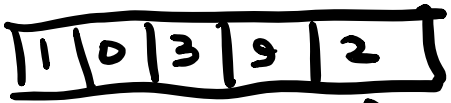
RANDOMLY REORDER ARRAY in O(N)

[1, 0, 3, 9, 2]

[3, 1, 9, 0, 2]



start here (i=4)

move backward through indexes

choose random num 0-3 and SWAP w/ value at that index

then i = i - 1

LONGEST CONSECUTIVE CHARACTER

YouTube CS Dojo

O(N)

"AABCDDBBBEA" → {'B': 3}

(at i=2)

current 'B'

max_count 2

max_char 'A'

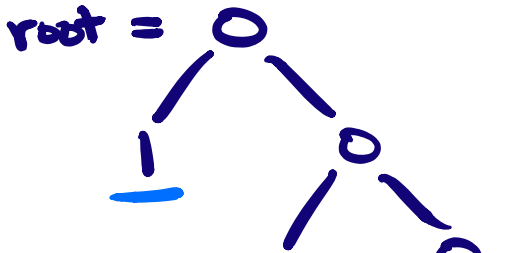
prev_char 'A'

count 1

UNIVERSAL VALUE TREE PROBLEM

Google interview YouTube

O(N)



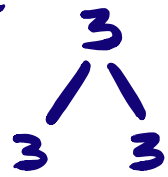
given: a tree

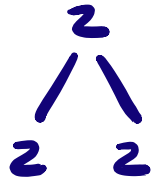
return # of non-empty universal value tree (unival tree)

Should return 5 for this tree

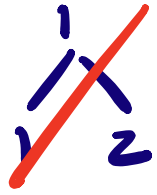
0

ex:





(empty)



NOT unival tree

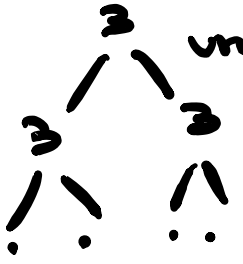
- when we hit a leaf node, this is by definition a unival tree
- if root.left != null and root.left.value != root.value return False
- if root.right != null and root.right.value != root.value return True
- if is_unival(root.left) and is_unival(root.right): return True
- return False

2 EXTRA CHALLENGES

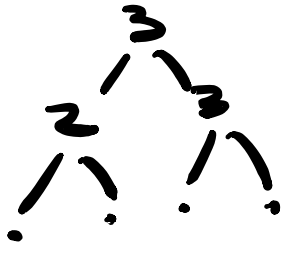
```
class Node {
    int value
    Node left
    Node right
}
```

what if Node has up to 5 child?

```
class Node {
    int value
    Node child 1
    :
    Node child 5
}
```



unival tree



semi-unival tree (only 2 values)

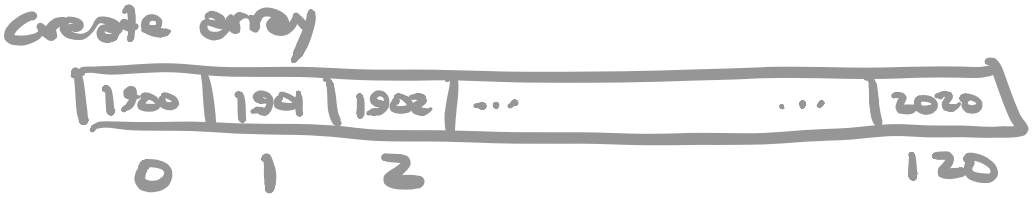
what if you want to count unival AND semi-unival?

YEAR WITH MAX PEOPLE ALIVE

given: group of people with birth year and death

year

find: year with max people alive



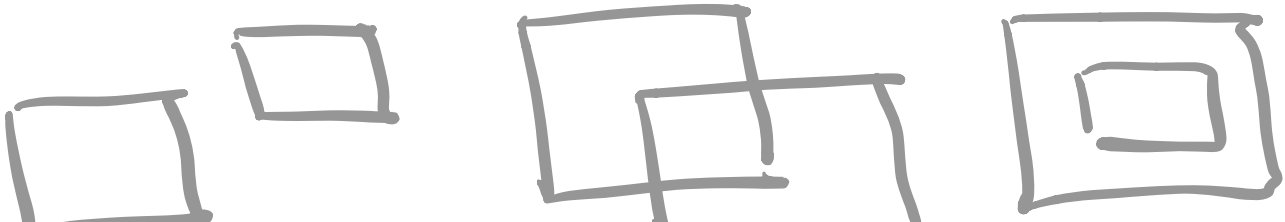
- for each person, iterate birth year to death year and add 1 to value in array
- find max value in array

* Turns out that we only need to look at years where a birth or death occurred.

AND we can ignore duplicates? Is this right? I guess if we hash the year and store the totals there...

OVERLAPPING RECTANGLES

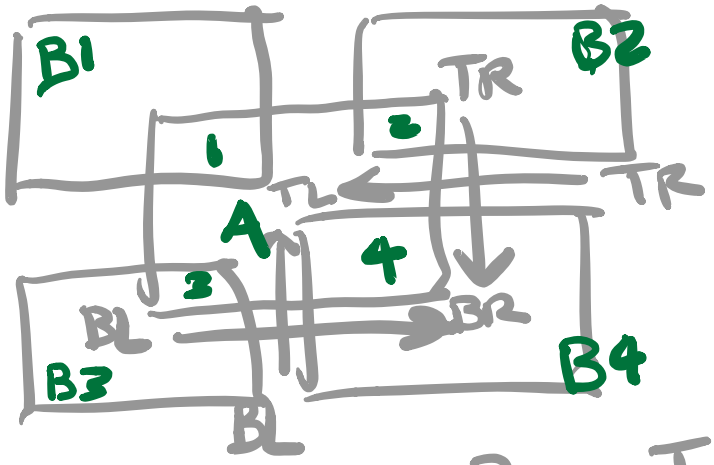
You're given 2 over-lapping rectangles on a plane. For each rectangle, you're given the bottom-left and top-right points. Find the area of their overlap.



no overlap

partial overlap

contained



rect A (L: $ax1$, B: $ay2$, R: $ax2$, T: $ay1$)

rect B ($bx1$, $by2$, $bx2$, $by1$)

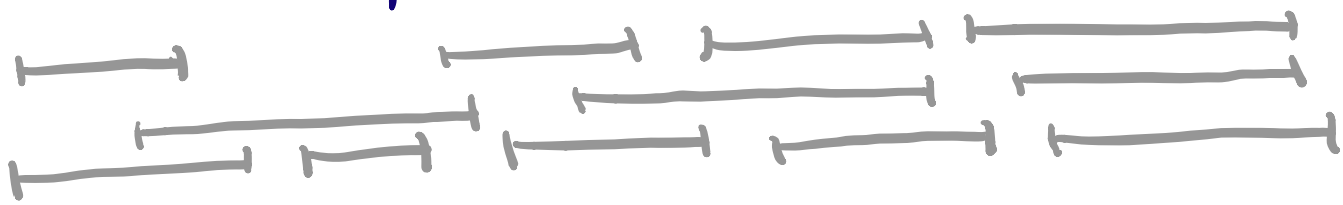
TL $(ax1, ay1)$ - BR $(bx2, by2)$ BL $(bx1, by2)$ - TR $(ax2, ay1)$

BL $(ax1, ay2)$ - TR $(bx2, by1)$ TL $(bx1, by1)$ - BR $(ax2, ay2)$

MOVIE SCHEDULING

Given start/end times of filming for various movies, choose set of movies which allow

for maximum made movies without overlapping.
(your fee is same for each movie)



0 days ⟶ increasing integer days

start end

[(0, 2), (0, 4), (1, 7), (2, 6), (8, 12), (9, 15)]

movie: 0 1 2 3 4 5

accept movie J with earliest completion date.
remove J and any movies that intersect the filming of J.
repeat.

SUM TWO LINKED-LIST NUMBERS

Given two numbers represented as linked lists, output sum as linked list.

5 → 6 → 3 // 365

8 → 4 → 2 // 248

output: 3 → 1 → 6 // 613

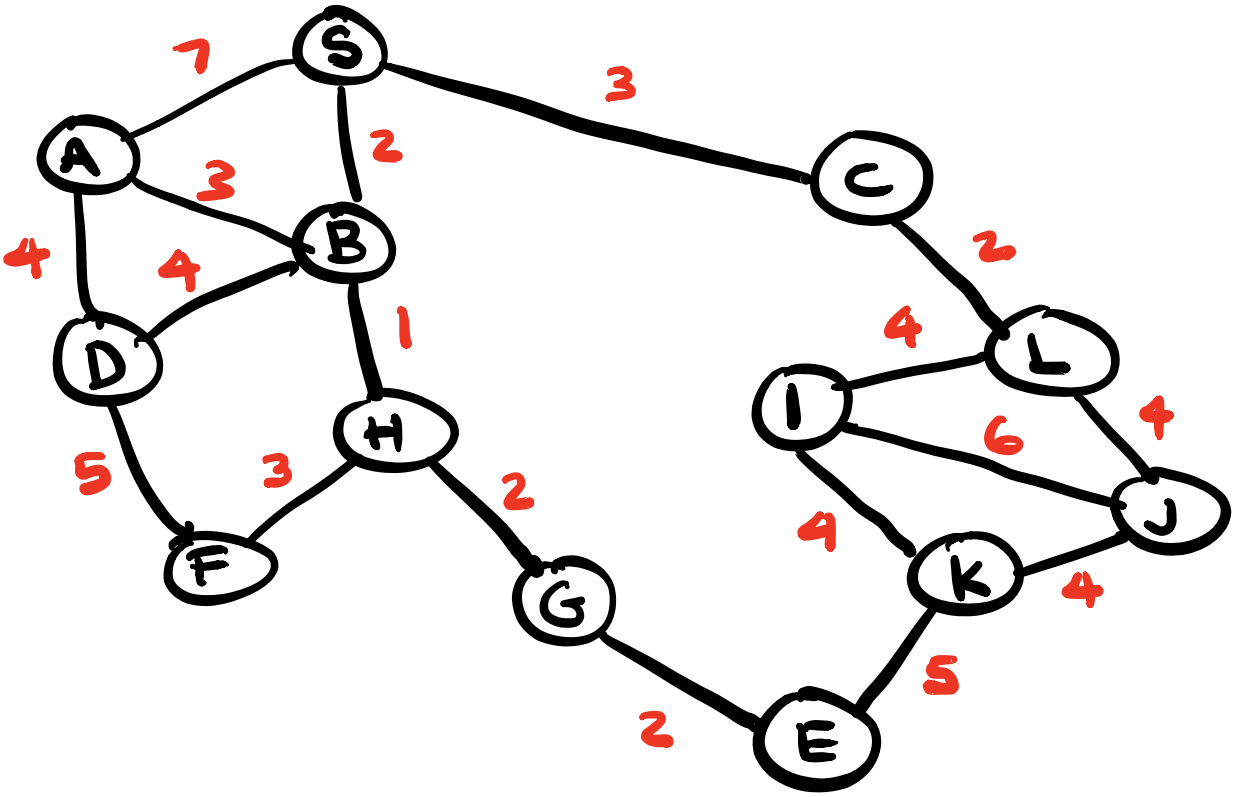
7 → 5 → 9 → 4 → 6 // 64957

8 → 4 // 48

output: 5 → 0 → 0 → 5 → 6 // 65005

DIJKSTRA'S ALGORITHM
(shortest path)

A* is extension of Dijkstra's



Find shortest path from S to E.

then remove S

	distance	
S	0	
A	~~∞~~	0+7=7
B	~~∞~~	0+2=2
C	~~∞~~	0+3=3
K	∞	
E	∞	

PRIORITY QUEUE

swap in priority queue

B dist: 2 path via: S

answer: S - B - H - G - E

MODIFICATION: look at all the fast roads

THAT HEAD IN APPROX THE RIGHT DIR.

A* — builds in heuristic that says "we're getting a bit closer"

example: add Euclidian distance to each node (from node to destination)
Order priority queue by weight + distance (instead of just weight).

combined heuristic