INSERTION SORT

SELCTION SORT ALGORITHM

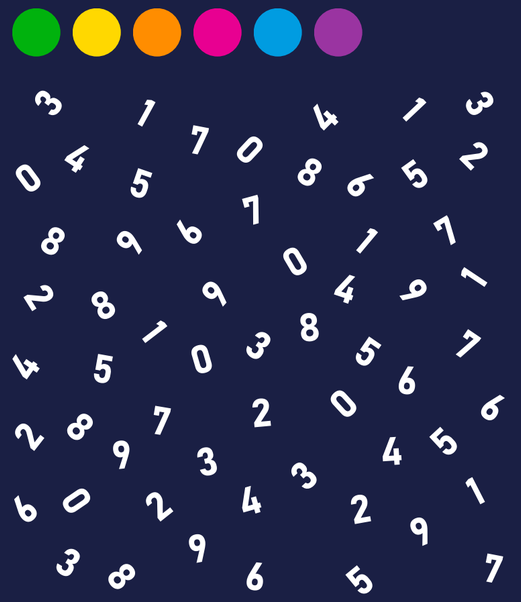
Last week we learned Selection Sort Algorithm.
It works by repeatedly finding the smallest element
and moving it to its correct position in the list

INSERTION SORT ALGORITHM

Today, we will be learning the Insertion Sort algorithm. It works by picking each element one by one and inserting it into its correct position in the already sorted part of the list. This process continues until the entire list is sorted.

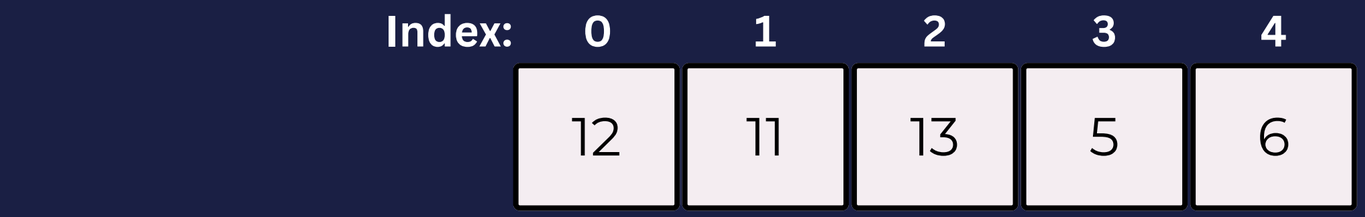
Imagine you're organizing a deck of playing cards in your hand.

You pick up one card at a time and insert it into the correct position in the already sorted part of your hand.

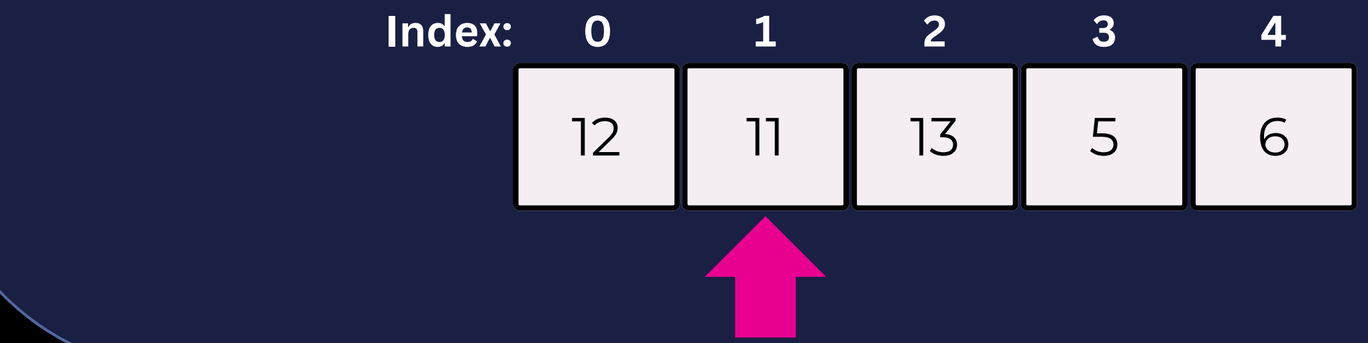


But Instead of Cards…
Let’s use a sequence of numbers instead!
We’ll organize them from smallest to largest by repeatedly picking a number and inserting it into its correct position

Lets consider an array of integer values:

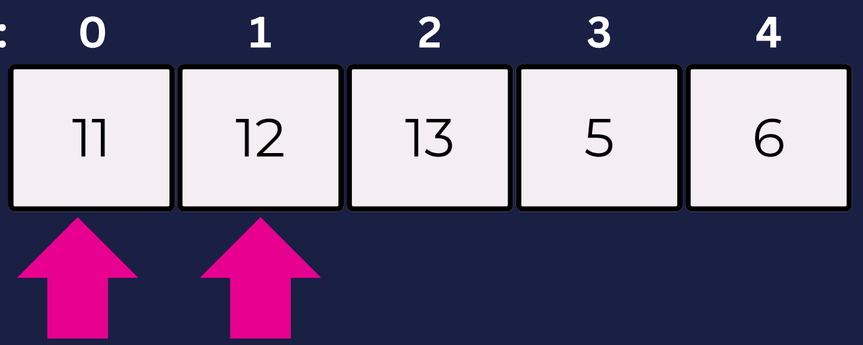


First we will start by picking a number

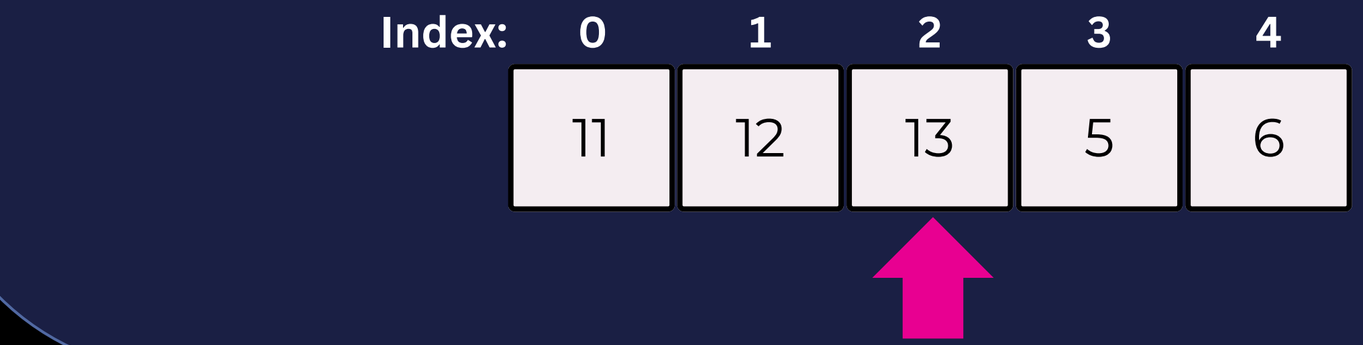


Next we will...

Compare with 12 → Move 12 → Insert 11

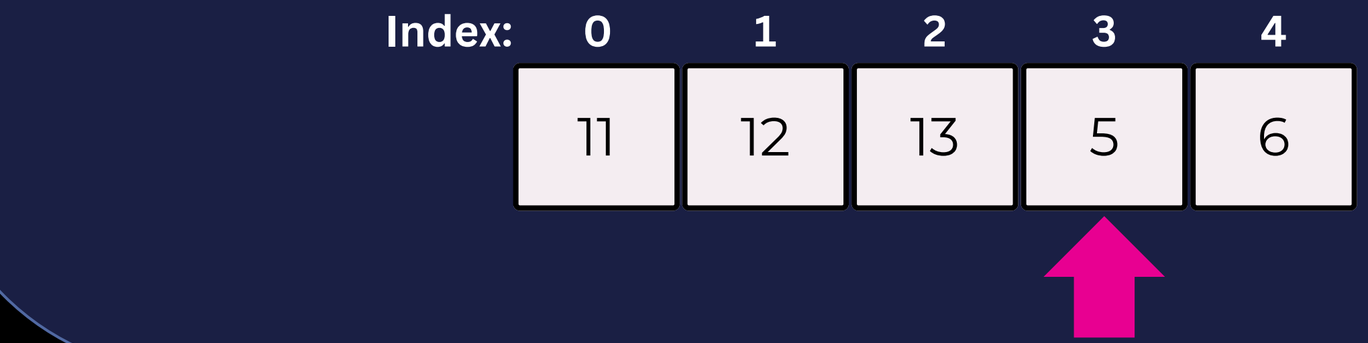


Next we will choose another number
13 is in the correct place move on

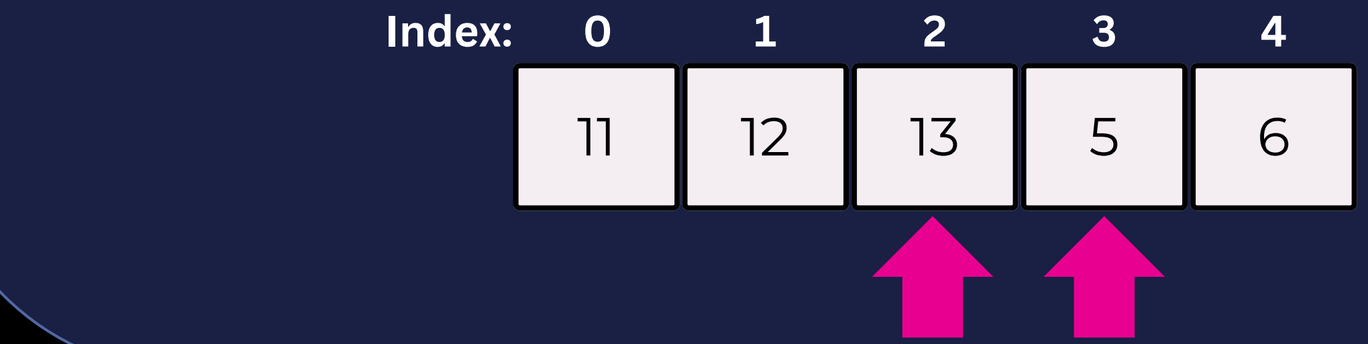


Next we will choose another number

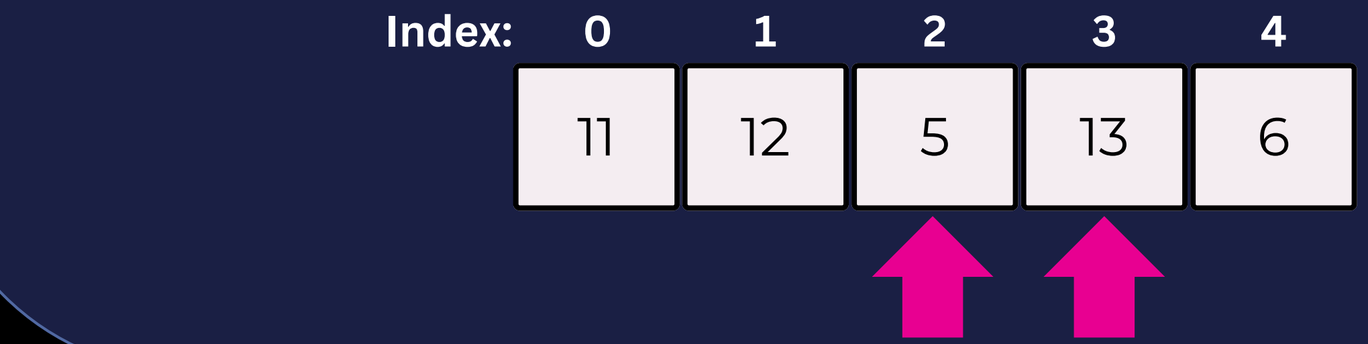
Let’s pick 5



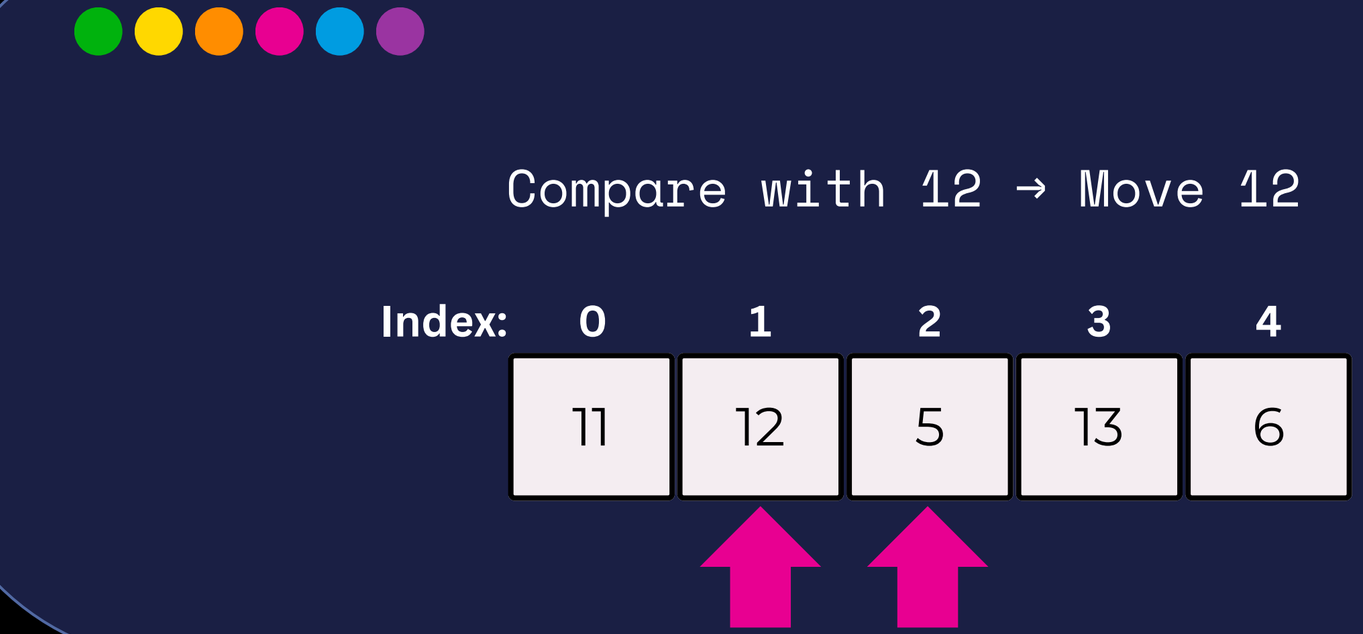
Compare with 13 → Move 13



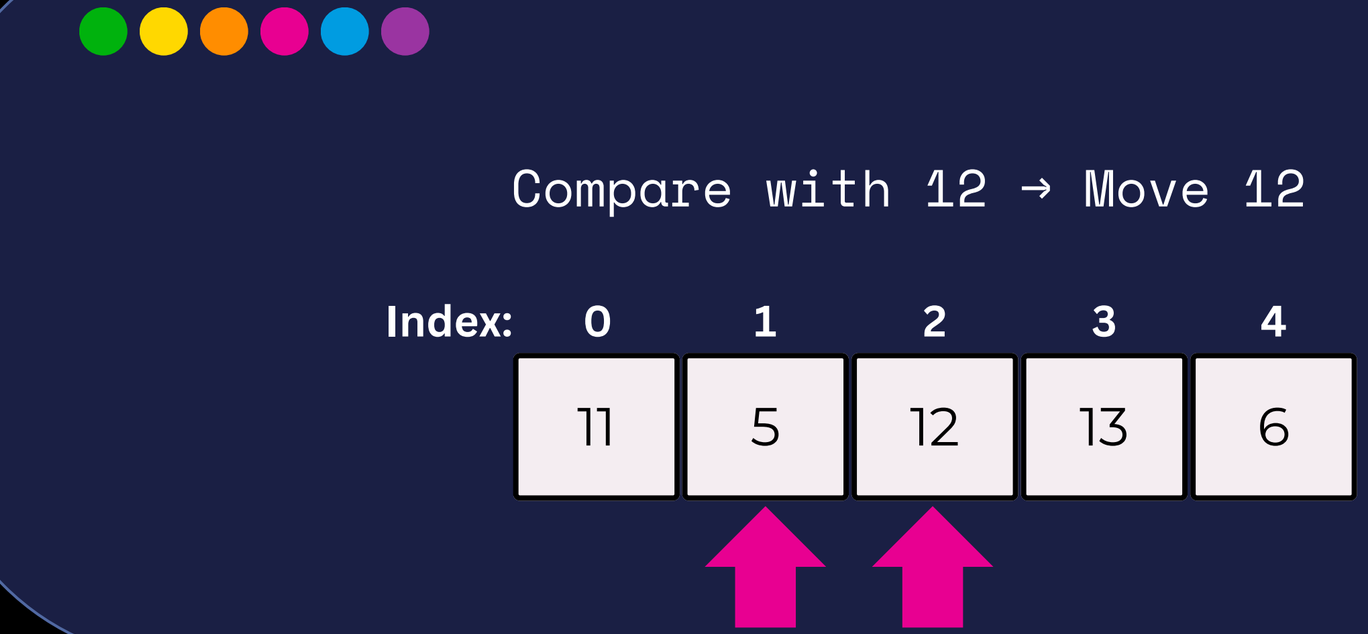
Compare with 13 → Move 13



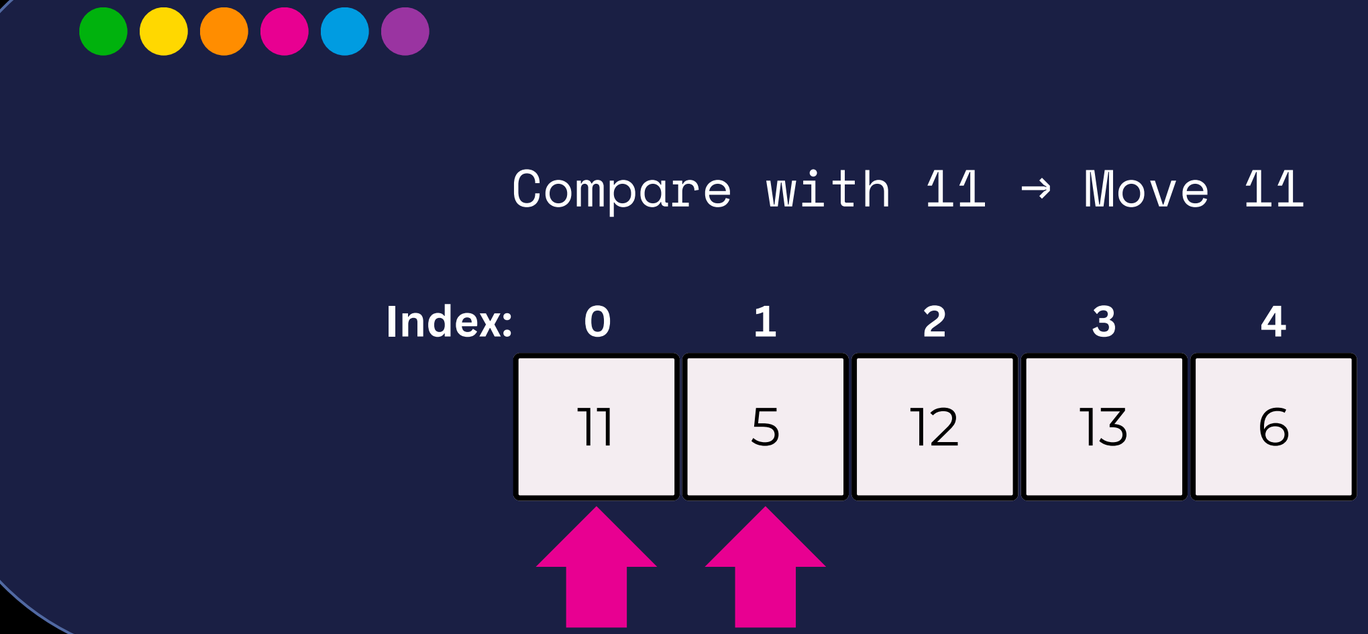
Compare with 12 → Move 12



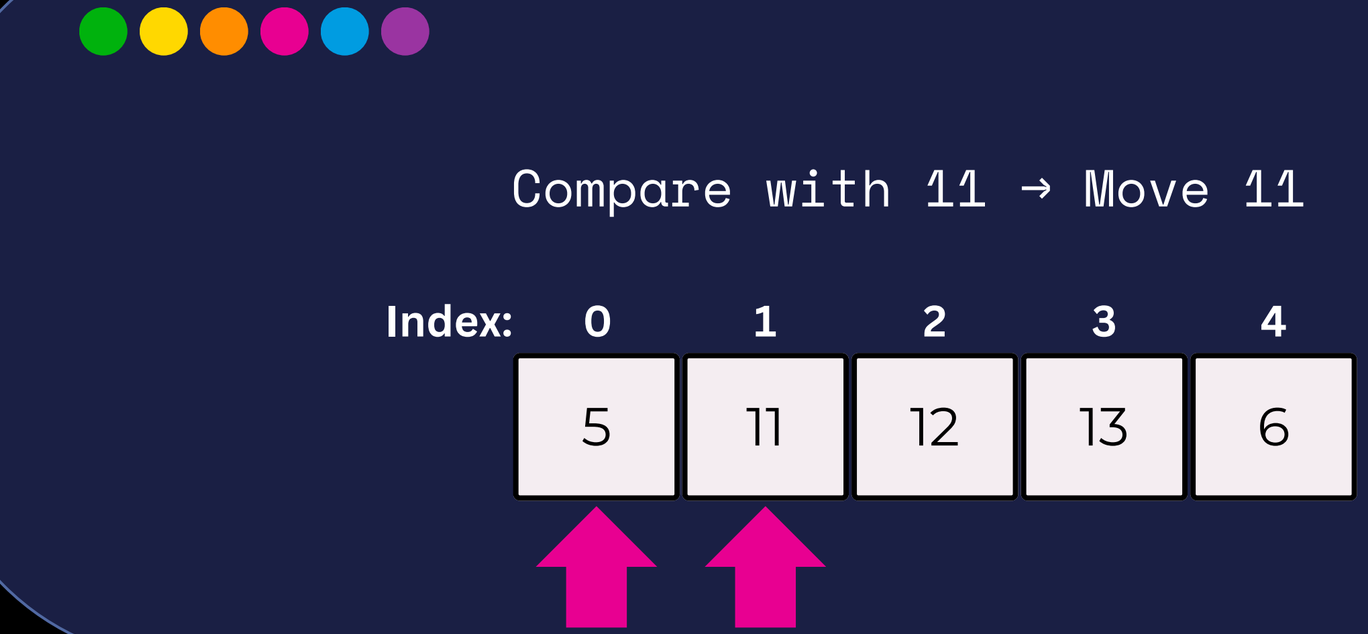
Compare with 12 → Move 12



Compare with 11 → Move 11



Compare with 11 → Move 11



REPEAT THE PROCESS

We would select the last number (6) in our case and repeat the process of comparing that number against the other sorted numbers and moving them as needed.

TIME COMPLEXITY

- Best case (already sorted list): O(n) (just one comparison per element)
- Worst case (reverse order): $O(n^2)$ (shifts every element)
- Average case: $O(n^2)$ (still quadratic in nature)

WHEN SHOULD YOU USE INSERTION SORT?

✅ Great for small datasets
✅ Efficient when the array is nearly sorted
✅ Stable sorting algorithm (preserves order of equal elements)
❌ Not efficient for large datasets (better use QuickSort or MergeSort

THE PSEUDOCODE

```
array = [12, 11, 13, 5, 6]

FOR i from 1 to length of array - 1
   key = array[i]
   j = i - 1

WHILE j >= 0 AND array[j] > key
 array[j + 1] = array[j]
 j = j - 1
END WHILE

array[j + 1] = key
```

EXAMPLES REPLIT! PLEASE GO TO:

HTTPS://REPLIT.COM/@RIKKIEHRHART/GRABABYTE

UP NEXT

Feb 26 - Insertion Sort
Mar 5 - Merge Sort
Mar 12 - Quick Sort
SPRING BREAK!
Mar 26 - Breadth-First Search (BFS)
Apr 2 - Depth-First Search (DFS)
Apr 9 Hashing
Apr 16 - Dijkstra's Algorithm
Apr 23 - Dynamic Programming (Knapsack Problem)
Apr 30 - Union-Find
May 7 - Kruskal's Algorithm
May 14 - Prim's Algorithm

Questions? - rikki.ehrhart@g.ausitncc.edu

If you'd like the opportunity to run a Grab a Byte algorithm workshop, please let me know!