### Algorithm review

By Richard

**Syllabus:**

### Searching and sorting algorithms

* Elementary searching – Linear and binary search
* Binary search trees.
* Hash Tables – Hashing, Hash functions

### Graph algorithms

* Directed acyclic graphs.
* PERTchart
* Dijkstra’s algorithm
* Bellman-Ford algorithm
* Floyd-Warshall algorithm

### Text algorithms

* String matching
* Longest Common Sequence.
* Sequence/subsequence

### Analysis of algorithms

* Algorithmic complexity.
* Upper and lower bounds & big O notation.
* Best, worst and average cases.
* Classifications of algorithms.

### Linked Lists

* Linked structures
* Singly-linked list, Doubly-linked list
* Circular linked list

### Abstract Data Types

* ADTs
* Defining & Implementing the ADT
* Bag ADT
* List-based Implementation
* Queue ADT

### Dynamic Programming

* Elements of dynamic programming
* Rod cutting
* Matrix-chain multiplication,
* Optimal binary search trees

### Recursion

* Properties
* Run time stack, tail recursion
* Recursive applications, recursive binary search, exponential operation

### Analysis of searching and sorting algorithms

* Binary search.
* Selection sort, Insertion sort, mergesort, quicksort, radix sort
* Bubble sort, lower bounds for sorting
* Working with sorted lists
* Heapsort, building a heap, heaspsort algorithm, priority queues

### Greedy Algorithms

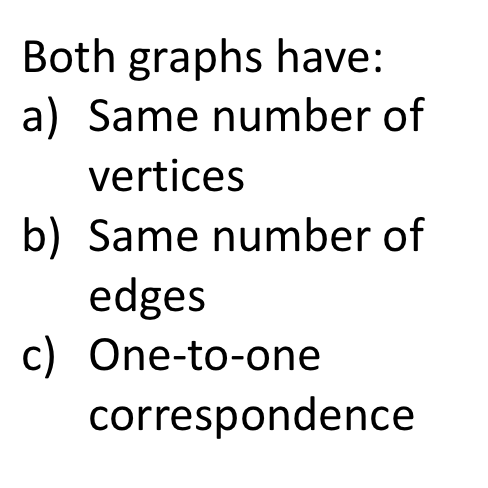
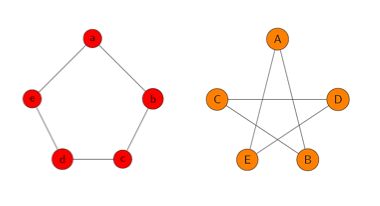
* Elements of greedy strategy
* Huffman codes
* Matroids

### Graph algorithms

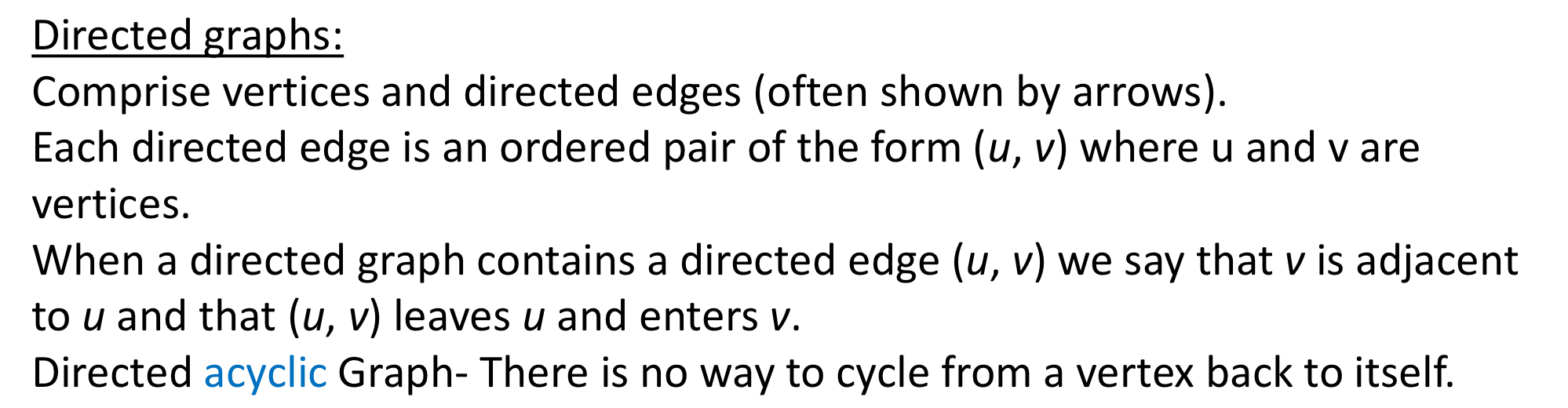
Definition of graph:

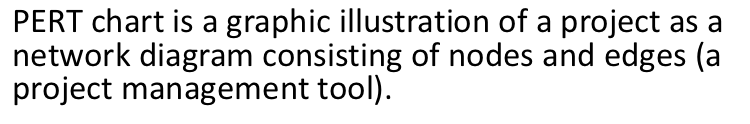
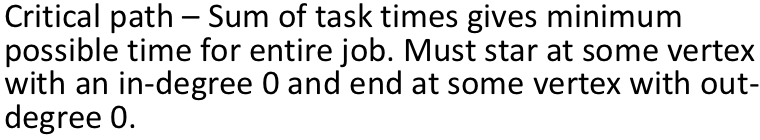
Data structure representation of connected nodes (vertices) by various edge.

Isomorphic graphs:

 Any two graphs which follow the three properties below are isomorphic.

Directed acyclic graphs.



PERT chart

Dijkstra’s algorithm

1) Create a set sptSet (shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.

2) Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.

3) While sptSet doesn’t include all vertices

a) Pick a vertex u which is not there in sptSet and has minimum distance value.

b) Include u to sptSet.

c) Update distance value of all adjacent vertices of u.

To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

Bellman-Ford algorithm

1) This step initializes distances from source to all vertices as infinite and distance to source itself as 0. Create an array dist[] of size |V| with all values as infinite except dist[src] where src is source vertex.

2) This step calculates shortest distances. Do following |V|-1 times where |V| is the number of vertices in given graph.

a) Do following for each edge u-v

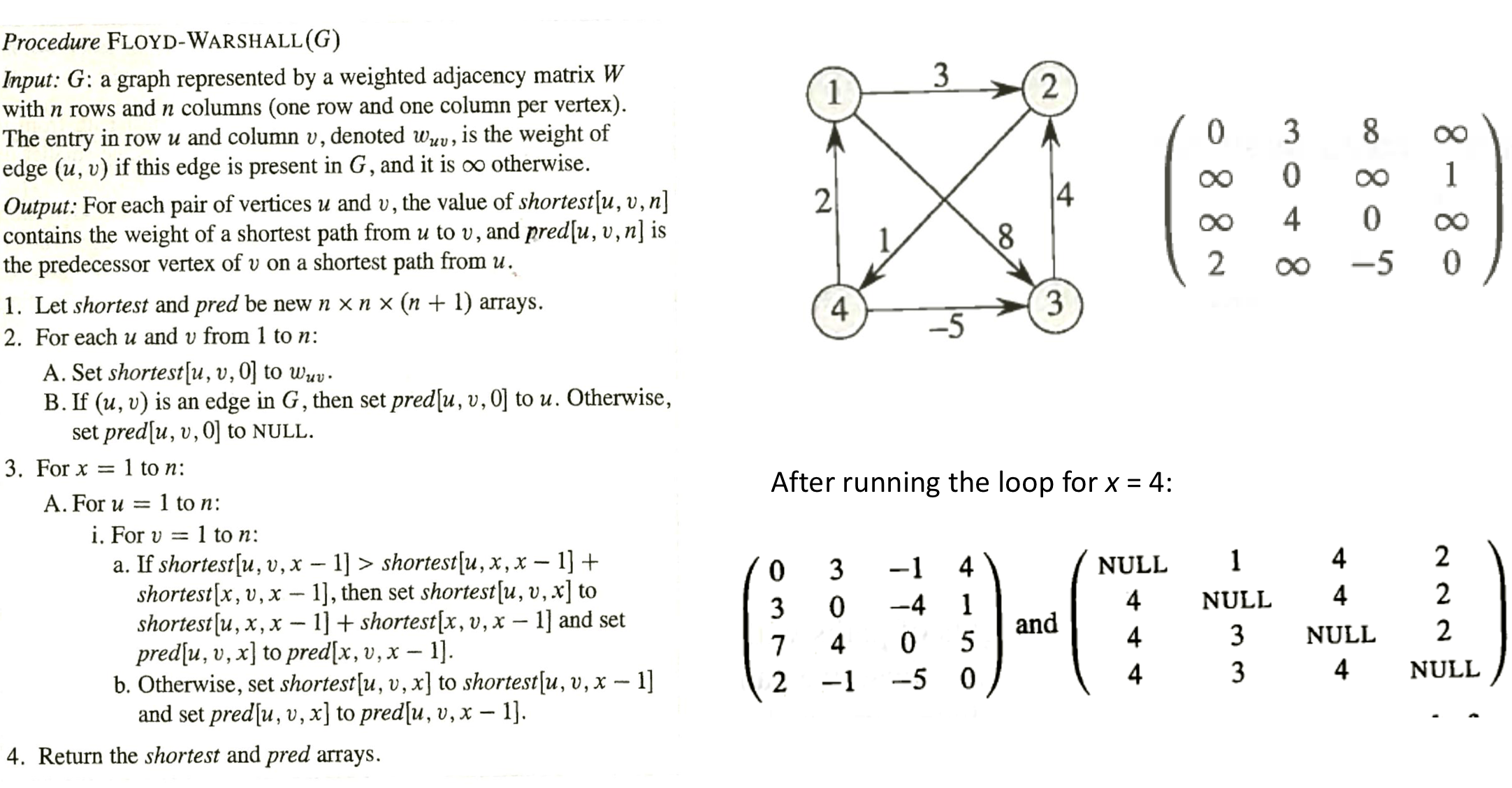
If dist[v] > dist[u] + weight of edge uv, then update dist[v]

dist[v] = dist[u] + weight of edge uv

3) This step reports if there is a negative weight cycle in graph. Do following for each edge u-v

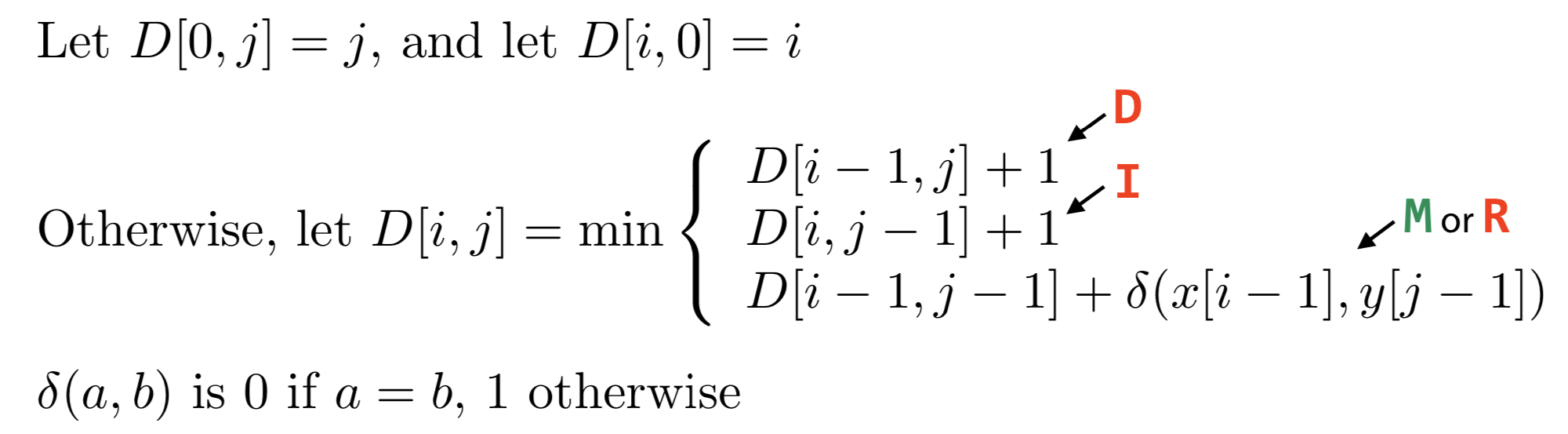
……If dist[v] > dist[u] + weight of edge uv, then “Graph contains negative weight cycle”

The idea of step 3 is, step 2 guarantees shortest distances if graph doesn’t contain negative weight cycle. If we iterate through all edges one more time and get a shorter path for any vertex, then there is a negative weight cycle

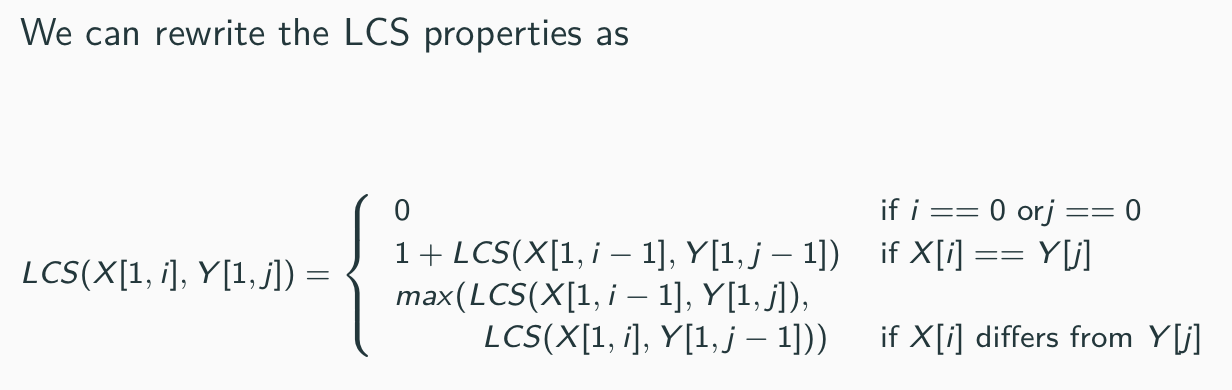
Floyd-Warshall algorithm

### Articulation point

### Text algorithms

String matching

Longest Common Sequence.

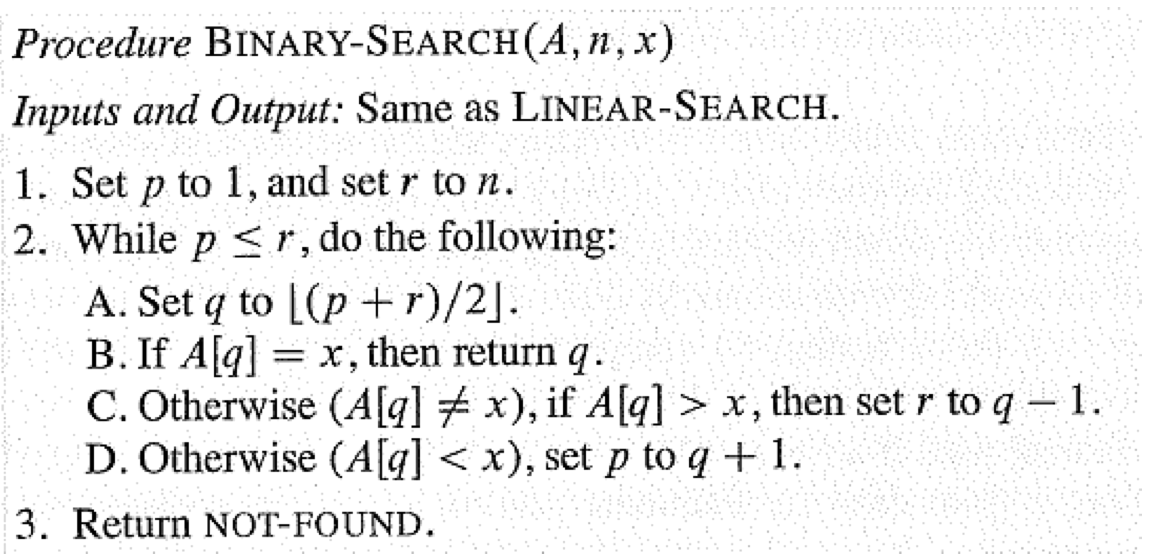


Sequence/subsequence

### Analysis of searching and sorting algorithms

Binary search

Procedures:



Best case: running time of (1)

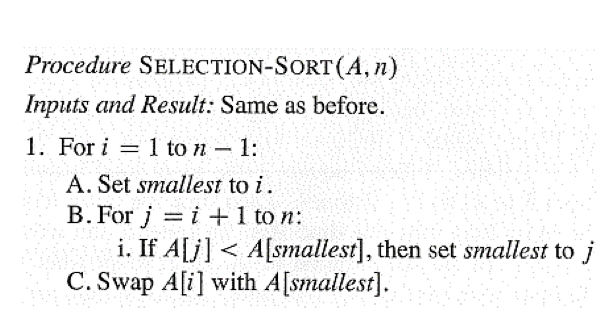
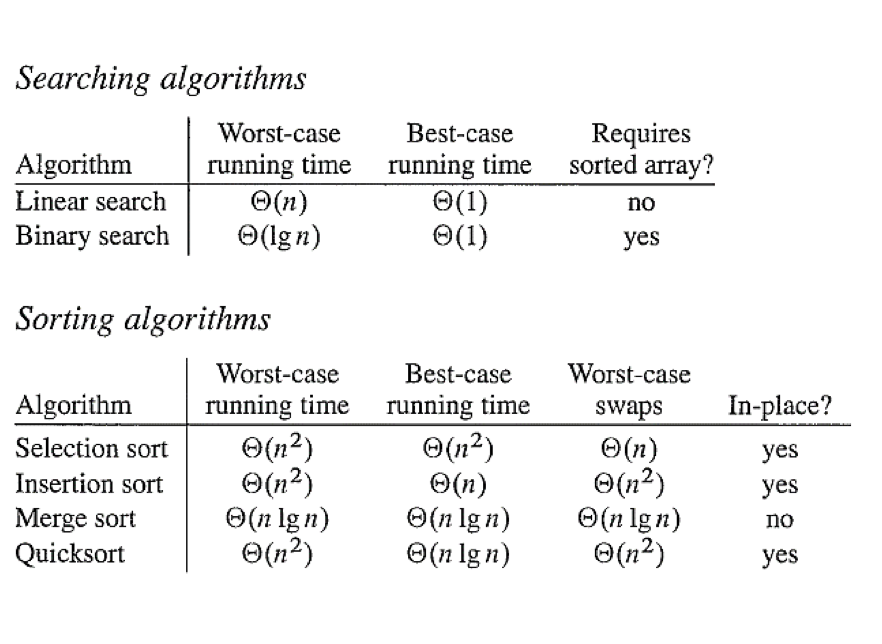


Worst case: running time of (log2(n))



General case: running time of (log2(n))



Selection sort, Insertion sort, merge sort, quick sort, radix sort

Bubble sort, lower bounds for sorting

Working with sorted lists

Heap sort, building a heap, heap sort algorithm, priority queues

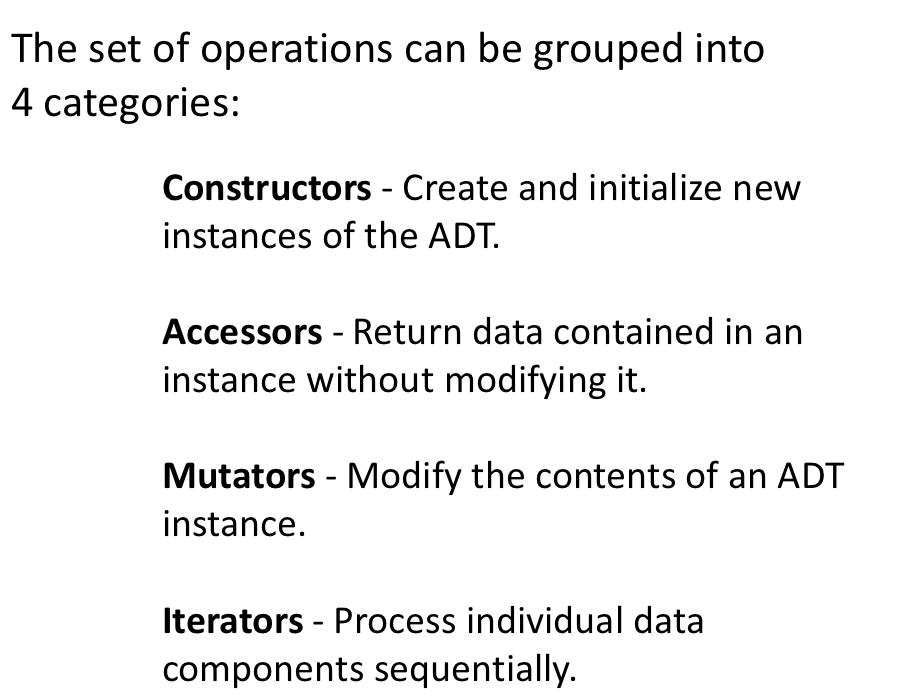
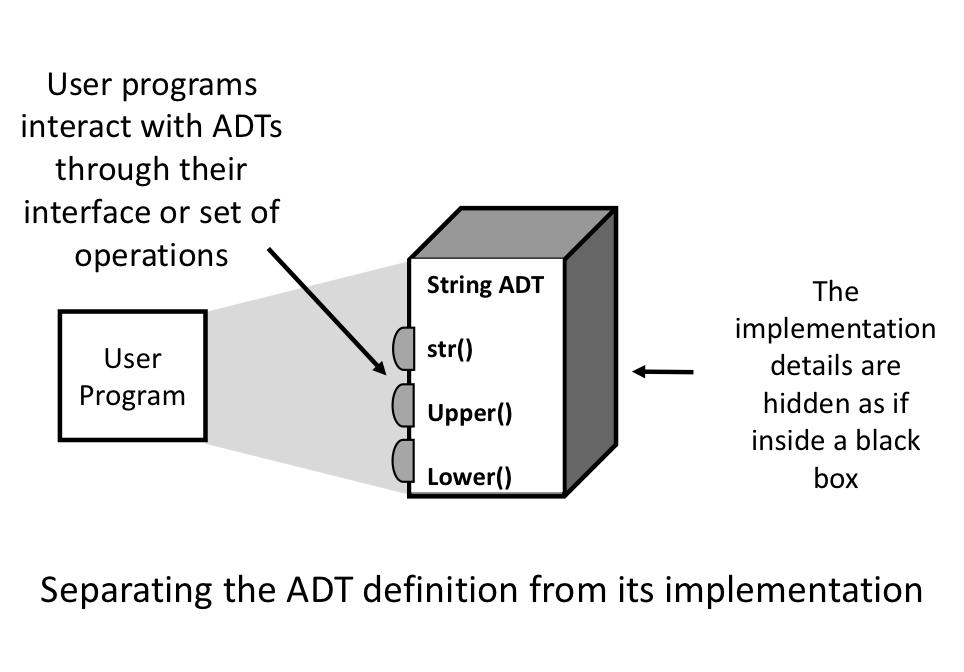
### Abstract Data Types

Defining the ADT

**Abstract data type is a program-defined data type that specifies a set of data values and a collection of well-defined operations that can be performed on those values.**

Implementing the ADT

**User programs interact with the instances of ADT by invoking on of the several operations defined in its interface**

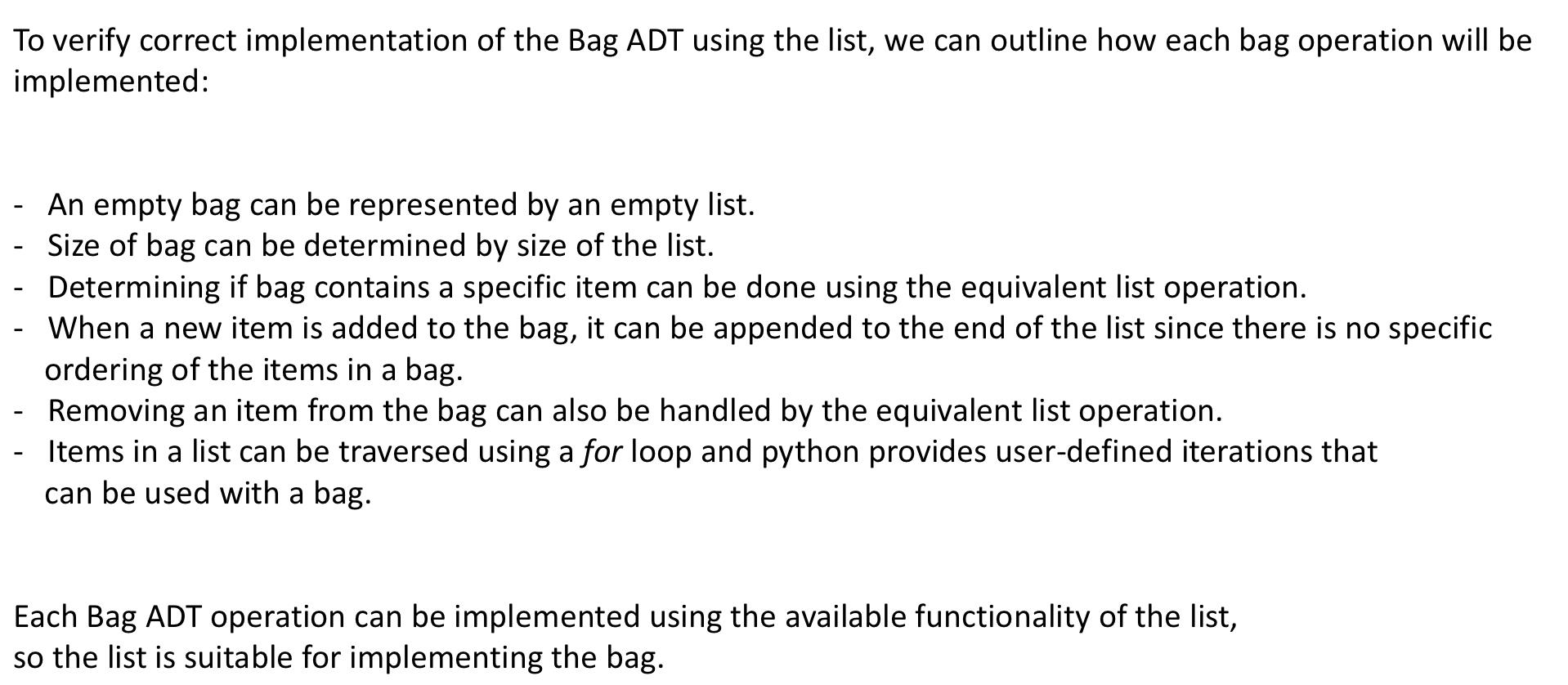


Bag ADT

**Bag ADT- is a container that stores a collection in which duplicate values are allowed.**

**The items, each of which is individually stored, have no particular order but they must be comparable.**

List-based Implementation



Queue ADT

**A queue is a specialized list with a limited number of operations in which items can only be added to one end and removed from the other.**

**A queue is also known as a first-in, first-out (FIFO) list.**

**Can be implemented as a circular array.**

### Linked Lists

Linked structures

**Linked structures contains a collection of objects called node, each of which contains data and at least one reference or link to another node.**

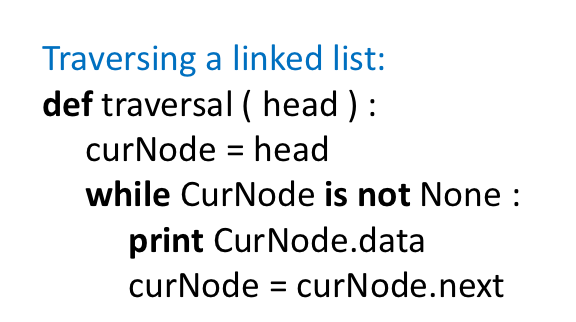
Singly-linked list, Doubly-linked list

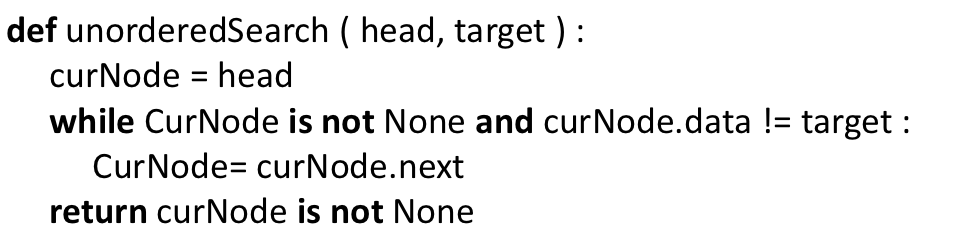
**Singly-linked list: Linear structure in which traversals start at front and progress one element at a time to the end. Nodes are linked in a single direction and sequence - each node contains a single link field and allows for a complete traversal from a distinctive first node to the last.**

**Doubly-linked list: Each node contains not only the data component and a link to next node as in singly linked list, but also a second link that points to preceding node.**



**Traversing a single linked list:**



**Searching in a single linked list:**

//Code below are written by myself, which is not accurate.

**Adding to a single linked list:**

def add(element):

newNode=new Node();

newNode.data=element;

newNode.next=head;

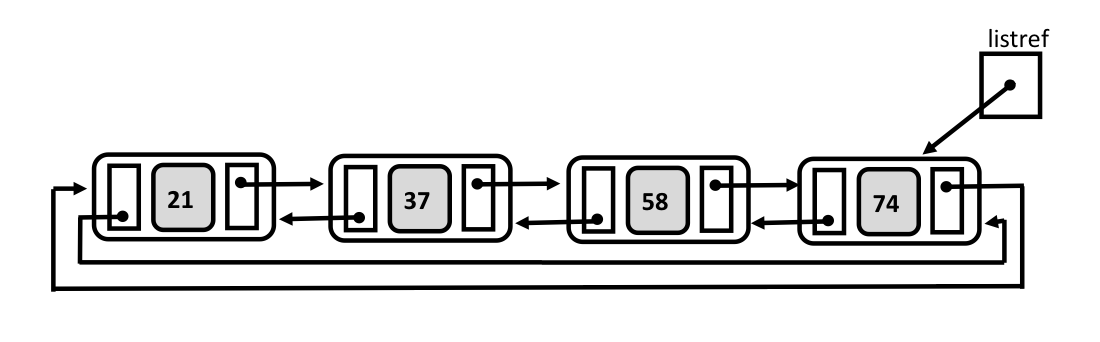
head=newNode;

**Remove from a single linked list:**

…

**Double linked list shares the same ideal with single, but it has a second link point to the precious node.(And it also has “tail” besides “head”)**

Circular linked list

**An example of double-linked circular list:**

**Operation of circular linked list can be concluded from non-circular linked list, except the list reference at the end of list, which indicates the starting point of a loop.**

**Big O Notation**

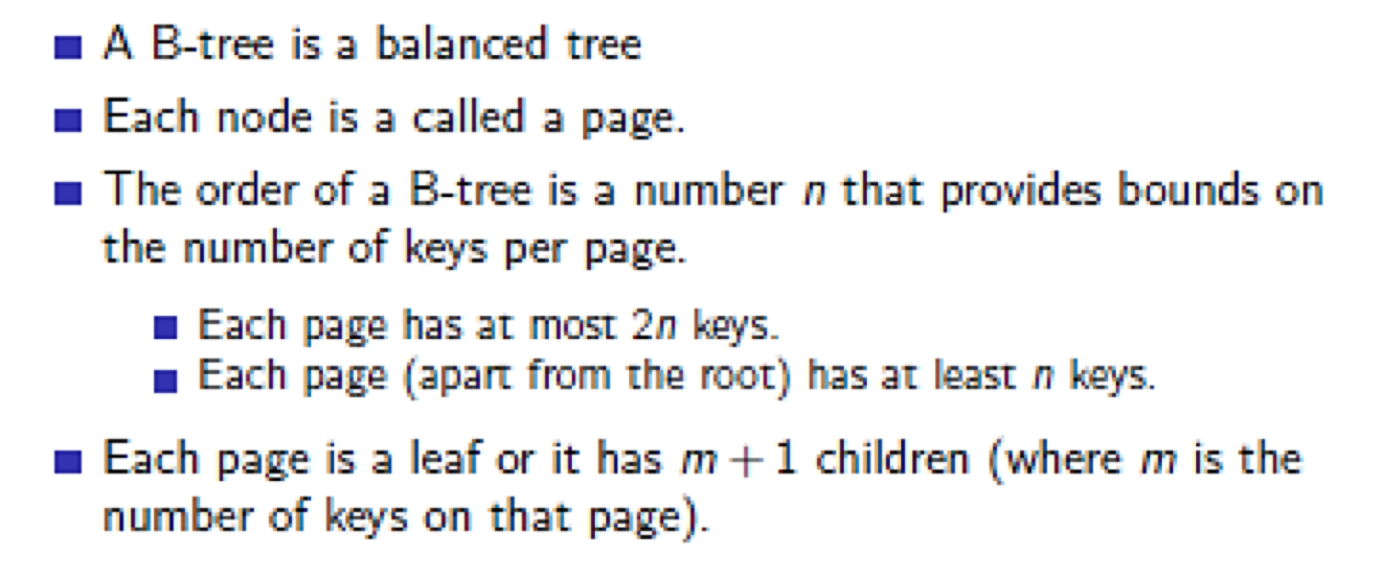
Algorithmic complexity

Upper and lower bounds

Big O notation

Best, worst and average cases

Classifications of algorithms

B-TREE

**Steps to build Huffman Tree**

**Input is array associated with unique characters with their frequency associated with occurrences as well as output is Huffman Tree.**

**Step-1: Make a leaf node for every unique character as well as develop a min heap of all leaf nodes.**

**Step-2: Get two nodes using the minimum frequency from the min heap.**

**Step-3: Make a new internal node together with frequency corresponding to the sum the two nodes frequencies. Help make the initial extracted node as its left child and the other extracted node because it's right child. Include this particular node to the min heap.**

**Step-4: Do it again steps2 and steps3 prior to the heap consists of just one node. The remaining node may be the root node and the tree is complete.**