Insertion Sort

September 24, 2023

1 Describe and Implement Insertion Sort

by Egle Group work

From Jing Wang, Marius Gnoth, Alexander Moshchev, Cheng Yu, Jialiang Gao and Zhuan Wu

Mented by Vahe Andonians Salmas

1.1 Introduction

Sort is the most commen algorithmus in our daily life. There are a lot of sort algorithmus, Insertion sort is one of the most intuitive and match the human mind.

1.2 Describe the Insertion Sort

The insertion sort is most like humankind sort a hand of pork card. It strat with the second card, campare it with the first card and insert in the right way (left when it smaller , right when it bigger). After the second card, campare and insert the third the forth untill end of the card.

The Algorithmus of Insertion Sort is think very close. Use a loop from the second list element to the last list element. In each loop compare the number list left of the selected number from left to right. When the selected number is smaller then the number in the list, insert the selected number left of the compared number and go to the next loop.

1.3 Implement

[ ]:
```
def insertion_sort(List):
    for i in range(0,len(List)):
        ii = 0
        while List[i] > List[ii]:
            ii=ii+1
        a = List[i]
        List[ii+1:i+1]=List[ii:i]
        List[ii]=a
    return List
```

1.3.1 test

with the List [3,4,6,1,2] & [5,3,6,7,2,5]

[ ]:
```
List1=[3,4,6,1,2]
List2=[5,3,6,7,2,5]
print(f'List1 is {List1}, after the Sort result is {insertion_sort(List1)}')
print(f'List2 is {List2}, after the Sort result is {insertion_sort(List2)}')
```

```
List1 is [3, 4, 6, 1, 2], after the Sort result is [1, 2, 3, 4, 6]
List2 is [5, 3, 6, 7, 2, 5], after the Sort result is [2, 3, 5, 5, 6, 7]
```

1.4 Performance Analysis

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import random
from datetime import datetime
```

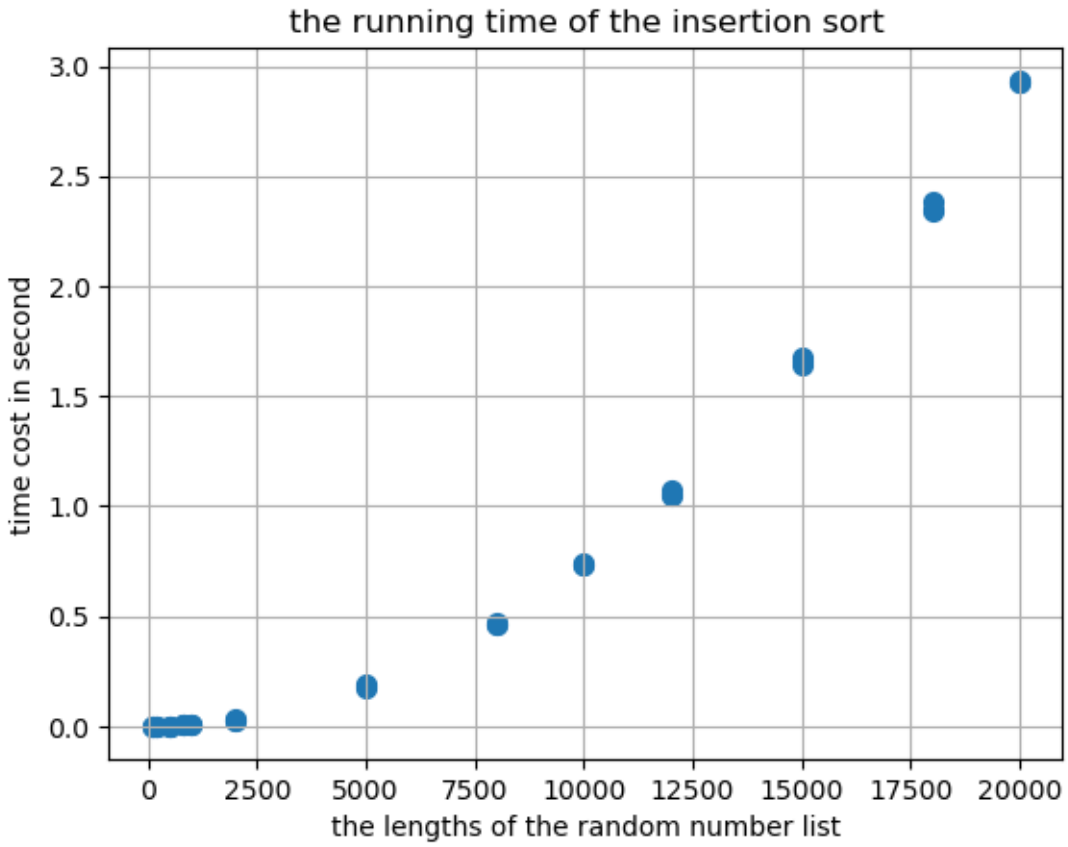
1.4.1 Generate random lists of varying sizes and Measure the time it takes to sort each list using insertion_sort function.

[ ]:
```
x=3
#x is for each len of the number how many round we run to estimate the random␣
 ↪error.

a=np.array([100,200,500,800,1000,2000,5000,8000,10000,12000,15000,18000,20000])
# a is a list of the len of random number list.

timeresult=np.zeros((x,len(a)))
#timeresult is the result array in size(x * len(a)) to record the running time,␣
 ↪first are all zero

for j in range(x):

    for i in range(len(a)):
        number = a[i]
        #take the len of the random list

        n = random.sample(range(0, number), number)
        #generate the random lists

        start_time = datetime.now()
        #use datetime to record the start time
        insertion_sort(n)
        #use insertion_sort function sort the random list n
        end_time = datetime.now()
        #record the end time

        timeresult[j,i]=(end_time - start_time).total_seconds()
```

```
        #calcaulate the total time in seconds and record it in the matrix␣
↪timeresult.
```

1.4.2 Plot a chart

[ ]:
```
x = np.array([a,a,a])
y = timeresult
# plot
fig, ax = plt.subplots()
ax.scatter(x, y, linewidth=2)
ax.set_ylabel('time cost in second')
ax.set_xlabel('the lengths of the random number list')
ax.set_title('the running time of the insertion sort')
ax.grid()
#ax.legend()
image_format = 'svg' # e.g .png, .svg, etc.
image_name = 'insertion_sort.svg'

fig.savefig(image_name, format=image_format, dpi=1200)
```



[ ]: