CS 30 Discussion 1A

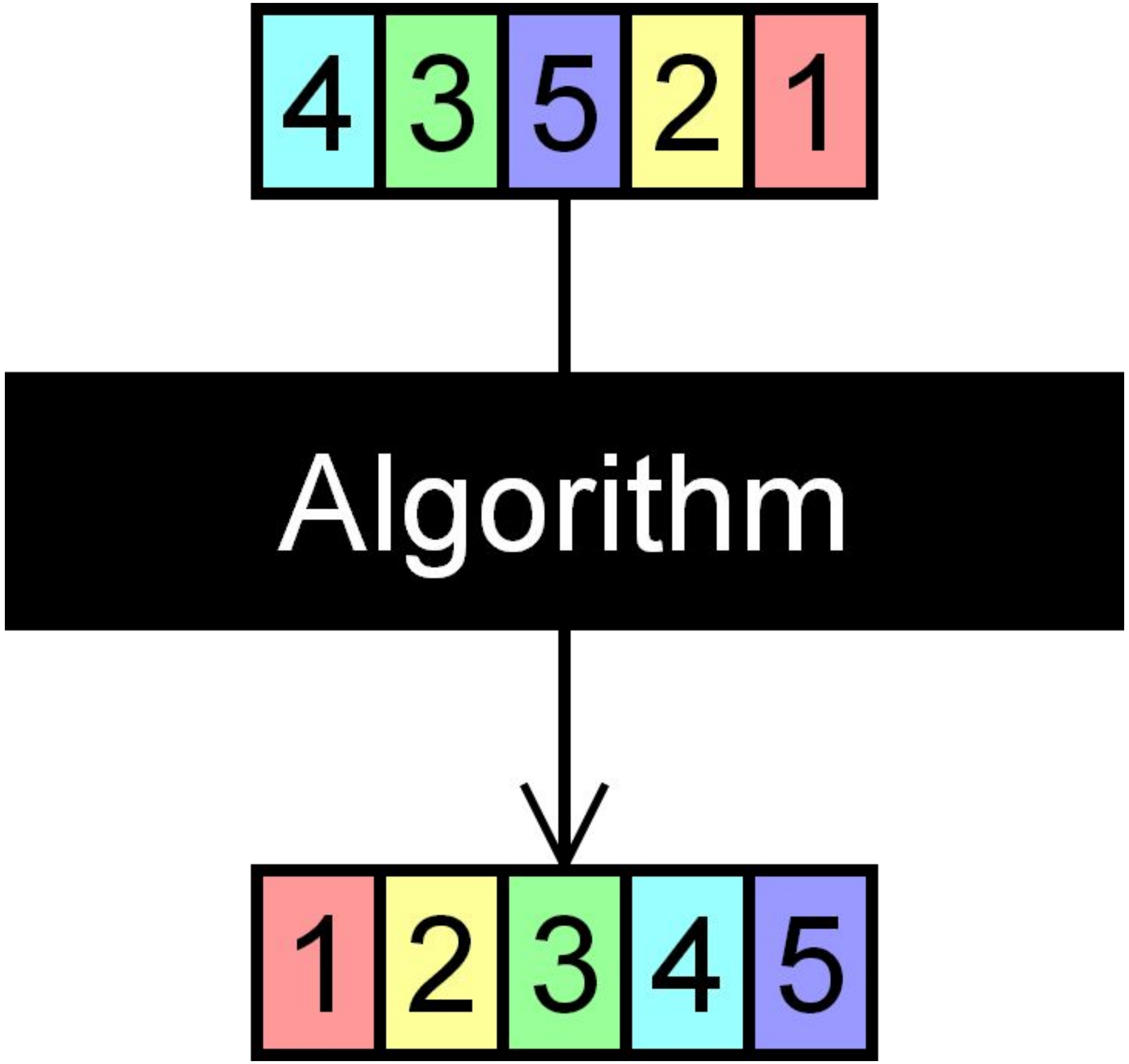
2020.10.30

UCLA Samueli Computer Science

Welcome back to CS30 Discussion!

- HW3 has been posted, dues Thursday, November 5, at 11:30pm.
- Mid-term Grading.

Sorting algorithm



Sorting algorithm

A sorting algorithm will put items in a list into an order, such as alphabetical or numerical order. You can decide either increasing or decreasing order.

For example, a list of customer names could be sorted into alphabetical order by surname, or a list of people could be put into numerical order by age.

Sorting algorithm

Sorting a list of items can take a long time, especially if it is a large list.

A computer program can be created to do this, making sorting a list of data much easier.

e.g. some_sorting_algorithm([4, 3, 5, 2, 1]) ➜ [1, 2, 3, 4, 5]

some_sorting_algorithm?

1. Selection Sort

Steps:

4	3	1	5	2
1	3	4	5	2
1	2	4	5	3
1	2	3	5	4
1	2	3	4	5

Idea: Find the smallest item in the list and place it in the front.

Recursive Thinking:

Find Minimum of the list: 1
Remove the minimum from the list: [4, 3, 5, 2]
Sort the removed list: [2, 3, 4, 5]
Append minimum to the head of the list: [1, 2, 3, 4, 5]

Implementation

```
def selectionSort(lst):
    if len(lst) <= 1:
        return lst
    else:
        minimum = minlist(lst)
        removed = removeSmallest(lst)
        return [minimum] + selectionSort(removed)
```

Implementation

```
def selectionSort(l):
    if len(l) <= 1:
        return l
    else:
        minimum = minlist(l)
        removed = removeSmallest(l)
        return [minimum] + selectionSort(removed)
```

```
call selectionSort([1, 5, 2])
input1 is [1, 5, 2]
  minimum : 1
  removed : [5, 2]
  call selectionSort(removed) : ?

  input2 is [5, 2]
    minimum : 2
    removed : [5]
    selectionSort(removed) : [5] (base case)
    return [2] + [5] -> [2, 5]

  return [1] + [2, 5] -> [1, 2, 5]
```

Implementation

Step 1: To find the minimum in a list.

```
def minlist(l):
    if len(l) == 1:
        return l[0]
    else:
        head = l[0]
        tail = l[1:]
        minTail = minlist(tail)
        return head if head < minTail else minTail
```

```
if head < minTail:
    return head
else:
    return minTail
```

Implementation

Step 2: Remove the minimum from the list

```
def removeSmallest(l):
    if len(l) == 0:
        return l
    else:
        minimum = minlist(l)
        return helper(l, minimum)
```

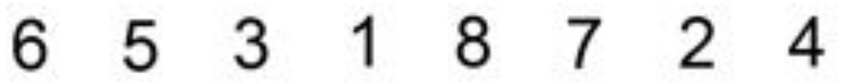
```
def helper(l, minimum):
    if l == []:
        return []
    else:
        head = l[0]
        tail = l[1:]
        if head == minimum:
            return tail
        else:
            return [head] + helper(tail, minimum)
```

```
def removeSmallest(l):
    if l == []:
        return []
    else:
        head = l[0]
        tail = l[1:]
        tail_removeSmallest = removeSmallest(tail)
        if minlist(l) == head:
            return l[1:]
        else:
            return [l[0]] + tail_removeSmallest
```

some_sorting_algorithm?

2. Insertion Sort

Steps:

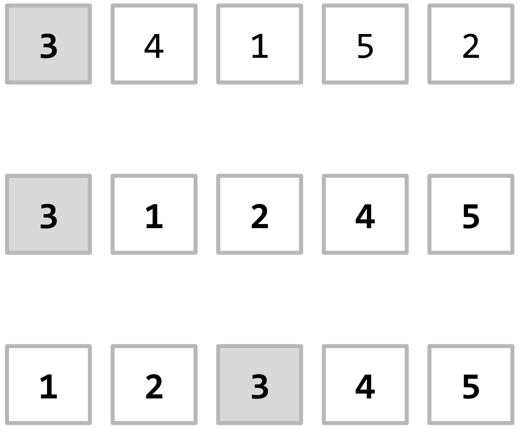


Idea: Pick one from the unsorted part and place it in the right position.

some_sorting_algorithm?

2. Insertion Sort

Steps:



3 4 1 5 2

3 1 2 4 5

1 2 3 4 5

Recursive Thinking:

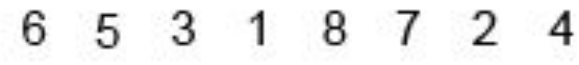
Pick the head to insert: 3
Sorted the tail: [1, 2, 4, 5]
Insert head to the correct position: [1, 2, 3, 4, 5]

some_sorting_algorithm?

3. Merge Sort

Idea: Divide and conquer

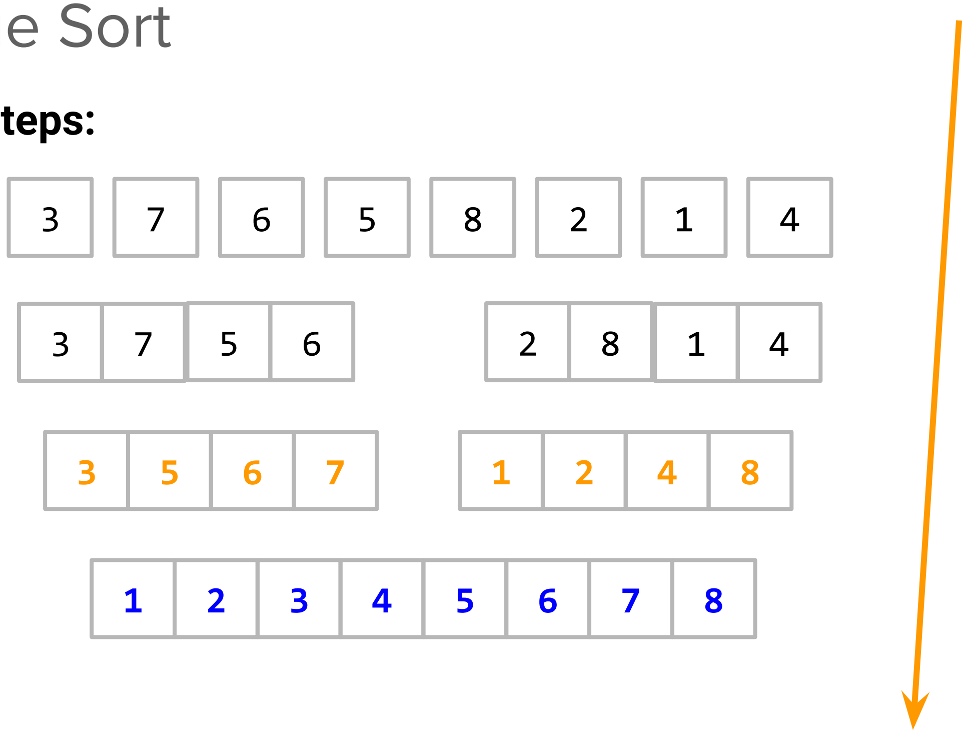
Steps:



some_sorting_algorithm?

3. Merge Sort

Steps:



3 7 6 5 8 2 1 4

3 7 5 6 2 8 1 4

3 5 6 7 1 2 4 8

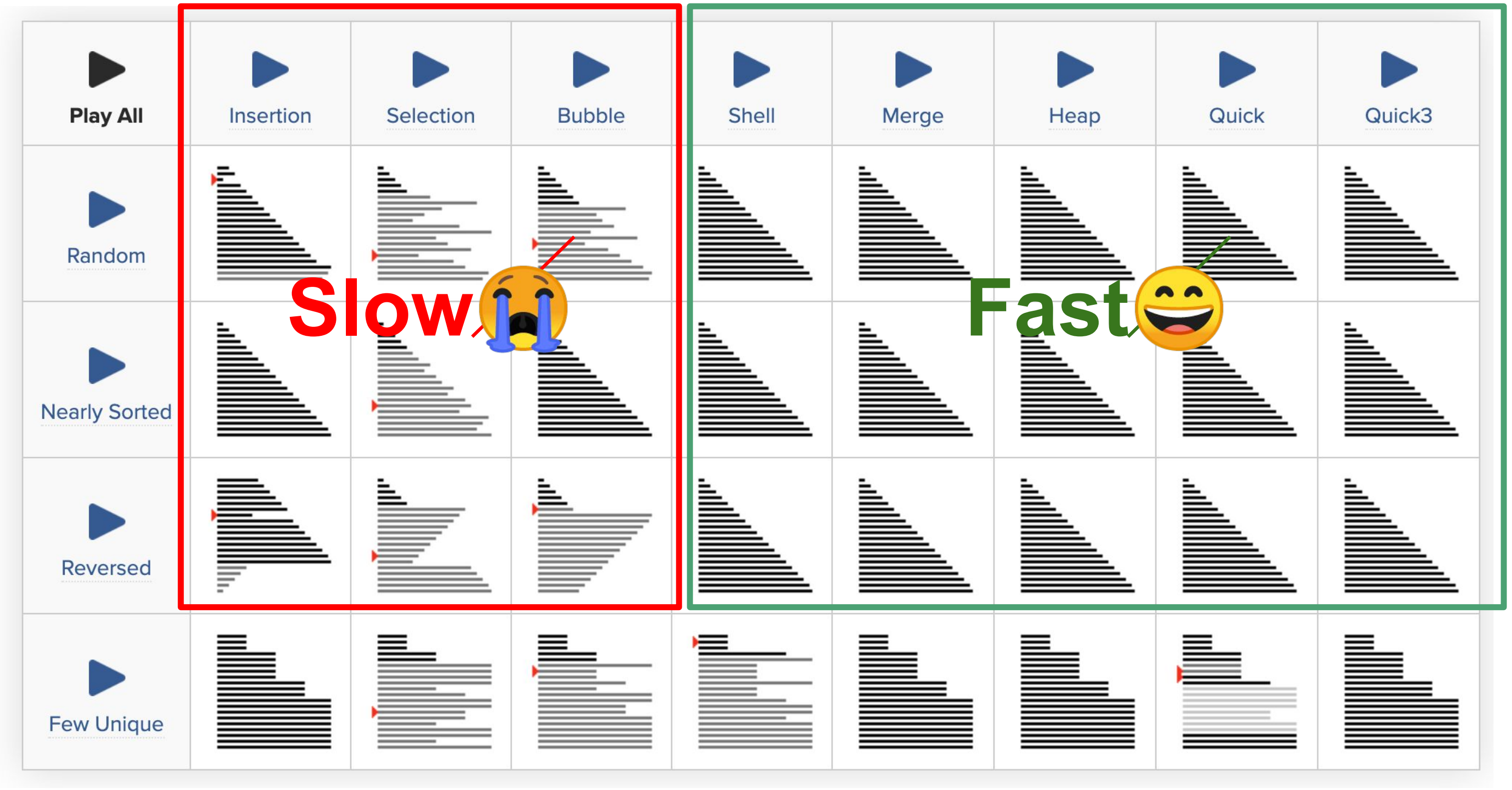
1 2 3 4 5 6 7 8

Recursive Thinking:

Split the list into two halves: [3, 7, 6, 5] , [2, 8, 1, 4]
Sort each of them: [3, 5, 6, 7] , [1, 2, 4, 8]
Merge two halfs: [1, 2, 3, 4, 5, 6, 7, 8]

Interesting Demos

1. https://www.toptal.com/developers/sorting-algorithms
2. https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html
3. http://sorting.at/



Problem set 4

Please work on Question 1, 2, 3 in groups.