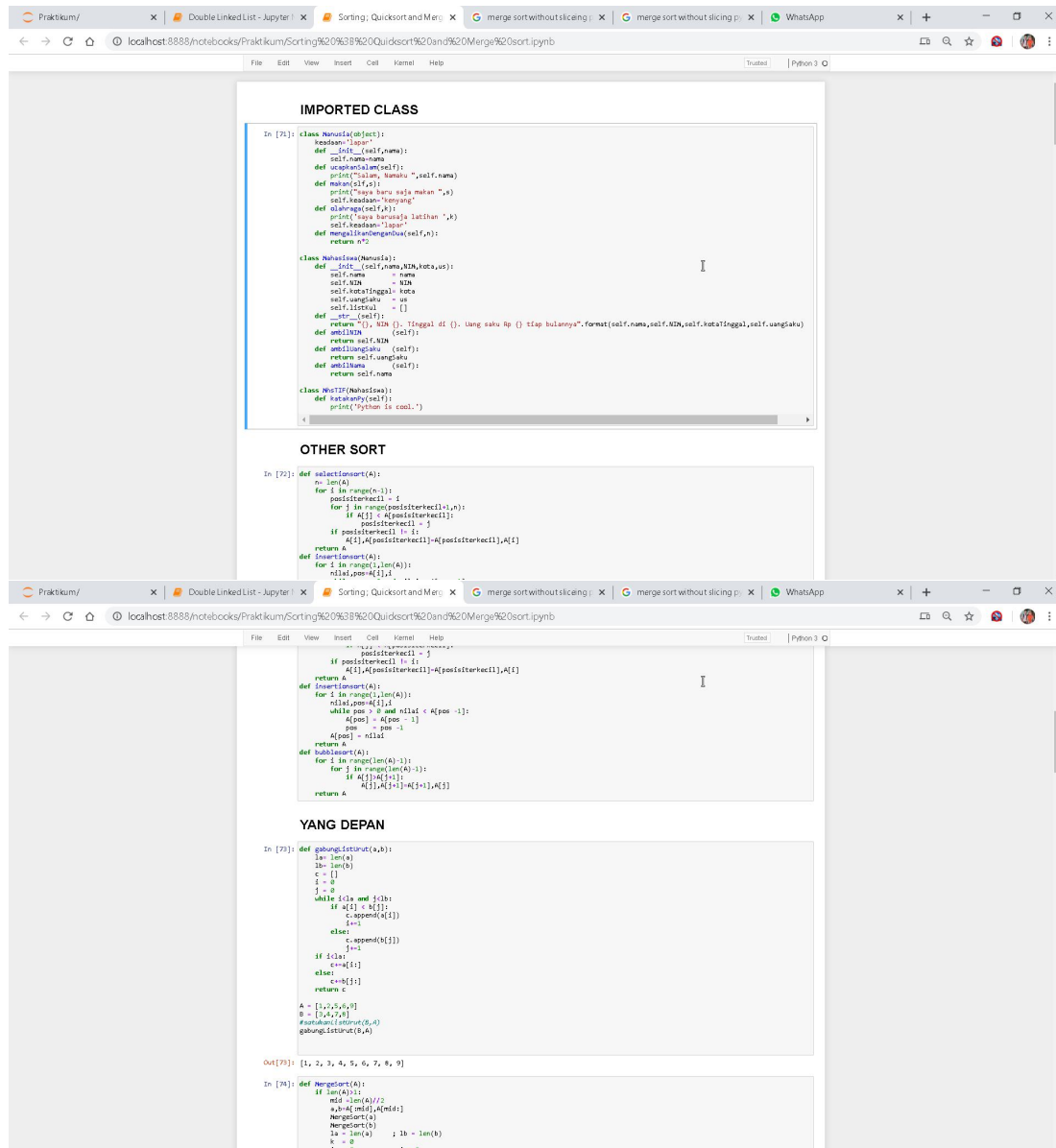


Modul 6 Runtime code (Jupyter notebook)



```
Praktikum/ x Double Linked List - Jupyter x Sorting, Quicksort and Merge x merge sort without slicing x merge sort without slicing p x WhatsApp x + - x
localhost:8888/notebooks/Praktikum/Sorting%20%38%20Quicksort%20and%20Merge%20sort.ipynb
File Edit View Insert Cell Kernel Help Trusted Python 3
In [74]:
def merge_sort(A):
    mid = len(A)//2
    a,b=A[:mid],A[mid:]
    merge_sort(a)
    merge_sort(b)
    la = len(a) ; lb = len(b)
    k = 0 ; i = 0 ; j = 0
    while i < la and j < lb:
        if a[i] < b[j]:
            A[k] = a[i]
            i += 1
        else:
            A[k] = b[j]
            j += 1
        k += 1
    if i < la :
        A[k:] = a[i:]
    elif j < lb :
        A[k:] = b[j:]
    return A
f=[9,7,6,5,4,3,2,1,7,9]
merge_sort(f)

Out[74]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [75]: def QuickSort(A):
    recursion(A,0,len(A)-1)
    return A
def recursion(A,start,last):
    if start < last:
        pivotposition = partition(A,start,last)
        recursion(A,start,pivotposition-1)
        recursion(A,pivotposition+1,last)
def partition(A,start,Right_cursor):
    pivot=A[start]
    Left_cursor=start+1
    while True:
        while Left_cursor < Right_cursor and A[Left_cursor] < pivot:
            Left_cursor += 1
        while A[Right_cursor] >= pivot and Left_cursor < Right_cursor:
            Right_cursor -= 1
        if Left_cursor < Right_cursor:
            break
    A[Left_cursor],A[Right_cursor]=A[Right_cursor],A[Left_cursor]
    A[start],A[Right_cursor]=A[Right_cursor],A[start]
    return Right_cursor
QuickSort([7,7,6,5,3,2,1,7,9])

Out[75]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

NUMBER 1 MERGESORT

In [76]: Anker object
c0 = MSTIP("Tika",10,"Sukoharjo",250000)
c1 = MSTIP("Dudi",11,"Sragen",230000)
c2 = MSTIP("Amad",12,"Surakarta",250000)
c3 = MSTIP("Chandra",18,"Surakarta",230000)
c4 = MSTIP("Eva",14,"Bojaleti",240000)
c5 = MSTIP("Fani",13,"Salatiga",250000)
c6 = MSTIP("Demi",13,"Latan",240000)
c7 = MSTIP("Diah",15,"Kongiri",245000)
c8 = MSTIP("Luna",19,"Latan",240000)
c9 = MSTIP("Hanan",16,"Karanganyar",270000)
c10 = MSTIP("Kholid",19,"Purwodadi",250000)

In [77]: def merge_sort_MSTIP(A):
    if len(A)>1:
        mid = len(A)//2
        a,b=A[:mid],A[mid:]
        merge_sort_MSTIP(a)
        merge_sort_MSTIP(b)
        la = len(a) ; lb = len(b)
        k = 0 ; i = 0 ; j = 0
        while i < la and j < lb:
            if a[i].ambILINAM() < b[j].ambILINAM():
                A[k] = a[i]
                i += 1
            else:
                A[k] = b[j]
                j += 1
            k += 1
        if i < la :
            A[k:] = a[i:]
        elif j < lb :
            A[k:] = b[j:]
        return A
Daftar=[c0,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10]
print("Mergesort_MSTIP(Daftar),sep='\n'")

Amad, NIM 2, Tinggal di Surakarta, Uang saku Rp 250000 tiap bulannya
Eva, NIM 4, Tinggal di Bojaleti, Uang saku Rp 240000 tiap bulannya
Galah, NIM 5, Tinggal di Kongiri, Uang saku Rp 245000 tiap bulannya
Ika, NIM 10, Tinggal di Sukoharjo, Uang saku Rp 240000 tiap bulannya
Demi, NIM 13, Tinggal di Klaten, Uang saku Rp 240000 tiap bulannya
Chandra, NIM 18, Tinggal di Surakarta, Uang saku Rp 230000 tiap bulannya
Janto, NIM 23, Tinggal di Klaten, Uang saku Rp 245000 tiap bulannya
Kholid, NIM 29, Tinggal di Purwodadi, Uang saku Rp 250000 tiap bulannya
Fendi, NIM 31, Tinggal di Salatiga, Uang saku Rp 250000 tiap bulannya
Bufti, NIM 51, Tinggal di Sragen, Uang saku Rp 235000 tiap bulannya
Hanan, NIM 64, Tinggal di Karanganyar, Uang saku Rp 270000 tiap bulannya

NUMBER 1 QUICKSORT

In [78]: def QuickSort_MSTIP(A):
    recursion_MSTIP(A,0,len(A)-1)
    return A
def recursion_MSTIP(A,start,last):
    if start < last:
        pivotposition = partition_MSTIP(A,start,last)
        recursion_MSTIP(A,start,pivotposition-1)
        recursion_MSTIP(A,pivotposition+1,last)
def partition_MSTIP(A,start,Right_cursor):
    pivot=A[start]
    Left_cursor=start+1
    while True:
        while Left_cursor < Right_cursor and A[Left_cursor].ambILINAM() < pivot.ambILINAM():
            Left_cursor += 1
        while A[Right_cursor].ambILINAM() >= pivot.ambILINAM() and Left_cursor < Right_cursor:
            Right_cursor -= 1
        if Left_cursor < Right_cursor:
            break
    A[Left_cursor],A[Right_cursor]=A[Right_cursor],A[Left_cursor]
    A[start],A[Right_cursor]=A[Right_cursor],A[start]
    return Right_cursor
print("QuickSort_MSTIP(Daftar),sep='\n'")

Amad, NIM 2, Tinggal di Surakarta, Uang saku Rp 250000 tiap bulannya
Eva, NIM 4, Tinggal di Bojaleti, Uang saku Rp 240000 tiap bulannya
Galah, NIM 5, Tinggal di Kongiri, Uang saku Rp 245000 tiap bulannya
Ika, NIM 10, Tinggal di Sukoharjo, Uang saku Rp 240000 tiap bulannya
Demi, NIM 13, Tinggal di Klaten, Uang saku Rp 240000 tiap bulannya
Chandra, NIM 18, Tinggal di Surakarta, Uang saku Rp 230000 tiap bulannya
Janto, NIM 23, Tinggal di Klaten, Uang saku Rp 245000 tiap bulannya
Kholid, NIM 29, Tinggal di Purwodadi, Uang saku Rp 250000 tiap bulannya
Fendi, NIM 31, Tinggal di Salatiga, Uang saku Rp 250000 tiap bulannya
Bufti, NIM 51, Tinggal di Sragen, Uang saku Rp 235000 tiap bulannya
Hanan, NIM 64, Tinggal di Karanganyar, Uang saku Rp 270000 tiap bulannya

NOMER 3

In [79]: from time import time
from random import shuffle
k=list(range(1,10000))
shuffle(k)
k1=k[1]
k2=k[2]
k3=k[3]
k4=k[4]
k5=k[5]
k6=k[6]
k7=k[7]
print("===== Sorting i) data =====",format(len(k)))
#AwalTime(j)bubbleSort(k1) & waktuTime(j) print("Bubble sort : ",(k1-awal)/2,"s")
#AwalTime(j)insertionSort(k2) & waktuTime(j) print("Insertion sort : ",(k2-awal)/2,"s")
#AwalTime(j)selectionSort(k3) & waktuTime(j) print("Selection sort : ",(k3-awal)/2,"s")
```

```
Praktikum/ x Double Linked List - Jupyter x Sorting, Quicksort and Merge x merge sort without slicing x merge sort without slicing p x WhatsApp x + - x
localhost:8888/notebooks/Praktikum/Sorting%20%38%20Quicksort%20and%20Merge%20sort.ipynb
File Edit View Insert Cell Kernel Help Trusted Python 3
In [76]: Anker object
c0 = MSTIP("Tika",10,"Sukoharjo",250000)
c1 = MSTIP("Dudi",11,"Sragen",230000)
c2 = MSTIP("Amad",12,"Surakarta",250000)
c3 = MSTIP("Chandra",18,"Surakarta",230000)
c4 = MSTIP("Eva",14,"Bojaleti",240000)
c5 = MSTIP("Fani",13,"Salatiga",250000)
c6 = MSTIP("Demi",13,"Latan",240000)
c7 = MSTIP("Diah",15,"Kongiri",245000)
c8 = MSTIP("Luna",19,"Latan",240000)
c9 = MSTIP("Hanan",16,"Karanganyar",270000)
c10 = MSTIP("Kholid",19,"Purwodadi",250000)

In [77]: def merge_sort_MSTIP(A):
    if len(A)>1:
        mid = len(A)//2
        a,b=A[:mid],A[mid:]
        merge_sort_MSTIP(a)
        merge_sort_MSTIP(b)
        la = len(a) ; lb = len(b)
        k = 0 ; i = 0 ; j = 0
        while i < la and j < lb:
            if a[i].ambILINAM() < b[j].ambILINAM():
                A[k] = a[i]
                i += 1
            else:
                A[k] = b[j]
                j += 1
            k += 1
        if i < la :
            A[k:] = a[i:]
        elif j < lb :
            A[k:] = b[j:]
        return A
Daftar=[c0,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10]
print("Mergesort_MSTIP(Daftar),sep='\n'")

Amad, NIM 2, Tinggal di Surakarta, Uang saku Rp 250000 tiap bulannya
Eva, NIM 4, Tinggal di Bojaleti, Uang saku Rp 240000 tiap bulannya
Galah, NIM 5, Tinggal di Kongiri, Uang saku Rp 245000 tiap bulannya
Ika, NIM 10, Tinggal di Sukoharjo, Uang saku Rp 240000 tiap bulannya
Demi, NIM 13, Tinggal di Klaten, Uang saku Rp 240000 tiap bulannya
Chandra, NIM 18, Tinggal di Surakarta, Uang saku Rp 230000 tiap bulannya
Janto, NIM 23, Tinggal di Klaten, Uang saku Rp 245000 tiap bulannya
Kholid, NIM 29, Tinggal di Purwodadi, Uang saku Rp 250000 tiap bulannya
Fendi, NIM 31, Tinggal di Salatiga, Uang saku Rp 250000 tiap bulannya
Bufti, NIM 51, Tinggal di Sragen, Uang saku Rp 235000 tiap bulannya
Hanan, NIM 64, Tinggal di Karanganyar, Uang saku Rp 270000 tiap bulannya

NUMBER 1 QUICKSORT

In [78]: def QuickSort_MSTIP(A):
    recursion_MSTIP(A,0,len(A)-1)
    return A
def recursion_MSTIP(A,start,last):
    if start < last:
        pivotposition = partition_MSTIP(A,start,last)
        recursion_MSTIP(A,start,pivotposition-1)
        recursion_MSTIP(A,pivotposition+1,last)
def partition_MSTIP(A,start,Right_cursor):
    pivot=A[start]
    Left_cursor=start+1
    while True:
        while Left_cursor < Right_cursor and A[Left_cursor].ambILINAM() < pivot.ambILINAM():
            Left_cursor += 1
        while A[Right_cursor].ambILINAM() >= pivot.ambILINAM() and Left_cursor < Right_cursor:
            Right_cursor -= 1
        if Left_cursor < Right_cursor:
            break
    A[Left_cursor],A[Right_cursor]=A[Right_cursor],A[Left_cursor]
    A[start],A[Right_cursor]=A[Right_cursor],A[start]
    return Right_cursor
print("QuickSort_MSTIP(Daftar),sep='\n'")

Amad, NIM 2, Tinggal di Surakarta, Uang saku Rp 250000 tiap bulannya
Eva, NIM 4, Tinggal di Bojaleti, Uang saku Rp 240000 tiap bulannya
Galah, NIM 5, Tinggal di Kongiri, Uang saku Rp 245000 tiap bulannya
Ika, NIM 10, Tinggal di Sukoharjo, Uang saku Rp 240000 tiap bulannya
Demi, NIM 13, Tinggal di Klaten, Uang saku Rp 240000 tiap bulannya
Chandra, NIM 18, Tinggal di Surakarta, Uang saku Rp 230000 tiap bulannya
Janto, NIM 23, Tinggal di Klaten, Uang saku Rp 245000 tiap bulannya
Kholid, NIM 29, Tinggal di Purwodadi, Uang saku Rp 250000 tiap bulannya
Fendi, NIM 31, Tinggal di Salatiga, Uang saku Rp 250000 tiap bulannya
Bufti, NIM 51, Tinggal di Sragen, Uang saku Rp 235000 tiap bulannya
Hanan, NIM 64, Tinggal di Karanganyar, Uang saku Rp 270000 tiap bulannya
```

```
Praktikum/ x Double Linked List - Jupyter x Sorting, Quicksort and Merge x merge sort without slicing x merge sort without slicing p x WhatsApp x + - x
localhost:8888/notebooks/Praktikum/Sorting%20%38%20Quicksort%20and%20Merge%20sort.ipynb
File Edit View Insert Cell Kernel Help Trusted Python 3
In [78]: def QuickSort_MSTIP(A):
    recursion_MSTIP(A,0,len(A)-1)
    return A
def recursion_MSTIP(A,start,last):
    if start < last:
        pivotposition = partition_MSTIP(A,start,last)
        recursion_MSTIP(A,start,pivotposition-1)
        recursion_MSTIP(A,pivotposition+1,last)
def partition_MSTIP(A,start,Right_cursor):
    pivot=A[start]
    Left_cursor=start+1
    while True:
        while Left_cursor < Right_cursor and A[Left_cursor].ambILINAM() < pivot.ambILINAM():
            Left_cursor += 1
        while A[Right_cursor].ambILINAM() >= pivot.ambILINAM() and Left_cursor < Right_cursor:
            Right_cursor -= 1
        if Left_cursor < Right_cursor:
            break
    A[Left_cursor],A[Right_cursor]=A[Right_cursor],A[Left_cursor]
    A[start],A[Right_cursor]=A[Right_cursor],A[start]
    return Right_cursor
print("QuickSort_MSTIP(Daftar),sep='\n'")

Amad, NIM 2, Tinggal di Surakarta, Uang saku Rp 250000 tiap bulannya
Eva, NIM 4, Tinggal di Bojaleti, Uang saku Rp 240000 tiap bulannya
Galah, NIM 5, Tinggal di Kongiri, Uang saku Rp 245000 tiap bulannya
Ika, NIM 10, Tinggal di Sukoharjo, Uang saku Rp 240000 tiap bulannya
Demi, NIM 13, Tinggal di Klaten, Uang saku Rp 240000 tiap bulannya
Chandra, NIM 18, Tinggal di Surakarta, Uang saku Rp 230000 tiap bulannya
Janto, NIM 23, Tinggal di Klaten, Uang saku Rp 245000 tiap bulannya
Kholid, NIM 29, Tinggal di Purwodadi, Uang saku Rp 250000 tiap bulannya
Fendi, NIM 31, Tinggal di Salatiga, Uang saku Rp 250000 tiap bulannya
Bufti, NIM 51, Tinggal di Sragen, Uang saku Rp 235000 tiap bulannya
Hanan, NIM 64, Tinggal di Karanganyar, Uang saku Rp 270000 tiap bulannya

NOMER 3

In [79]: from time import time
from random import shuffle
k=list(range(1,10000))
shuffle(k)
k1=k[1]
k2=k[2]
k3=k[3]
k4=k[4]
k5=k[5]
k6=k[6]
k7=k[7]
print("===== Sorting i) data =====",format(len(k)))
#AwalTime(j)bubbleSort(k1) & waktuTime(j) print("Bubble sort : ",(k1-awal)/2,"s")
#AwalTime(j)insertionSort(k2) & waktuTime(j) print("Insertion sort : ",(k2-awal)/2,"s")
#AwalTime(j)selectionSort(k3) & waktuTime(j) print("Selection sort : ",(k3-awal)/2,"s")
```

Praktikum/Double Linked List - Jupyter/Sorting, Quicksort and Merge/merge sort without slicing/merge sort without slicing p/WhatsApp

localhost8888/notebooks/Praktikum/Sorting%20%38%20Quicksort%20and%20Merge%20sort.ipynb

FileEditViewInsertCellKernelHelp

Python 3

```
k4=k[i]
k5=k[i]
k6=k[i]
print("----- Sorting () data -----",format(len(v)))
#bubble sort
#insertion sort
#selection sort
#merge sort
#quicksort
print("----- Sorting 100000 data -----")
Merge sort : 2.571256037972842 s
Quick sort : 1.7481529460221004 s
Tunsort : 0.10403084754943848 s
```

NOMER 4 TRACE MERGESORT

```
In [80]: def MergeSort_Trace(A):
        print("Memulai t:",A)
        if len(A)>1:
            mid=len(A)//2
            a,b=A[:mid],A[mid:]
            MergeSort(a)
            MergeSort(b)
            la=len(a)
            lb=len(b)
            k=0
            l=0
            j=0
            print("Menggabungkan t:",a,b)
            while l<la and j<lb:
                if a[l]<=b[j]:
                    A[k]=a[l]
                    l+=1
                else:
                    A[k]=b[j]
                    j+=1
                k+=1
            if l<la:
                A[k:]=a[l:]
            elif j<lb:
                A[k:]=b[j:]
            print("hasil Menggabungan t:",A)
            return A
        f=[80, 7, 34, 16, 43, 91, 35, 2, 19, 72]
        MergeSort(f)

Out[80]: [2, 7, 16, 19, 24, 35, 43, 72, 80, 91]
```

NOMER 4 QUICKSORT TRACE

```
In [81]: def Quicksort_Trace(A):
        Recursion_Trace(A,0,len(A)-1)
        print("Data telah urut")
        return A
        def Recursion_Trace(A,start,last):
            if start<last:
                pivotposition = partition_Trace(A,start,last)
                Recursion_Trace(A,start,pivotposition-1)
                Recursion_Trace(A,pivotposition+1,last)
            def partition_Trace(A,start,Right_cursor):
                pivot=A[start]
                print("Memilih pivot",A[start])
                Left_cursor=start+1
                while True:
                    while Left_cursor<Right_cursor and A[Left_cursor]<=pivot:
                        Left_cursor+=1
                    print(A[Left_cursor],"lebih besar dari pivot, mencari penukar")
                    while A[Right_cursor]>=pivot and Left_cursor<Right_cursor:
                        print("Membandingkan",A[Right_cursor],"dengan",pivot)
                        Right_cursor-=1
                    if Left_cursor<Right_cursor:
                        break
                    else:
                        print("Tukar",A[Left_cursor],"dengan",A[Right_cursor])
                        A[Left_cursor],A[Right_cursor]=A[Right_cursor],A[Left_cursor]
                        print("Tukar",A[Left_cursor],"dengan",A[Right_cursor])
                        A[start],A[Right_cursor]=A[Right_cursor],A[start]
                        return Right_cursor
            f=[80, 7, 34, 16, 43, 91, 35, 2, 19, 72]
            Quicksort(f)

Out[81]: [2, 7, 16, 19, 24, 35, 43, 72, 80, 91]
```



NOMER 5 IMPROVING MERGESORT

```
In [ ]: #masukkan fungsi dalam return

In [ ]:
```

NOMER 6 QUICKSORT WITH 3 MEDIAN PIVOT

```
In [82]: from statistics import median
        def Quicksort(A):
            Recursion(A,0,len(A)-1)
            return A
        def Recursion(A,start,last):
            if start<last:
                pivotposition = partition2(A,start,last)
                Recursion(A,start,pivotposition-1)
                Recursion(A,pivotposition+1,last)
        def partition2(A,start,Right_cursor):
```



Praktikum/Double Linked List - Jupyter | Sorting, Quicksort and Merge | merge sort without slicing | merge sort without slicing p | WhatsApp

localhost:8888/notebooks/Praktikum/Sorting%20and%20Merge%20sort.ipynb

File Edit View Insert Cell Kernel Help

```
def partition(a, start, last):
    pivot = a[start]
    left = start
    right = last
    while True:
        while left < right and a[left] < pivot:
            left += 1
        while right > left and a[right] > pivot:
            right -= 1
        if left < right:
            a[left], a[right] = a[right], a[left]
        left += 1
        right -= 1
    a[start], a[right] = a[right], a[start]
    return right

def quicksort(a, start, last):
    if start < last:
        pivot = partition(a, start, last)
        quicksort(a, start, pivot-1)
        quicksort(a, pivot+1, last)
    return a

a = [10, 5, 8, 3, 2, 7, 9, 4, 6]
quicksort(a, 0, len(a)-1)
print(a)
```

Quick sort

1.78032886809368 s

Mergesort on Linked List

```
class Node:
    def __init__(self, data, next=None):
        self.data = data
        self.next = next

class LinkedList:
    def __init__(self, head=None):
        self.head = head

    def append(self, data):
        new_node = Node(data)
        if self.head is None:
            self.head = new_node
        else:
            current = self.head
            while current.next is not None:
                current = current.next
            current.next = new_node

    def merge_sort(self, a, b):
        if a is None:
            return b
        if b is None:
            return a
        if a.data < b.data:
            result = a
            result.next = self.merge_sort(a.next, b)
        else:
            result = b
            result.next = self.merge_sort(a, b.next)
        return result

    def merge_sort_linked_list(self, self):
        self.head = self.merge_sort(self.head)

    def merge_sort(self, h):
        if h is None or h.next is None:
            return h
        middle = self.find_middle(h)
        next_middle = middle.next
        middle.next = None
        left = self.merge_sort(h)
        right = self.merge_sort(next_middle)
        return self.merge_sort(left, right)

    def find_middle(self, h):
        pointer = self.head
        buffer = []
        while pointer is not None:
            buffer.append(pointer.data)
            pointer = pointer.next
        print(buffer, sep=" ")

ll = LinkedList()
ll.append(Node(8))
ll.append(Node(7))
ll.append(Node(1))
ll.append(Node(5))
ll.append(Node(4))
ll.append(Node(3))
ll.append(Node(2))
ll.merge_sort_linked_list()
ll.printer()
```

0 1 2 3 4 5 7