Lecture 06 - Sorting

February 17, 2021

[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 it to new_list
        new_list.append(L.pop(min_index))

    return new_list
```

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

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

[4]:
```
tt = time()
R1 = insertion_sort(L)
print(time()-tt)
```

```
5.3559629917144775
```

[5]:
```
tt = time()
R2 = sorted(L)
print(time()-tt)
```

```
0.00138092041015625
```

[6]:
```
def merge_sort(L):

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

    # split the list (roughly) in hald
    mid_point = math.ceil(len(L)/2)
```

```
    left_half = L[:mid_point]
    right_half = L[mid_point:]

    # recursively apply the function to the two halfs (divide)
    LS = merge_sort(left_half)
    RS = merge_sort(right_half)

    # time to conquer
    full_list = []

    # while the two halves still have something left
    while len(LS) + len(RS) > 0:
        # either LS[0] or RS[0] is the smallest of all elements
        # in LS and RS. Find it, remove it from its list, and
        # add it to full_list
        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
```

[7]:
```
tt = time()
R3 = merge_sort(L)
print(time()-tt)
```

```
0.06351804733276367
```

[8]:
```
R3 == R2
```

[8]:
```
True
```

[12]:
```
times = []
for p in range(1,6):
    L = [random.randint(1,1000000) for n in range(10**p)]

    tt = time()
    R = insertion_sort(L)
    times.append(time()-tt)

    print(f"{10**p} {time()-tt}")
    if len(times) > 1:
```

```
        print(f"\t{times[-1]/times[-2]}")
```

```
10 0.000209808349609375
100 0.00075531005859375
        3.6775320139697323
1000 0.06404900550842285
        85.0360873694207
10000 5.391217947006226
        84.17702854122228
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-12-5a7d85f24383> in <module>
      4
      5     tt = time()
----> 6     R = insertion_sort(L)
      7     times.append(time()-tt)
      8

<ipython-input-1-19bac1fad570> in insertion_sort(list_to_sort)
      5     while len(L) > 0:
      6         # find the index of the smallest thing
----> 7         min_index = min(range(len(L)), key=lambda i : L[i])
      8
      9         # remove it from L and add it to new_list

<ipython-input-1-19bac1fad570> in <lambda>(i)
      5     while len(L) > 0:
      6         # find the index of the smallest thing
----> 7         min_index = min(range(len(L)), key=lambda i : L[i])
      8
      9         # remove it from L and add it to new_list

KeyboardInterrupt:
```

[11]:
```
times = []
for p in range(1,6):
    L = [random.randint(1,1000000) for n in range(10**p)]

    tt = time()
    R2 = merge_sort(L)
    times.append(time()-tt)

    print(f"{10**p} {time()-tt}")
    if len(times) > 1:
        print(f"\t{times[-1]/times[-2]}")
```

```
10 0.0023648738861083984
100 0.0004220008850097656
        0.17847397675593735
1000 0.005793094635009766
        13.746319365798415
10000 0.07274603843688965
        12.568215521502719
100000 1.458730936050415
        20.05314546419933
```

[13]:
```python
times = []
for p in range(1,8):
    L = [random.randint(1,1000000) for n in range(10**p)]

    tt = time()
    R3 = sorted(L)
    times.append(time()-tt)

    print(f"{10**p} {time()-tt}")
    if len(times) > 1:
        print(f"\t{times[-1]/times[-2]}")
```

```
10 9.799003601074219e-05
100 1.5020370483398438e-05
        0.13819095477386933
1000 0.0001399517059326172
        10.50909090909091
10000 0.0017790794372558594
        12.865051903114187
100000 0.017344951629638672
        9.780661646046262
1000000 0.3232860565185547
        18.643429718544184
10000000 4.332744836807251
        13.402582604552782
```

[ ]:
```python

```