

Sorts

CSC 505 Spring 2022 (001)

Nathan Vercaemert

2022-03-16

Contents

| | | |
|----------|--------------------------|----------|
| 1 | Insertion Sort | 1 |
| 1.1 | Stack Overflow | 1 |
| 1.2 | Changes | 2 |
| 1.2.1 | Note | 2 |
| 2 | Merge Sort | 2 |
| 2.1 | Changes | 3 |

1 Insertion Sort

MIT OCW Introduction to Algorithms 6.046J/18.401J LECTURE 1

“pseudocode” { INSERTION-SORT (A, n) $\triangleright A[1 \dots n]$
 for $j \leftarrow 2$ **to** n
 do $key \leftarrow A[j]$
 $i \leftarrow j - 1$
 while $i > 0$ and $A[i] > key$
 do $A[i+1] \leftarrow A[i]$
 $i \leftarrow i - 1$
 $A[i+1] = key$

1.1 Stack Overflow

<https://stackoverflow.com/questions/12755568/how-does-python-insertion-sort-work>
Title: How does Python insertion sort work?

```
def insertion_sort(seq):
    for i in range(1, len(seq)):
        j = i
        while j > 0 and seq[j - 1] > seq[j]:
            seq[j - 1], seq[j] = seq[j], seq[j - 1]
            j -= 1
```

1.2 Changes

- Variable names were changed for consistency with mergeSort.
- Operator was switched to "less than" for consistency.

1.2.1 Note

Pseudocode does not exactly match implementation, but close inspection will reveal that the execution matches pseudocode.

2 Merge Sort

MIT OCW SEARCHING AND SORTING ALGORITHMS 6.0001 LECTURE 12

```
def merge(left, right):
    result = []
    i, j = 0, 0
    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1
    while (i < len(left)):
        result.append(left[i])
        i += 1
    while (j < len(right)):
        result.append(right[j])
        j += 1
    return result
```

- left and right sublists are ordered
- move indices for sublists depending on which sublist holds next smallest element

when right sublist is empty

when left sublist is empty

```
def merge_sort(L):  
    if len(L) < 2:  
        return L[:]  
    else:  
        middle = len(L)//2  
        left = merge_sort(L[:middle])  
        right = merge_sort(L[middle:])  
        return merge(left, right)
```

base case

divide

conquer with the merge step

▪ **divide list** successively into halves

2.1 Changes

- Variable names were changed for consistency.
- Assignment of "middle" incorporates int() for readability.
- "<=" used instead of "<" in "merge" to make sort stable.