Bellman ford Implementation

Relaxation

V-1 time

$$\boxed{\begin{array}{l} d(u) + c(u,v) < d(v): \\ \qquad d(v) = d(u) + c(u,v) \end{array}}$$

ONE CHECK

Changing

(flag)

Negative Path cycle

Not changing

Report

Catalan Numbers

0 1 2 3 4 5 6

1, 1, 2, 5, 14, 42, 132, - - - - - - - - - -

$$C_n = C_0 C_{n-1} + C_1 C_{n-2} + C_2 C_{n-3} + \cdots + C_{n-1} C_0$$

Base case

$$C_0 = C_1 = 1$$

$$C_2 = C_0 C_1 + C_1 C_0$$
$$= 1*1 + 1*1$$
$$= 1 + 1 = 2$$

$$C_3 = C_0 C_2 + C_1 C_1 + C_2 C_0$$
$$= 1*2 + 1*1 + 2*1$$
$$= 2 + 1 + 2 = 5$$

$$C_4 = C_0 C_3 + C_1 C_2 + C_2 C_1 + C_3 C_0$$
$$= 1*5 + 1*2 + 2*1 + 5*1$$
$$= 5 + 2 + 2 + 5$$
$$= 14$$

Applications

catalan number C_n

1) To find out the total number of BSTs given the value of n by the user.

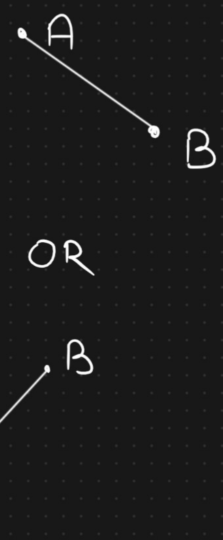
Base case condition

		Total Number of Ways BST	
$n = 0$		1	C_0
$n = 1$	• A	1	C_1

AB

$A < B$

$n = 2$ — 2 — C_2

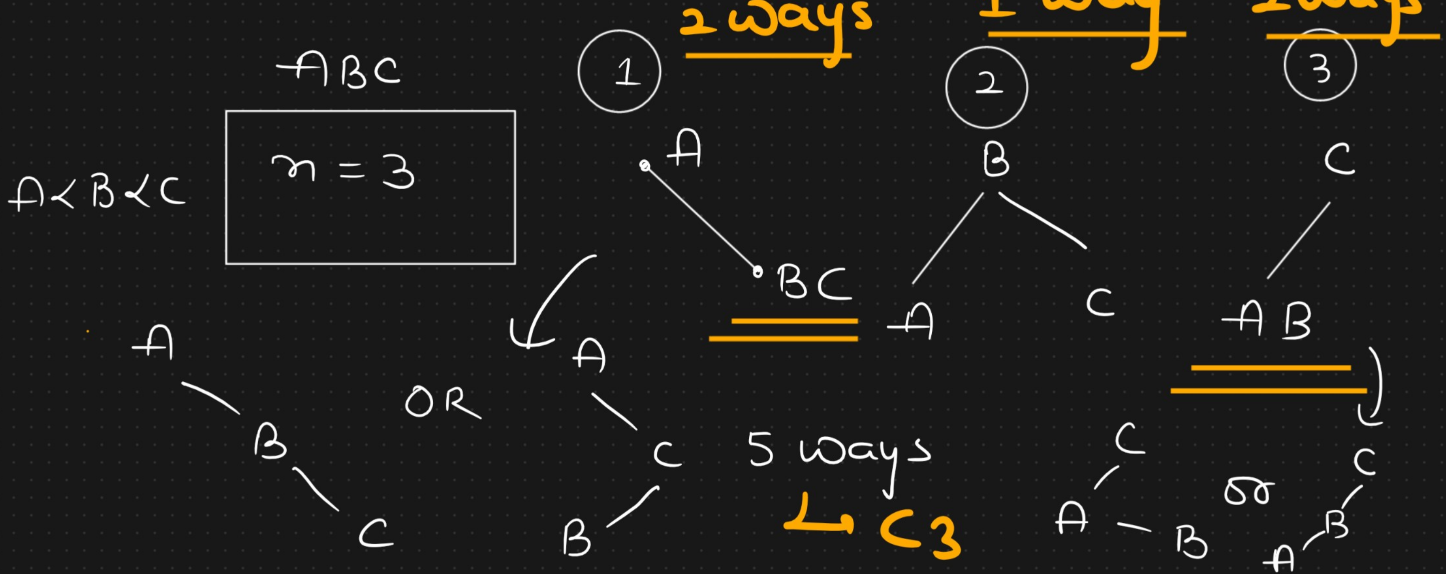


ABC

$A < B < C$

$n = 3$

2 ways — 1 way — 2 ways



$n = 6$

$\hookrightarrow \# BST's = 132$

$C_0 = C_1 = \underline{1}$

Non-comparison Sorting

↳ count sort

max(arr) → 7 1000

max value in arr ↓ k = 7

0 to 9

array →

0	1	2	3	4	5	6
1	4	1	2	7	5	2

Sorted array →

0	1	2	3	4	5	6
1	1	2	2	4	5	7

cnt_arr

count array

0	2	2	0	1	1	0	1
0	1	2	3	4	5	6	7

Psuedocode

```
for i in range(n):
    cnt_arr[a(i)] += 1
```

i = 1 to 7

cum_arr(i) = count(i) + count(i-1)

cumu. sum array

0	1	2	3	4	5	6	7
0	2	4	4	5	6	6	7
	~~1~~ 0	~~3~~ ~~2~~ 2		4	5		6

→ Position of element

Sorted array

0	1	2	3	4	5	6	7
1	1	2	2	4	5	7	

sorted order

```
for i in range(n-1, -1, -1):
    sorted_arr[--cum_arr[a[i]]] = a[i]
```

Stable & Unstable Sorting

1^a 1^b

Stability

Maintain the Stability in

1^a 1^b

Sorting algorithm

Time complexity

↳ $\Theta(n)$

Space complexity

↳ $\Theta(n)$

Drawback

↳ Range → max element value in array

Range → Small range

↳ count sort is preferable

$\#\text{elements} \approx \max(\text{arr}(i))$

↳ Count sort

↑ Rare in Real life applications