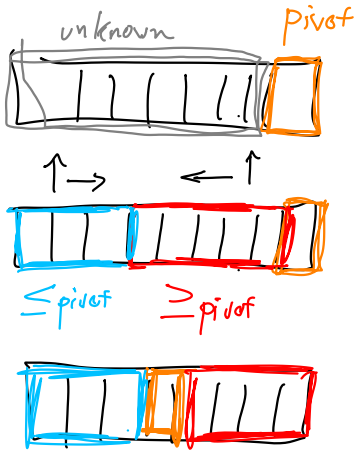
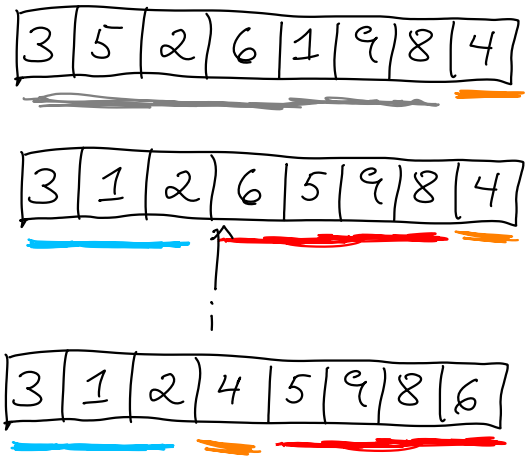
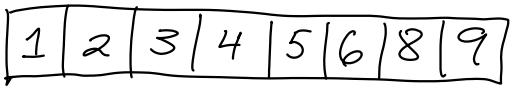
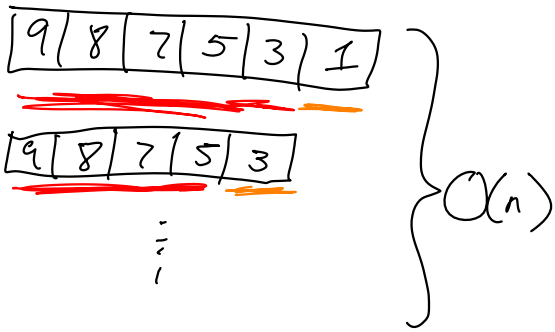
Quick Sort








```
Function quickSort(A, start, end)
    size ← end - start + 1
    If size > 1:
        pivotIdx ← partition(A, start, end)
        quickSort(A, start, pivotIdx - 1)
        quickSort(A, pivotIdx + 1, end)
    EndIf
EndFunction
```



Quick Sort worst-case: $O(n^2)$

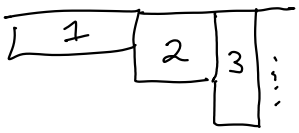
Imagine:

"Gambler's Sort"

```
Flip a coin
If coin is heads:
    array is magically sorted
Else:
    Repeat
```

worst-case: $O(\infty)$

expected: $O(1)$



```
Function RandomizedQuickSort(A, start, end):
    swap a randomly-chosen element in A with A[end]
    do what Quick Sort does
```

expected: $O(n \log n)$

Lists

Arrays

- put things in (at index)
- get things (from index)

Lists

- ask size
- add things (changes size)
- remove things (changes size)
- get things