

Lecture 23

Dynamic Programming (Longest Common Subsequence): Problem Analysis, Notations, Designing DP Algorithm for LCS & its Time Complexity, and Applications of LCS





Problem Statement

- › Imran is too good with string observations but now the Asif thought to defeat Imran.
- › So, Asif gave him two strings and told him to find the length of the longest common subsequence.



Subsequence

- › The question demands you to find the longest **subsequence** .
- › What is a subsequence?
 - A subsequence is a sequence that can be derived from another sequence by zero or more elements, without changing the order of the remaining elements.
 - Suppose X and Y are two sequences over a finite set of elements.
 - We can say that Z is a common subsequence of X and Y, if Z is a subsequence of both X and Y.



Longest Common Subsequence

- › If a set of sequences are given, then the longest common subsequence problem is
 - to find a common subsequence of all the sequences that is of maximal length.
- › The longest common subsequence problem is a classic computer science problem, the basis of data comparison programs such as Diff-Utility (a data comparison tool that calculates and displays the differences between the two texts. It tries to determine the smallest set of deletions and insertions and create one text from the other. Diff is line-oriented rather than character-oriented).
- › Applications
 - Bioinformatics.
 - Revision control systems, such as SVN (Apache Subversion) and Git, for reconciling multiple changes made to a revision-controlled collection of files.
 - › SVN is used to manage and track changes to code and assets across projects.
 - › Git is a distributed version control system,
 - › SVN is a centralized version control system.



Longest common subsequence (LCS)

For a sequence $X = x_1, x_2, \dots, x_n$, a subsequence is a subset of the sequence defined by a set of increasing indices (i_1, i_2, \dots, i_k) where $1 \leq i_1 < i_2 < \dots < i_k \leq n$

$X = A B A C D A B A B$

$ABA?$



Longest common subsequence (LCS)

For a sequence $X = x_1, x_2, \dots, x_n$, a subsequence is a subset of the sequence defined by a set of increasing indices (i_1, i_2, \dots, i_k) where $1 \leq i_1 < i_2 < \dots < i_k \leq n$

$X = \text{A B A C D A B A B}$

A B A



Longest common subsequence (LCS)

For a sequence $X = x_1, x_2, \dots, x_n$, a subsequence is a subset of the sequence defined by a set of increasing indices (i_1, i_2, \dots, i_k) where $1 \leq i_1 < i_2 < \dots < i_k \leq n$

$X = A B A C D A B A B$

$ACA?$



Longest common subsequence (LCS)

For a sequence $X = x_1, x_2, \dots, x_n$, a subsequence is a subset of the sequence defined by a set of increasing indices (i_1, i_2, \dots, i_k) where $1 \leq i_1 < i_2 < \dots < i_k \leq n$

$X = A B A C D A B A B$

ACA



Longest common subsequence (LCS)

For a sequence $X = x_1, x_2, \dots, x_n$, a subsequence is a subset of the sequence defined by a set of increasing indices (i_1, i_2, \dots, i_k) where $1 \leq i_1 < i_2 < \dots < i_k \leq n$

$X = A B A C D A B A B$

DCA?



Longest common subsequence (LCS)

For a sequence $X = x_1, x_2, \dots, x_n$, a subsequence is a subset of the sequence defined by a set of increasing indices (i_1, i_2, \dots, i_k) where $1 \leq i_1 < i_2 < \dots < i_k \leq n$

$X = A B A C D A B A B$

~~DCA~~



Longest common subsequence (LCS)

For a sequence $X = x_1, x_2, \dots, x_n$, a subsequence is a subset of the sequence defined by a set of increasing indices (i_1, i_2, \dots, i_k) where $1 \leq i_1 < i_2 < \dots < i_k \leq n$

$X = A B A C D A B A B$

$A A D A A ?$



Longest common subsequence (LCS)

For a sequence $X = x_1, x_2, \dots, x_n$, a subsequence is a subset of the sequence defined by a set of increasing indices (i_1, i_2, \dots, i_k) where $1 \leq i_1 < i_2 < \dots < i_k \leq n$

$X = A B A C D A B A B$

$A A D A A$



LCS problem

Given two sequences X and Y , a common subsequence is a subsequence that occurs in both X and Y

Given two sequences $X = x_1, x_2, \dots, x_n$ and $Y = y_1, y_2, \dots, y_n$,

What is the longest common subsequence?

$X = A B C B D A B$

$Y = B D C A B A$



LCS problem

Given two sequences X and Y , a common subsequence is a subsequence that occurs in both X and Y

Given two sequences $X = x_1, x_2, \dots, x_n$ and $Y = y_1, y_2, \dots, y_n$,

What is the longest common subsequence?

$X = A B C B D A B$

$Y = B D C A B A$



Step 1: Define the problem with respect to subproblems

$X = A B C B D A B$

$Y = B D C A B A$

Assume you have a solver for smaller problems



Step 1: Define the problem with respect to subproblems

$X = A B C B D A ?$



$Y = B D C A B ?$



Is the last character part of the LCS?



Step 1: Define the problem with respect to subproblems

$X = A B C B D A ?$



$Y = B D C A B ?$



Two cases: either the characters
are the same or they're different

Step 1: Define the problem with respect to subproblems

$X = \boxed{A B C B D A} A$

LCS

$Y = \boxed{B D C A B} A$

The characters are
part of the LCS

What is the recursive
relationship?

If they're the same

$$LCS(X, Y) = LCS(X_{1...n-1}, Y_{1...m-1}) + x_n$$

Step 1: Define the problem with respect to subproblems

$X = \boxed{A B C B D A} B$

LCS

$Y = \boxed{B D C A B A}$

If they're different

$$LCS(X, Y) = LCS(X_{1...n-1}, Y)$$

Step 1: Define the problem with respect to subproblems

$X = \boxed{A B C B D A B}$

LCS

$Y = \boxed{B D C A B} A$

If they're different

$$LCS(X, Y) = LCS(X, Y_{1...m-1})$$

Step 1: Define the problem with respect to subproblems

$X = \boxed{A B C B D A} B$

$Y = \boxed{B D C A B A}$

$X = \boxed{A B C B D A B}$

$Y = \boxed{B D C A B} A$

?

If they're different



Step 1: Define the problem with respect to subproblems

X = A B C B D A B



Y = B D C A B A



$$LCS(X, Y) = \begin{cases} 1 + LCS(X_{1\dots n-1}, Y_{1\dots m-1}) & \text{if } x_n = y_m \\ \max(LCS(X_{1\dots n-1}, Y), LCS(X, Y_{1\dots m-1})) & \text{otherwise} \end{cases}$$

(for now, let's just worry about counting the length of the LCS)

DR. KHALID IQBAL

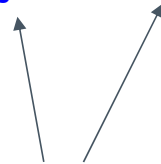


Step 2: Build the solution from the bottom up

$$LCS(X, Y) = \begin{cases} 1 + LCS(X_{1\dots n-1}, Y_{1\dots m-1}) & \text{if } x_n = y_m \\ \max(LCS(X_{1\dots n-1}, Y), LCS(X, Y_{1\dots m-1})) & \text{otherwise} \end{cases}$$

What types of subproblem?
solutions do we need to store?

$LCS(X_{1\dots j}, Y_{1\dots k})$



two different indices



Step 2: Build the solution from the bottom up

$$LCS(X, Y) = \begin{cases} 1 + LCS(X_{1\dots n-1}, Y_{1\dots m-1}) & \text{if } x_n = y_m \\ \max(LCS(X_{1\dots n-1}, Y), LCS(X, Y_{1\dots m-1})) & \text{otherwise} \end{cases}$$

What types of subproblem?
solutions do we need to store?

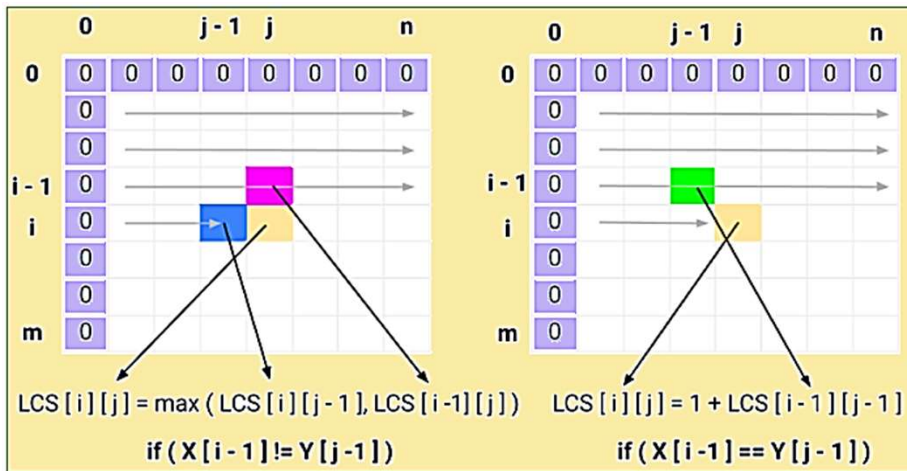
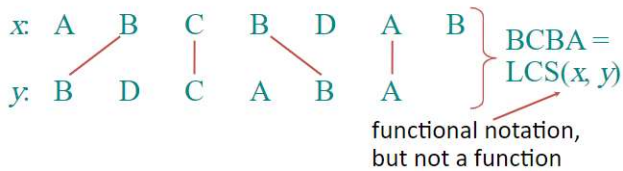
$$LCS(X_{1\dots j}, Y_{1\dots k})$$

$$LCS[i, j] = \begin{cases} 1 + LCS[i-1, j-1] & \text{if } x_i = y_j \\ \max(LCS[i-1, j], LCS[i, j-1]) & \text{otherwise} \end{cases}$$

Longest Common Subsequence

- Given two sequences $x[1 \dots m]$ and $y[1 \dots n]$, find a longest subsequence common to them both.

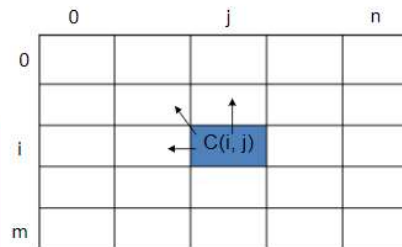
"a" not "the"



DP Algorithm

- Key: find out the correct order to solve the sub-problems
- Total number of sub-problems: $m * n$

$$c[i, j] = \begin{cases} c[i-1, j-1] + 1 & \text{if } x[i] = y[j], \\ \max\{c[i-1, j], c[i, j-1]\} & \text{otherwise.} \end{cases}$$



DP Algorithm

LCS-Length(X, Y)

1. $m = \text{length}(X)$ // get the # of symbols in X
2. $n = \text{length}(Y)$ // get the # of symbols in Y
3. for $i = 1$ to m $c[i, 0] = 0$ // special case: $Y[0]$
4. for $j = 1$ to n $c[0, j] = 0$ // special case: $X[0]$
5. for $i = 1$ to m // for all $X[i]$
6. for $j = 1$ to n // for all $Y[j]$
7. if ($X[i] == Y[j]$)
8. $c[i, j] = c[i-1, j-1] + 1$
9. else $c[i, j] = \max(c[i-1, j], c[i, j-1])$
10. return c



$X_1 = A B C B D A B$
 $Y_1 = B D C A B A$

$|X_1| = 7$
 $|Y_1| = 6$

Example

Define $c[i, j]$ = length of LCS of X_i and Y_j .

We want $c[m, n]$.

Recursive Solution

$$c[i, j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ c[i-1, j-1] + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j, \\ \max(c[i-1, j], c[i, j-1]) & \text{if } i, j > 0 \text{ and } x_i \neq y_j. \end{cases}$$

DR. KHALID IQBAL

25

$$LCS[i, j] = \begin{cases} 1 + LCS[i-1, j-1] & \text{if } x_i = y_j \\ \max(LCS[i-1, j], LCS[i, j-1]) & \text{otherwise} \end{cases}$$

		j	0	1	2	3	4	5	6
		i	y _j	B	D	C	A	B	A
0	x _i								
1	A								
2	B								
3	C								
4	B								
5	D								
6	A								
7	B								

For Fibonacci and tree counting,
we had to initialize some entries in
the array. Any here?



$$LCS[i, j] = \begin{cases} 1 + LCS[i-1, j-1] & \text{if } x_i = y_j \\ \max(LCS[i-1, j], LCS[i, j-1]) & \text{otherwise} \end{cases}$$

		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						

Need to initialize values within 1 smaller in either dimension.

$$LCS[i, j] = \begin{cases} 1 + LCS[i-1, j-1] & \text{if } x_i = y_j \\ \max(LCS[i-1, j], LCS[i, j-1]) & \text{otherwise} \end{cases}$$

		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						

LCS(A, B)



$$LCS[i, j] = \begin{cases} 1 + LCS[i-1, j-1] & \text{if } x_i = y_j \\ \max(LCS[i-1, j], LCS[i, j-1]) & \text{otherwise} \end{cases}$$

		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					
2	B		0						
3	C		0						
4	B		0						
5	D		0						
6	A		0						
7	B		0						

$$LCS[i, j] = \begin{cases} 1 + LCS[i-1, j-1] & \text{if } x_i = y_j \\ \max(LCS[i-1, j], LCS[i, j-1]) & \text{otherwise} \end{cases}$$

		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						

LCS(A, BDCA)



$$LCS[i, j] = \begin{cases} 1 + LCS[i-1, j-1] & \text{if } x_i = y_j \\ \max(LCS[i-1, j], LCS[i, j-1]) & \text{otherwise} \end{cases}$$

		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						

LCS(A, BDCA)



$$LCS[i, j] = \begin{cases} 1 + LCS[i-1, j-1] & \text{if } x_i = y_j \\ \max(LCS[i-1, j], LCS[i, j-1]) & \text{otherwise} \end{cases}$$

		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	?	
5	D		0						
6	A		0						
7	B		0						

LCS(ABCB, BDCAB)



$$LCS[i, j] = \begin{cases} 1 + LCS[i-1, j-1] & \text{if } x_i = y_j \\ \max(LCS[i-1, j], LCS[i, j-1]) & \text{otherwise} \end{cases}$$

		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	
5	D		0						
6	A		0						
7	B		0						

LCS(ABCB, BDCAB)



$$LCS[i, j] = \begin{cases} 1 + LCS[i-1, j-1] & \text{if } x_i = y_j \\ \max(LCS[i-1, j], LCS[i, j-1]) & \text{otherwise} \end{cases}$$

		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

Where's the
final answer?



The algorithm

```
LCS-LENGTH( $X, Y$ )
1   $m \leftarrow \text{length}[X]$ 
2   $n \leftarrow \text{length}[Y]$ 
3   $c[0, 0] \leftarrow 0$ 
4  for  $i \leftarrow 1$  to  $m$ 
5       $c[i, 0] \leftarrow 0$ 
6  for  $j \leftarrow 1$  to  $n$ 
7       $c[0, j] \leftarrow 0$ 
8  for  $i \leftarrow 1$  to  $m$ 
9      for  $j \leftarrow 1$  to  $n$ 
10         if  $x_i = y_j$ 
11              $c[i, j] \leftarrow 1 + c[i - 1, j - 1]$ 
12         elseif  $c[i - 1, j] > c[i, j - 1]$ 
13              $c[i, j] \leftarrow c[i - 1, j]$ 
14         else
15              $c[i, j] \leftarrow c[i, j - 1]$ 
16  return  $c[m, n]$ 
```

The algorithm

LCS-LENGTH(X, Y)

```
1   $m \leftarrow \text{length}[X]$ 
2   $n \leftarrow \text{length}[Y]$ 
3   $c[0,0] \leftarrow 0$ 
4  for  $i \leftarrow 1$  to  $m$ 
5       $c[i,0] \leftarrow 0$ 
6  for  $j \leftarrow 1$  to  $n$ 
7       $c[0,j] \leftarrow 0$ 
8  for  $i \leftarrow 1$  to  $m$ 
9      for  $j \leftarrow 1$  to  $n$ 
10         if  $x_i = y_i$ 
11              $c[i,j] \leftarrow 1 + c[i-1,j-1]$ 
12         elseif  $c[i-1,j] > c[i,j-1]$ 
13              $c[i,j] \leftarrow c[i-1,j]$ 
14         else
15              $c[i,j] \leftarrow c[i,j-1]$ 
16  return  $c[m,n]$ 
```

Base case initialization

The algorithm

LCS-LENGTH(X, Y)

```
1   $m \leftarrow \text{length}[X]$ 
2   $n \leftarrow \text{length}[Y]$ 
3   $c[0, 0] \leftarrow 0$ 
4  for  $i \leftarrow 1$  to  $m$ 
5       $c[i, 0] \leftarrow 0$ 
6  for  $j \leftarrow 1$  to  $n$ 
7       $c[0, j] \leftarrow 0$ 
8  for  $i \leftarrow 1$  to  $m$ 
9      for  $j \leftarrow 1$  to  $n$ 
10         if  $x_i = y_i$ 
11              $c[i, j] \leftarrow 1 + c[i - 1, j - 1]$ 
12         elseif  $c[i - 1, j] > c[i, j - 1]$ 
13              $c[i, j] \leftarrow c[i - 1, j]$ 
14         else
15              $c[i, j] \leftarrow c[i, j - 1]$ 
16  return  $c[m, n]$ 
```

Fill in the matrix

The algorithm

LCS-LENGTH(X, Y)

```
1   $m \leftarrow \text{length}[X]$ 
2   $n \leftarrow \text{length}[Y]$ 
3   $c[0, 0] \leftarrow 0$ 
4  for  $i \leftarrow 1$  to  $m$ 
5       $c[i, 0] \leftarrow 0$ 
6  for  $j \leftarrow 1$  to  $n$ 
7       $c[0, j] \leftarrow 0$ 
8  for  $i \leftarrow 1$  to  $m$ 
9      for  $j \leftarrow 1$  to  $n$ 
10         if  $x_i = y_i$ 
11              $c[i, j] \leftarrow 1 + c[i - 1, j - 1]$ 
12         elseif  $c[i - 1, j] > c[i, j - 1]$ 
13              $c[i, j] \leftarrow c[i - 1, j]$ 
14         else
15              $c[i, j] \leftarrow c[i, j - 1]$ 
16  return  $c[m, n]$ 
```

The algorithm

LCS-LENGTH(X, Y)

```
1   $m \leftarrow \text{length}[X]$ 
2   $n \leftarrow \text{length}[Y]$ 
3   $c[0, 0] \leftarrow 0$ 
4  for  $i \leftarrow 1$  to  $m$ 
5       $c[i, 0] \leftarrow 0$ 
6  for  $j \leftarrow 1$  to  $n$ 
7       $c[0, j] \leftarrow 0$ 
8  for  $i \leftarrow 1$  to  $m$ 
9      for  $j \leftarrow 1$  to  $n$ 
10         if  $x_i = y_i$ 
11              $c[i, j] \leftarrow 1 + c[i - 1, j - 1]$ 
12         elseif  $c[i - 1, j] > c[i, j - 1]$ 
13              $c[i, j] \leftarrow c[i - 1, j]$ 
14         else
15              $c[i, j] \leftarrow c[i, j - 1]$ 
16  return  $c[m, n]$ 
```



The algorithm

LCS-LENGTH(X, Y)

```
1   $m \leftarrow \text{length}[X]$ 
2   $n \leftarrow \text{length}[Y]$ 
3   $c[0, 0] \leftarrow 0$ 
4  for  $i \leftarrow 1$  to  $m$ 
5       $c[i, 0] \leftarrow 0$ 
6  for  $j \leftarrow 1$  to  $n$ 
7       $c[0, j] \leftarrow 0$ 
8  for  $i \leftarrow 1$  to  $m$ 
9      for  $j \leftarrow 1$  to  $n$ 
10         if  $x_i = y_j$ 
11              $c[i, j] \leftarrow 1 + c[i - 1, j - 1]$ 
12         elseif  $c[i - 1, j] > c[i, j - 1]$ 
13              $c[i, j] \leftarrow c[i - 1, j]$ 
14         else
15              $c[i, j] \leftarrow c[i, j - 1]$ 
16 return  $c[m, n]$ 
```




Running time?

LCS-LENGTH(X, Y)

```
1   $m \leftarrow \text{length}[X]$ 
2   $n \leftarrow \text{length}[Y]$ 
3   $c[0, 0] \leftarrow 0$ 
4  for  $i \leftarrow 1$  to  $m$ 
5       $c[i, 0] \leftarrow 0$ 
6  for  $j \leftarrow 1$  to  $n$ 
7       $c[0, j] \leftarrow 0$ 
8  for  $i \leftarrow 1$  to  $m$ 
9      for  $j \leftarrow 1$  to  $n$ 
10         if  $x_i = y_i$ 
11              $c[i, j] \leftarrow 1 + c[i - 1, j - 1]$ 
12         elseif  $c[i - 1, j] > c[i, j - 1]$ 
13              $c[i, j] \leftarrow c[i - 1, j]$ 
14         else
15              $c[i, j] \leftarrow c[i, j - 1]$ 
16  return  $c[m, n]$ 
```

$\Theta(nm)$




Keeping track of the solution

Our LCS algorithm only calculated the length of the LCS between X and Y

What if we wanted to know the actual sequence?

Keep track of this as well...

```
8  for  $i \leftarrow 1$  to  $m$ 
9      for  $j \leftarrow 1$  to  $n$ 
10         if  $x_i = y_j$ 
11              $c[i, j] \leftarrow 1 + c[i - 1, j - 1]$ 
12         elseif  $c[i - 1, j] > c[i, j - 1]$ 
13              $c[i, j] \leftarrow c[i - 1, j]$ 
14         else
15              $c[i, j] \leftarrow c[i, j - 1]$ 
16  return  $c[m, n]$ 
```





$$LCS[i, j] = \begin{cases} 1 + LCS[i, j-1] & \text{if } x_i = y_j \\ \max(LCS[i-1, j], LCS[i, j-1]) & \text{otherwise} \end{cases}$$

		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

We can follow the arrows to generate the solution

$$LCS[i, j] = \begin{cases} 1 + LCS[i, j-1] & \text{if } x_i = y_j \\ \max(LCS[i-1, j], LCS[i, j-1]) & \text{otherwise} \end{cases}$$

		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

We can follow the arrows to generate the solution

BCBA

Thank You!!!

Have a good day

