SEARCH AND SORT

Marija Venta

CMSC 461

Spring 2020 (YAY!)

OUTLINE

- WHY SEARCH AND SORT
- SEARCHING
  - Linear
  - Binary
- SORTING
  - Selection
  - Insertion
  - Merge
  - Quick

WHY SEARCH AND SORT?

- Search: find an item or a group from a collection
  - Used by search engines, AI algorithms

- ordering a list of objects (numerical, lexicographical)
  - helps search increase it efficiency
  - {1,4,3,13} -> {1,3,4,13}

SEARCHING

LINEAR

- Searching for x
- looking at each value in turn
- It quits and returns true if the current value is x
- it quits and returns false if it has looked at all of the values in the array without finding x

O(n)

UNSORTED LIST ALGORITHM

```
bool linearSearch(vector<int> nums, int x) {
bool found = false;
    for (int i = 0; i < nums.size(); i++) {
        if (nums[i] == x)
       found = true;
}
    return found;
}
```

SORTED LIST ALGORITHM

```
bool linearSearch(vector<int> nums, int x) {
bool found = false;
   for (int i = 0; i < nums.size(); i++) {
      if (nums[i] == x)
       return = true;
If(nums[x] > x)
       return false;
       }
```

O(n)

BINARY

- starts by looking at the middle item n
- If n is equal to x, it quits and returns true
- Otherwise, eliminates half of the array
- algorithm repeats on the remaining half until an element is found and returns true or returns false
- The call is reduced by a factor of two every time:

O(lg(n))

```python
def bisect_search2(L, e):
    def bisect_search_helper(L, e, low, high):
        if high == low:
            return L[low] == e
        mid = (low + high)//2
        if L[mid] == e:
            return True
        elif L[mid] > e:
            if low == mid: #nothing left to search
                return False
            else:
                return bisect_search_helper(L, e, low, mid - 1)
        else:
            return bisect_search_helper(L, e, mid + 1, high)
    if len(L) == 0:
        return False
    else:
        return bisect_search_helper(L, e, 0, len(L) - 1)
```

Source: https://www.youtube.com/watch?v=6LOwPhPDwVc&t=2560s

```python
def bisect_search2(L, e):
    def bisect_search_helper(L, e, low, high):
        if high == low:
            return L[low] == e
        mid = (low + high)//2
        if L[mid] == e:
            return True
        elif L[mid] > e:
            if low == mid: #nothing left to search
                return False
            else:
                return bisect_search_helper(L, e, low, mid - 1)
        else:
            return bisect_search_helper(L, e, mid + 1, high)
    if len(L) == 0:
        return False
    else:
        return bisect_search_helper(L, e, 0, len(L) - 1)
```

Source: https://www.youtube.com/watch?v=6LOwPhPDwVc&t=2560s

```python
def bisect_search2(L, e):
    def bisect_search_helper(L, e, low, high):
        if high == low:
            return L[low] == e
        mid = (low + high)//2
        if L[mid] == e:
            return True
        elif L[mid] > e:
            if low == mid: #nothing left to search
                return False
            else:
                return bisect_search_helper(L, e, low, mid - 1)
        else:
            return bisect_search_helper(L, e, mid + 1, high)
    if len(L) == 0:
        return False
    else:
        return bisect_search_helper(L, e, 0, len(L) - 1)
```

Source: https://www.youtube.com/watch?v=6LOwPhPDwVc&t=2560s

SORTING

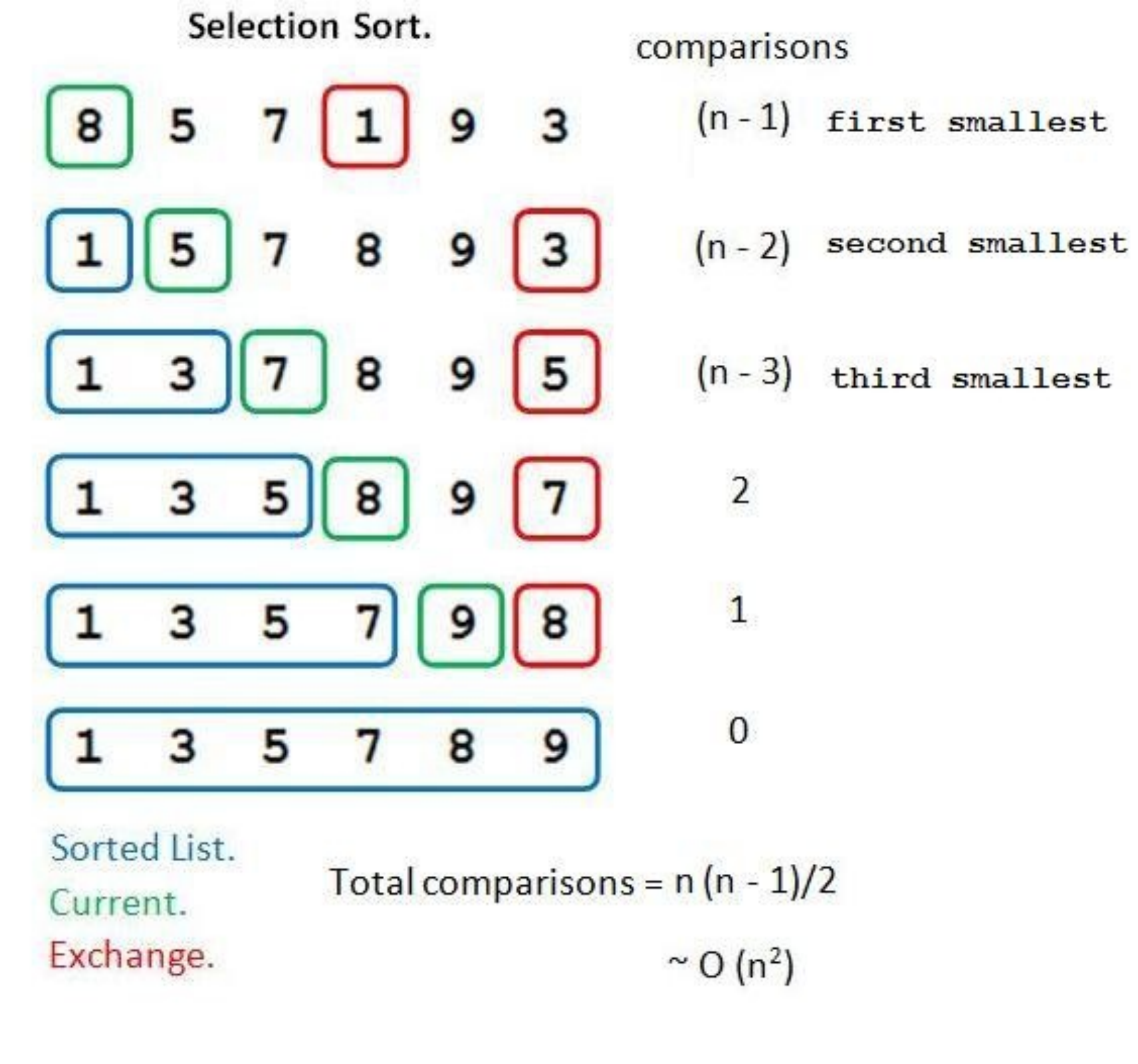
WHEN TO DO IT?

- When Big O is less than n
- IMPOSSIBLE
- Multiple searches

SELECTION

- Selecting the smallest one
- To start, find smallest element of the list
- Put it in the beginning of the list by swapping it with the element at position 0
- Find the smallest element in the remaining list
- Swap it with the element at position 1
- All elements to the left of the current element are sorted
- Keep going until list is sorted

O(n^2)



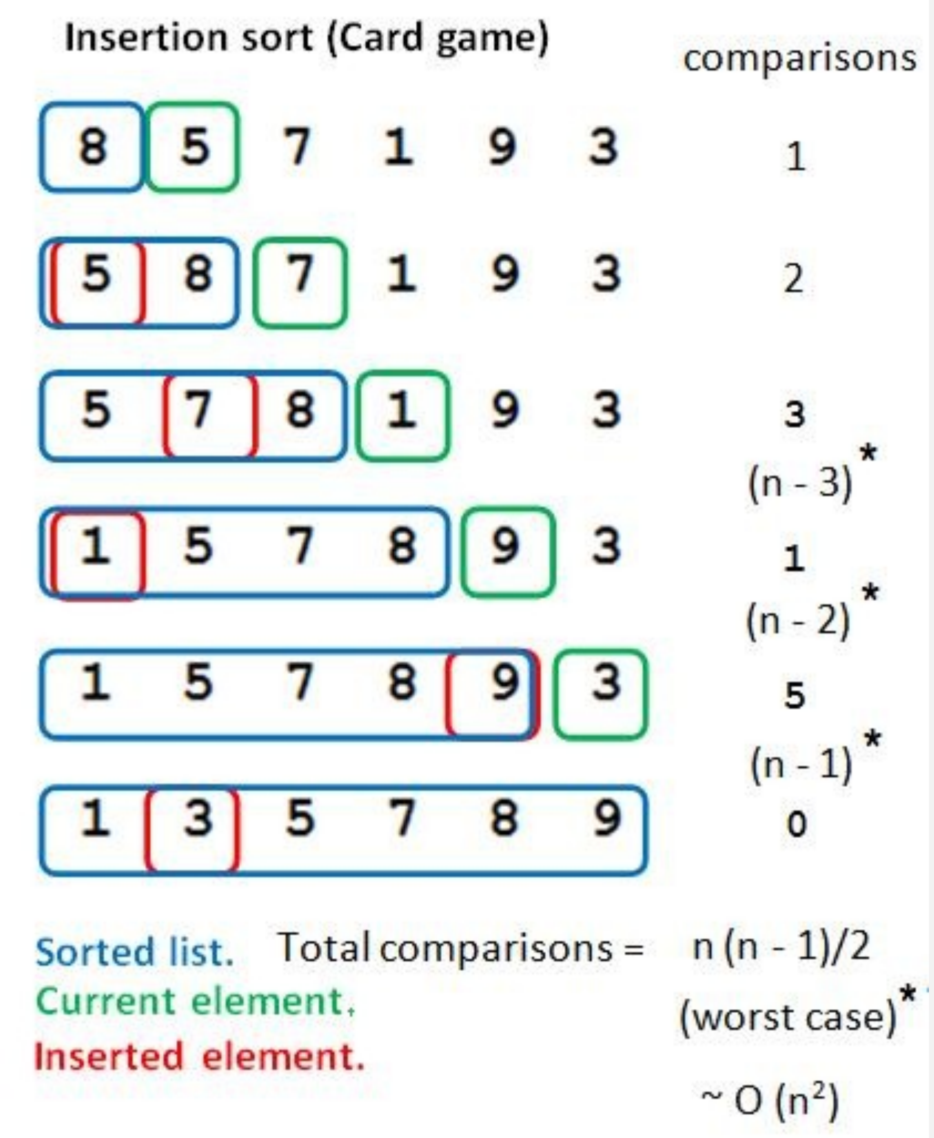
Source: https://stackoverflow.com/questions/15799034/insertion-sort-vs-selection-sort

INSERTION

- Two lists, start and empty
- Start with element at position 0
- Insert into the new list
- Insert element into the correct position in the list

O(n²)

For new insertions:O(n)



Source: https://stackoverflow.com/questions/15799034/insertion-sort-vs-selection-sort

MERGE

- List of 1 or 0 elements are sorted by definition
- Split list in half, until subset of 1 element is remaining
- Merges them back up

split

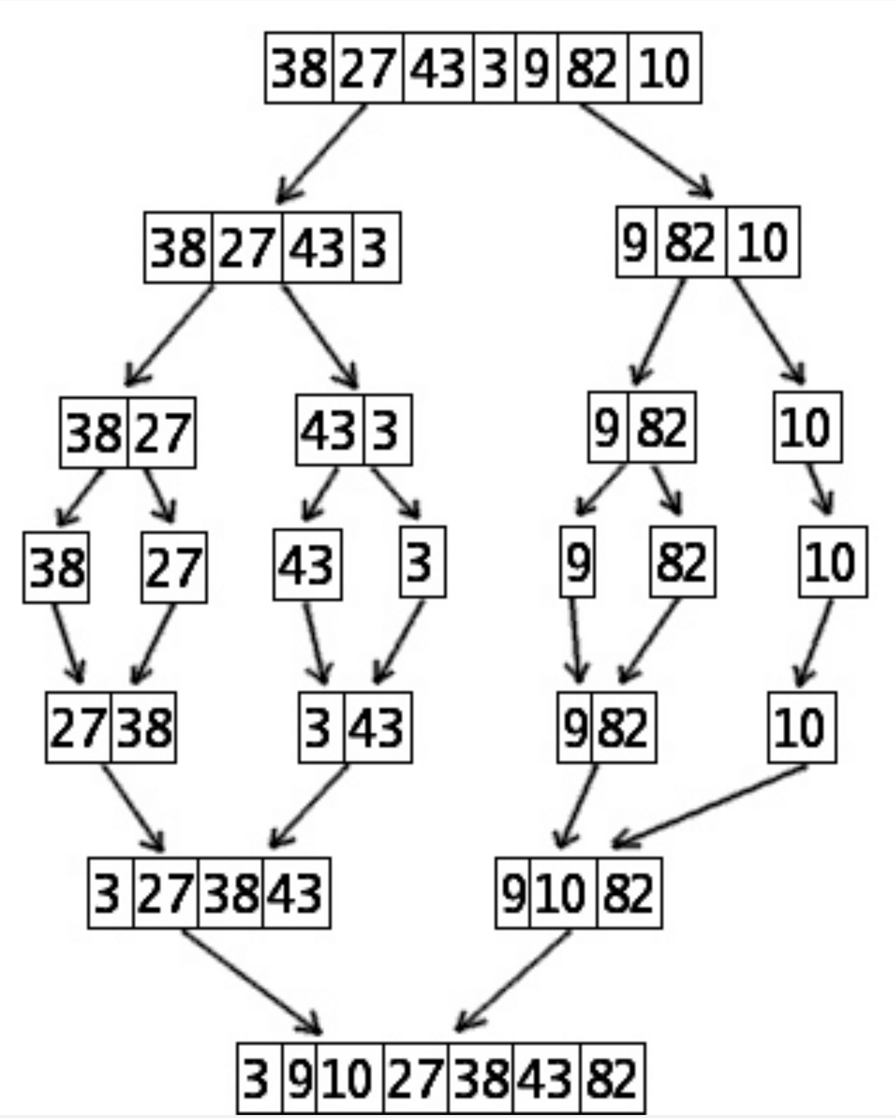
split

split

merge

merge

merge



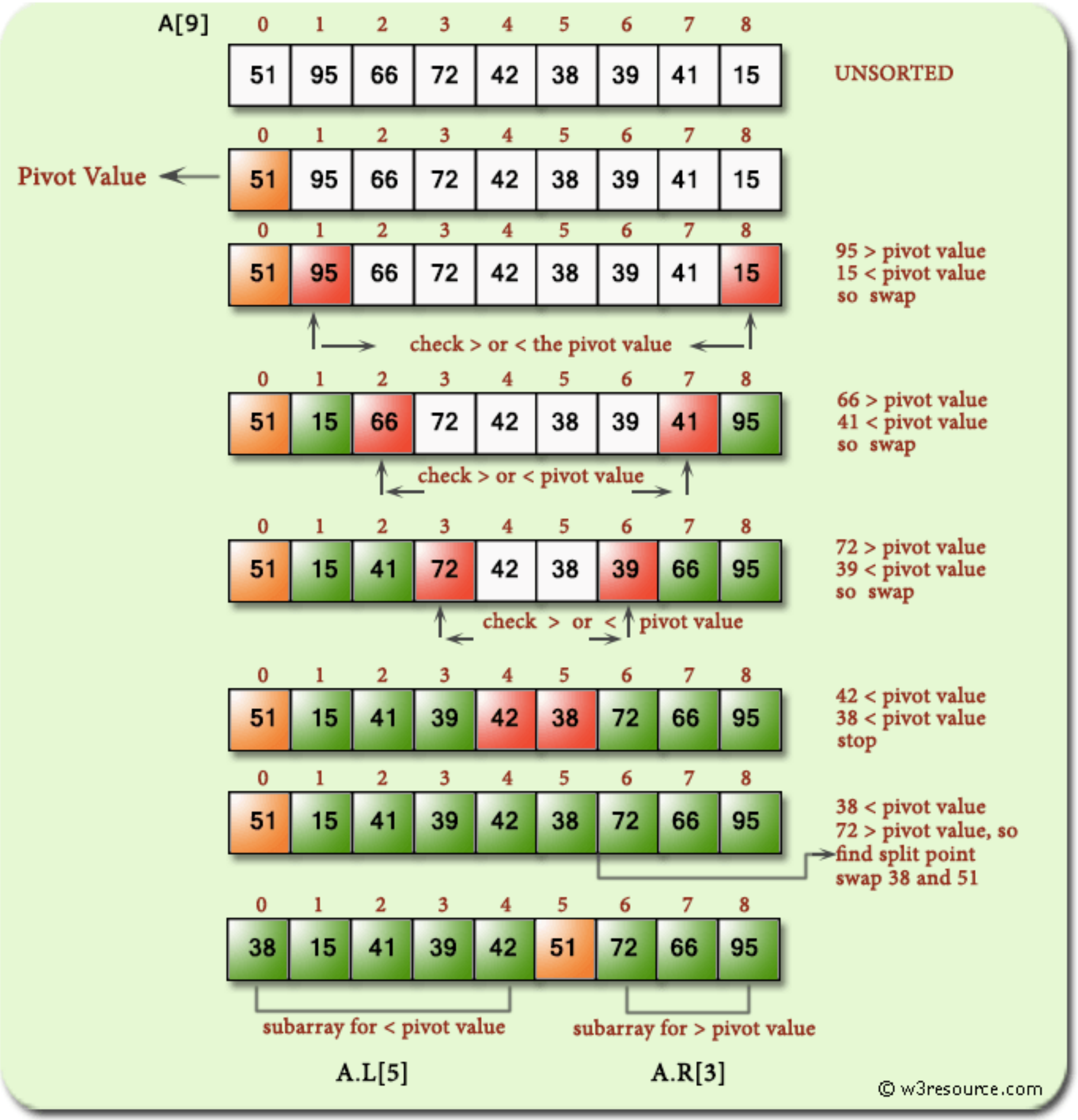
O(n lg(n))

Source: https://commons.wikimedia.org/wiki/File:Mergesort_algorithm_diagram.png

QUICKSORT

- PIVOT
  - Items to the left are smaller
  - Items to the right are bigger
- Sort the items on the left and right

Quick Sort



Average:
O(n lg(n))

Worst case:
$O(n^2)$

Source: https://www.w3resource.com/csharp-exercises/searching-and-sorting-algorithm/searching-and-sorting-algorithm-exercise-9.php

ACKNOWLEDGMENTS

- My family and friends
- All the Longwood Computer Science faculty
- Especially Dr. Julian Dymacek (160, 162)