FIT1045 Algorithmic Problem Solving – Tutorial 10.

Objectives

The **objectives of this tutorial** are:

- To become familiar with heaps.
- To become familiar with Binary Search Trees.
- To introduce a simple computational model for counting the number of instructions used by a program.

Task 1

Part A

A *tree* is a graph that has no cycles. Discuss, as a class, the special properties of trees that are *Binary Search Trees*, or BST.

Write an algorithm for inserting an element into a BST. Use your algorithm to insert the following items (in the order they are listed) into the BST:

goat, dog, frog, elephant, rhino, sheep, ant, zebra.

Complete the following table:

Item	Root Node (YES/NO)	Leaf Node (YES/NO)	List of children (if Parent Node)
ant			
dog			
elephant			
frog			
goat			
rhino			
sheep			
zebra			

Part B

Write an algorithm to search for an item in a BST. Your algorithm should take as input the BST and the target item. It should return **true** if the item is in the BST and **false** otherwise.

1. How many nodes in the BST do you need to visit when searching for "dog" in the BST in Task 1?
2. How many nodes in the BST do you need to visit when searching for "goat" in the BST in Task 1?
3. How many nodes in the BST do you need to visit when searching for "monkey" in the BST in Task 1?
4. What is the maximum number of nodes you would need to visit when searching for an item in this BST?

Task 2

A *heap* is a binary tree with some additional properties. Discuss these properties.

Task A

Insert the following items (in the order they are listed) into a max-heap:

$$10,\ 16,\ 2,\ 29,\ 6,\ 14,\ 11,\ 4.$$

What value is stored at the top of the heap?

Task B

Insert the following items (in the order they are listed) into a min-heap:

$$10,\ 16,\ 2,\ 29,\ 6,\ 14,\ 11,\ 4.$$

What node is stored at the top of the heap?

Task 3

Discuss how a heap can be used to sort a list. Using a heap, sort the list $[10, 16, 2, 29, 6, 14, 11, 4]$ in ascending order. Show the state of the heap at every step.

Task 4

In this task you will count the number of steps taken by each program to run using the following rules:

Simple Instructions Each read, print, return and assignment instruction takes 1 time step.
Each comparison takes 1 time step.

Sequences of Instructions The running time of a sequence of instructions is the sum of the running times of the statements.

Loops and Functions Loops and functions are considered as the composition of many simple operations, and their running time depends upon how many times each of these simple operations are performed.

Count the number of steps for the following two code fragments:

```
total=0
i=0
while i < 3:
    j=0
    while j< 3:
         total=total+1
         j=j+1
    i=i+1
return total
```

and

```
total=0
i=0
while i< 3:
    j=i+1
    while j<4:
        total=total+1
        j=j+1
    i=i+1
return total
```

Which code fragment takes more steps to run?

Count the number of steps in the function `power`. What is the total number of steps for the program?

```
def power(x,N):
    ans=1
    i=0
    while i<N:
        ans=ans*x
        i=i+1
    return ans


L1=[2, 4, 8]
i=0
while i<3:
    L1[i]=power(i, L1[i])
    i=i+1
print(L1)
```

Puzzle of the week

Given a sorted list of distinct elements where the list is rotated at an unknown position (e.g. $[1, 3, 5, 6, 10]$ is rotated 2 places to right to $[6, 10, 1, 3, 5, 6]$). Find a minimum element in the list with the minimum number of comparisons.