sorting

February 9, 2026

[1]:
```
def insertion_sort(list_to_sort): # O(n^2)
    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(1,1_000_000) for n in range(10_000)]
```

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

```
2.4214320182800293
```

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

```
0.0008769035339355469
```

[6]:
```
def merge_sort(L):
    # assume L is a list of numbers
    # output will be a sorted list of those numbers

    # 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 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 [list]" means "if the list is non-empty".
        #       same as "if len([list]) > 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
```

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

```
0.02702784538269043
```

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

[8]:
```
times = []
for p in range(1,20):
    L = [random.randint(1,1000000) for n in range(2**p)]

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

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

```
2 5.9604644775390625e-06
4 3.814697265625e-06
        0.5714285714285714
8 5.245208740234375e-06
        1.5
16 1.0967254638671875e-05
        2.5555555555555554
32 3.0994415283203125e-05
        2.8260869565217392
64 0.00010180473327636719
        3.2846153846153845
128 0.00036597251892089844
        3.5948477751756442
256 0.0019021034240722656
        5.188925081433225
512 0.0071680545806884766
        3.7730069052102952
1024 0.024302959442138672
        3.3915213629708507
2048 0.10291194915771484
        4.234924746374679
4096 0.41745686531066895
        4.05651341883827
8192 1.6518058776855469
        3.9568638447993942
16384 6.449176073074341
        3.9043272120315997
32768 26.217708110809326
        4.065282212193308
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[8], line 6
      3 L = [random.randint(1,1000000) for n in range(2**p)]
      5 tt = time()
----> 6 R = insertion_sort(L)
      7 times.append(time()-tt)
      9 print(f"{2**p} {time()-tt}")

Cell In[1], line 7, in insertion_sort(list_to_sort)
      3 new_list = []
      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])
      9     # remove it from L and add it to new_list
     10     new_list.append(L.pop(min_index))

Cell In[1], line 7, in insertion_sort.<locals>.<lambda>(i)
      3 new_list = []
      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])
      9     # remove it from L and add it to new_list
     10     new_list.append(L.pop(min_index))

KeyboardInterrupt:
```

[ ]:
```
times = []
for p in range(0,7):
    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]}")
```

[ ]:
```
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]}")
```

[ ]:
```

```