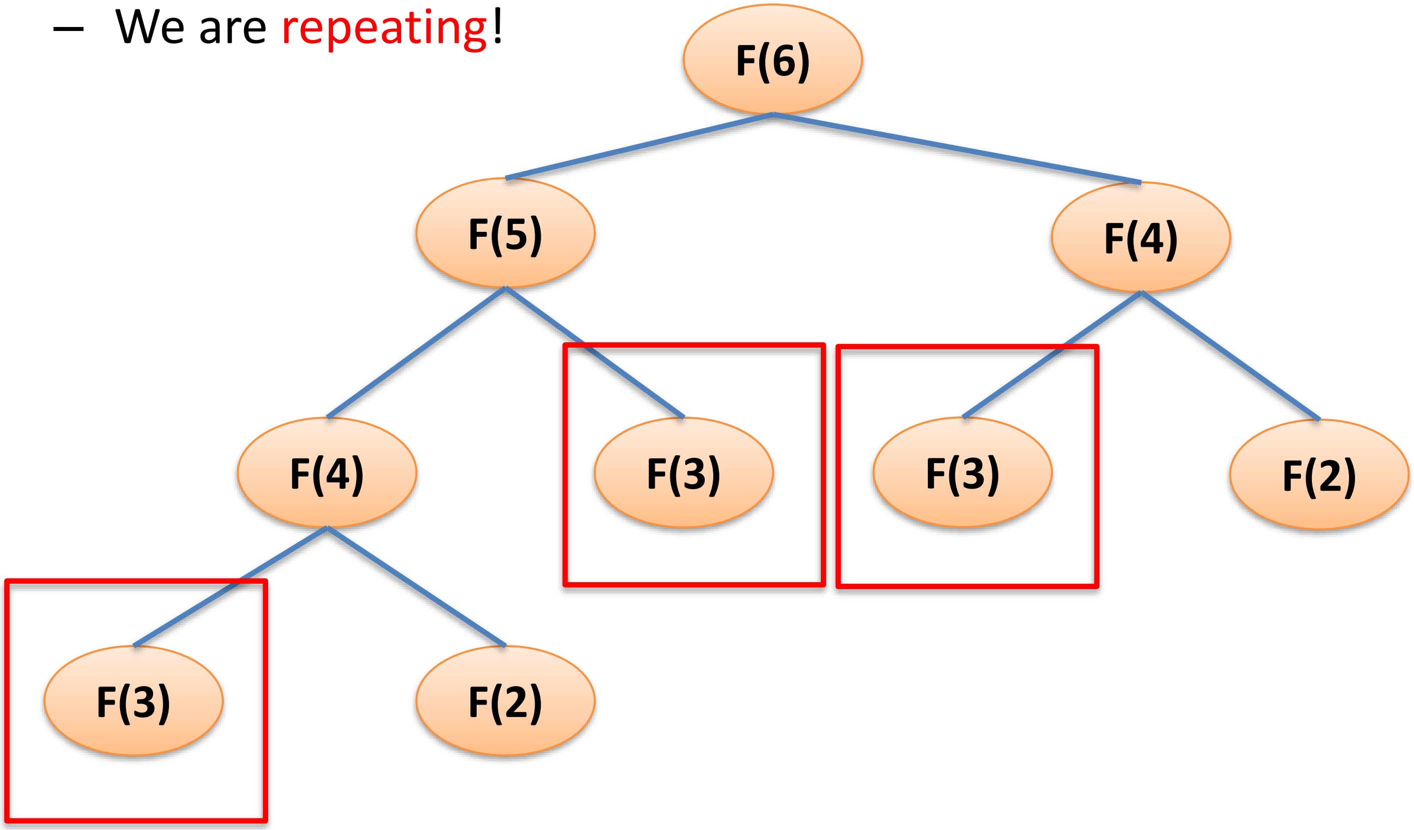


- Now if we want the 6th Fibonacci number…
  - How would we get it?



Fibonacci
n-th number in the series

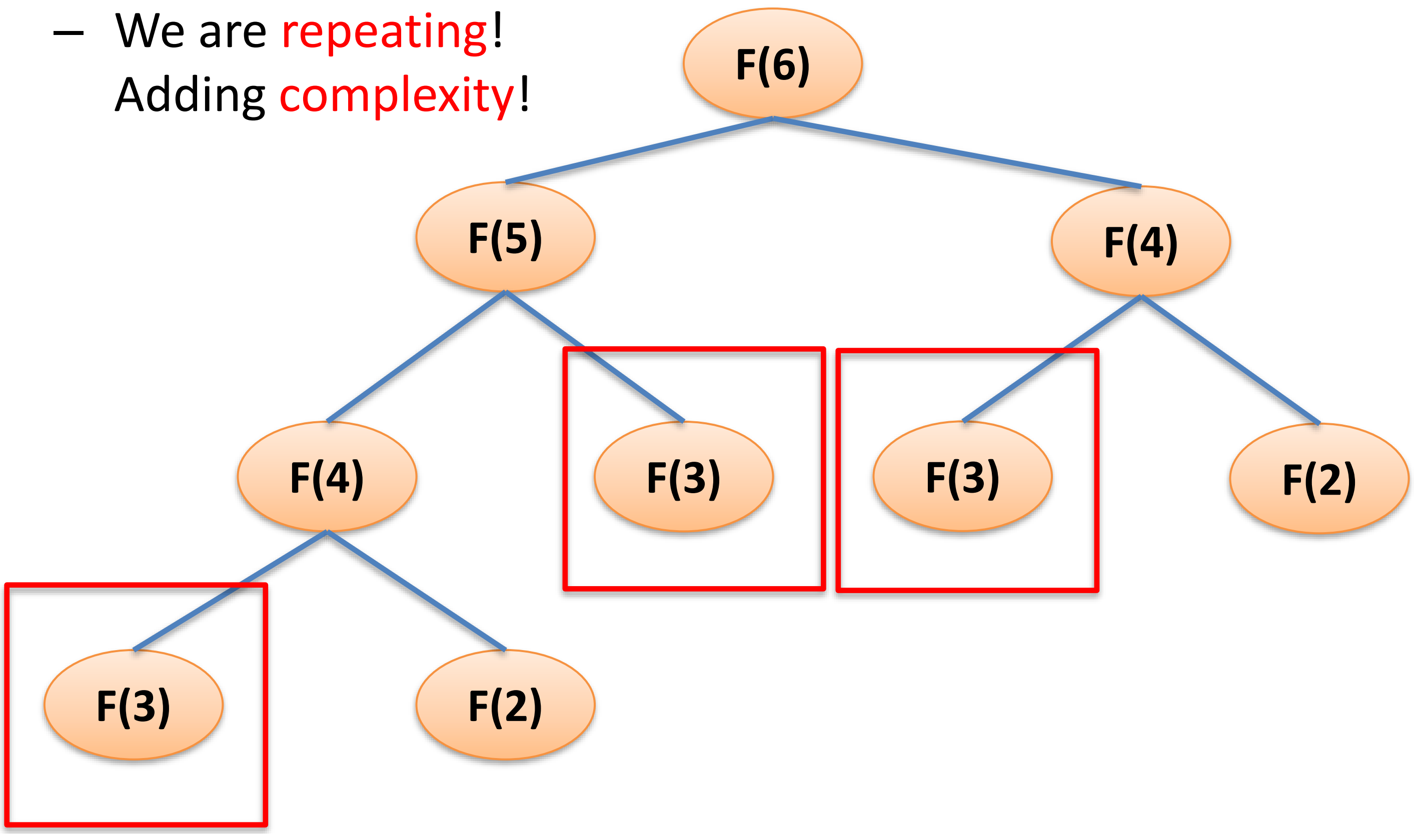
- Now if we want the 6th Fibonacci number…
  - How would we get it?
  - We are repeating!



Fibonacci

n-th number in the series

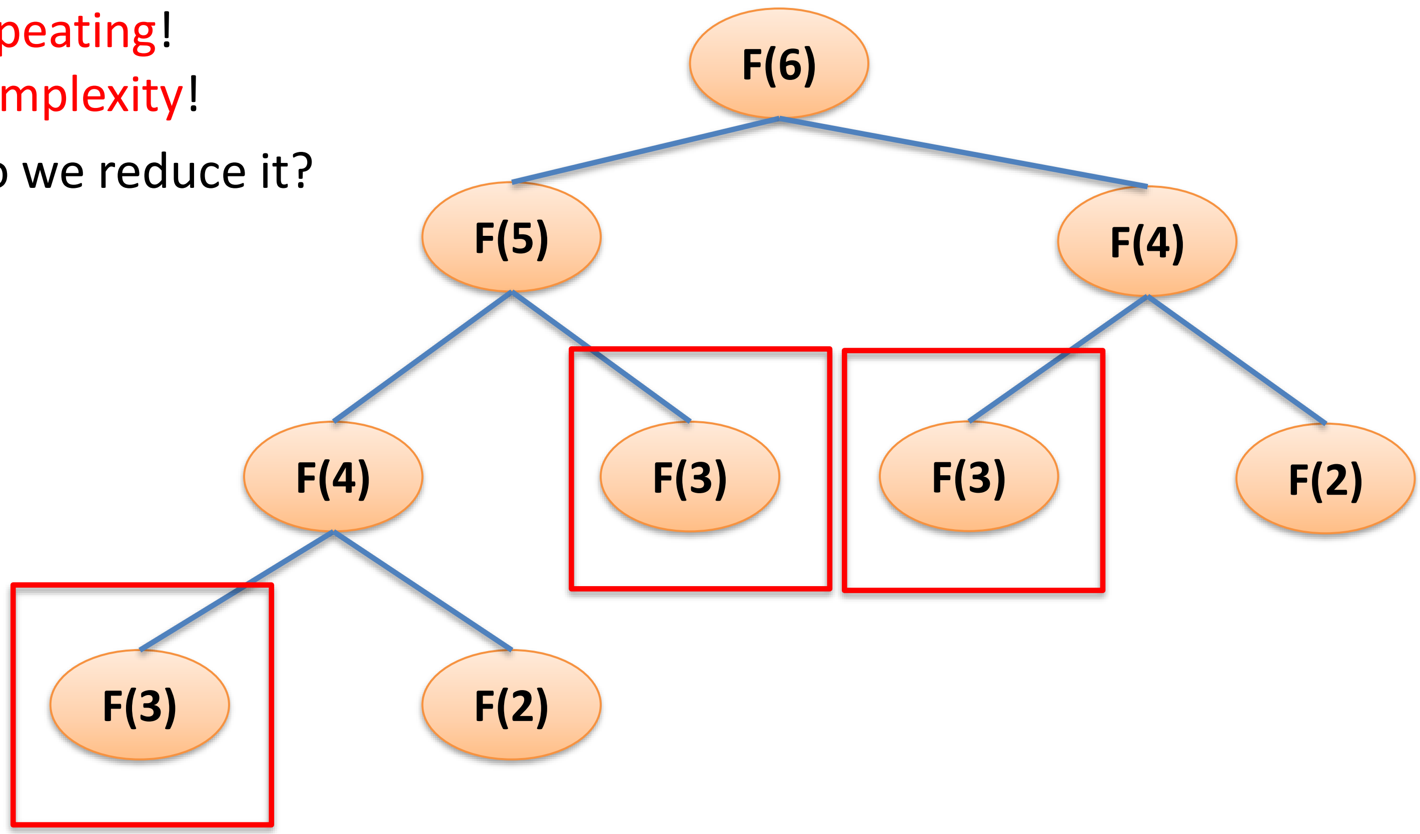
- Now if we want the 6th Fibonacci number…
  - How would we get it?
  - We are repeating! Adding complexity!



Fibonacci

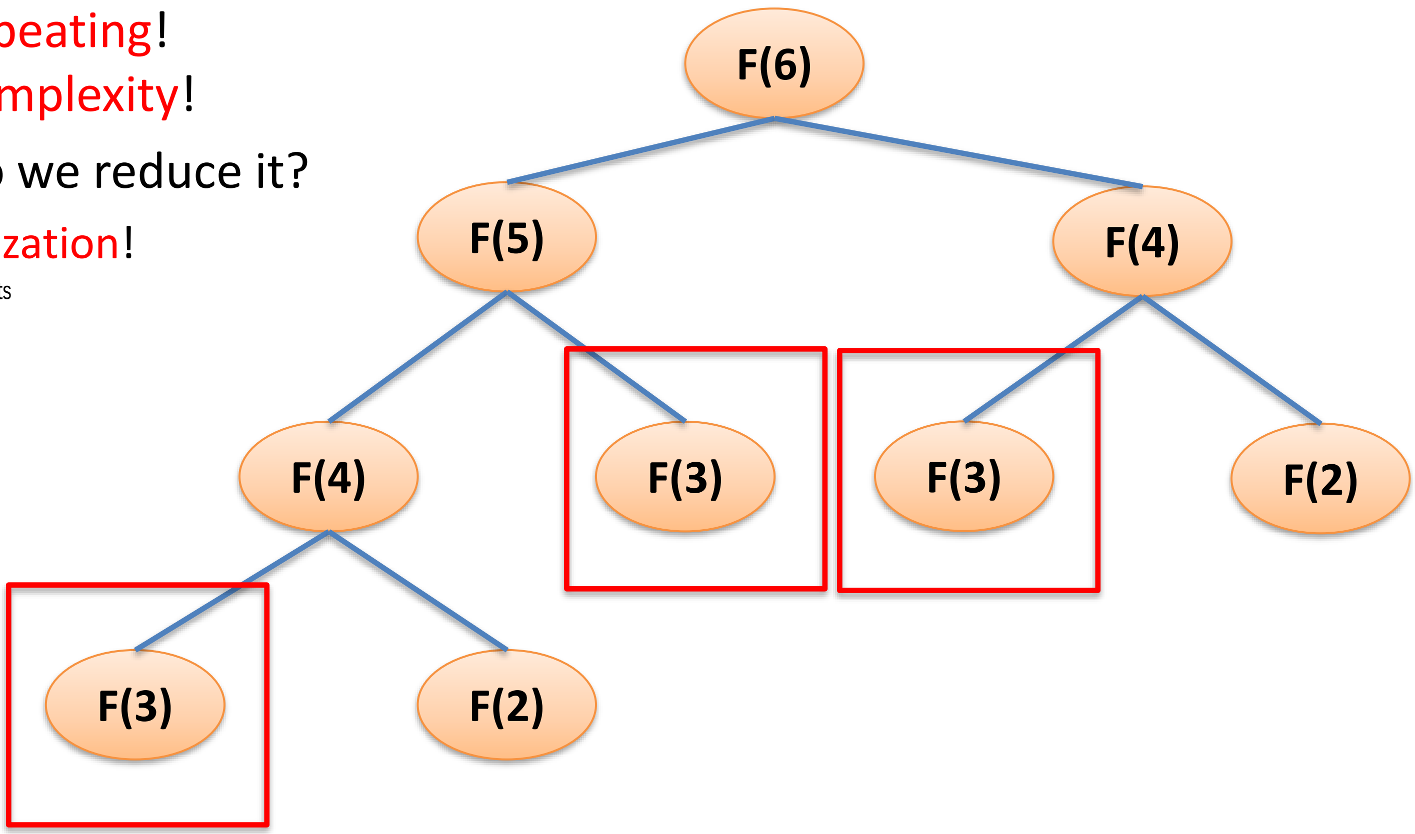
n-th number in the series

- Now if we want the 6th Fibonacci number…
  - How would we get it?
  - We are repeating!
    Adding complexity!
  - So how do we reduce it?



- Now if we want the 6th Fibonacci number…
  - How would we get it?
  - We are repeating!
    Adding complexity!
  - So how do we reduce it?
    - Memoization!

memorise results



Questions?

Fibonacci
n-th number in the series

- Now if we want the 6th Fibonacci number…
  - How would we get it?
  - We are repeating!
    Adding complexity!
  - So how do we reduce it?
    - Memoization!
    - Quick real world examples…

Fibonacci

n-th number in the series

- Now if we want the 6th Fibonacci number…
  - How would we get it?
  - We are repeating!
    Adding complexity!
  - So how do we reduce it?
    - Memoization!
    - Quick real world examples…
      - What is 12x12?

Fibonacci

n-th number in the series

- Now if we want the 6^{th} Fibonacci number…
  - How would we get it?
  - We are repeating!
    Adding complexity!
  - So how do we reduce it?
    - Memoization!
    - Quick real world examples…
      - What is 12x12?
      - What is 12x13?

Fibonacci
n-th number in the series

- Now if we want the 6th Fibonacci number…
  - How would we get it?
  - We are repeating!
    Adding complexity!
  - So how do we reduce it?
    - Memoization!
    - Quick real world examples…
      - What is 12x12?
      - What is 12x13?
      - What is 12x14?

Fibonacci

n-th number in the series

- Now if we want the 6^{th} Fibonacci number…
  - How would we get it?
  - We are repeating!
    Adding complexity!
  - So how do we reduce it?
    - Memoization!
    - Quick real world examples…
      - What is 12x12? Remember this
      - What is 12x13? Used for this…
      - What is 12x14? Used for this…

Fibonacci
n-th number in the series

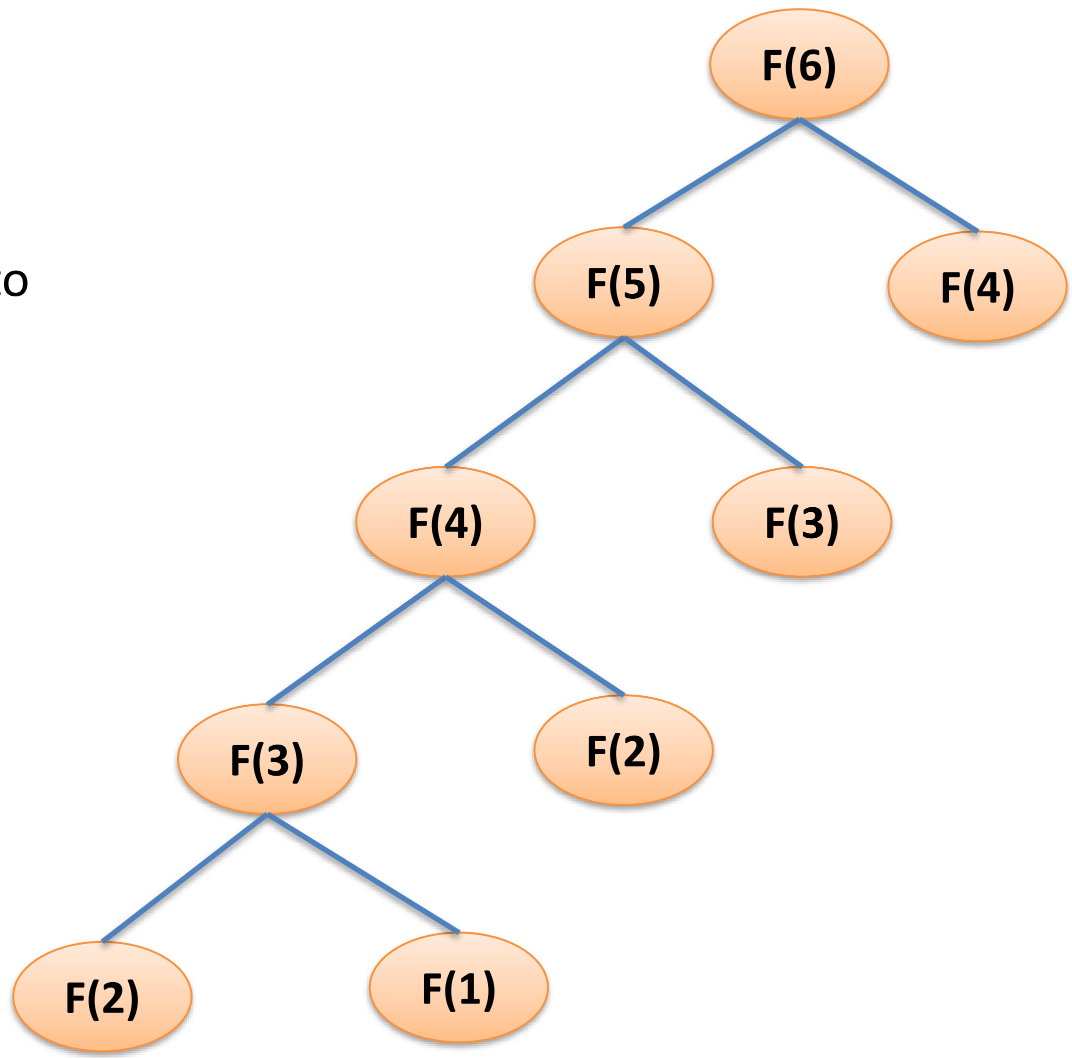
- Now if we want the 6th Fibonacci number…
  - How would we get it?
  - We are repeating!
    Adding complexity!
  - So how do we reduce it?
    - Memoization!
    - Quick real world examples…
      - What is 12x12? Remember this
      - What is 12x13? Used for this…
      - What is 12x14? Used for this…
    - Let’s apply to Fibonacci

Questions?

- Top-down approach
  - Start from top

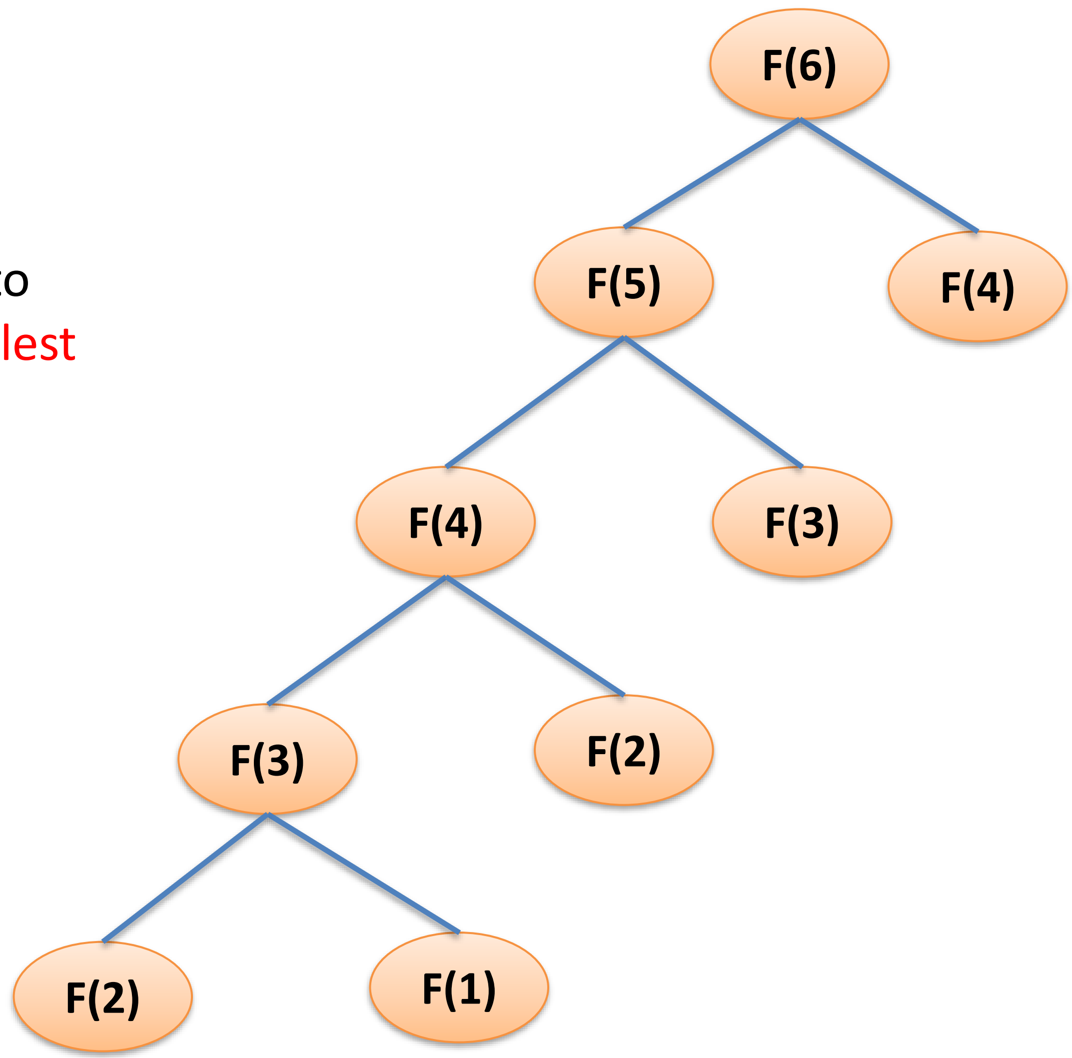
Fibonacci
n-th number in the series

- Top-down approach
  - Start from top
  - Keep on breaking it into smaller problem



Fibonacci
n-th number in the series

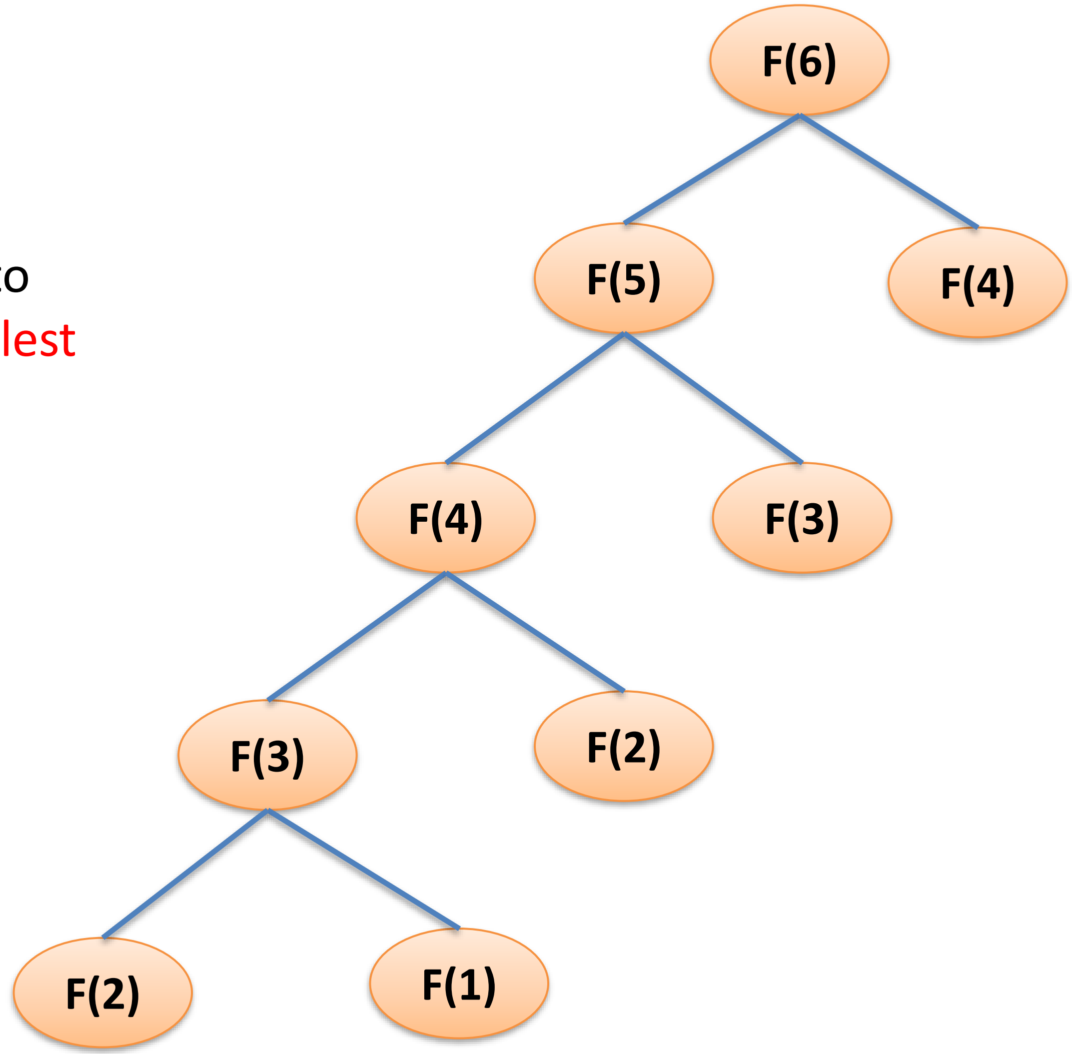
- Top-down approach
  - Start from top
  - Keep on breaking it into smaller problem. Smallest is the base case!



Fibonacci

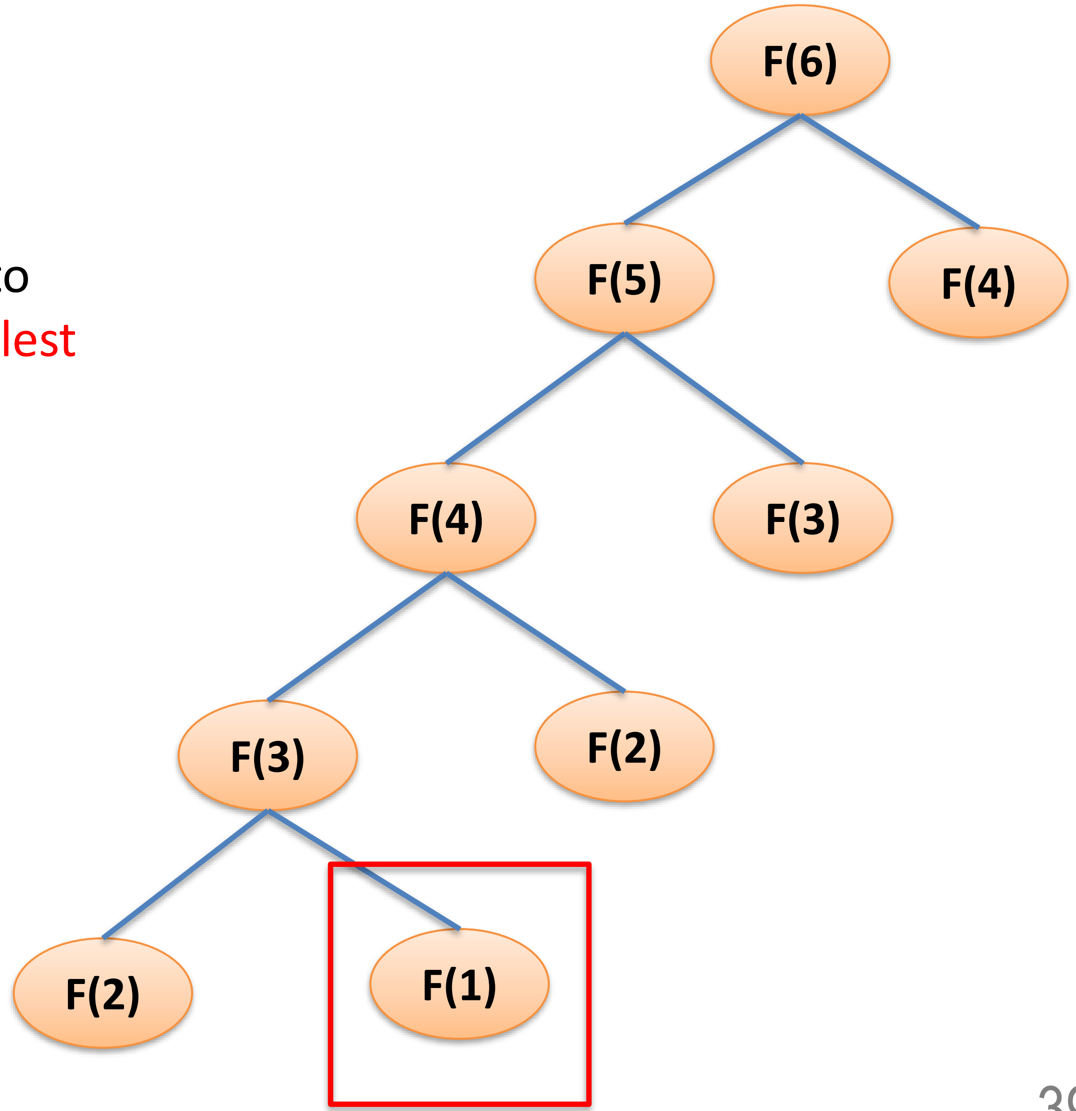
n-th number in the series

- Top-down approach
  - Start from top
  - Keep on breaking it into smaller problem. Smallest is the base case!
  - Then solve it, reusing the results



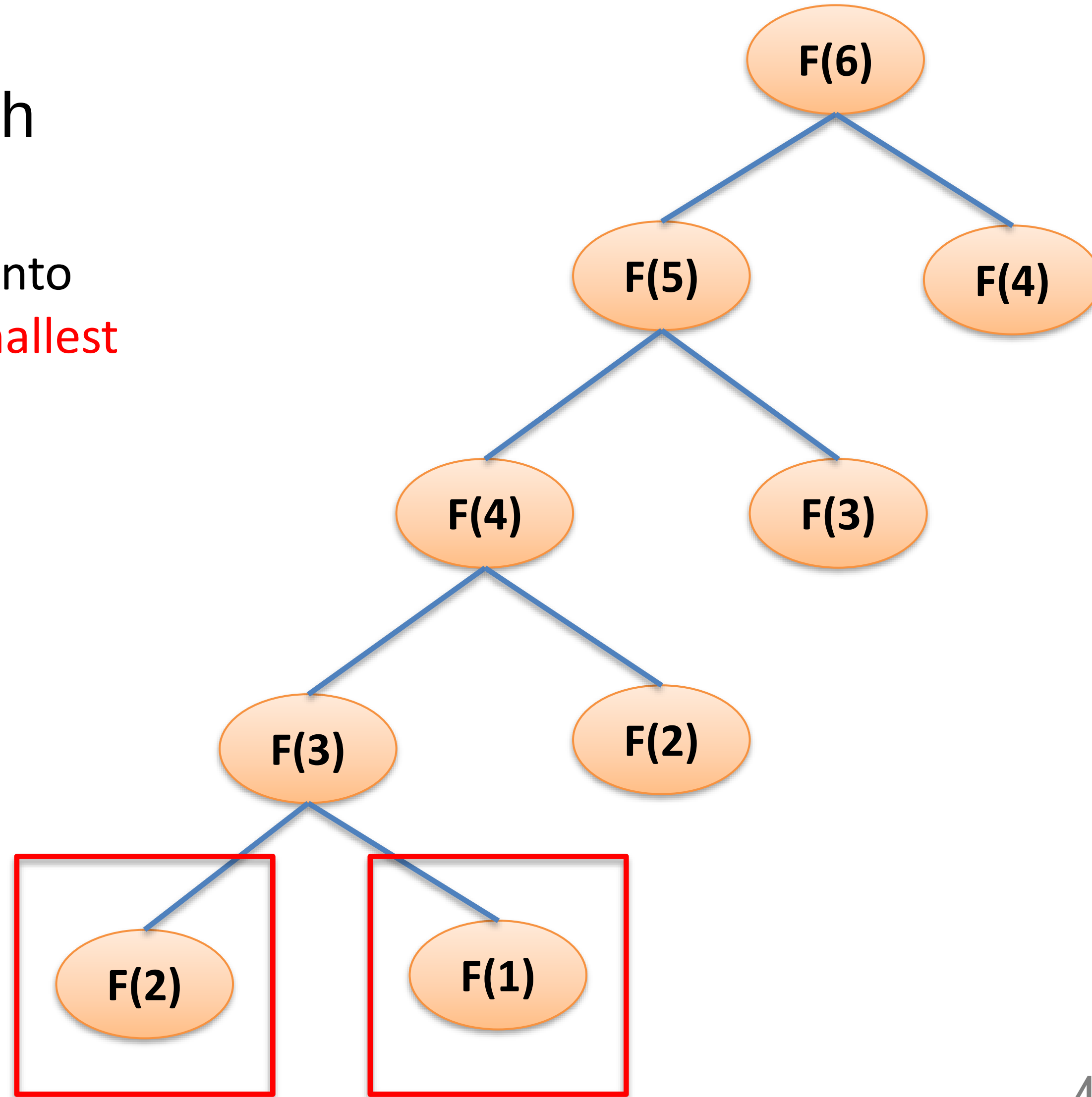
Fibonacci
n-th number in the series

- Top-down approach
  - Start from top
  - Keep on breaking it into smaller problem. Smallest is the base case!
  - Then solve it, reusing the results



Fibonacci
n-th number in the series

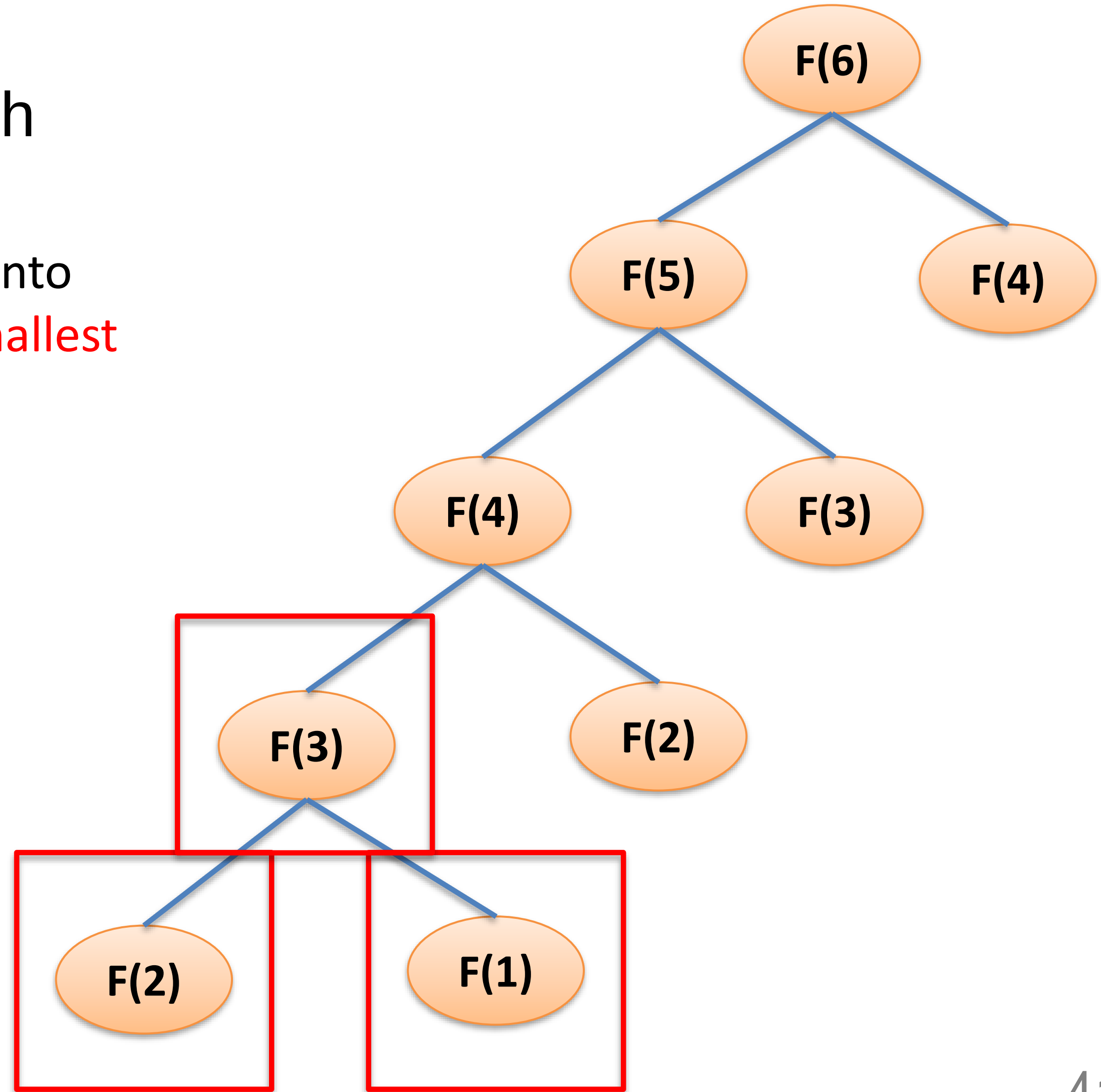
- Top-down approach
  - Start from top
  - Keep on breaking it into smaller problem. Smallest is the base case!
  - Then solve it, reusing the results



Fibonacci

n-th number in the series

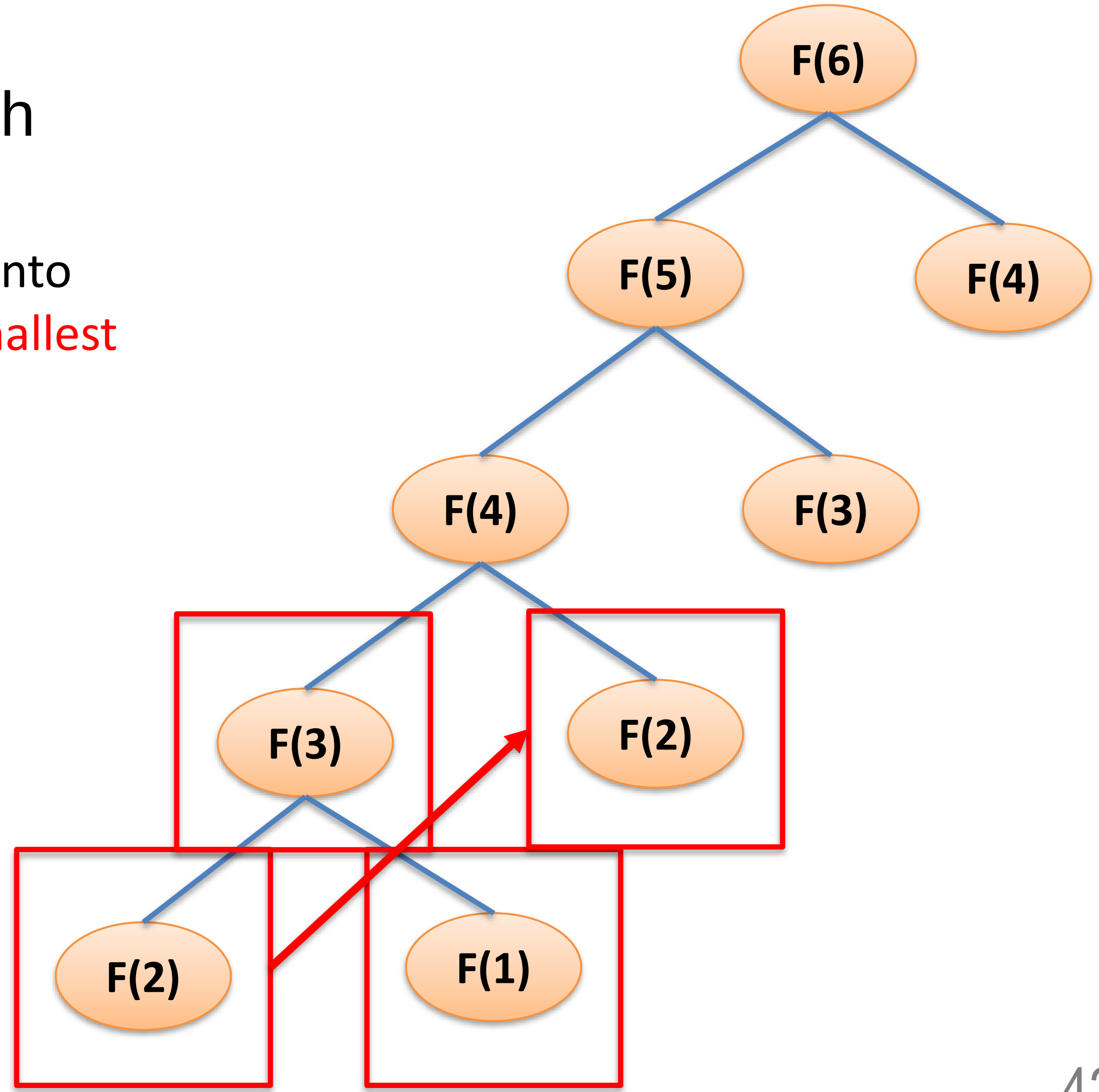
- Top-down approach
  - Start from top
  - Keep on breaking it into smaller problem. Smallest is the base case!
  - Then solve it, reusing the results



Fibonacci

n-th number in the series

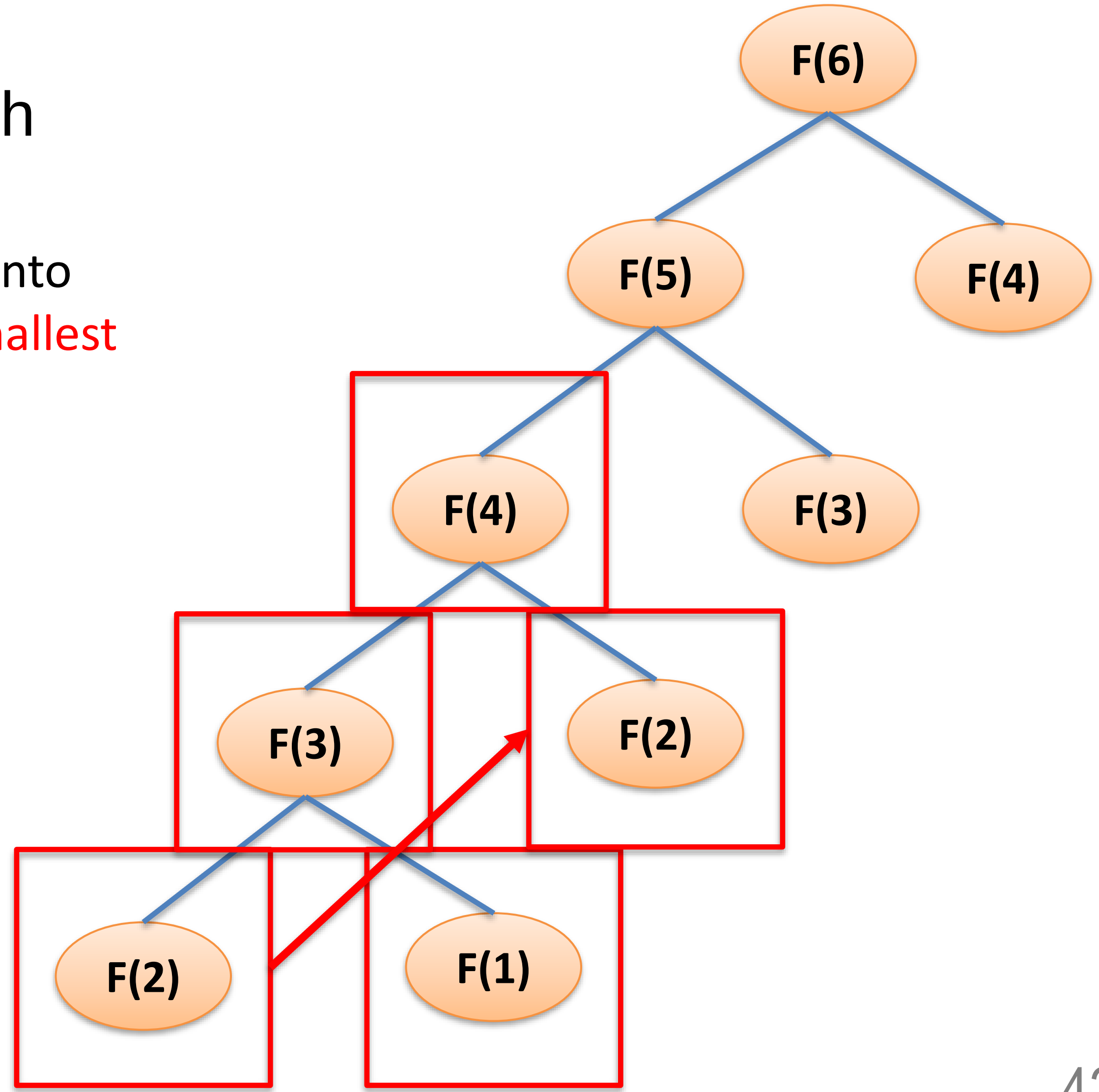
- Top-down approach
  - Start from top
  - Keep on breaking it into smaller problem. Smallest is the base case!
  - Then solve it, reusing the results



Fibonacci

n-th number in the series

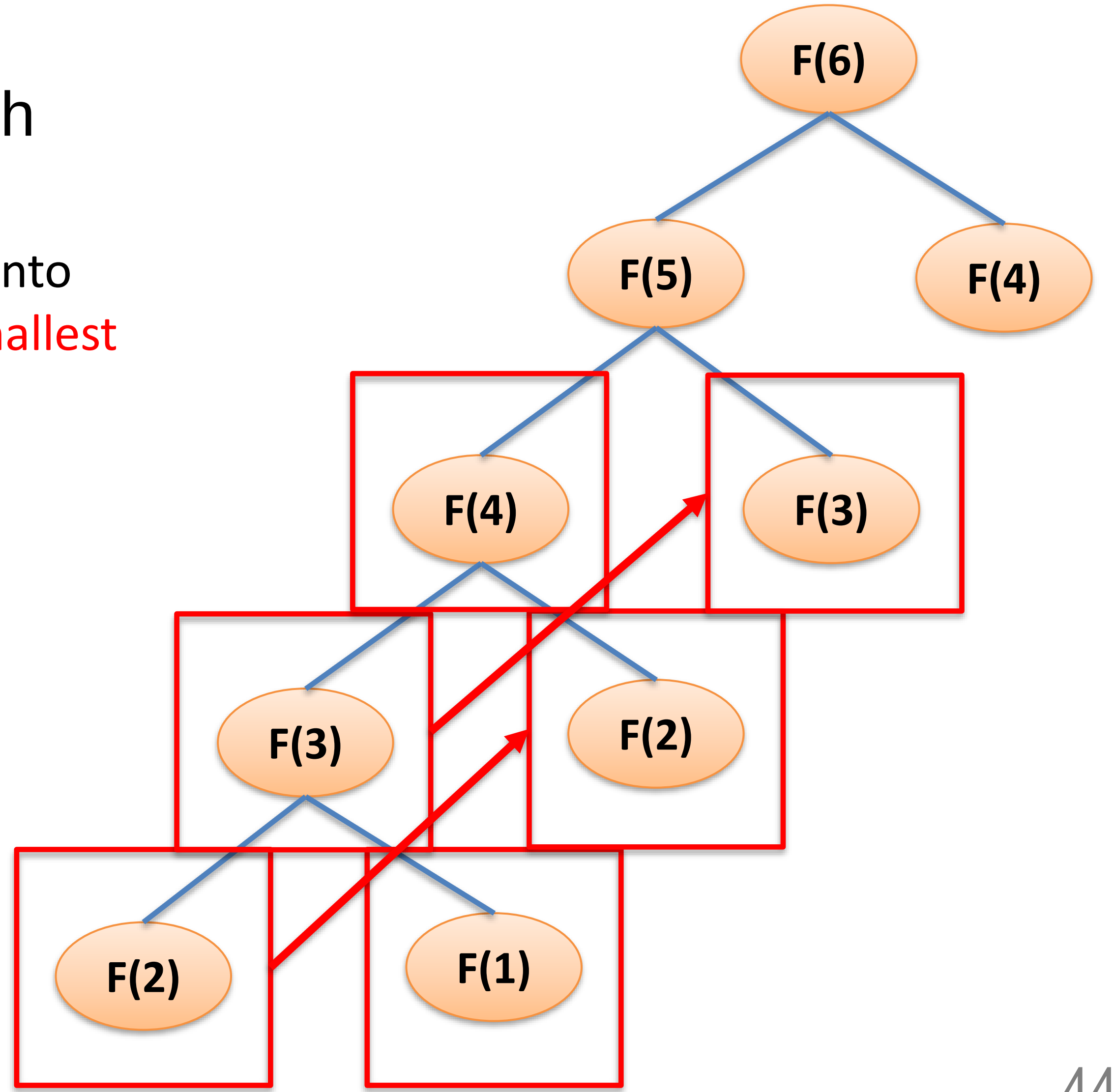
- Top-down approach
  - Start from top
  - Keep on breaking it into smaller problem. Smallest is the base case!
  - Then solve it, reusing the results



Fibonacci

n-th number in the series

- Top-down approach
  - Start from top
  - Keep on breaking it into smaller problem. Smallest is the base case!
  - Then solve it, reusing the results



Fibonacci

n-th number in the series

- Top-down approach
  - Start from top
  - Keep on breaking it into smaller problem. Smallest is the base case!
  - Then solve it, reusing the results

```
memo = [-1] * N
memo[0] = 0              # in CompSci, we start from 0
memo[1] = 1

def fib_dpa(N):
    if memo[N] != -1:    # if we have computed before
        return memo[N]
    # computer the fibonacci
    memo[N] = fib_dpa(N-1) + fib_dpa(N-2)
    return memo[N]
```

if the N index inside memo list is not -1(there is value computed and stored< directly reuse it)

Questions?

- Is it the same?

- Is it the same?
  - Take a big thing
  - Divide to a smaller one
  - Solve the smaller one

Dynamic Programming
vs Divide and Conquer

- Is it the same?
  - Take a big thing
  - Divide to a smaller one
  - Solve the smaller one
  - Combine solutions up to the big thing…

Dynamic Programming
vs Divide and Conquer

- Is it the same?
  - Take a big thing
  - Divide to a smaller one
  - Solve the smaller one
  - Combine solutions up to the big thing…

- NO!... But why?

Dynamic Programming
vs Divide and Conquer

- Is it the same?
  - Take a big thing
  - Divide to a smaller one
  - Solve the smaller one.
    These solutions are reusable due to overlapping sub problems
  - Combine solutions up to the big thing…

- NO!... But why?

MONASH University

- Is it the same?
  - Take a big thing
  - Divide to a smaller one
  - Solve the smaller one.
    These solutions are reusable due to overlapping sub problems
  - Combine solutions up to the big thing…
- NO!... But why?
- Let us look at bottom-up to understand better…

Dynamic Programming
vs Divide and Conquer

- Is it the same?
  - Take a big thing
  - Divide to a smaller one
  - Solve the smaller one.
    These **OPTIMAL** solutions are reusable due to overlapping sub problems
  - Combine solutions up to the big thing…

differentiate between divide and conquer
and dynamic programming

since only computed once and used forever

merge sort has not repetitive subpart solution
can not be reused
so merge sort is devide and conquer(recursion)
dynamic programming has subpart solution to remember
to trade for faster computation speed

- NO!... But why?

- Let us look at bottom-up to understand better…

Questions?

- Bottom-up approach
  - Can you explain it?

Fibonacci
n-th number in the series

- Bottom-up approach
  - Can you explain it?

Fibonacci

n-th number in the series

- Bottom-up approach
  - Start from the base case
  - Solve it
  - Use it to solve bigger case
  - Until we reach the final one…

Fibonacci
n-th number in the series

- Bottom-up approach
  - Start from the base case
  - Solve it
  - Use it to solve bigger case
  - Until we reach the final one…

```
memo = [-1] * N
memo[0] = 0              # in CompSci, we start from 0
memo[1] = 1

for i in range(2,N):
    memo[i] = memo[i-1] + memo[i-2]
```

Questions?

Fibonacci
n-th number in the series

- What is the complexity?
  - Bottom-up
  - Top-down

Fibonacci

n-th number in the series

- What is the complexity?
  - Bottom-up
  - Top-down

recursice (Bottom-up)

```
memo = [-1] * N
memo[0] = 0            # in CompSci, we start from 0
memo[1] = 1

def fib_dpa(N):
    if memo[N} != -1:    # if we have computed before
        return memo[N]
    # computer the fibonacci
    memo[N] = fib_dpa(N-1) + fib_dpa(N-2)
    return memo[N]
```

iterative

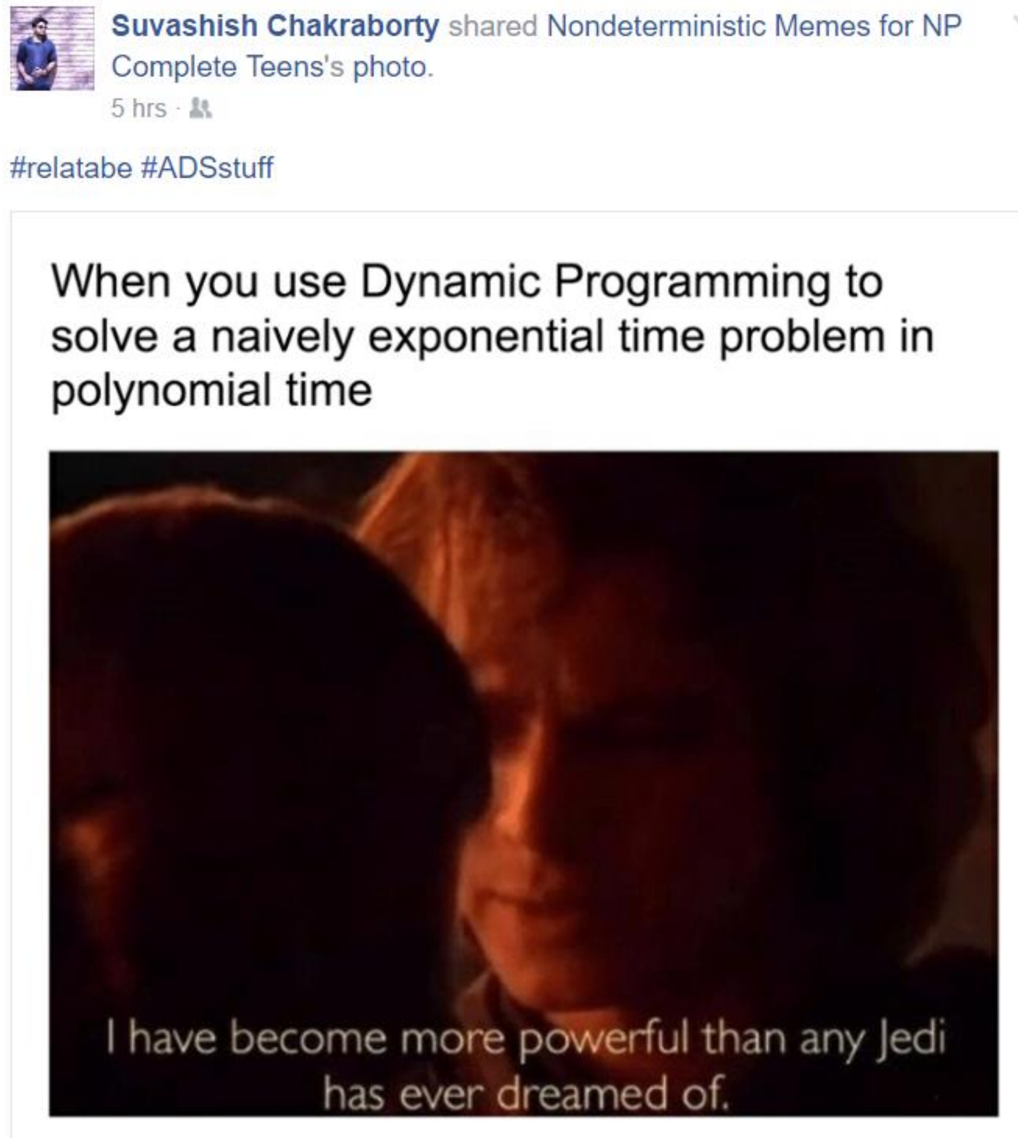
```
memo = [-1] * N
memo[0] = 0            # in CompSci, we start from 0
memo[1] = 1

for i in range(2,N):
    memo[i] = memo[i-1] + memo[i-2]
```

Fibonacci

n-th number in the series

- What is the complexity?
  - Bottom-up
  - Top-down

brute force $O(2^n)$



Questions?

- Sounds easy right?

- Sounds easy right?
- Can be used to solve a lot of problem
  - Especially finding combinations

- Sounds easy right?
- Can be used to solve a lot of problem
  - Especially finding combinations
  - Popular in interviews

The superpower

MONASH University

- Sounds easy right?
- Can be used to solve a lot of problem
  - Especially finding combinations
  - Popular in interviews
  - Very popular in programming competitions!

- Sounds easy right?
  - It isn’t that easy however… Why?

- Can be used to solve a lot of problem
  - Especially finding combinations
  - Popular in interviews
  - Very popular in programming competitions!

- Sounds easy right?
  – It isn't that easy however… Why? Not easy to break problems down

- Can be used to solve a lot of problem
  – Especially finding combinations
  – Popular in interviews
  – Very popular in programming competitions!

- Sounds easy right?
  - It isn’t that easy however… Why? Not easy to break problems down

- So how?

MONASH University

- Sounds easy right?
  - It isn’t that easy however… Why? Not easy to break problems down

- So how?
  - Practice

- Sounds easy right?
  - It isn’t that easy however… Why? Not easy to break problems down

- So how?
  - Coin change
  - Knapsack
  - Edit-distance

Questions?

Coin Change problem

The less number of coins…

- Consider the following scenario

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}

Coin Change problem
The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}
  - I want 110 doge coin, so what is the possible coin combination?

Coin Change problem
The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}
  - I want 110 doge coin, so what is the possible coin combination?
    - 2x50 + 1x10
    - 11x10
    - … and many more

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}
  - I want 110 doge coin, so what is the possible coin combination?
    - 2x50 + 1x10 (3 coins)
    - 11x10
    - … and many more
    - But we want the smallest number of coins!

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}
  - I want 110 doge coin, so what is the possible coin combination?
    - 2x50 + 1x10 (3 coins)
    - 11x10
    - … and many more
    - But we want the smallest number of coins!

- Let us now explore the possible solutions…

Questions?

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}
  - I want 110 doge coin, so what is the possible coin combination?

- Brute force?

Coin Change problem
The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}
  - I want 110 doge coin, so what is the possible coin combination?

- Brute force?
  - Try every combination!

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}
  - I want 110 doge coin, so what is the possible coin combination?
- Brute force?
  - Try every combination!
  - Choose the smallest number of coin…

Coin Change problem
The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}
  - I want 110 doge coin, so what is the possible coin combination?
- Brute force?
  - Try every combination!
  - Choose the smallest number of coin…
  - Will it work? Of course!

Questions?

Coin Change problem
The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}
  - I want 110 doge coin, so what is the possible coin combination?
- Greedy solution?

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}
  - I want 110 doge coin, so what is the possible coin combination?

- Greedy solution?
  - Try with the biggest number of coin
  - Then fill the balance with the rest #ez

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}
  - I want 110 doge coin, so what is the possible coin combination?
- Greedy solution?
  - Try with the biggest number of coin
  - Then fill the balance with the rest #ez
  - Doesn't always work (greed is not good)

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}
  - I want 110 doge coin, so what is the possible coin combination?

- Greedy solution?
  - Try with the biggest number of coin
  - Then fill the balance with the rest #ez
  - Doesn’t always work (greed is not good)
    {1,5,6,9} and I want 12

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}
  - I want 110 doge coin, so what is the possible coin combination?

- Greedy solution?
  - Try with the biggest number of coin
  - Then fill the balance with the rest #ez
  - Doesn’t always work (greed is not good)
    {1,5,6,9} and I want 12
    - 1x9 + 3x1 = 12 for 4 coins

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}
  - I want 110 doge coin, so what is the possible coin combination?

- Greedy solution?
  - Try with the biggest number of coin
  - Then fill the balance with the rest #ez
  - Doesn't always work (greed is not good) {1,5,6,9} and I want 12
    - 1x9 + 3x1 = 12 for 4 coins
    - 2x6 = 12 for 2 coins…

Questions?

- Consider the following scenario
  - Dogecoin currency is {1,5,10,50}
  - I want 110 doge coin, so what is the possible coin combination?
- Dynamic solution?
  - Let us try a smaller problem (easier for me to visualize)

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
  - I want 12 doge coin value, so what is the possible coin combination?

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
  - I want 12 doge coin value, so what is the possible coin combination?

base case

N+1

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins													

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
  - I want 12 doge coin value, so what is the possible coin combination?

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0												

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
  - I want 12 doge coin value, so what is the possible coin combination?

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

Value	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Number of coins	0	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

or -1

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

0+1 = 1

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

want smallest number of coin then initialise with inf

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

0+1 = 1

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

-5+5 = 1 coin

Value	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Number of coins	0	1	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

-5+5 = 1, and so on….

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

Repeat the process…

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

1+1=2

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

optimal

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

1+1=2, 2 coins as well

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

just use result here

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	inf	inf	inf	inf	inf	inf	inf	inf

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

4+1=5, for 5 coins

Value	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Number of coins	0	1	2	3	4	inf	inf	inf	inf	inf	inf	inf	inf

4 coins

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

4+1=5, for 5 coins

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	5	inf	inf	inf	inf	inf	inf	inf

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

0+5=5, for 1 coin

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	5	inf	inf	inf	inf	inf	inf	inf

MONASH University

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

0+5=5, for 1 coin

0 + 5

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	5 vs 1	inf	inf	inf	inf	inf	inf	inf

optimal 0 coin

= 1 coin

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

0+5=5, for 1 coin

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	inf	inf	inf	inf	inf	inf	inf

Coin Change problem
The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

5+1=6, for 2 coins

Value	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Number of coins	0	1	2	3	4	1	inf	inf	inf	inf	inf	inf	inf

Coin Change problem
The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

5+1=6, for 2 coins

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	2	inf	inf	inf	inf	inf	inf

2 < inf

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

1+5=6, for 2 coins

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	2	inf	inf	inf	inf	inf	inf

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

0+6=6, for 1 coin

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	inf	inf	inf	inf	inf	inf

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

  - So keep on running it and eventually we would be done

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	inf	inf	inf	inf	inf	inf

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

  - So keep on running it and eventually we would be done

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	inf	inf	inf	inf	inf

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?
  - So keep on running it and eventually we would be done

Value	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Number of coins	0	1	2	3	4	1	1	2	3	inf	inf	inf	inf

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

  - So keep on running it and eventually we would be done

Value	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Number of coins	0	1	2	3	4	1	1	2	3	4	inf	inf	inf

Coin Change problem
The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?
  - So keep on running it and eventually we would be done

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	3	1	?	inf	inf

Coin Change problem

The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

  - So keep on running it and eventually we would be done

optimal case

base case

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	3	1	2	inf	inf

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?

  - So keep on running it and eventually we would be done

7 - 1 =0
7 - 5 = 2
7 - 6 = 1

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	3	1	2	2	inf

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?
  - So keep on running it and eventually we would be done

Value	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Number of coins	0	1	2	3	4	1	1	2	3	1	2	2	2

Questions?

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9} O(M)
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?
  - So keep on running it and eventually we would be done
    - Complexity?

O(N)   each location on list loop through {1,5,6,9} = O(NM)

Value	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Number of coins	0	1	2	3	4	1	1	2	3	4	2	2	2

Coin Change problem
The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?
  - So keep on running it and eventually we would be done
    - Complexity? O(NM)

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	3	4	2	2	2

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?
  - So keep on running it and eventually we would be done
    - Complexity? O(NM) still much faster than brute force…

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	3	4	2	2	2

Coin Change problem
The less number of coins…

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?
  - So keep on running it and eventually we would be done
    - Complexity? O(NM) still much faster than brute force…
      - Note: If the list is sorted, we can terminate earlier on the smaller values

3 < 5, skip 6, 9

Value	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Number of coins	0	1	2	3	4	1	1	2	3	4	2	2	2

Questions?

- Consider the following scenario
  - Dogecoin currency is {1,5,6,9}
    - We will loop through this over and over considering the coins…
  - I want 12 doge coin value, so what is the possible coin combination?
  - So keep on running it and eventually we would be done
    - Complexity? O(NM) still much faster than brute force…
    - Can you code it?

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	3	4	2	2	2

Coin Change problem

The less number of coins…

- I'll give the algorithm here

```
memo = [inf] * (N+1)
memo[0] = 0
for value in range(1,N+1):
    for j in range(M):                        # this is to go through coins
        if coin[j] <= value  if 1 <= 3
            balance = value - coin[j]  2 = 3 -1
            count = 1 + memo[balance]  memo[2] = 2, reuse
            if count < memo[value]:          # if we have new optimal
                memo[value] = count
return memo[N]
```

Coin Change problem

The less number of coins…

- I'll give the algorithm here
  - Is this top-down or bottom-up?

```
memo = [inf] * (N+1)
memo[0] = 0
for value in range(1,N+1):
    for j in range(M):                    # this is to go through coins
        if coin[j] <= value
            balance = value - coin[j]
            count = 1 + memo[balance]
            if count < memo[value]:       # if we have new optimal
                memo[value] = count
return memo[N]
```

Coin Change problem

The less number of coins…

- I'll give the algorithm here
  - Is this top-down or bottom-up?

```
memo = [inf] * (N+1)
memo[0] = 0
for value in range(1,N+1):
    for j in range(M):                    # this is to go through coins
        if coin[j] <= value
            balance = value - coin[j]
            count = 1 + memo[balance]
            if count < memo[value]:       # if we have new optimal
                memo[value] = count
return memo[N]
```

Coin Change problem
The less number of coins…

- I'll give the algorithm here
  - Is this top-down or bottom-up?
  - The following is the top-down from Aamir (usually recursive)

```
memo = [inf] * (N+1)
memo[0] = 0
for value in range(1,N+1):
    for j in range(M):                  # this is to go through coins
        if coin[j] <= value
            balance = value - coin[j]
            count = 1 + memo[balance]
            if count < memo[value]:     # if we have new optimal
                memo[value] = count
return memo[N]
```

Top-down Solution

```
Initialize Memo[ ] to contain -1 for all indices # -1 indicates the solution
for this index has not been computed yet
Memo[0] = 0
Function CoinChange(value)
  if Memo[value] != -1: \\ DISCUSS infinity is incorrect
    return Memo[value]
  else:
    minCoins = Infinity
    for i=1 to N
      if Coins[ i ] <= value
        c = 1 + CoinChange(value - Coins[ i ])
        if c < minCoins
          minCoins = c
    Memo[value] = minCoins
    return Memo[value]
```

watch clayton's

recursive

Bottom up solution:

1 + Memo[value – Coins[i]]

Coin Change problem
The less number of coins…

- Top-down vs Bottom up
  - Top-down might save some computations
  - Bottom-up might save space especially since no recursion

- Top-down vs Bottom up
  - Top-down might save some computations
  - Bottom-up might save space especially since no recursion
  - I only use bottom-up

Coin Change problem

The less number of coins…

- Top-down vs Bottom up
  - Top-down might save some computations
  - Bottom-up might save space
    especially since no recursion

  - I only use bottom-up
    - But some problems could be easier with top-down as it is more intuitive
    - Technically both are interchangeable…

Questions?

Kapsack Problem

Min-max like a boss

- A problem that can be applicable to a lot of real life scenario

Kapsack Problem

Min-max like a boss

- A problem that can be applicable to a lot of real life scenario
  - Given a limitation (cost)

- A problem that can be applicable to a lot of real life scenario
  - Given a limitation (cost)
  - Optimize something (profit)

Kapsack Problem

Min-max like a boss

- A problem that can be applicable to a lot of real life scenario
  - Given a limitation (cost)
  - Optimize something (profit)
  - 18th most popular algorithmic problem

- A problem that can be applicable to a lot of real life scenario
  - Given a limitation (cost)
  - Optimize something (profit)
  - 18th most popular algorithmic problem

- Given a capacity C and a set of items with weights and value… can you find a combination of item such as the total weight < C but the total value is maximized?

Kapsack Problem

Min-max like a boss

- A problem that can be applicable to a lot of real life scenario
  - Given a limitation (cost)
  - Optimize something (profit)
  - 18th most popular algorithmic problem
- Given a capacity C and a set of items with weights and value… can you find a combination of item such as the total weight < C but the total value is maximized?
  - Unbounded = items are unlimited
  - Bounded = each item can only be taken once

Kapsack Problem

Min-max like a boss

- Let say you have these items

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

Kapsack Problem

Min-max like a boss

- Let say you have these items

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

- And you only have 12 kg… what would you loot?

Kapsack Problem

Min-max like a boss

- Let say you have these items

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

- And you only have 12 kg… what would you loot?
  - 2xB + 2xD = $780

Kapsack Problem

Min-max like a boss

- Let say you have these items

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

- And you only have 12 kg… what would you loot?
  - 2xB + 2xD = $780
  - So how do we code it?
    This is very similar to the coin change!

Questions?

- Let us try the bottom-up approach

Kapsack Problem
Unbounded

- Let us try the bottom-up approach
  - Start with 0 weight till 12 weight

Kapsack Problem
Unbounded

- Let us try the bottom-up approach
  - Start with 0 weight till 12 weight
  - Find the optimal for each weight…

Kapsack Problem
Unbounded

- Let us try the bottom-up approach
  - Start with 0 weight till 12 weight
  - Find the optimal for each weight…
  - Use the earlier optimal for the new weight

Kapsack Problem
Unbounded

- Let us try the bottom-up approach
  - Start with 0 weight till 12 weight
  - Find the optimal for each weight…
    Maximum profit!
  - Use the earlier optimal for the new weight

from lowest to maximum

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0												

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	0	0	0	0	0	0	0	0	0	0	0	0

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	0	0	0	0	0	0	0	0	0	0	0	0

Kapsack Problem
Unbounded

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

9kg, too heavy

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	0	0	0	0	0	0	0	0	0	0	0	0

Kapsack Problem

Unbounded

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

5kg, too heavy

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	0	0	0	0	0	0	0	0	0	0	0	0

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

6kg, too heavy

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	0	0	0	0	0	0	0	0	0	0	0	0

Kapsack Problem
Unbounded

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

1kg, can fit

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	0	0	0	0	0	0	0	0	0	0	0	0

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

1kg, can fit… so we find the optimal of 0kg + 1kg

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	0	0	0	0	0	0	0	0	0	0	0	0

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

1kg, can fit… so we find the optimal of 0kg + 1kg

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	0	0	0	0	0	0	0	0	0	0	0

Kapsack Problem

Unbounded

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	0	0	0	0	0	0	0	0	0	0	0

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

9kg too heavy

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	0	0	0	0	0	0	0	0	0	0	0

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	\$550	\$350	\$180	\$40

5kg too heavy

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	0	0	0	0	0	0	0	0	0	0	0

Kapsack Problem
Unbounded

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

6kg too heavy

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	0	0	0	0	0	0	0	0	0	0	0

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

1kg is OK

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	0	0	0	0	0	0	0	0	0	0	0

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

1kg is OK… optimal of 1kg + profit from this item

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	0	0	0	0	0	0	0	0	0	0	0

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

1kg is OK… optimal of 1kg + profit from this item

Weight	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Profit	0	40	0	0	0	0	0	0	0	0	0	0	0

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

1kg is OK… optimal of 1kg + profit from this item

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	0	0	0	0	0	0	0	0	0	0

Kapsack Problem
Unbounded

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

We just repeat the process

Weight	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Profit	0	40	80	120	0	0	0	0	0	0	0	0	0

MONASH University

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

We just repeat the process

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	0	0	0	0	0	0	0	0

Kapsack Problem
Unbounded

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

Now let us try here

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	0	0	0	0	0	0	0	0

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

9kg too heavy…

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	0	0	0	0	0	0	0	0

Kapsack Problem
Unbounded

MONASH University

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

5kg can fit. Optimal 0kg + current item

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	0	0	0	0	0	0	0	0

Kapsack Problem
Unbounded

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

5kg can fit. Optimal 0kg + current item

Weight	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Profit	0	40	80	120	160	350	0	0	0	0	0	0	0

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

6kg too heavy

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	350	0	0	0	0	0	0	0

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

1kg is OK. Optimal 4kg + current item

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	350	0	0	0	0	0	0	0

Kapsack Problem
Unbounded

MONASH University

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

1kg is OK. Optimal 4kg + current item but it is only 200

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	350	0	0	0	0	0	0	0

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	\$550	\$350	\$180	\$40

Moving on….

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	350	0	0	0	0	0	0	0

Kapsack Problem

Unbounded

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

heavy

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	350	0	0	0	0	0	0	0

Kapsack Problem
Unbounded

MONASH University

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

Optimal 1kg + current item

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	350	0	0	0	0	0	0	0

Kapsack Problem
Unbounded

MONASH University

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

Optimal 1kg + current item

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	350	390	0	0	0	0	0	0

Kapsack Problem
Unbounded

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

Optimal 0kg + current item = 180 only

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	350	390	0	0	0	0	0	0

Kapsack Problem
Unbounded

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

Optimal 5kg + current item = same 390

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	350	390	0	0	0	0	0	0

Kapsack Problem

Unbounded

- Let us run through it

Item	A	B	C	D
Weight	9kg	5kg	6kg	1kg
Value	$550	$350	$180	$40

Eventually....

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	350	390	430	470	550	700	740	780

Questions?

Have a break!

- So what is our algorithm?
  – Let us try to produce it now as part of the class activity!

- So what is our algorithm?
  - Very similar to the coin change except
    - Finding maximum instead of minimum
    - Initialized to 0

Kapsack Problem
Unbounded

- So what is our algorithm?
  - Very similar to the coin change except
    - Finding maximum instead of minimum
    - Initialized to 0
  - You have the 1 array, called items
    - N number of items
    - Items[i].weight for the weight
    - Items[i].profit for the profit

Questions?

Kapsack Problem
Unbounded

- So what is our algorithm?
  - Very similar to the coin change except
    - Finding maximum instead of minimum
    - Initialized to 0

```
memo = [0] * (N+1)                          # N is the total weight
memo[0] = 0
for bag_weight in range(1,N+1):
    for j in range(M):                      # this is to go through items
        if items[j].weight <= bag_weight:
            balance = bag_weight - items[j].weight
            profit = item[j].profit + memo[balance]
            if profit > memo[bag_weight]:   # if we have new optimal
                memo[bag_weight] = profit
return memo[N]
```

Kapsack Problem
Unbounded

- So what is our algorithm?
  - Very similar to the coin change except
    - Finding maximum instead of minimum
    - Initialized to 0
  - Complexity?

```
memo = [0] * (N+1)                          # N is the total weight
memo[0] = 0
for bag_weight in range(1,N+1):
    for j in range(M):                      # this is to go through items
        if items[j].weight <= bag_weight:
            balance = bag_weight - items[j].weight
            profit = item[j].profit + memo[balance]
            if profit > memo[bag_weight]:   # if we have new optimal
                memo[bag_weight] = profit
return memo[N]
```

Kapsack Problem
Unbounded

- So what is our algorithm?
  - Very similar to the coin change except
    - Finding maximum instead of minimum
    - Initialized to 0
  - Complexity? O(NM)

see top-down from clayton side

```
memo = [0] * (N+1)                          # N is the total weight
memo[0] = 0
for bag_weight in range(1,N+1):
    for j in range(M):                      # this is to go through items
        if items[j].weight <= bag_weight:
            balance = bag_weight - items[j].weight
            profit = item[j].profit + memo[balance]
            if profit > memo[bag_weight]:   # if we have new optimal
                memo[bag_weight] = profit
return memo[N]
```

Kapsack Problem

Unbounded

- So what is our algorithm?
  - Very similar to the coin change except
    - Finding maximum instead of minimum
    - Initialized to 0
  - Complexity?
  - Top-down? See Nathan's slides

Questions?

Kapsack Problem
Unbounded

- But is the code correct?

```
memo = [0] * (N+1)                              # N is the total weight
memo[0] = 0
for bag_weight in range(1,N+1):
    for j in range(M):                          # this is to go through items
        if items[j].weight <= bag_weight:
            balance = bag_weight - items[j].weight
            profit = item[j].profit + memo[balance]
            if profit > memo[bag_weight]:       # if we have new optimal
                memo[bag_weight] = profit
return memo[N]
```

- But is the code correct?
  - Currently, we assume the maximum value is when you find items with a total weight of 12 kg

Eventually….

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	350	390	430	470	550	700	740	780

```
memo = [0] * (N+1)                          # N is the total weight
memo[0] = 0
for bag_weight in range(1,N+1):
    for j in range(M):                      # this is to go through items
        if items[j].weight <= bag_weight:
            balance = bag_weight - items[j].weight
            profit = item[j].profit + memo[balance]
            if profit > memo[bag_weight]:   # if we have new optimal
                memo[bag_weight] = profit
return memo[N]
```

Kapsack Problem
Unbounded

- But is the code correct?
  - Currently, we assume the maximum value is when you find items with a total weight of 12 kg
  - What if we can't reach 12 kg? even bag is not up to 12kg, can still use 12kg list for that bag
  - What if the optimal is at 10 kg instead of 12 kg?

Eventually….

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	40	80	120	160	350	390	430	470	550	700	740	780

```
memo = [0] * (N+1)                          # N is the total weight
memo[0] = 0
for bag_weight in range(1,N+1):
    for j in range(M):                      # this is to go through items
        if items[j].weight <= bag_weight:
            balance = bag_weight - items[j].weight
            profit = item[j].profit + memo[balance]
            if profit > memo[bag_weight]:   # if we have new optimal
                memo[bag_weight] = profit
return memo[N]
```

Kapsack Problem

Unbounded

- But is the code correct?
  - Currently, we assume the maximum value is when you find items with a total weight of 12 kg
  - What if we can't reach 12 kg?
  - What if the optimal is at 10 kg instead of 12 kg?
  - So what must we change?

```
memo = [0] * (N+1)                           # N is the total weight
memo[0] = 0
for bag_weight in range(1,N+1):
    for j in range(M):                        # this is to go through items
        if items[j].weight <= bag_weight:
            balance = bag_weight - items[j].weight
            profit = item[j].profit + memo[balance]
            if profit > memo[bag_weight]:     # if we have new optimal
                memo[bag_weight] = profit
return memo[N]
```

Kapsack Problem
Unbounded

- But is the code correct?
  - Currently, we assume the maximum value is when you find items with a total weight of 12 kg
  - What if we can't reach 12 kg?
  - What if the optimal is at 10 kg instead of 12 kg?
  - So what must we change?
    - memo[i] = memo[i-1]   # copy the previous optimal

```
memo = [0] * (N+1)                          # N is the total weight
memo[0] = 0
for bag_weight in range(1,N+1):
    for j in range(M):                      # this is to go through items
        if items[j].weight <= bag_weight:
            balance = bag_weight - items[j].weight
            profit = item[j].profit + memo[balance]
            if profit > memo[bag_weight]:   # if we have new optimal
                memo[bag_weight] = profit
return memo[N]
```

Kapsack Problem
Unbounded

- But is the code correct?
  - Currently, we assume the maximum value is when you find items with a total weight of 12 kg
  - What if we can't reach 12 kg?
  - What if the optimal is at 10 kg instead of 12 kg?
  - So what must we change?
    - memo[i] = memo[i-1]   # copy the previous optimal
    - or, linear search through memo for the maximum

```
memo = [0] * (N+1)                              # N is the total weight
memo[0] = 0
for bag_weight in range(1,N+1):
    for j in range(M):                          # this is to go through items
        if items[j].weight <= bag_weight:
            balance = bag_weight - items[j].weight
            profit = item[j].profit + memo[balance]
            if profit > memo[bag_weight]:   # if we have new optimal
                memo[bag_weight] = profit
return memo[N]
```

Questions?

Kapsack Problem

0/1 items

- Same problem, but you can’t repeat the item

Kapsack Problem
0/1 items

- Same problem, but you can’t repeat the item
  - So how would we solve it?

Kapsack Problem

0/1 items

- Same problem, but you can't repeat the item
  - So how would we solve it?
- This is where we can see the growing of problems…

Kapsack Problem

0/1 items

- Same problem, but you can't repeat the item
  - So how would we solve it?

- This is where we can see the growing of problems…
  - Grow from 0 weight to N weight

different weight campacity

Kapsack Problem
0/1 items

MONASH University

- Same problem, but you can't repeat the item
  - So how would we solve it?
- This is where we can see the growing of problems…
  - Grow from 0 weight to N weight
  - Grow from a set of 0 items till M items

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

item 1, either pick or not pick

Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Profit	0	?	?	?	?	?	?	?	?	?	?	?	?

not decision make

Kapsack Problem

0/1 items

- We use a matrix 2 constraint weight, item

Increasing weight

	0	1	2	3	4	5	6	7	8	9	10	11	12
0													
A													
B													
C													
D													

- We use a matrix

Increasing weight

Increasing items in set

	0	1	2	3	4	5	6	7	8	9	10	11	12
0													
A													
B													
C													
D													

Kapsack Problem

0/1 items

- We use a matrix

Increasing weight weight capacity

Increasing items in set

	0	1	2	3	4	5	6	7	8	9	10	11	12
{}													
{A}													
{A,B}													
{A,B, C}													
{A,B, C,D}													

Kapsack Problem

0/1 items

- We use a matrix
  - So we fill up the matrix

	0	1	2	3	4	5	6	7	8	9	10	11	12
0													
A													
B													
C													
D													

Kapsack Problem

0/1 items

- We use a matrix
  - Base cases

	0	1	2	3	4	5	6	7	8	9	10	11	12
0													
A													
B													
C													
D													

Kapsack Problem
0/1 items

- We use a matrix
  - Base cases
    - No item to choose from…

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A													
B													
C													
D													

Kapsack Problem

0/1 items

- We use a matrix
  - Base cases
    - No item to choose from…
    - Max weight is 0….

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0												
B	0												
C	0												
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0												
B	0												
C	0												
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0												
B	0												
C	0												
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0							
B	0												
C	0												
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	?						
B	0												
C	0												
D	0												

Kapsack Problem

0/1 items

MONASH University

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230						
B	0												
C	0												
D	0												

Kapsack Problem
0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	?					
B	0												
C	0												
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230					
B	0												
C	0												
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230				
B	0												
C	0												
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230			
B	0												
C	0												
D	0												

Kapsack Problem
0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0												
C	0												
D	0												

reuse from this row

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40											
C	0												
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

all item just 1

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

		0	1	2	3	4	5	6	7	8	9	10	11	12
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
just itemA	A	0	0	0	0	0	0	230	230	230	230	230	230	230
	B	0	40	40										
	C	0												
	D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40							
C	0												
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	40						
C	0												
D	0												

Kapsack Problem
0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	40	?					
C	0												
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	40	270					
C	0												
D	0												

Kapsack Problem
0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	40	270	270				
C	0												
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?
    - Is this correct?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	40	270	270	270	270	270	270
C	0												
D	0												

MONASH University

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?
    - Is this correct? Here, we can choose not to include B, having only A in bag

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	$\boxed{40}$	270	270	270	270	270	270
C	0												
D	0												

Kapsack Problem
0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?
    - Is this correct? Here, we can choose not to include B, having only A in bag

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0												
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

exclude item B

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	?											
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	?											
D	0												

Kapsack Problem
0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40											
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40								
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350							
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390						
D	0												

Kapsack Problem
0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390					
D	0												

Kapsack Problem
0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	
D	0												

Kapsack Problem
0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0												

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	?											

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390				

Kapsack Problem
0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550			

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590		

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	?

Kapsack Problem

0/1 items

- We use a matrix
  - So row by row…
    - Start with item A
    - Should we add it?

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

j

i

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

Questions?

Kapsack Problem

0/1 items

- We use a matrix
  – What is the algorithm/ code?

j

i

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

Kapsack Problem

0/1 items

MONASH University

- We use a matrix
  - What is the algorithm/ code?

```
# for every row (item)
for i=1 to M:    if i = 1
    # for every column (weight)
    for j=1 to N:    j = 6
        # get the excluded at current weight (row above)
        exclude = memo[i-1][j]    current weight from the last row in the same column
        # calculate the include                         j is total weight
        include = 0
        if item[i].weight <= j:    available item's value    memo[1,0] = 0
            include = item[i].value + memo[i-1][j-item[i].weight]
                     third row for third item
        memo[i][j] = max(exclude,include)          6 - 6 (item A. weight)= 0
```

Kapsack Problem

0/1 items

- We use a matrix
  - What is the algorithm/ code?
  - Complexity?

```
# for every row (item)
for i=1 to M:
    # for every column (weight)
    for j=1 to N:
        # get the excluded at current weight (row above)
        exclude = memo[i-1][j]
        # calculate the include
        include = 0
        if item[i].weight <= j:
            include = item[i].value + memo[i-1][j-item[i].weight]
        memo[i][j] = max(exclude,include)
```

Kapsack Problem
0/1 items

- We use a matrix
  - What is the algorithm/ code?
  - Complexity?
    - O(NM) time from filling matrix
    - O(NM) space for the matrix

```
# for every row (item)
for i=1 to M:
    # for every column (weight)
    for j=1 to N:
        # get the excluded at current weight (row above)
        exclude = memo[i-1][j]
        # calculate the include
        include = 0
        if item[i].weight <= j:
            include = item[i].value + memo[i-1][j-item[i].weight]
        memo[i][j] = max(exclude,include)
```

choose the bigger one

Kapsack Problem

0/1 items

- We use a matrix
  - What is the algorithm/ code?
  - Complexity?
    - O(NM) time from filling matrix
    - O(NM) space for the matrix… we can reduce this however

```
# for every row (item)
for i=1 to M:
    # for every column (weight)
    for j=1 to N:
        # get the excluded at current weight (row above)
        exclude = memo[i-1][j]
        # calculate the include
        include = 0
        if item[i].weight <= j:
            include = item[i].value + memo[i-1][j-item[i].weight]
        memo[i][j] = max(exclude,include)
```

Kapsack Problem
0/1 items

- We use a matrix
  - What is the algorithm/ code?
  - Complexity?
    - O(NM) time from filling matrix
    - O(NM) space for the matrix… we can reduce this however
      We realize we do not need the entire matrix! We get the current value by looking at the row above only. So we can just store the latest 2 row…

Kapsack Problem

0/1 items

- We use a matrix
  - What is the algorithm/ code?
  - Complexity?
    - O(NM) time from filling matrix
    - O(NM) space for the matrix… we can reduce this however
      We realize we do not need the entire matrix! We get the current value by looking at the row above only. So we can just store the latest 2 row…
      Reducing complexity to O(2N+M) just a list for items

Kapsack Problem

0/1 items

- We use a matrix
  - What is the algorithm/ code?
  - Complexity?
    - O(NM) time from filling matrix
    - O(NM) space for the matrix… we can reduce this however
      We realize we do not need the entire matrix! We get the current value by looking at the row above only. So we can just store the latest 2 row… Reducing complexity to O(2N+M)
    - But in reality, we can't do this space saving… because we need it to reconstruct the solution…

Questions?

Take a break!

Kapsack Problem

0/1 items

- So what are the items?

Kapsack Problem

0/1 items

- So what are the items?

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

Kapsack Problem

0/1 items

- So what are the items?
  - Recall that we compare the current value (inclusive) with the value in the row above (exclusive).

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

- So what are the items?
  - Recall that we compare the current value (inclusive) with the value in the row above (exclusive).

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

Kapsack Problem

0/1 items

- So what are the items?
  - Recall that we compare the current value (inclusive) with the value in the row above (exclusive). If same value, means we do not include…

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	$\boxed{620}$
D	0	40	40	40	40	350	390	390	390	550	590	590	$\boxed{620}$

Kapsack Problem
0/1 items

- So what are the items?
  – Recall that we compare the current value (inclusive) with the value in the row above (exclusive). If same value, means we do not include…

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

  - {}

Kapsack Problem

0/1 items

- So what are the items?
  - Recall that we compare the current value (inclusive) with the value in the row above (exclusive). If different value, means we include…

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

  - {C}

Kapsack Problem

0/1 items

- So what are the items?
  - Recall that we compare the current value (inclusive) with the value in the row above (exclusive). If different value, means we include…
  - Then we update to the suitable weight of the included item

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

- {C} just leave item C in the bag if the value is not increased

Kapsack Problem

0/1 items

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

- So what are the items?
  - Recall that we compare the current value (inclusive) with the value in the row above (exclusive). If different value, means we include…
  - Then we update to the suitable weight of the included item

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

- {C}

Kapsack Problem
0/1 items

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

- So what are the items?
  - Recall that we compare the current value (inclusive) with the value in the row above (exclusive). If different value, means we include…
  - Then we update to the suitable weight of the included item

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

  - {C}

Kapsack Problem

0/1 items

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

- So what are the items?
  - Recall that we compare the current value (inclusive) with the value in the row above (exclusive). So now do the same…

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

- {C}

Kapsack Problem

0/1 items

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

- So what are the items?
  - Recall that we compare the current value (inclusive) with the value in the row above (exclusive). So now do the same…

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

- {C,B}

12 - 5 = 7 (column 7)

since unique item, so no longer can choose C and D, so A and B
if row A and row B have the same values, then just includes itemA

Kapsack Problem
0/1 items

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

- So what are the items?
  - Recall that we compare the current value (inclusive) with the value in the row above (exclusive). So now do the same…

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

    - {C,B}

Kapsack Problem
0/1 items

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

- So what are the items?
  - Recall that we compare the current value (inclusive) with the value in the row above (exclusive). So now do the same…

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

  - {C,B}

Kapsack Problem

0/1 items

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

- So what are the items?
  - Recall that we compare the current value (inclusive) with the value in the row above (exclusive). So now do the same…

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

- {C,B,A}

Kapsack Problem

0/1 items

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

- So what are the items?
  - Recall that we compare the current value (inclusive) with the value in the row above (exclusive). So now do the same…

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

  - {C,B,A} makes up the item for total value of 620

Kapsack Problem

0/1 items

Item	A	B	C	D
Weight	6kg	1kg	5kg	9kg
Value	$230	$40	$350	$550

- So what are the items?
  - Recall that we compare the current value (inclusive) with the value in the row above (exclusive). So now do the same…

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	230	230	230	230	230	230	230
B	0	40	40	40	40	40	230	270	270	270	270	270	270
C	0	40	40	40	40	350	390	390	390	390	390	580	620
D	0	40	40	40	40	350	390	390	390	550	590	590	620

  - {C,B,A} makes up the item for total value of 620
  - This is what we call backtracking!

Questions?

- We often need to reconstruct solutions

- We often need to reconstruct solutions
  - Coin change = what are the coins?

- We often need to reconstruct solutions
  - Coin change = what are the coins?
  - Knapsack = what are the items?

- We often need to reconstruct solutions
  - Coin change = what are the coins?
  - Knapsack = what are the items?
  - Edit distance = what are the modifications made (insert/delete/replace)

- We often need to reconstruct solutions
  - Coin change = what are the coins?
  - Knapsack = what are the items?
  - Edit distance = what are the modifications made (insert/delete/replace)

  - When we update the optimal value, we can store the decision we made…

- We often need to reconstruct solutions
  - Coin change = what are the coins?
  - Knapsack = what are the items?
  - Edit distance = what are the modifications made (insert/delete/replace)

  - When we update the optimal value, we can store the decision we made... But this waste a lot of memory as we need to store decisions/ combinations at every step!

Backtracking
Reconstruction solution

- We often need to reconstruct solutions
  - Coin change = what are the coins?
  - Knapsack = what are the items?
  - Edit distance = what are the modifications made (insert/delete/replace)
  - When we update the optimal value, we can store the decision we made… But this waste a lot of memory as we need to store decisions/ combinations at every step!
  - So, we leave bread crumbs to backtrack

- We often need to reconstruct solutions
  - Coin change = what are the coins?
  - Knapsack = what are the items?
  - Edit distance = what are the modifications made (insert/delete/replace)

  - When we update the optimal value, we can store the decision we made… But this waste a lot of memory as we need to store decisions/ combinations at every step!
  - So, we leave bread crumbs to backtrack or only the final decision made!

Questions?

- Let say we store our decisions…

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	3	1	2	2	2

- Let say we store our decisions…

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	3	1	2	2	2
The coins	{}	1	1, 1	1, 1, 1	1, 1, 1, 1	5	6	6, 1	6, 1, 1	9	5, 5	6, 5	6, 6

- Let say we store our decisions…
  – Space complexity?

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	3	1	2	2	2
The coins	{}	1	1, 1	1, 1, 1	1, 1, 1, 1	5	6	6, 1	6, 1, 1	9	5, 5	6, 5	6, 6

- Let say we store our decisions…
  - Space complexity? Can be O(N^2)

if just 1, 10 coin

O(N) * O(N) = O(N^2)

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	3	1	2	2	2
The coins	{}	1	1, 1	1, 1, 1	1, 1, 1, 1	5	6	6, 1	6, 1, 1	9	5, 5	6, 5	6, 6

1,1,1,1,1 = O(N)

- Let say we store our decisions…
  - Space complexity? Can be O(N^2)
  - Improve it further?

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	3	1	2	2	2
The coins	{}	1	1, 1	1, 1, 1	1, 1, 1, 1	5	6	6, 1	6, 1, 1	9	5, 5	6, 5	6, 6

- Let say we store our decisions…
  - Space complexity? Can be O(N^2)
  - Improve it further? Remember the last coin you added

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	3	1	2	2	2
The coins	{}	1	1	1	1	5	6	1	1	9	5	6	6

- Let say we store our decisions…
  - Space complexity? Can be O(N^2)
  - Improve it further? Remember the last coin you added
    So space complexity now?

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	3	1	2	2	2
The coins	{}	1	1	1	1	5	6	1	1	9	5	6	6

6-6 =0

5 -5 =0
{} null set coin

- Let say we store our decisions…
  - Space complexity? Can be O(N^2)
  - Improve it further? Remember the last coin you added
    So space complexity now? O(N)

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	3	1	2	2	2
The coins	{}	1	1	1	1	5	6	1	1	9	5	6	6

- Let say we store our decisions…
  - Space complexity? Can be O(N^2)
  - Improve it further? Remember the last coin you added
    So space complexity now? O(N)
    Coins = {6,6}

backtracking always go back to base case

12 -6 = 6 (go to column 6)

Value	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of coins	0	1	2	3	4	1	1	2	3	1	2	2	2
The coins	{}	1	1	1	1	5	6	1	1	9	5	6	6

last item added

Questions?

Backtracking
vs Decision Array

- Backtracking save space
- Decision array save time

Backtracking
vs Decision Array

- Backtracking save space
  - Less auxiliary space
  - Same space complexity

not saving decision made but save the result
and backtrack to compare to retrieve back the decision

- Decision array save time

Backtracking
vs Decision Array

- Backtracking save space
  - Less auxiliary space
  - Same space complexity
- Decision array save time
  - Faster
  - But time complexity lies in finding the solution still…

Questions?

Edit Distance

Cost to convert string

- Edit-distance
- We will skip this since this is similar to the Knapsack really…

Edit Distance

Cost to convert string

- Edit-distance
- We will skip this since this is similar to the Knapsack really…
  - Will be covered in the tutorial, linking it up with the longest common subsequence (LCS) problem there

- Edit-distance
- We will skip this since this is similar to the Knapsack really…
  - Will be covered in the tutorial, linking it up with the longest common subsequence (LCS) problem there
  - Note: Nathan's slide at the end

watch Ian W5 tutorial

Questions?

- Take problem
- Break it down

Summary

Dynamic Programming Algorithm (DPA)

- Take problem
- Break it down
  - Sub-problem has optimal solution
  - Sub-problem overlap (thus can be reused)

Summary

Dynamic Programming Algorithm (DPA)

- Take problem
- Break it down
  - Sub-problem has optimal solution
  - Sub-problem overlap (thus can be reused)
  - Store these solutions (memoization) for faster computing

Summary
Dynamic Programming Algorithm (DPA)

- Take problem
- Break it down
  - Sub-problem has optimal solution
  - Sub-problem overlap (thus can be reused)
  - Store these solutions (memoization) for faster computing
- Reconstruct solution
  - Decision array
  - Backtracking

Summary

Dynamic Programming Algorithm (DPA)

- Take problem
- Break it down
  - Sub-problem has optimal solution
  - Sub-problem overlap (thus can be reused)
  - Store these solutions (memoization) for faster computing
- Reconstruct solution
  - Decision array
  - Backtracking

Questions?

MONASH University

Thank You