Intro to Dynamic Programming (DP)

Presented by Alex Li

What is DP?

Solving a problem by first making it harder.

A classical conundrum

You can only go RIGHT or UP. How many ways are there to get from START to END?

EVERY OTHER SQUARE?

					END
START					

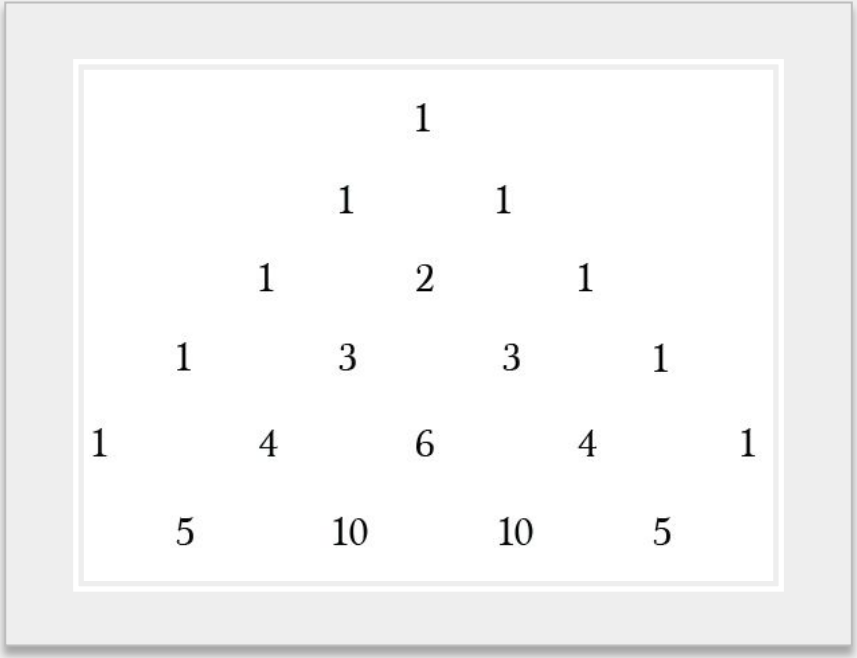
A classical conundrum

You can only go RIGHT or UP. How many ways are there to get from START to EVERY OTHER SQUARE?

1	5	15			
1	4	10			
1	3	6	10	15	21
1	2	3	4	5	6
START	1	1	1	1	1

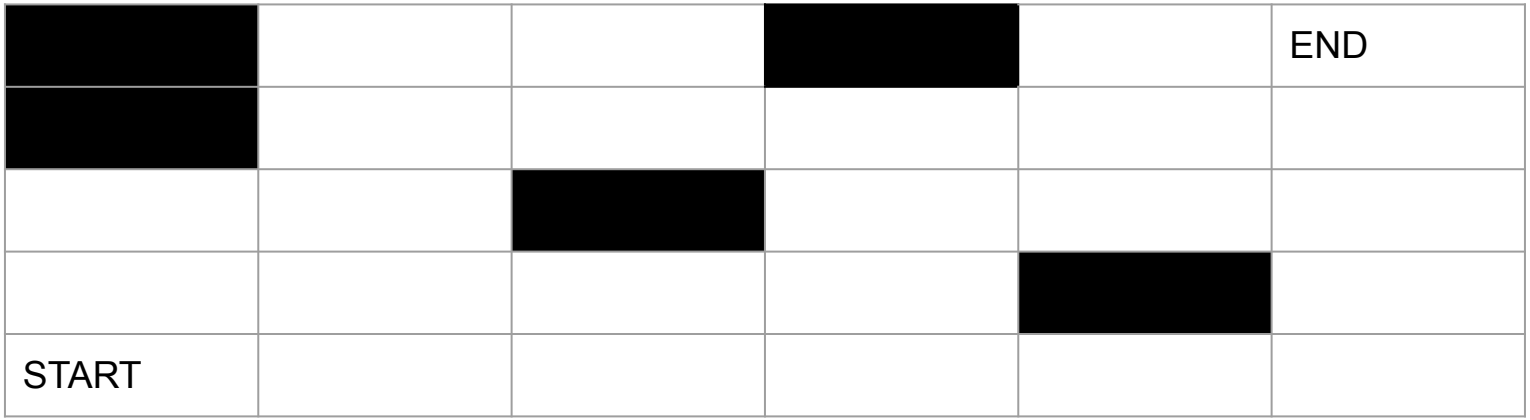
The antihero - STATIC PROGRAMMING!

- I see Pascal's triangle...
- Answer is just $\binom{9}{5}$
- Equal to number of ways to rearrange the sequence RRRRRUUUU



The Generalizable approach - Blockades

You can only go RIGHT or UP, but not through black squares. How many ways are there to get from START to END?



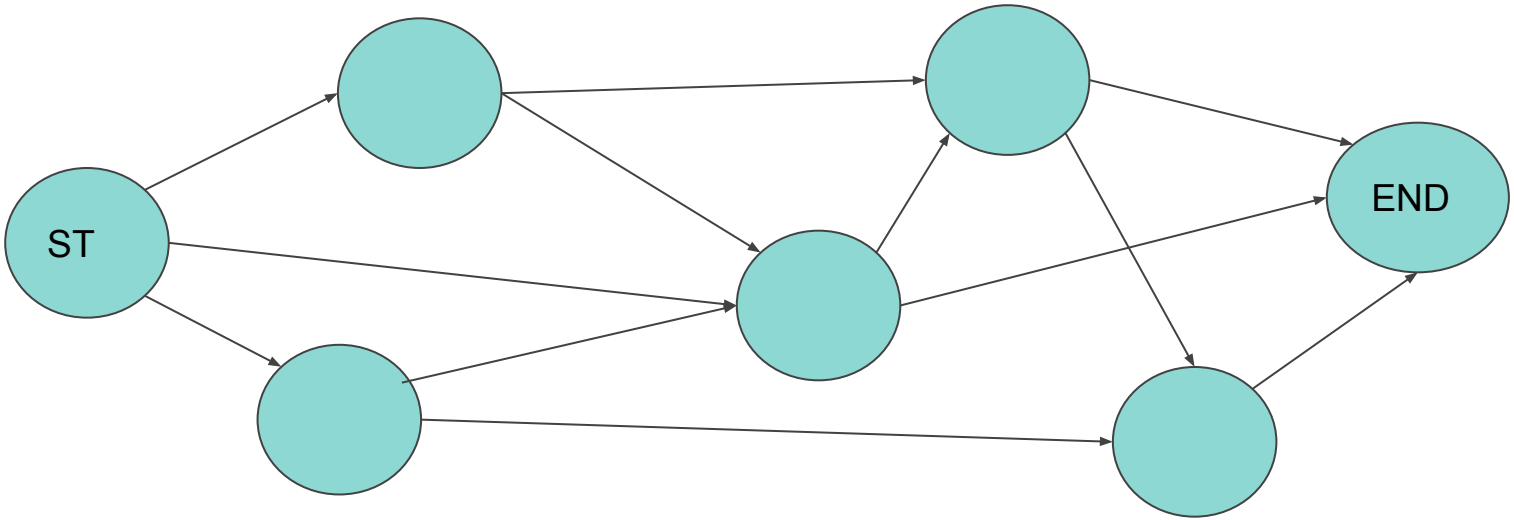
The Generalizable approach - Blockades

You can only go RIGHT or UP, but not through black squares. How many ways are there to get from START to END?

	3	6		11	27
	3	3	7	11	16
1	3		4	4	5
1	2	3	4		1
START	1	1	1	1	1

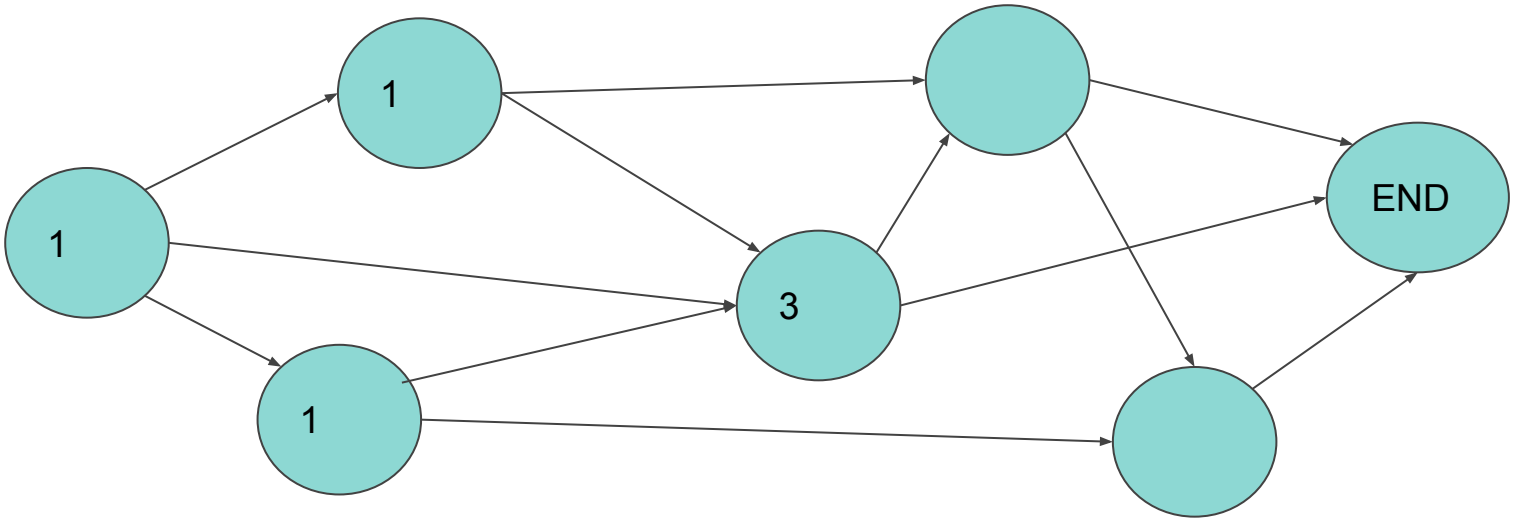
The Generalizable approach - Blockades 2.0

How many ways are there to get from START to END?



The Generalizable approach - Blockades 2.0

How many ways are there to get from START to END?



The Generalizable approach - Optimizing

You can only go RIGHT or UP. What's the maximum score you can get, if your score is the sum of the squares your path crossed?

Score

+0	+2	0
+3	+1	+4
+2	+5	+1
0	+3	+4

The Generalizable approach - Optimizing

You can only go RIGHT or UP. What's the maximum score you can get, if your score is the sum of the squares your path crossed?

Best Score

5		
5		
2	8	
0	3	7

Score

+0	+2	0
+3	+1	+4
+2	+5	+1
0	+3	+4

Takeaways

If you can write a recurrence relation, then we can do DP by saving the states in a table.

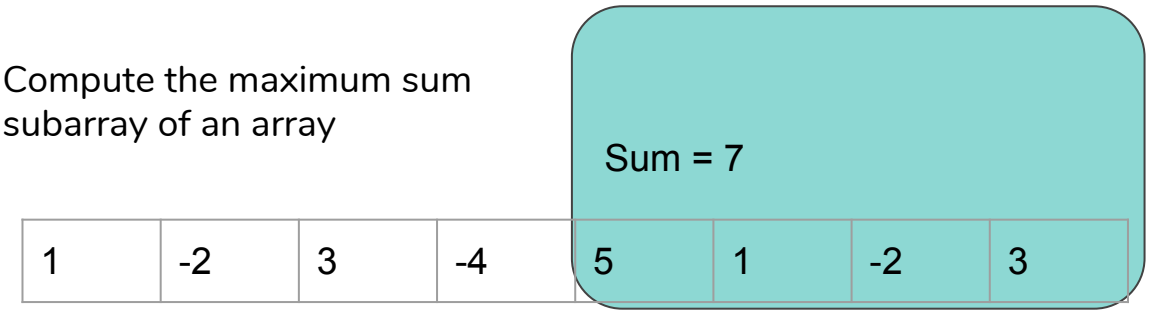
Distinctways[i][j] = Distinctways[i-1][j] + Distinctways[i][j-1]

BestScore[i][j] = Score[i][j] + max(BestScore[i-1][j], BestScore[i][j-1])

Contrast to memoization

This looks different

Compute the maximum sum subarray of an array



Max sum subarray

1	-2	3	-4	5	1	-2	3

Best sum in the first i positions

0	1	1					

How to fill in the next entry?
Not obvious, since we are extending from the array before it

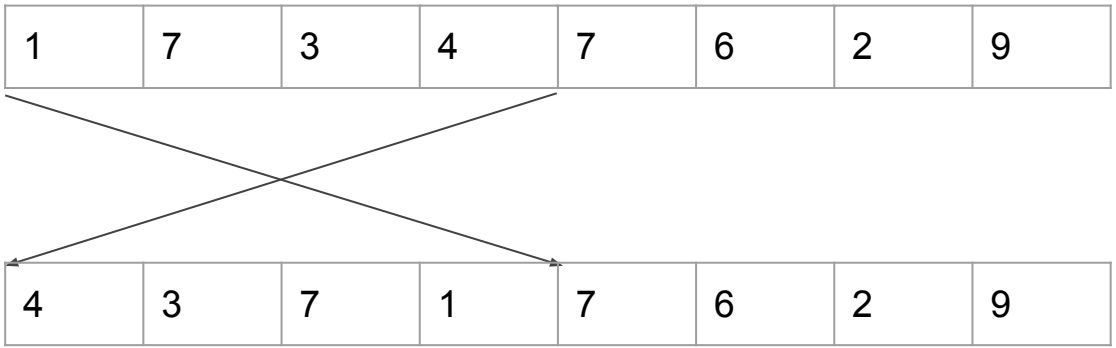
Max sum subarray

	1	-2	3	-4	5	1	-2	3
Best sum in the first i positions	0	1	1					
Best sum ending in this position	0	1	-1	2				

https://github.com/cpcosu/Weekly_Events/blob/master/Autumn%202020/2020-09-22/BestSum.cpp

Application in a problem

Reverse a subarray of this array so as to maximize the sum of the even indexed elements

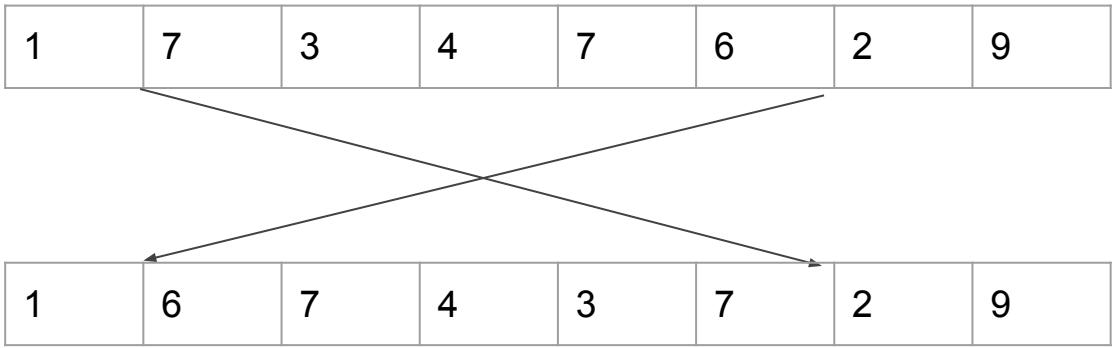


score = 1 + 3 + 7 + 2 = 13

score = 4 + 7 + 7 + 2 = 20

Observations?

Reverse a subarray of this array so as to maximize the sum of the even indexed elements

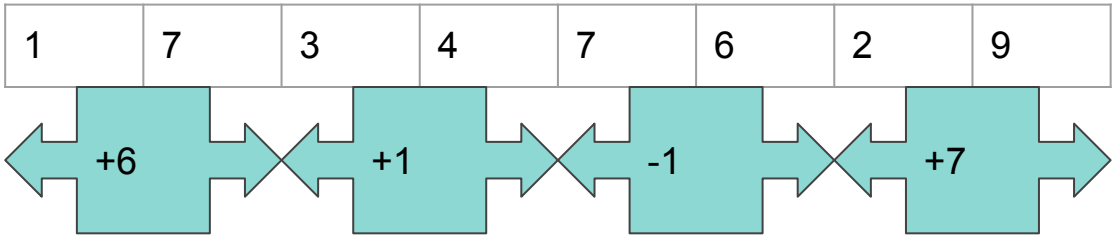


score = 1 + 3 + 7 + 2 = 13

score = 1 + 7 + 3 + 2 = 13

Reduction - Case 1

Start from an even index, end before an even index. We must maximize the subarray sum along the new array (6, 1, -1, 7).

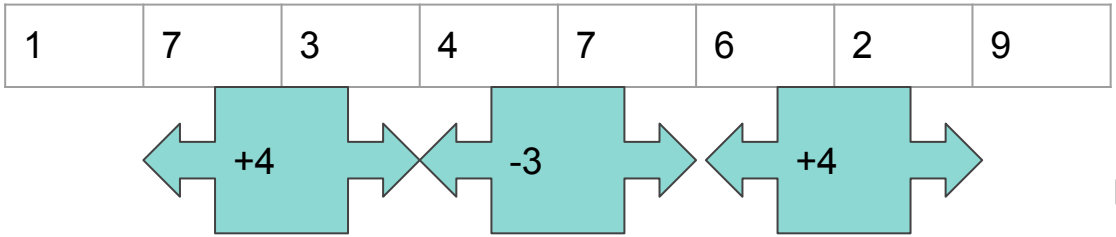


score = 1 + 3 + 7 + 2 = 13

Best score 26

Reduction - Case 2

Start from an odd index, end before an odd index. We must maximize the subarray sum along the new array (4, -3 , 4).



score = 1 + 3 + 7 + 2 = 13

Best score 18

https://github.com/cpcosu/Weekly_Events/blob/master/Autumn%202020/2020-09-22/MaxSumEvenPositions.cpp

A hard problem: Inversion Sum

https://atcoder.jp/contests/agc030/tasks/agc030_d

Given a sequence and a list of Q swaps, there are 2^Q ways to perform swaps. What's the sum of the inversion numbers of all 2^Q final sequences (mod 10^9 +7)?

A: 10, 20, 30, Swaps: (1,2) then (1,3)

	Nothing	(1,2)
Nothing	10, 20, 30 -> 0	20, 10, 30 -> 1
(1,3)	30, 20, 10 -> 3	30, 10, 20 -> 2

Total inversion sum: 6

Given a sequence and a list of Q swaps, there are 2^Q ways to perform swaps. What's the sum of the inversion numbers of all 2^Q final sequences (mod 10^9 +7)?

- Just simulate all 2^Q pairs of swaps, then check all pairs at the end to see if they are inverted.
- Time $\Theta(2^Q(n^2+Q))$

DP solution

Given a sequence and a list of Q swaps, there are 2^Q ways to perform swaps. What's the sum of the inversion numbers of all 2^Q final sequences (mod 10^9 +7)?

- Create tables where the i^{th} row and j^{th} column in is the number of sequences where $a_i < a_j$ after the first k operations.
- We need a way to construct each table from the last one, this seems plausible.

0	1	0
0	0	0
1	1	0

$2, 3, 1$

Swap(1,3)

0	0	1
0	0	2
1	2	0

$2, 3, 1 \quad 1, 3, 2$

Swap(1,2)

0	2	1
2	0	1
3	3	0

$2, 3, 1 \quad 1, 3, 2$
$3, 2, 1 \quad 3, 1, 2$

Recurrence Relation

- DP[k][i][j] - number of sequences where the i and jth elements are swapped after the first k operations.
- Suppose the kth operation swaps indices x < y

$$\Theta(Q * n^2)$$

Equal to the size of the DP array, so there's no way we can go faster

```
for (int i = 0; i < n; i++)

      for (int j = 0; j < n; j++)

            DP[k][i][j] = DP[k-1][i][j]*2

DP[k][x][y] = DP[k][y][x] = DP[k-1][x][y] + DP[k-1][y][x];

for (int j = 0; j < n; j++) {

   if (j != x && j != y) {

       DP[k][j][x] = DP[k][j][y] = DP[k-1][j][x] + DP[k-1][j][y];

       DP[k][x][j] = DP[k][y][j] = DP[k-1][x][j] + DP[k-1][y][j];

   }

}
```

We need to go faster

In the original problem statement, we have

$$1 \leq N \leq 3000$$
$$0 \leq Q \leq 3000$$

Has DP failed us??

Inspiration!

If we instead compute the expected number of inversions, then we are only updating about 2n terms in the array at each update.

We can compress the first dimension of our code since we don't really use it.

https://github.com/cpcosu/Weekly_Events/blob/master/Autumn%202020/2020-09-22/InversionSum.cpp

$$\Theta(nQ)$$

Weekly Challenge

https://codeforces.com/contest/1418/problem/C