Divide and conquer - merge sort

February 25, 2022

[1]:
```
def insertion_sort(list_to_sort):
    L = list(list_to_sort)
    new_list = []

    while len(L) > 0:
        # find the index of the smallest thing
        min_index = min(range(len(L)), key=lambda i : L[i])

        # remove it from L and add to new_list
        new_list.append(L.pop(min_index))
    return new_list
```

[2]:
```
import math, random
from time import time
```

[3]:
```
L = [random.randint(1, 1000000) for n in range(10000)]
```

[4]:
```
L[:5]
```

[4]:
```
[427197, 114724, 62702, 642190, 841575]
```

[5]:
```
tt = time()
R1 = insertion_sort(L)
print(time() - tt)
```

```
4.119019985198975
```

[6]:
```
tt = time()
R2 = sorted(L)
print(time()-tt)
```

```
0.0018157958984375
```

[7]:
```
R1 == R2
```

[7]:
```
True
```

[19]:
```
def merge_sort(L):
    #print("Calling with list:",L)
```

```
    # base case: a list of length 1 is already sorted
    if len(L) == 1:
        return L

    # split the list (roughly) in half
    mid_point = math.ceil(len(L)/2)
    left_half = L[:mid_point]
    right_half = L[mid_point:]

    # recursively apply the function to both halves
    LS = merge_sort(left_half)
    RS = merge_sort(right_half)

    full_list = []

    while len(LS) + len(RS) > 0:
        # either LS[0] or RS[0] is the smallest of
        # all elements left. Find it, remove it, and
        # add to full_list.

        # if len(LS) > 0
        if LS:
            if RS:
                if LS[0] <= RS[0]:
                    full_list.append(LS.pop(0))
                else:
                    full_list.append(RS.pop(0))
            else:
                full_list.append(LS.pop(0))
        else:
            full_list.append(RS.pop(0))

    return full_list
```

[20]:
```
merge_sort([3, 19, -7, 2, 1, 6, 0, -10])
```

[20]:
```
[-10, -7, 0, 1, 2, 3, 6, 19]
```

[21]:
```
tt = time()
R3 = merge_sort(L)
print(time()-tt)
```

```
0.07052326202392578
```

[10]:
```
R1 == R3
```

[10]:
```
True
```

insertion sort: 100 elements - 0.00075 1000 elements - 0.064 – 85 times bigger than 100 10000 elements - 5.39 – 84 times bigger than 1000

our merge sort: 100 elements - 0.0004 1000 elements - 0.005 - 13 times bigger 10000 elements - 0.07 - 12.5 times bigger 100000 elements - 1.46 - 20 times bigger

pythons sort (timsort): 10000 elements - 0.0018 100000 elements - 0.017 1000000 elements - 0.32 10000000 elements - 4.3

[ ]: