DYNAMIC PROGRAMMING

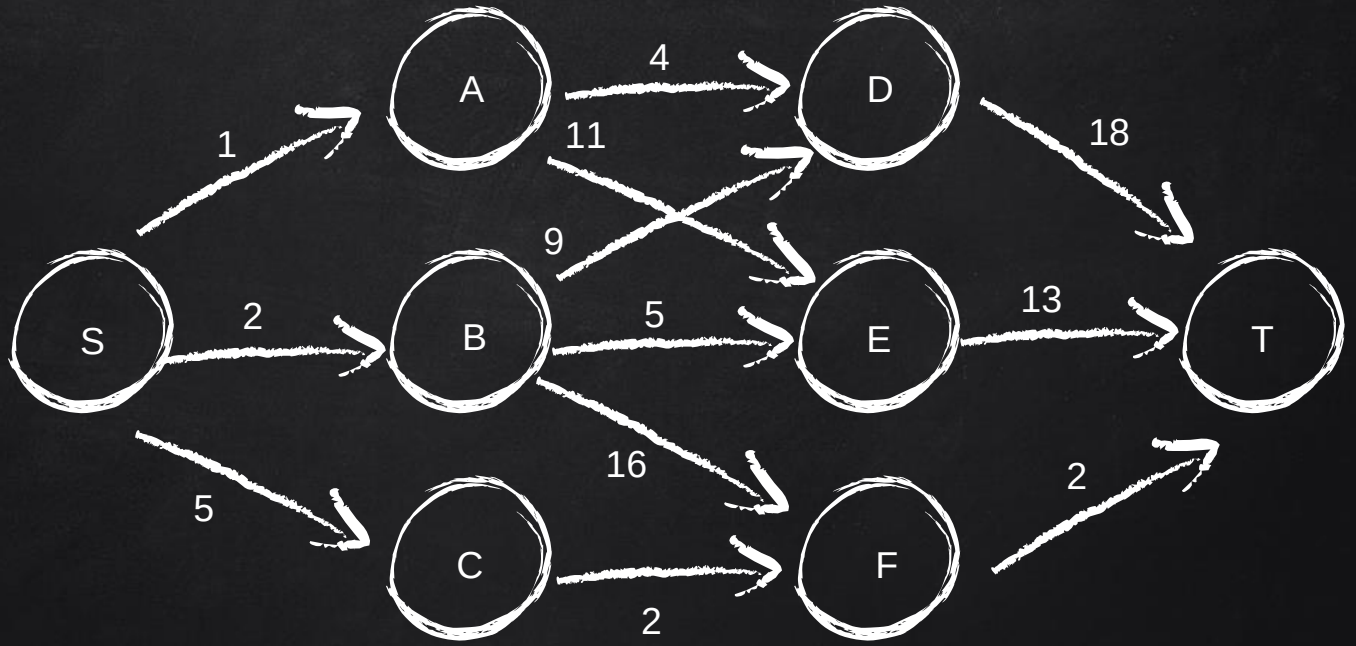
DP

DESIGN AND ANALYSIS OF ALGORITHMS

LECTURER: Nguyễn Thanh Sơn

CS112.L23.KHCL.N12

The shortest path



Apply the Greedy approach, the shortest path from S to T is?

The shortest path

The Greedy approach can not be applied to this case:
(S, A, D, T) 1+4+18=23

The real shortest path is:
(S, C, F, T) 5+2+2=9

So today we will learn about a new algorithm

CONTENT

1. What is Dynamic Programming?
2. Characteristics of Dynamic Programming
3. Dynamic Programming Methods
4. Compare with other algorithms
5. Steps in Dynamic Programming

1. What is Dynamic programMing?

Dynamic programming (DP) approach is similar to Divide and Conquer in breaking down the problem in smaller and yet smaller possible sub-problems. But unlike Divide and Conquer, results of these smaller sub-problems are remembered and used for similar or overlapping sub-problems.

Characteristics of Dynamic Programming

- Overlapping Subproblems
- Optimal Substructure

- **Overlapping Subproblems**

There exist some places where we solve the same subproblem more than once

EX:

```python
def fib(n):
    if n <= 1:
        return n
    return fib(n -1) + fib(n - 2)
```

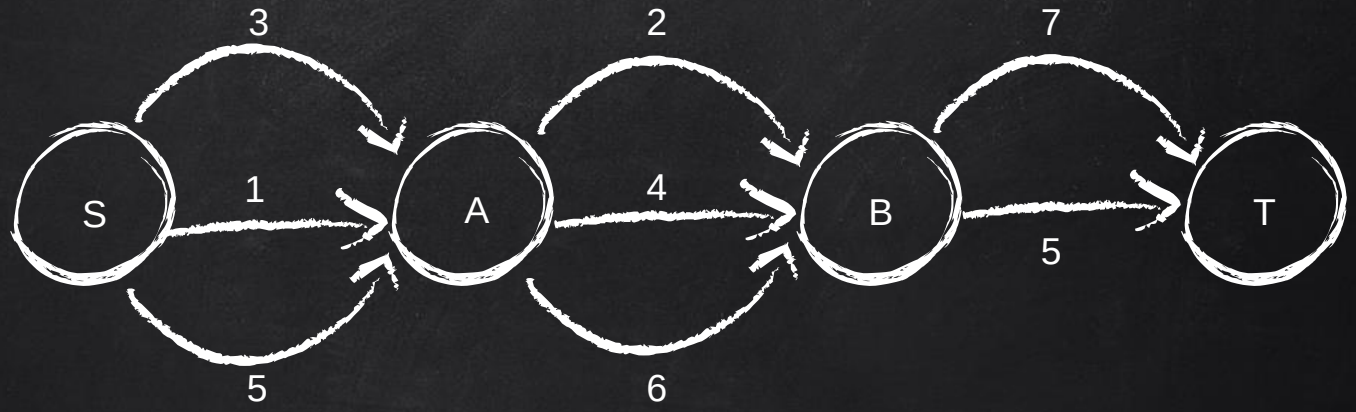
DP:

```python
def fib(n):
    dp = [0] * (n + 1)
    dp[1] = 1
    for i in range(2, n + 1):
        dp[i] = dp[i - 1] + dp[i - 2]
    return dp[n]
```

➢ Optimal Substructure

The optimal solution to the problem contains within optimal solutions to its subproblems.

EX:



$d_{min}(S, T) = d_{min}(S, A) + d_{min}(A, B) + d_{min}(B, T)$

$= 1 + 2 + 5$

$= 8$

Dijkstra

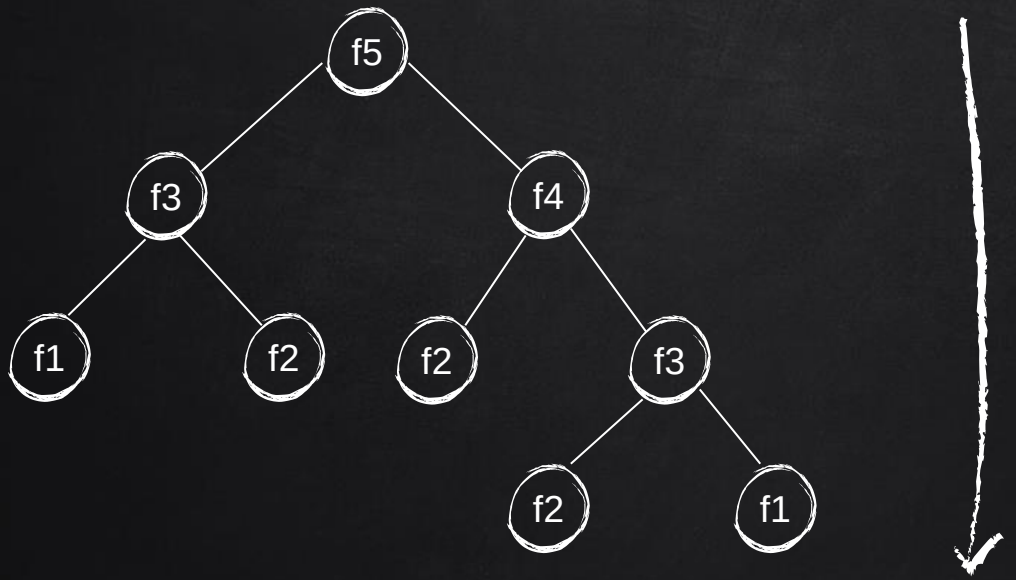
3. Dynamic Programming Methods

DP offers two methods to solve a problem:

- Top-down with Memoization
- Bottom-up with Tabulation

3. Dynamic Programming Methods

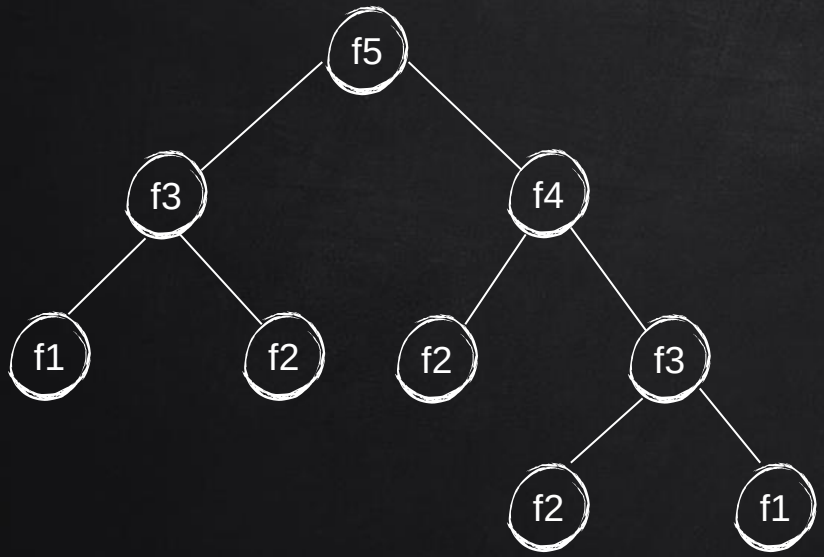
- Top-down with Memoization



In this approach, we try to solve the bigger problem by recursively finding the solution to smaller sub-problems. Whenever we solve a sub-problem, we cache its result so that we don't end up solving it repeatedly if it's called multiple times. Instead, we can just return the saved result. This technique of storing the results of already solved subproblems is called **Memoization.**

3. Dynamic Programming Methods

- Top-down with Memoization



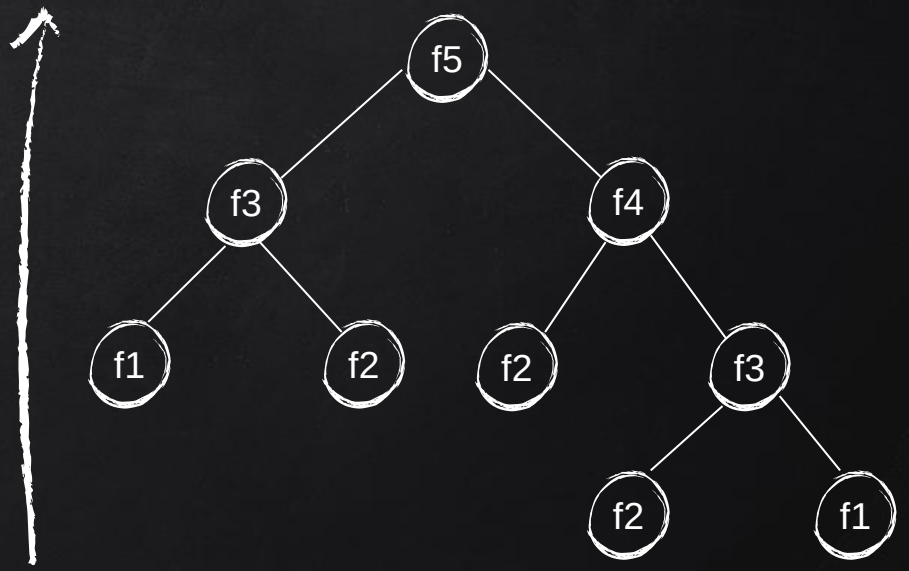
```
def Fib(n):
    if (n < 2):
        result = n
    else:
        result = Fib(n-2) + Fib(n-1)
    F[n] = result
    return F[n]
```

$O(2^n)$

3. Dynamic Programming Methods

- Bottom-up with Tabulation

Tabulation is the opposite of the top-down approach and avoids recursion. In this approach, we solve the problem "bottom-up" (i.e. by solving all the related sub-problems first). This is typically done by filling up an n-dimensional table. Based on the results in the table, the solution to the top/original problem is then computed.

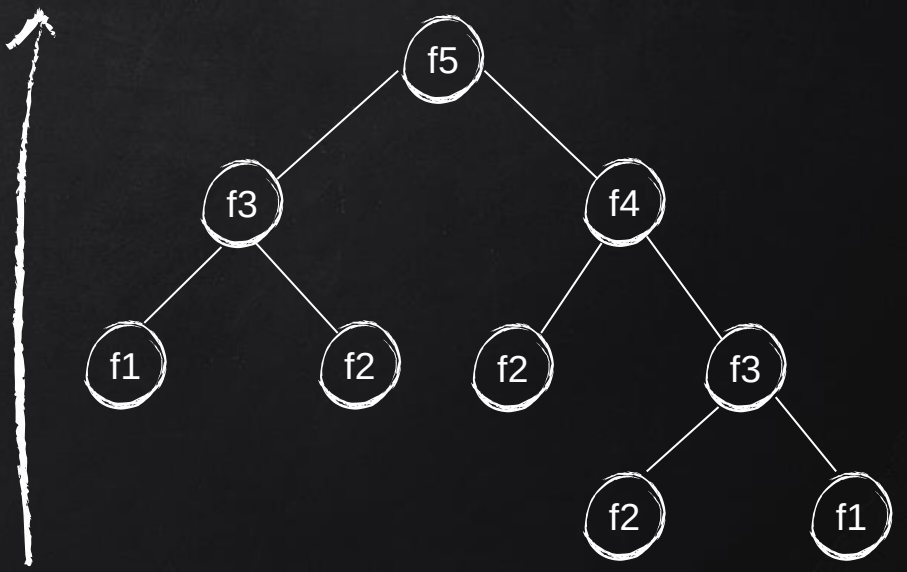


3. Dynamic Programming Methods

- Bottom-up with Tabulation

O(n)

```
def Fib(n):
    F[0] = 0
    F[1] = 1
    for i in 2...n
        F[i] = F[i-2] + F[i-1]
    return F[n]
```



Comparison

Top-down with Memoization	Bottom-up with Tabulation
Easy to set up	Gets complicated if there are multiple conditions
Must be solved from the top down	Must be solved from the bottom up
Slower due to recursion	Faster, due to direct access to the results stored in the table
Just solve the necessary problems	Must solve all subproblems

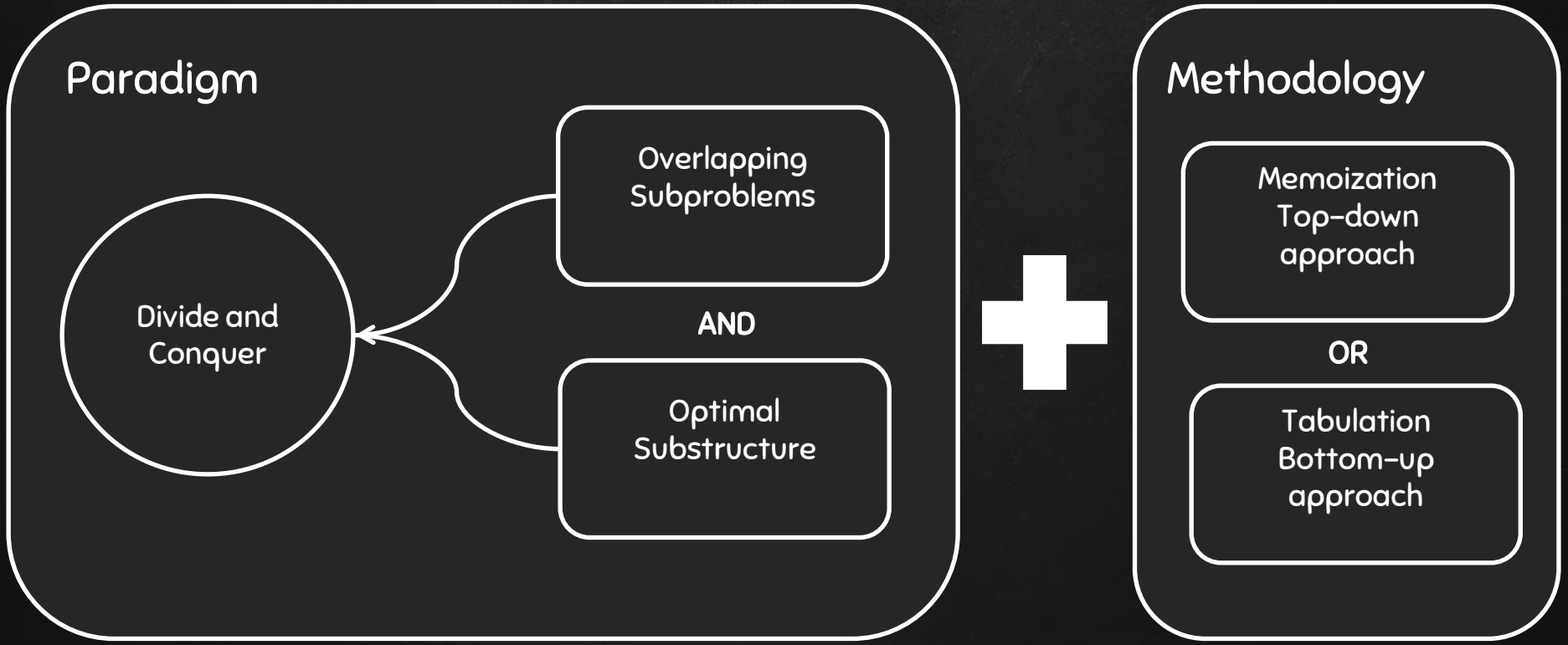
4. Compare with other algorithms

Dynamic programming	Greedy approach
DP are motivated for an overall optimization of the problem	Local optimization is addressed

4. Compare with other algorithms

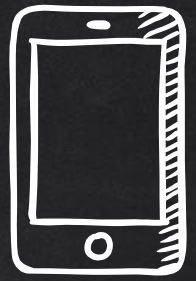
Dynamic Programming	Divide and Conquer
DP use the output of a smaller sub-problem and then try to optimize a bigger sub-problem and use Memoization to remember the output of already solved sub-problems.	Solutions are combined to achieve an overall solution

Dynamic Programming



Steps in Dynamic Programming

1. Characterize structure of an optimal solution.
2. Define value of optimal solution recursively.
3. Compute optimal solution values either top-down with caching or bottom-up in a table.
4. Construct an optimal solution from computed values.

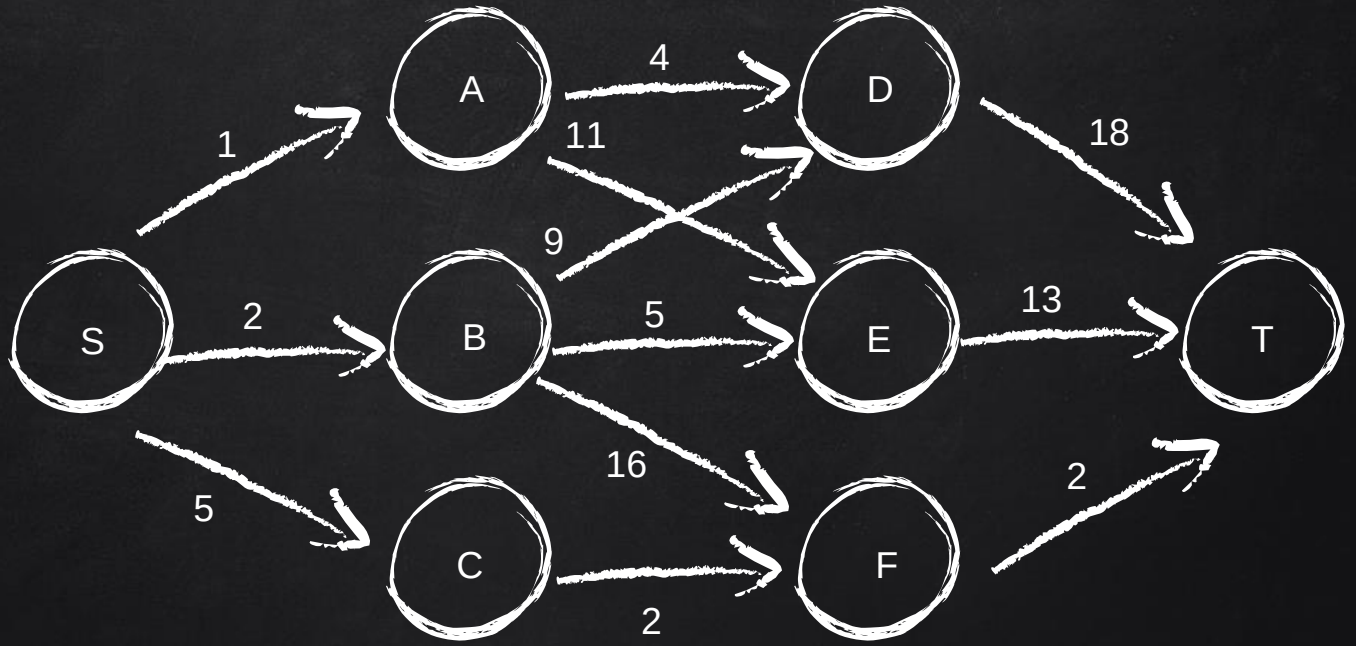


KAHOOT!

HOMEWORK

Gmail: 19521482@gm.uit.edu.vn

The shortest path

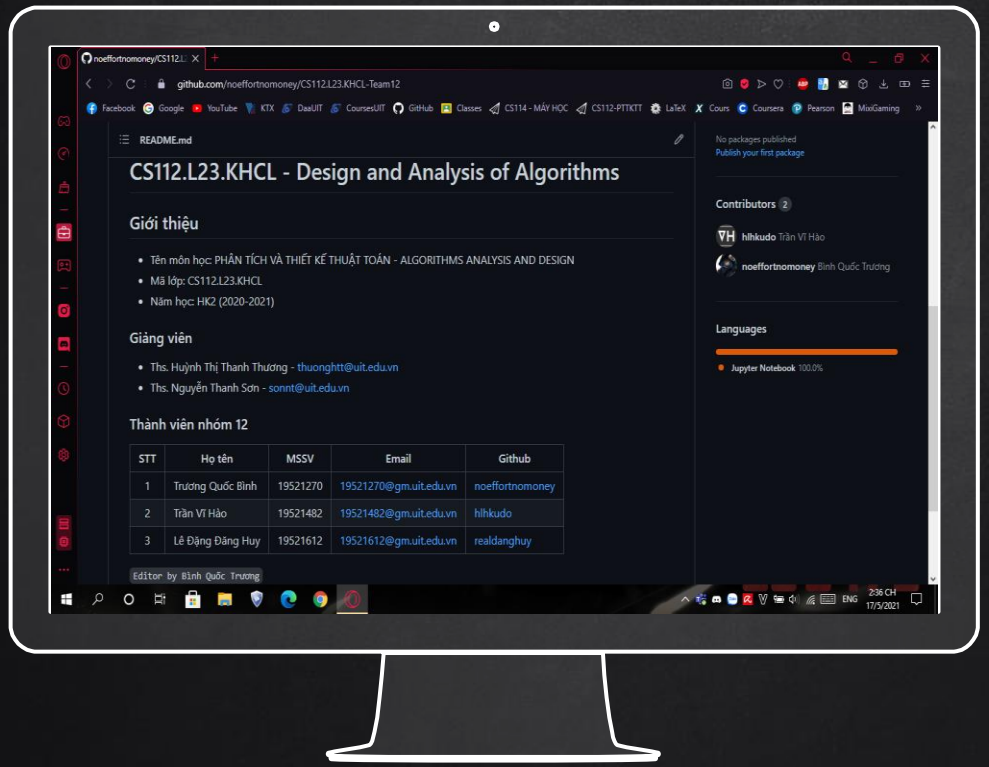


Apply the Dynamic programming, the shortest path from S to T is?

THANKS!

Any questions?

You can review these slides on our team's GitHub



Our team's github link:

https://github.com/noeffortnomoney/CS112.L23.KHCL-Team12

Credits

Special thanks to all the people who made and released these awesome resources for free:

- ✘ Presentation template by SlidesCarnival
- ✘ TOPDev.vn
- ✘ educative.io
- ✘ tutorialspoint.com
- ✘ geeksforgeeks.org
- ✘ gacsach.vn
- ✘ Some PowerPoint files in the Reference section (GitHub)