Lecture 10

Dynamic Programming II: Bellman-Ford (cont), Interval Scheduling, Longest Common Subsequence

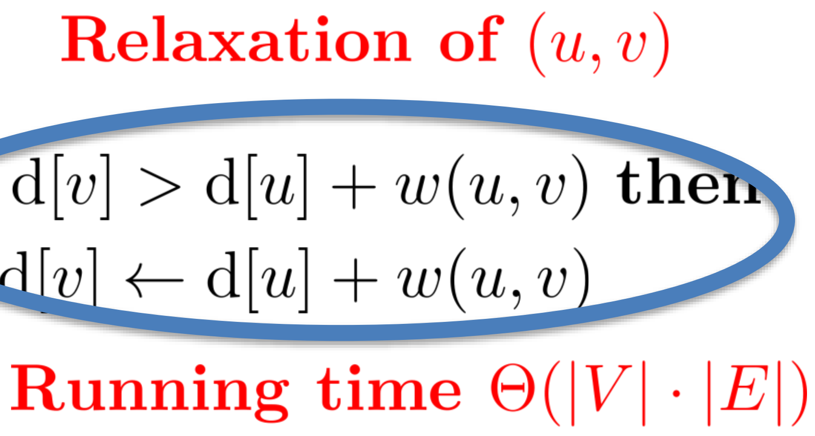
CS 161 Design and Analysis of Algorithms

Ioannis Panageas

Case study III: Bellman-Ford

In words: $d[s] = 0, d[u] = +\infty$ for $u \neq s$.

For $n - 1$ times, relax all the edges (u, v).



Relaxation of (u, v)

```
If $d[v] > d[u] + w(u,v)$ then
    $d[v] \leftarrow d[u] + w(u,v)$
```

Running time $\Theta(|V| \cdot |E|)$

Case study III: Bellman-Ford

In words: $d[s] = 0, d[u] = +\infty$ for $u \neq s$.

For $n - 1$ times, relax all the edges (u, v).

Relaxation of (u, v)

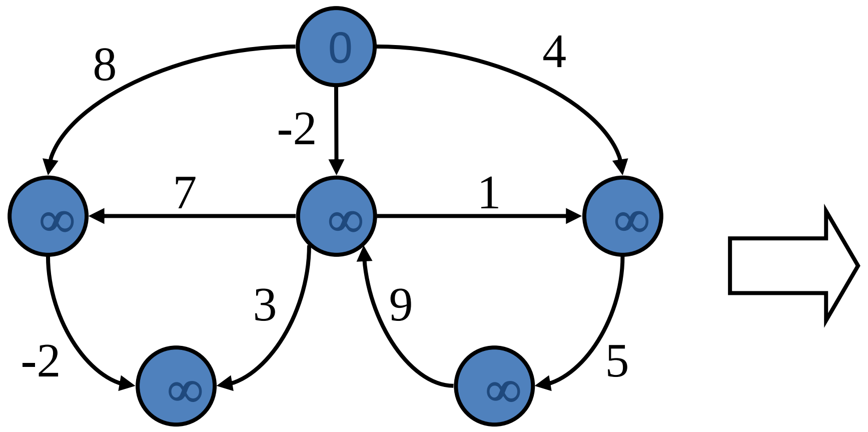
```
**If** $\mathrm{d}[v] > \mathrm{d}[u] + w(u, v)$ **then**
    $\mathrm{d}[v] \leftarrow \mathrm{d}[u] + w(u, v)$
```

Running time $\Theta(|V| \cdot |E|)$

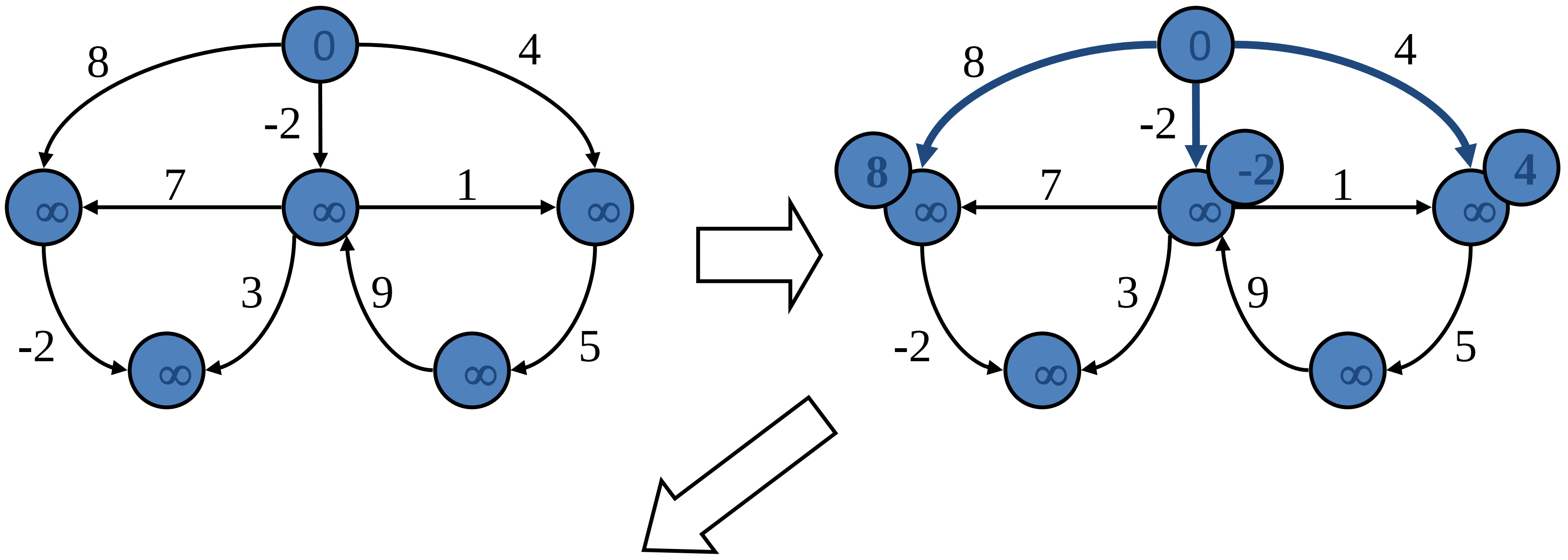
Property: Suppose we relax all edges one more time. If $d[]$ decreases for a vertex then there is a negative cycle. If $d[]$ remains the same, no negative cycle.

Case study III: Bellman-Ford

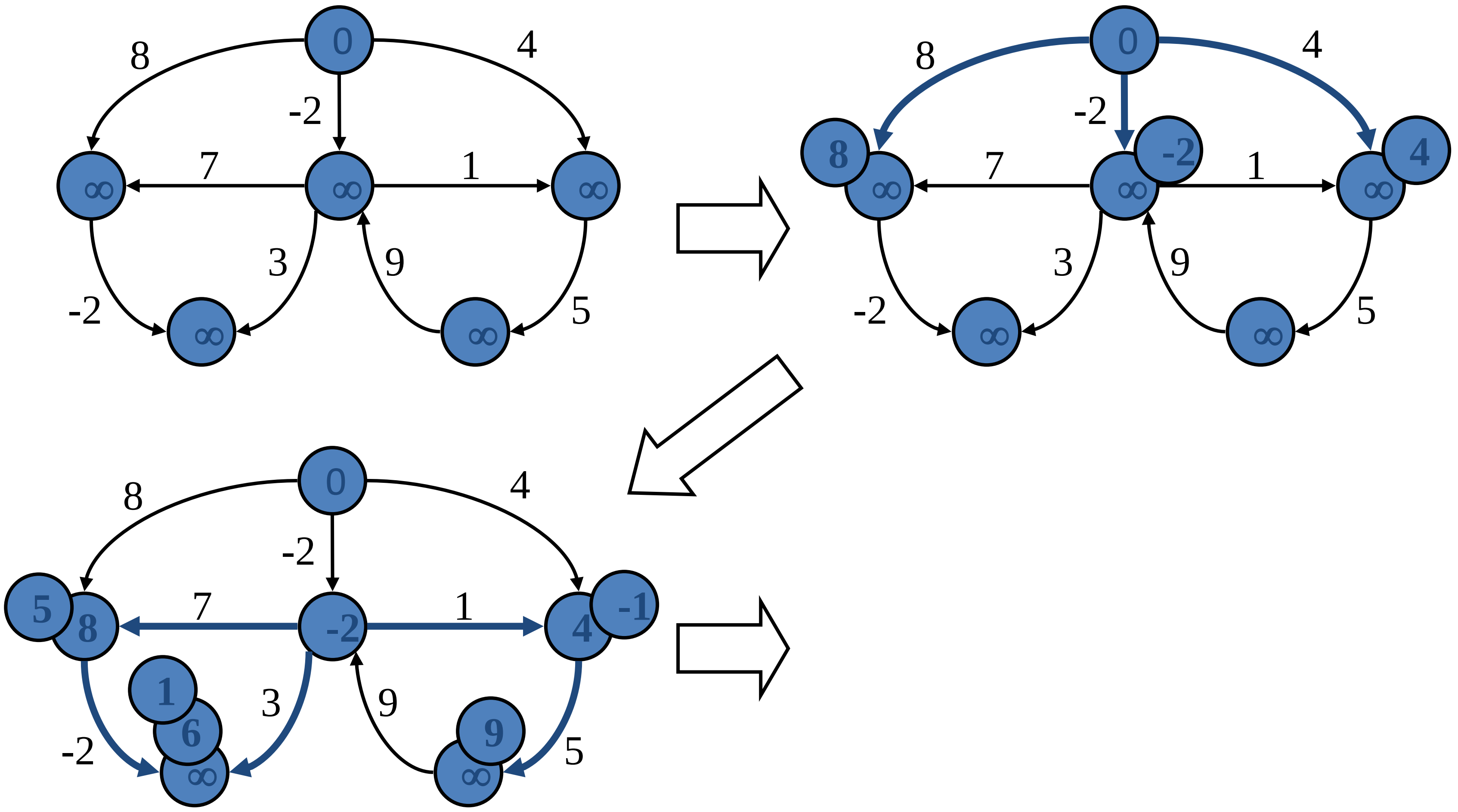
Find the shortest weight path from node 0.



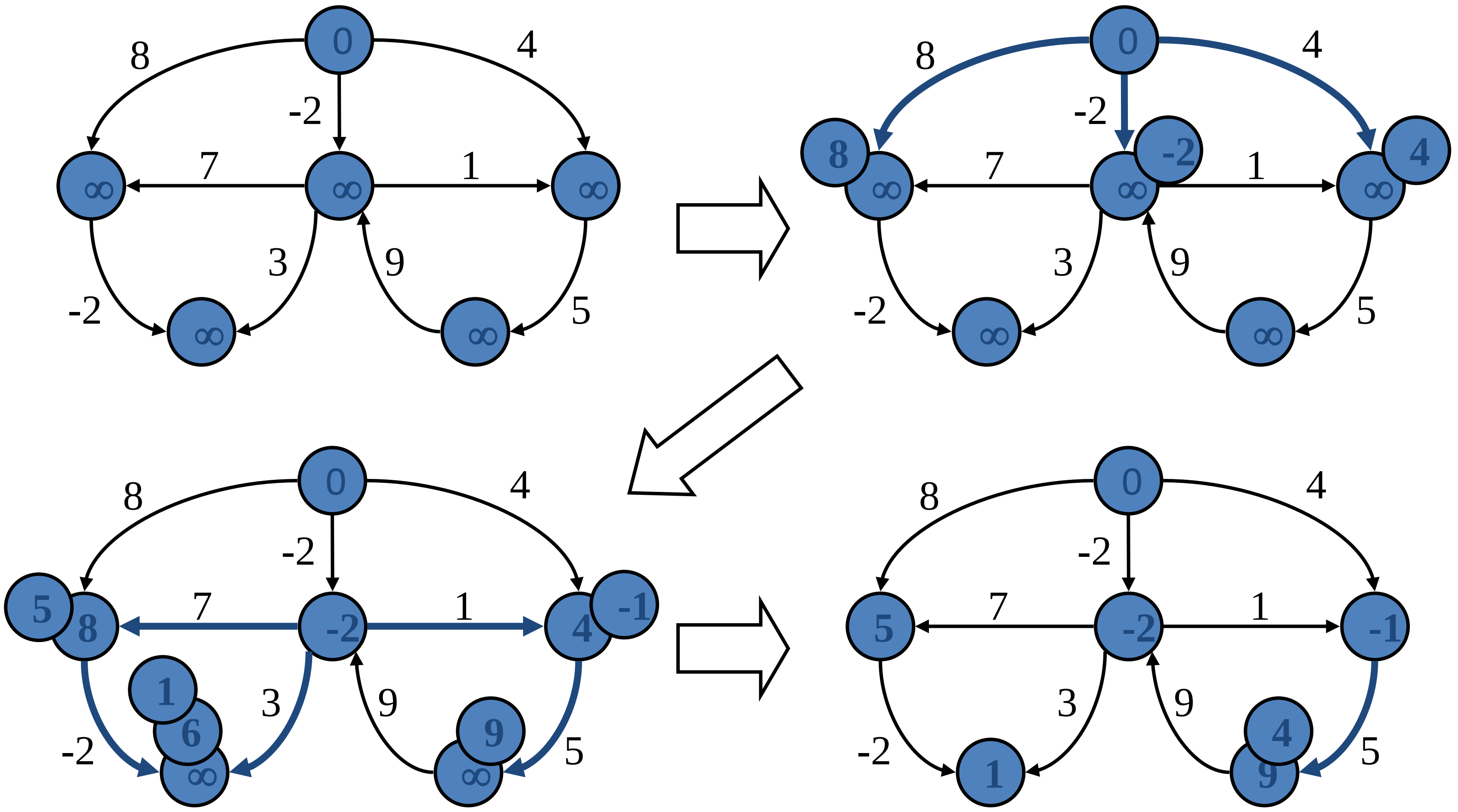
Case study III: Bellman-Ford



Case study III: Bellman-Ford



Case study III: Bellman-Ford

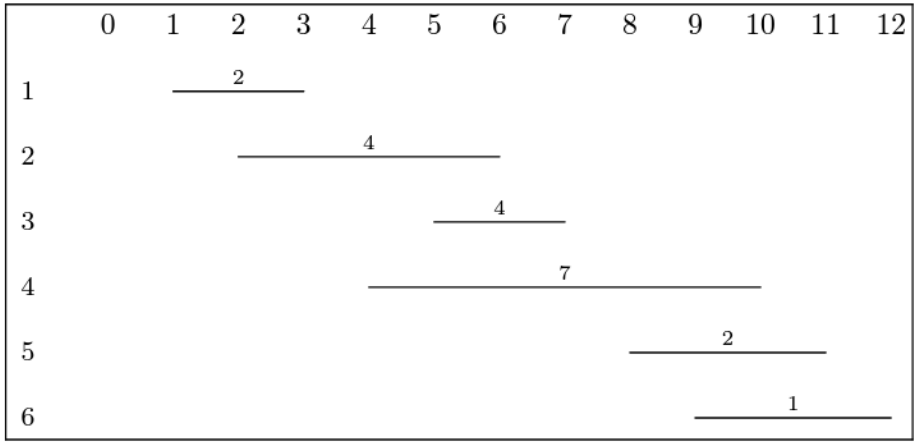


Case study IV: Interval Scheduling

Problem: You are given a collection of n intervals represented by start time, finish time, and value: (s_j, f_j, v_j), sorted w.r.t f_j. Find a non-overlapping set of intervals with maximum total value.

Example:

j	$s(j)$	$f(j)$	$v(j)$
1	1	3	2
2	2	6	4
3	5	7	4
4	4	10	7
5	8	11	2
6	9	12	1

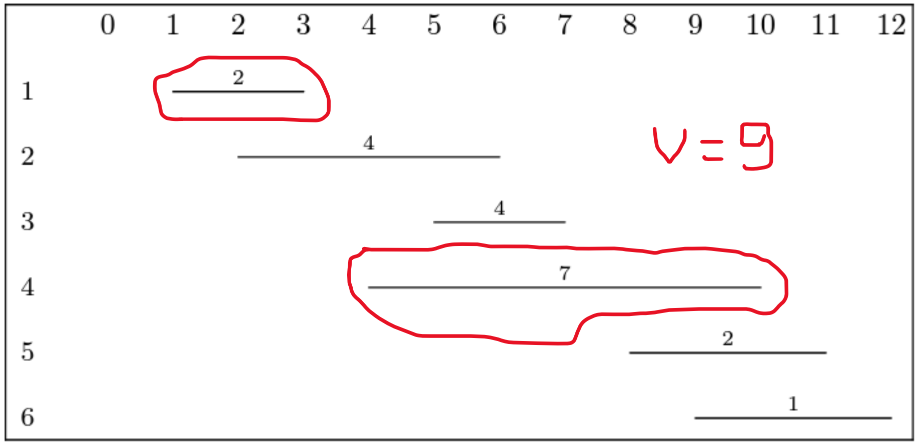


Case study IV: Interval Scheduling

Problem: You are given a collection of n intervals represented by start time, finish time, and value: (s_j, f_j, v_j), sorted w.r.t f_j. Find a non-overlapping set of intervals with maximum total value.

Example:

j	$s(j)$	$f(j)$	$v(j)$
1	1	3	2
2	2	6	4
3	5	7	4
4	4	10	7
5	8	11	2
6	9	12	1



Case study IV: Interval Scheduling

Step 1: Define the problem and subproblems.

Answer: Let $DP[j]$ be the maximum value that can be obtained from a set of non-overlapping intervals with indices in the range $\{1, \ldots, j\}$

Case study IV: Interval Scheduling

Step 1: Define the problem and subproblems.

Answer: Let $DP[j]$ be the maximum value that can be obtained from a set of non-overlapping intervals with indices in the range $\{1, \ldots, j\}$

Step 2: Define the goal/output given Step 1.

It is $\boldsymbol{DP[n]}$.

Case study IV: Interval Scheduling

Step 1: Define the problem and subproblems.

Answer: Let $DP[j]$ be the maximum value that can be obtained from a set of non-overlapping intervals with indices in the range $\{1, \ldots, j\}$

Step 2: Define the goal/output given Step 1.

It is $\boldsymbol{DP[n]}$.

Step 3: Define the base cases

It is $DP[0] = 0$.

Step 4: Define the recurrence

Case study IV: Interval Scheduling

Step 4: Define the recurrence

Interval j belongs to the optimal solution or **not**.

$$DP[j] \;=\; \max(\mathbf{DP}[\$] + \mathbf{v_j}, \mathbf{DP}[\mathbf{j-1}])$$

What is **$** ?

Case study IV: Interval Scheduling

Step 4: Define the recurrence

Interval j belongs to the optimal solution or **not**.

$$DP[j] \ = \ \max(\mathbf{DP[\$] + v_j}, \mathbf{DP[j-1]})$$

$ should be the interval with highest index in $\{1, \ldots, j-1\}$ that does not intersect with j (since j is chosen).

Let $p[j]$ be the highest index in $\{1, \ldots, j-1\}$ that does not intersect with j. Then the recurrence becomes

$$DP[j] \ = \ \max(\mathbf{DP[p[j]] + v_j}, \mathbf{DP[j-1]})$$

Case study IV: Interval Scheduling

Pseudocode:

```
Array DP[]
DP[0] $\leftarrow$ 0
**For** $k = 1$ to $n$ **do**
    DP[$k$] $\leftarrow \max(\text{DP}[k-1], \text{DP}[p[k]] + v[k])$
**return** DP[$n$]
```

Initialization

Bottom up filliing DP

Goal

Case study IV: Interval Scheduling

Pseudocode:

```
Array DP[]
DP[0] $\leftarrow 0$
**For** $k = 1$ to $n$ **do**
    DP[$k$] $\leftarrow \max(\mathrm{DP}[k-1], \mathrm{DP}[p[k]] + v[k])$
**return** DP[$n$]
```

Initialization

Bottom up filliing DP

Goal

Question: How can we compute $p[j]$ for $1 \le j \le n$ in $\Theta(n \log n)$ time?

Case study IV: Interval Scheduling

Question: How can we compute $p[j]$ for $1 \leq j \leq n$ in $\Theta(n \log n)$ time?

Answer:

- Sort first the intervals in increasing order of finishing times.

Case study IV: Interval Scheduling

Question: How can we compute $p[j]$ for $1 \leq j \leq n$ in $\Theta(n \log n)$ time?

Answer:

- Sort first the intervals in increasing order of finishing times.
- For every j, do binary search to find the interval before j with finishing time at most s_j

Case study V: Longest Common Subsequence

Problem: You are given two strings $x = X_1 \dots X_n$ and $y = Y_1 \dots Y_m$ of sizes n, m and you are asked to find the size of a longest common substring z of x and y.

Example:

x = H I E R O G L Y P H O L O G Y

y = M I C H E L A N G E L O

Case study V: Longest Common Subsequence

Problem: You are given two strings $x = X_1 \ldots X_n$ and $y = Y_1 \ldots Y_m$ of sizes n, m and you are asked to find the size of a longest common substring z of x and y.

Example:

x = H I E R O G L Y P H O L O G Y

y = M I C H E L A N G E L O

z = H E G L O

Case study V: LCS

Step 1: Define the problem and subproblems.

Answer: Let $DP[i, j]$ be the longest common substring that can be obtained from substrings $X_1X_2 \dots X_i$ and $Y_1Y_2 \dots Y_j$.

Case study V: LCS

Step 1: Define the problem and subproblems.

Answer: Let $DP[i, j]$ be the longest common substring that can be obtained from substrings $X_1X_2 \dots X_i$ and $Y_1Y_2 \dots Y_j$.

Step 2: Define the goal/output given Step 1.

It is $\boldsymbol{DP[n, m]}$.

Case study V: LCS

Step 1: Define the problem and subproblems.

Answer: Let $DP[i, j]$ be the longest common substring that can be obtained from substrings $X_1X_2 \dots X_i$ and $Y_1Y_2 \dots Y_j$.

Step 2: Define the goal/output given Step 1.

It is $\boldsymbol{DP[n, m]}$.

Step 3: Define the base cases. "One of two strings is empty". $DP[0, j] = 0$ for all j, $DP[i, 0] = 0$ for all i.

Case study V: LCS

Step 1: Define the problem and subproblems.

Answer: Let $DP[i, j]$ be the longest common substring that can be obtained from substrings $X_1X_2 \dots X_i$ and $Y_1Y_2 \dots Y_j$.

Step 2: Define the goal/output given Step 1.

It is $\boldsymbol{DP[n, m]}$.

Step 3: Define the base cases. “One of two strings is empty”. $DP[0, j] = 0$ for all j, $DP[i, 0] = 0$ for all i.

Step 4: Define the recurrence

Case study V: LCS

Step 4: Define the recurrence

Case 1: $x_i = X_1 X_2 \dots X_{i-1} A$
$y_j = Y_1 Y_2 \dots Y_{j-1} A$

Question: What is the LCS of x_i, y_j?

Case study V: LCS

Step 4: Define the recurrence

Case 1: $x_i = X_1 X_2 \ldots X_{i-1}\, A$

$y_j = Y_1 Y_2 \ldots Y_{j-1}\, A$

Question: What is the LCS of x_i, y_j?

Answer: **1** + the LCS of $\boldsymbol{x_{i-1}, y_{j-1}}$

Case study V: LCS

Step 4: Define the recurrence

Case 2: $x_i = X_1 X_2 \dots X_{i-1}\, A$

$y_j = Y_1 Y_2 \dots Y_{j-1}\, B$

Question: What is the LCS of x_i, y_j?

Case study V: LCS

Step 4: Define the recurrence

Case 2: $x_i = X_1 X_2 \ldots X_{i-1}\, A$

$y_j = Y_1 Y_2 \ldots Y_{j-1}\, B$

Question: What is the LCS of x_i, y_j?

Answer: the maximum of the

LCS of $\boldsymbol{x_{i-1}, y_j}$ and LCS of $\boldsymbol{x_i, y_{j-1}}$

Case study V: LCS

Step 4: Define the recurrence

(case 1)

$$DP[i,j] = \begin{cases} \text{if } \mathrm{X_i} == \mathrm{Y}_j \text{ then } DP[i-1,j-1]+1 \\ \text{if } \mathrm{X_i} \neq \mathrm{Y}_j \text{ then } \max(DP[i-1,j], DP[i,j-1]) \end{cases}$$

(case 2)

Case study V: LCS

Pseudocode:

```
Array DP[][], X[], Y[]
For $i = 1$ to $n$ do
  $\mathrm{DP}[i, 0] \leftarrow 0$
For $j = 1$ to $m$ do
  $\mathrm{DP}[0, j] \leftarrow 0$
For $i = 1$ to $n$ do
  For $j = 1$ to $m$ do
    If $X[i] == Y[j]$ then
      $\mathrm{DP}[i, j] \leftarrow \mathrm{DP}[i-1, j-1] + 1$
    else $\mathrm{DP}[i, j] \leftarrow \max(\mathrm{DP}[i-1, j], \mathrm{DP}[i, j-1])$
return $\mathrm{DP}[n, m]$
```

Initialization

Bottom up filliing DP

Goal

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0						
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0			
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1		
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1					
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1				
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1			
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1		
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	1	1	2	2	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	↖ 1	← 1	1	1	2	2
3	C	0	1	1	↖ 2	← 2	2	2
4	B	0	1	1	2	2	↖ 3	3
5	D	0	1	2	2	2	↑ 3	3
6	A	0	1	2	2	3	3	↖ 4
7	B	0	1	2	2	3	4	↑ 4

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	1	1	2	2	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	↖ 1	1	2	2	3	3
5	D	0	1	↖ 2	← 2	2	3	3
6	A	0	1	2	2	↖ 3	3	4
7	B	0	1	2	2	3	↖ 4	← 4

Case study V: LCS

Example: x is the string "ABCBDAB" and y is the string "BDCABA".

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	1	1	2	2	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4