AlgoHack #9

PROGRAMING DATA STRUCTURES

Authors
Niranjan Meegammana
N P Vishwa Kumara

Reviewers
Ravindu Ramesh Perera, Devanjith De Silva,
Prabhashana Hasthidhara, Yamuna Ratnayake.

AlgoHack aims to teach Computer Science and Programing to young people, initiated by Shilpa Sayura Foundation, supported by GOOGLE RISE and Computer Society of Sri Lanka..

Data Structures help us collect and organise data to perform operations effectively.We use relationships between data for better organization and storage.

If "Ramesh" obtained 66 marks, We know "Ramesh" is a String data type and 66 is an integer data type.
We can store all students records in a file as a data structure like "Ramesh" 66, "Nimi" 56, "Roshini" 56.

In simple, Data Structures are structures programmed to store ordered data, so that various operations can be performed on them easily.

What is Algorithm ?
An algorithm is a method we use to solve a problem.
An Algorithm is not our program.
It is our logic to solve the problem.
We first study how we would solve a problem manually.
Then design our logic. Finally we code and test.

Explain an algorithm is solving a daily life problem.
How do we get to school?
What we do when it rains?
How would you learn a science topic?
How do we make a Sambol?
How does plants grow?

An algorithm need to be efficient and fast.
We measure the performance of an algorithm on time it takes to execute and memory space consumed.

Study this program 1.

```
N = 1
print (N)
```

Does program execution time change if N is 2 or 100?

Study this program 2.

```
N = 10
for x in range (N):
  print (x)
```

How long will it take to execute the program What if N ==100? Will it take longer time if N ==1000? What if N==0? **Does the program running time depend on N ?**

Here running time is related to N.
N can start from 0 and reach infinity.
The time complexity of this algorithm is **Linear.**
If N doubles, running time also doubles.

Draw a graph with N on x axis and number of steps on y axis

Study this program 3

```
N = 10
M = 10
for x in range (N):
    for y in range (N):
          print (x, y, x * y)
```

Is the time complexity same this time? What happens if you double N.

We have two loops here,
so the time complexity is N * N.
The time complexity of above algorithm is **Quadratic**.

Study this program 4.

```
N = 6
list1=[1, 2, 3, 4, 5, 6]
list2 = []
while (len(list1) > 1):
    print("List1 :", list1)
    # find half the length of list1
    half_length = int(len(list1)/2)
    print("Half Length :", half_length)
    # add half the values of n
    list2 = list2 + list1[0:half_length]
    print("List2 :", list2)
    # make list1 half length of itself
    list1 = list1[0:half_length]
print ("Number of operations: " , str(len(list2)))
```

Check how program works with printed values
Explain what happens in each statement.
Increase list1 elements and record number of operations.

This algorithm breaks a set of numbers into halves with each iteration.
The running time is proportional to the number of times.
N can be divided by 2.
So we this algorithm is **Logarithmic .**
Therefore the time complexity will be **N*log(N)**

Record time complexity for each of the programs 1 to 4

N	1	2	3	4	5	6	7	8	9	10
Program 1										
Program 2										
Program 3										
Program 4										

Draw a graph for programs 1 to 4 with number of elements in x axis and number of operations in y axis.
Are the graphs same for all programs?
If not, how do they vary?

Bubble Sort

The sorting takes place by stepping through list items one-by-one in pairs and comparing adjacent data items and swapping each pair if they are out of order.
It is like a water bubble rising to the water surface.

```
list1 = [5, 1, 6, 2, 4, 3]
iteration = 0
print ("iteration", iteration, list1)
for i in range(len(list1)):
    for j in range(len(list1)-1-i):
        iteration=iteration+1
        if list1[j]> list1[j+1]:
            temp = list1[j]
            list1[j] = list1[j+1]
            list1[j+1] = temp # Swap!
        print ("iteration", iteration, list1)
print(list1)
```

Write your list of elements in a pieces of paper.
Process them according to your program.
Record the list value at the end each iteration.
How many iterations you need to sort data?

How many iterations will take to sort a list ?

Elements	3	4	5	6	9
Iterations				15	

Will the number of iterations depend on data?
What happens if you provide a sorted list?

We can see the for loop will keep going for same number of iterations even a sorted array is provided.
We have to compare each value with all other values using two loops.
So we say the complexity of **Bubble Sort** is $O(N^2)$.
We state algorithm complexity for the worst case, thinking all data is out of place, so that any given case it would perform.

So based on what we learn
Program 1 has complexity of O(1)
Program 2 has complexity of O(N)
Program 3 has complexity of $O(N^2)$
Program 4 has complexity of N log(N)
Bubble Sort has complexity of $O(N^2)$

Can you modify Bubble Sort program to reduce the number of iterations? **Tip:** Can the second loop break if a swapping takes place?

Insertion Sort
This algorithm shifts elements one by one to sort a list.

```
list1= [5, 1, 6, 2, 4, 3]
for i in range(len(list1)):
```

```
    key = list1[i] # first item
    j = i-1
    while(j>=0 and key < list1[j]):
        list1[j+1] = list1[j]
        j=j-1

    list1[j+1] = key
print(list1)
```

Write a list of 5 numbers on a pieces of paper.
Sort them using quick sort algorithm.
Record your operations.
Change element order and sort again.
What happens with an ordered list?
Does the program complexity depend on data?

This **Insertion Sort** algorithm takes elements from a list and inserts them at the right place

```
list=[7,6,18,12,0,4,3,21]
N=len(list)
for i in range(1,N):
    x=list[i]
    j=i
    while(j>0 and list[j-1]>x):
        list[j]=list[j-1]
        j=j-1
        list[j]=x
print(list)
```

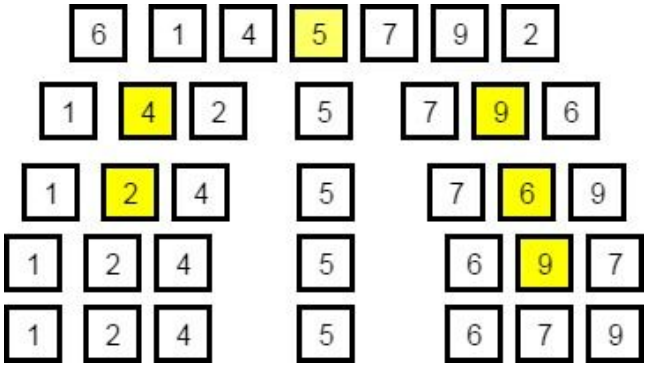
Step through above program and explain.
Use numbers on pieces of paper to do insertion sort.
Does complexity depend on data in list?

Quick Sort

This is algorithm is based on Divide and Conquer rule.
It divides the list into three main parts.

1. Pivot element
2. Elements less than the Pivot element
3. Elements greater than the pivot element

Consider list =[6,1,4,5, 7,9, 2]
On first pass we take 15 as the pivot element .
Then [6, 1, 4] and [7,9,2] become left and right lists



All items lower than 5 are moved to the left.
All items higher than 5 are moved to the left.
We take new pivot for left continue the process.
We take new pivot for right continue the process.
Finally the list is sorted.

Create list=[12,5,9,8,7,14,16,0,9] with paper.
Do Quicksort and record your operations.
Change the position of data and sort again.
Sort a sorted list
Compare time taken in all three occasions.
What will happen if you select first element as pivot?
What will happen if you select last element as pivot?
Does Quicksort complexity depend on data?

Study following Quicksort Program

It uses **recursion** for solving the problem.
Recursion is a function calling it self on and on until it find a solution, then return the result to the function it was called.

```
def qsort(L1):
    smaller = []
    equal = []
    bigger = []
    if len(L1) == 0:
        return []
    pivot = L1[0]
    for x in L1:
        if (x < pivot):
            smaller.append(x)
        elif (x > pivot):
            bigger.append(x)
        else :
            equal.append(x)
```

```
        a1=qsort(smaller)
        a2=qsort(bigger)
        L1=a1+equal+ a2
    return L1

L1=[7,6,18,12,0,4,3,21]
print(L1)
L1=qsort(L1)
print(L1)
```

Step though above program and explain.
This program uses 1st element as pivot.
What if we use last element for pivot ?
How do you find middle element for pivot?

Merge Sort

This algorithm follows the rule of Divide and Conquer. But it doesn't divides the list into two halves.
Merge sort divides an unsorted list into N sublists, each having one element, and repeatedly merge sublists, to produce a sorted list.

```
def mergesort(list):
    mid = len(list)//2
    left, right = list[:mid], list[mid:]
    if len(left) > 1:
        left = mergesort(left)
    if len(right) > 1:
        right = mergesort(right)
```

```
    res = []
    while left and right:
        if left[-1] >= right[-1]:
            res.append(left.pop())
        else:
            res.append(right.pop())
    res.reverse()
    if (left):
        list=left+res
    else:
        list=right+res
    return list

list=[4,7,2,6,12,9,5]
list =mergesort(list)
```

Merge Sort has time complexity of O(n log n).

Study above program and explain how it works?
Draw a diagram to show how L=[6,3,9,12,5] sorted.
Does the time taken to sort depend on data?

Dictionary.

In a language, each word maps to a meaning.
In a dictionary we map keys (words) to values (meanings).

Python dictionary is an unordered collection of items.
The array elements has a single value.
dictionary has a key: value pair.

If you want to store names of students and their marks you need two arrays.
students=["Isha", "Rani", "Uma"]
marks=[65, 80, 58]
If we use a dictionary we can write
student_marks={"Isha":65, "Rani":80, "Uma":58}
student_marks["Uma"] will output 58.
"Uma" is the key and 58 is the value in this element.
Dictionary is therefore a sorted list.

We can add an element to a dictionary
student_marks["Jeewa"] =45

We can update dictionary
student_marks["Uma"] =65

We can process dictionary

```
student_marks={"Isha":65, "Rani":80, "Uma":58}
student_marks["Jeewa"] =45
for x in student_marks:
    print (x, student_marks[x])
```

What will be the output of above program?

Some of methods in Python dictionaries

clear()	Remove all items in the dictionary.

copy()	Return a shallow copy of the dictionary.
items()	Returns dictionary's items (key, value).
keys()	Return a list of dictionary keys.
pop(key)	Remove the item with key and return value
values()	Returns dictionary's values
len()	Return the length of dictionary
cmp()	Compares items of two dictionaries.
sorted()	Return a sorted list of keys in the dictionary

We can test if a key is in a dictionary using in keyword.

```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
print (7 in squares)
print (11 in squares)
print (11 not in squares)
```

Write a program to store numbers 1 to 5 and their squares. The output dictionary should be {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
Tip : squares = {} is an empty dictionary

Write a program that takes 10 countries and population, Store them in a dictionary, print them sorted by country. Which country has the highest population.

Professor Crow has a text file. He wants to count how many different words in the file. He also want to know how many times each word occurs. Can you write a program for him. Professor crow will be thankful to you if you order the list by words.

Professor Crow has a text file. He wants to count how many different words in the file. He also want to know how many times each word occurs. Can you write a program for him. Professor crow will be thankful to you if you order the list by words.

Suggest a method to order word list by their values.

Numbers of Professor Crow
Professor crow has an array of numbers.
He wants to know the maximum sum of any two elements in a list of integers. He wants to know which pair of numbers will give a certain sum.

arr = [8, 7, 2, 5, 3, 1]
sum = 10
Among list of numbers, find sublist with 0 sum is exists or not. arr = [8, 7, -1, 5, -2, -3,7]

Find a duplicate elements in an array of 30 numbers.
Find largest sub list formed by adjacent integers.
Find all subarrays having given sum.
Merge two sorted arrays and output a sorted array.
Find maximum product of two integers in an array
Find maximum difference between two elements..

AlgoHack aims to teach Computer Science and Programing to young people, initiated by Shilpa Sayura Foundation, supported by GOOGLE RISE and Computer Society of Sri Lanka..

AlgoHack #9

PROGRAMING DATA STRUCTURES