Data structures: How data is represented

Data types

List
Tuple
Set
Dict
Queue
Int
Str

Stack

- Push()
- Pop()

LIFO

Push & Pop are to top of the stack

- Peek()
- Clear()

undo

Queue

- Enqueue()
- Dequeue()

FIFO

enque to end
deque from start

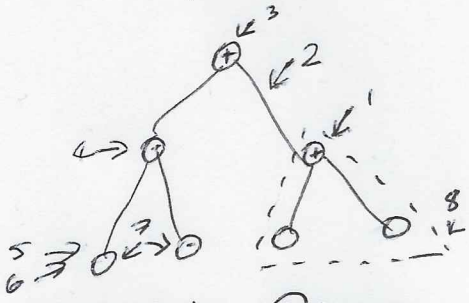
Supermarket checkout

Linked List

- root
- size

- find ()
- add ()
- remove()

TREE



1. node, Vertex
2. edge, Link
3. root
4. Parent
5. child
6. Leaf
7. siblings
8. sub tree

- Desendants
+ Ancestors

Graph

Undirected —
Directed →
Simple o—o
Non simple o—o
Weighted o—5—o
Dense
cyclic clockwise
acyclic both dir
topological
Labelled A o—o B

Adj list

List of neighbours in each node/vertex

A : B, C
B : C
C : A

Pros
faster and less space for sparse graphs
Con
slow on dense graphs

Adj matrix

matrix of neighbours stored centrally in graph obj

0	1	1
0	0	1
1	0	0

1. weights replace 1

Pros
fast on dense graphs
simpler for weighted
cons
uses more space

Edge List

Stores link of all edges

NodeA, NodeB
NodeA, NodeC
NodeB, NodeC

numbers = [4, 30, 2, 10]
Other = numbers

numbers ▫ → [0: 4 | 1: 30 | 2: 10]
other ▫ →

ADT

Array	Queue	table
List	set	Tree
Map	Stack	vector

Linked List

Single
double
Circular
doubly circular

Primative Data types

Primative
- Int
- float
- String
- Bool

Non-Prim
- Array
- List
- Tuple
- Dict
- Set
- file

Linear
stack
queue

non-lin
Graph
tree

ADT is the operations that can be performed

Tree

- Outline
- Nested bracket
- Nested area
- Icicle
- Node/Link
- Parent pointers

Tree Data Struct

- List of List
- List of children
- Parent pointer

Data Structures

- Array List
- Linked List
- Parent pointer tree
- Adj List
- List-of-Lists
- edge list
- List-of-children
- Tree