
FUNDAMENTALS OF ALGORITHMS

Graph traversal

Graph traversals

- ✚ Depth first: implemented recursively or using a stack
- ✚ Breadth first: implemented using a queue

Binary tree traversal

- ✚ Pre-order: visit node, traverse left subtree and traverse right subtree
- ✚ In-order: traverse left subtree, visit node and traverse right subtree
- ✚ Post-order: traverse left subtree, traverse right subtree and visit node

Recursion: a technique where a function can call itself in order to complete a task, each time a call is made the current state of the program must be stored on the stack

Dijkstra's shortest path algorithm

1. Give working value 0 at starting vertex
2. Give final label at vertex X, which is the one with lowest working value without final label
3. For all the vertices Y connected to X, if lower than the one present give working value $X + XY$
4. Repeat steps 2 and 3 until the destination receives final value

Search algorithms

- ✚ Linear search: look through data one item at the time until the search term is found
- ✚ Binary search: split the ordered dataset in half repeatedly until the search data is found
- ✚ Binary tree search: traverses the tree until the search item is found

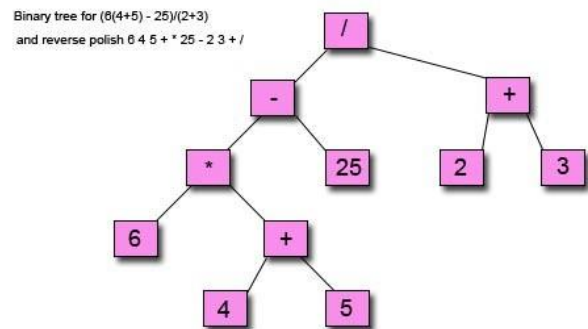
Reverse Polish Notation

3 + 4 - infix expression (normal mathematical expression, needs brackets)



+ 3 4 - prefix expression (or Polish Notation)

3 4 + - post Fix expression also called Reverse Polish Notation

A reverse polish expression can be easily evaluated using a stack: we go through the expression from left to right, every time we come across a number in the expression we push it to the stack, when we come across an operator we pop the correct number of numbers from the stack and substitute them with the result of the operation.



Sorting algorithms

-  Bubble sort: Repeatedly stepping through an array, comparing adjacent elements and swapping them if necessary until the array is in order.
-  Merge sort: 'Divide and conquer' algorithm, splits lists into single elements and then merging them back together again.

Big O notation

The efficiency of a solution is usually measured in terms of time and space:

- *time*: how long does the algorithm take to run
- *space*: how much space (memory) is required by the algorithm




Domain: all the values that may be input to a mathematical function

Codomain: all the values that may be output from a mathematical function. The set of values that are actually produced is called range.

Big O calculated the upper bound of the complexities, the notation refers to the order of growth (aka order of complexity) as the input size increases.

- *Constant time*: time taken to run does not vary with the input size
- *Logarithmic time*: time taken to run an algorithm increases in line with a logarithm
- *Linear time*: time taken in direct proportion with the input size
- *Polynomial time*: time taken to run is a polynomial function of the input size
- *Exponential time*: time taken increases as an exponential function

Tractable and intractable problems

-  Tractable problems: a problem that can be solved in polynomial time
-  Intractable problem: a problem that is theoretically solvable but not in polynomial time (e.g. travelling salesman) to face these problems programmers produce heuristic algorithms which find in approximate solutions.
-  Unsolvable problem: a problem that has been proved cannot be solved on a computer (e.g. halting problem)