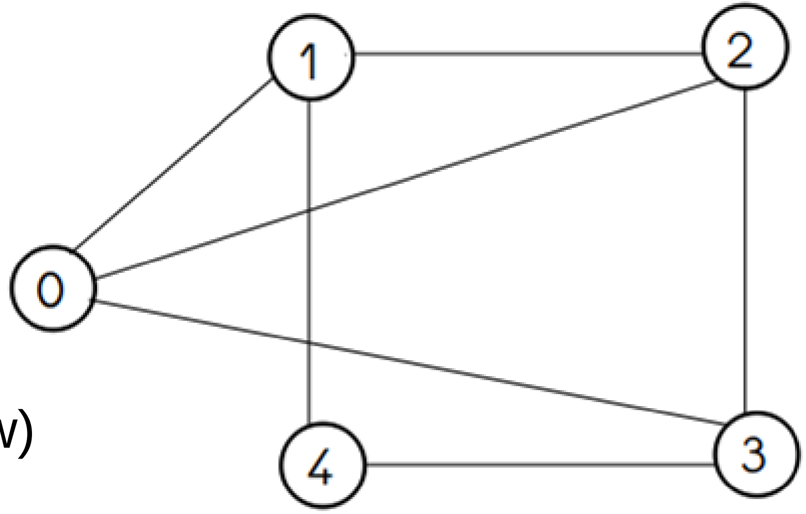
Data Structures: Graph

Graph: Graph is **non-linear** data structure, defined as set of vertices and edges.

- Vertices (or nodes) holds the data.
- Edges (or arcs) represent relation between vertices.
  - Edges may have direction and/or value assigned to them called as weight or cost.
- Applications of graph
  - Electronic circuits
  - Social media
  - Communication network
  - Road network
  - Flight/Train/Bus services
  - Bio-logical & Chemical experiments
  - Deep learning (Neural network, Tensor flow)
  - Graph databases (Neo4j)

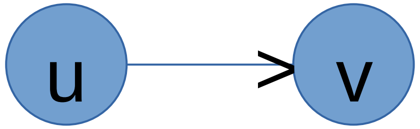
G(V,E): V={0,1,2,3,4}; E={ (0,1),(0,2),(0,3),(1,2),(1,4),(2,3),(3,4) }



Data Structures: Graph

- If there exists a direct edge between two vertices then those vertices are referred as an **adjacent vertices** otherwiese **non-adjacent**.

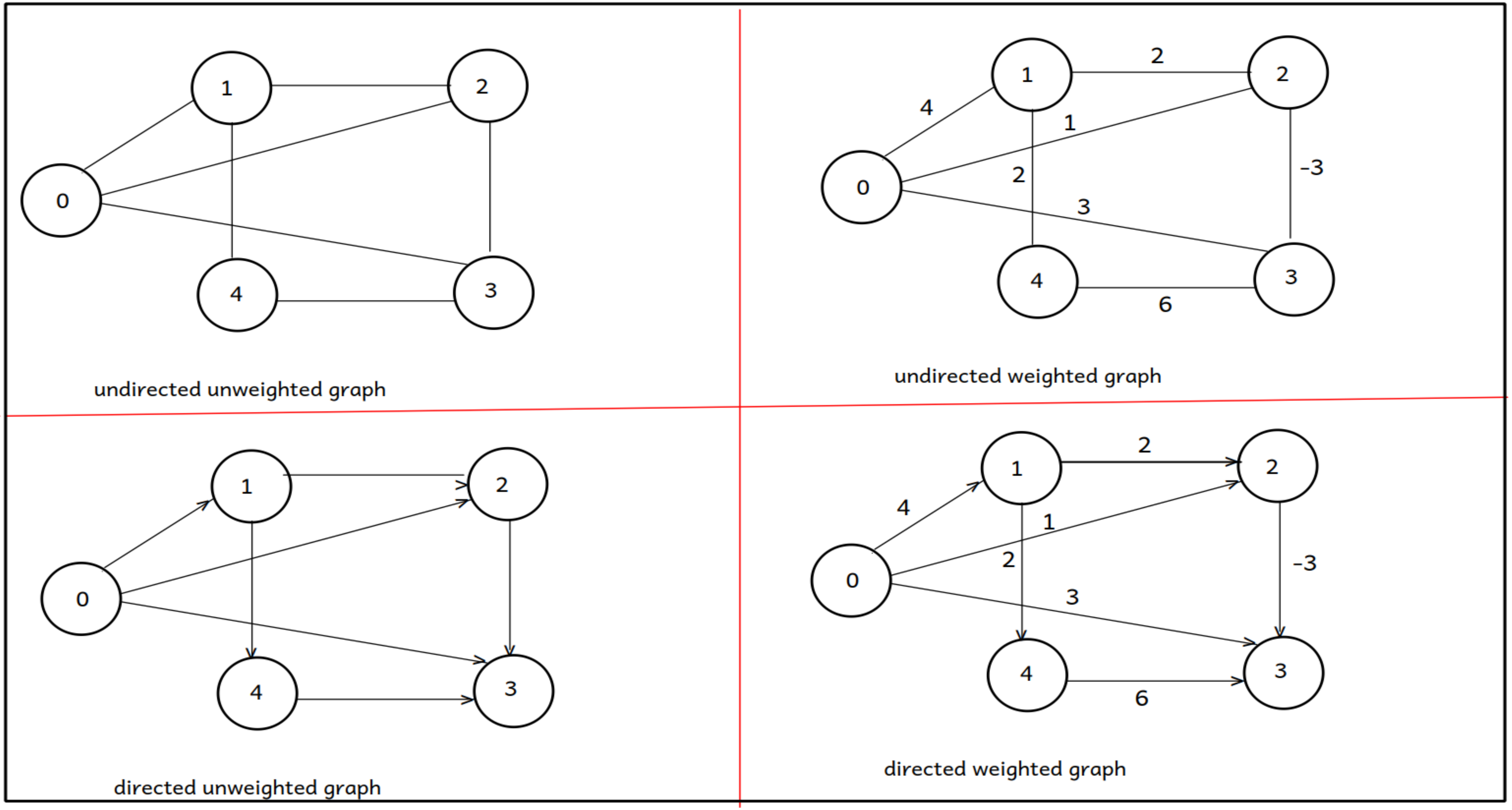
(u, v) == (v, u)



(u, v) != (v, u)

- If we can represent any edge either (u,v) OR (v,u) then it is referred as **unordered pair of vertices i.e. undirected edge.**
- **e.g. (u,v) == (v,u) => unordered pair of vertices => undirected edge => graph which contanis undirected edges referred as undirected graph.**
- If we cannot represent any edge either (u,v) OR (v,u) then it is referred as an **unordered pair of vertices** i.e. directed edge.
- **<u, v> != <v, u> => ordered pair of vertices => directed edge -> graph which contains set of directed edges referred as directed graph (di-graph).**

Data Structures: Graph

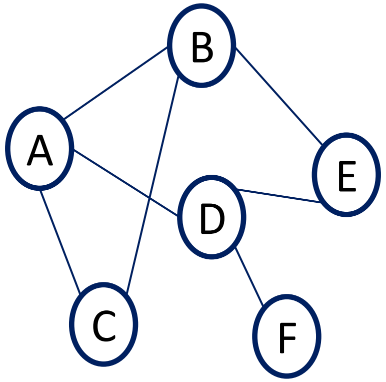


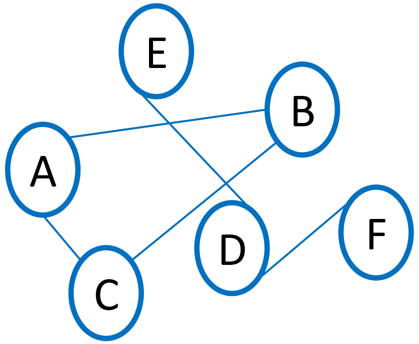
Data Structures: Graph

- **Path:** Path is set of edges connecting two vertices.
- **Cycle:** in a given graph, if in any path starting vertex and end vertex are same, such a path is called as a cycle.
- **Loop:** if there is an edge from any vertex to that vertex itself, such edge is called as a loop. Loop is the smallest cycle.
- **Connected Vertices:** if there exists a direct/indirect path between two vertices then those two vertices are referred as a connected vertices otherwise not-connected.
    - Adjacent vertices are always connected but vice-versa is not true.

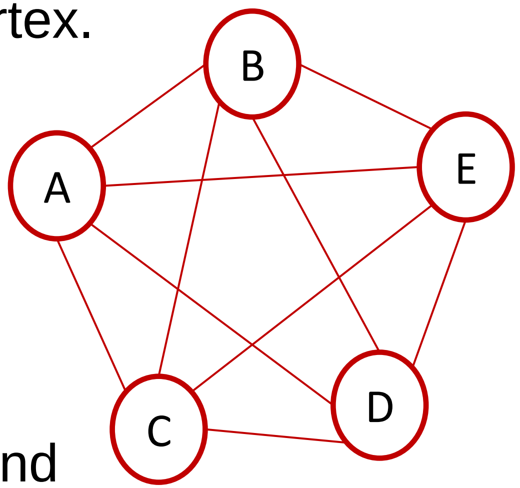
Graph

- Connected graph
  - From each vertex some path exists for every other vertex.
  - Can traverse the entire graph starting from any vertex.

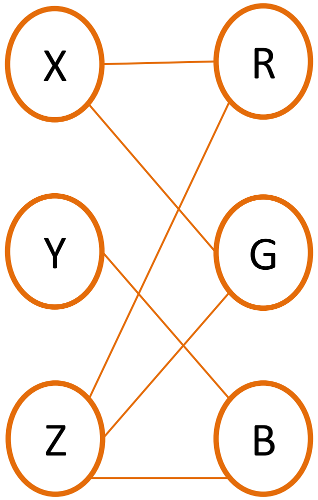




- Complete graph
  - Each vertex of a graph is adjacent to every other vertex.
  - Un-directed graph: Number of edges = n (n-1) / 2
  - Directed graph: Number of edges = n (n-1)

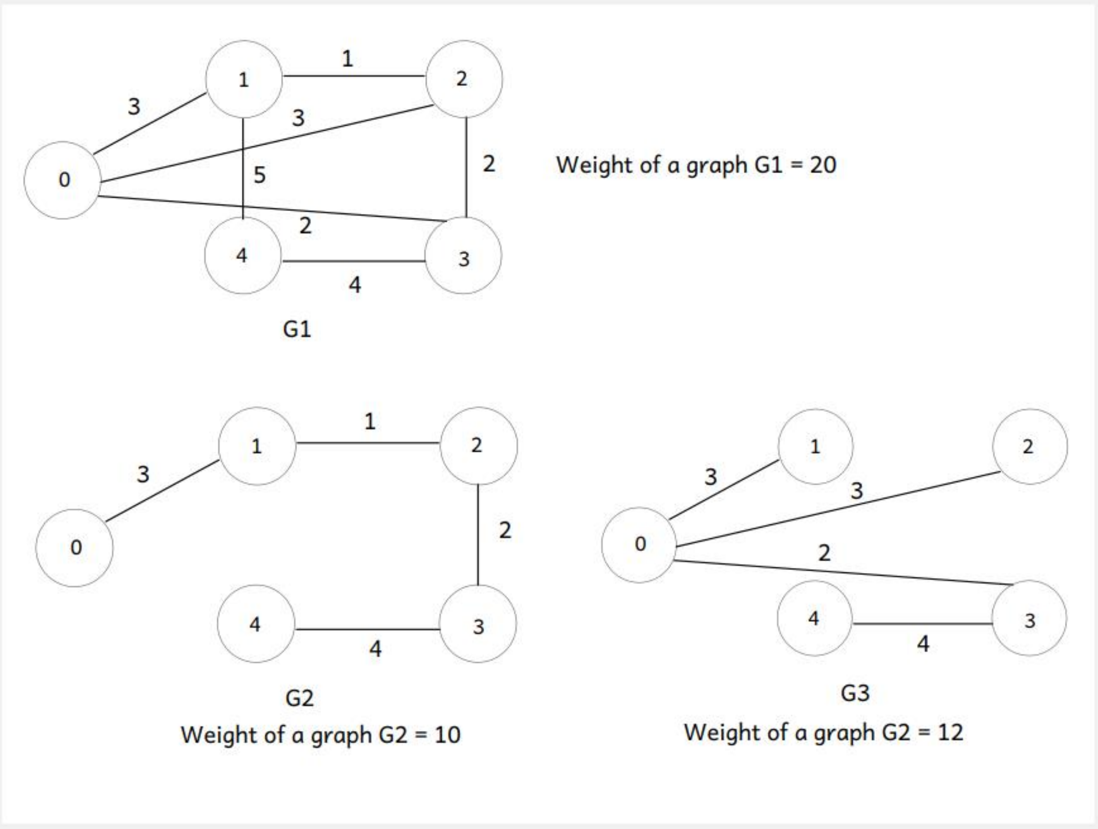


- Bi-partite graph
  - Vertices can be divided in two disjoint sets.
  - Vertices in first set are connected to vertices in second set.
  - Vertices in a set are not directly connected to each other.



Data Structures: Graph

Spanning Tree:



- **Weight of a graph =** sum of weights of all its edge.
- **Spanning Tree:**
  - Connected subgraph of a graph.
  - Includes all V vertices and V-1 edges.
  - Do not contain cycle.
  - A graph may have multiple spanning trees.
- **Minimum Spanning Tree:** Spanning tree of a given graph having minimum weight.
  - Used to minimize resources/cost.
  - **MST Algorithms**:
    - Prim’s Algorithm => O(E log V)
    - Kruskal’s Algorithm => O(E log V)

Data Structures: Graph

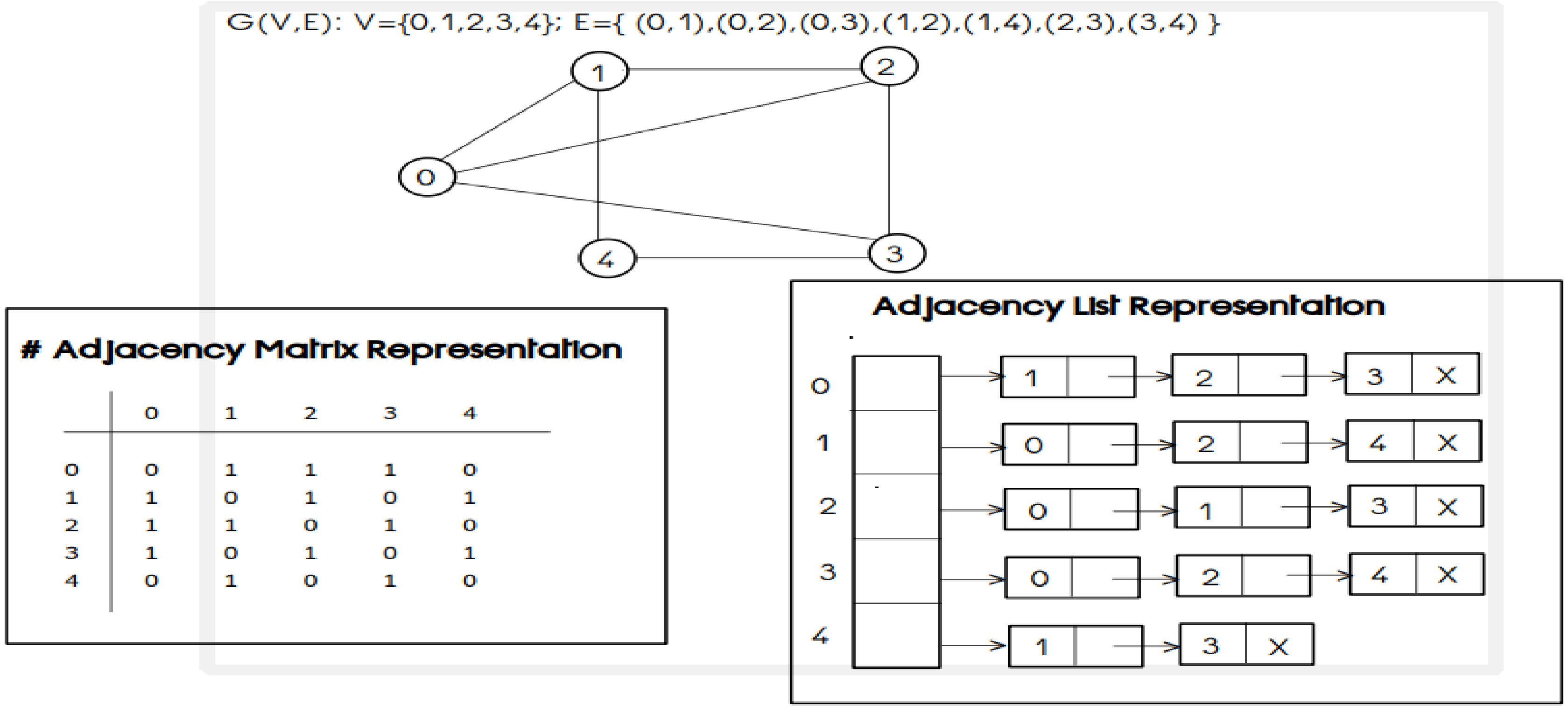
- **Graph Traversal Algorithms:**
  - Used to traverse all vertices in the graph.
  - DFS Traversal (using Stack) and BFS Traversal (by using Queue)
- **Shortest Path Algorithm:**
  - Single source SPT algorithm used to find minimum distance from the given vertex to all other vertices.
  - Dijsktra’s Algorithm (Doesn’t work for -ve weight edges) => O(V log V).
  - Bellman Ford Algorithm => O(VE).
- **All pair Shortest Path Algorithm:**
  - To find minimum distance from each vertex to all other vertices.
  - Floyd Warshall Algorithm => $O(V^3)$
  - Johnson’s Algorithm => $O(V^2 \log V + VE)$

Data Structures: Graph

There are two graph representation methods:

1. Adjacency Matrix Representation (2-D Array)

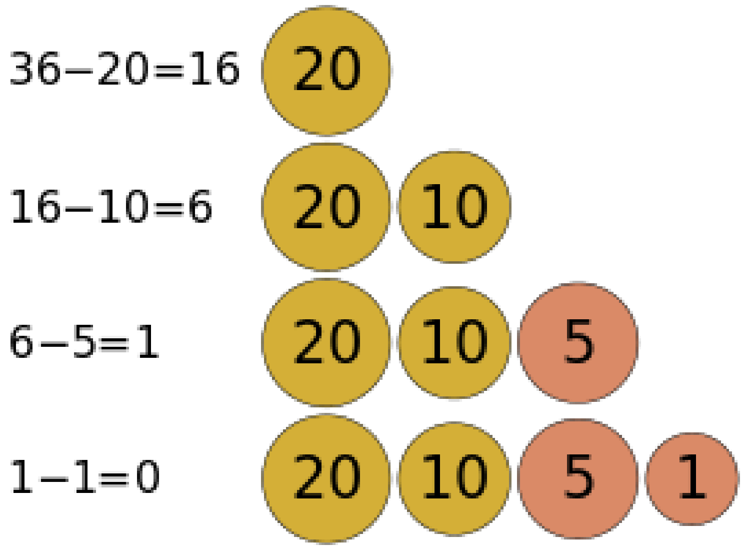
2. Adjacency List Representation (Array of Linked Lists)



	0	1	2	3	4
0	0	1	1	1	0
1	1	0	1	0	1
2	1	1	0	1	0
3	1	0	1	0	1
4	0	1	0	1	0

Problem solving technique: Greedy approach

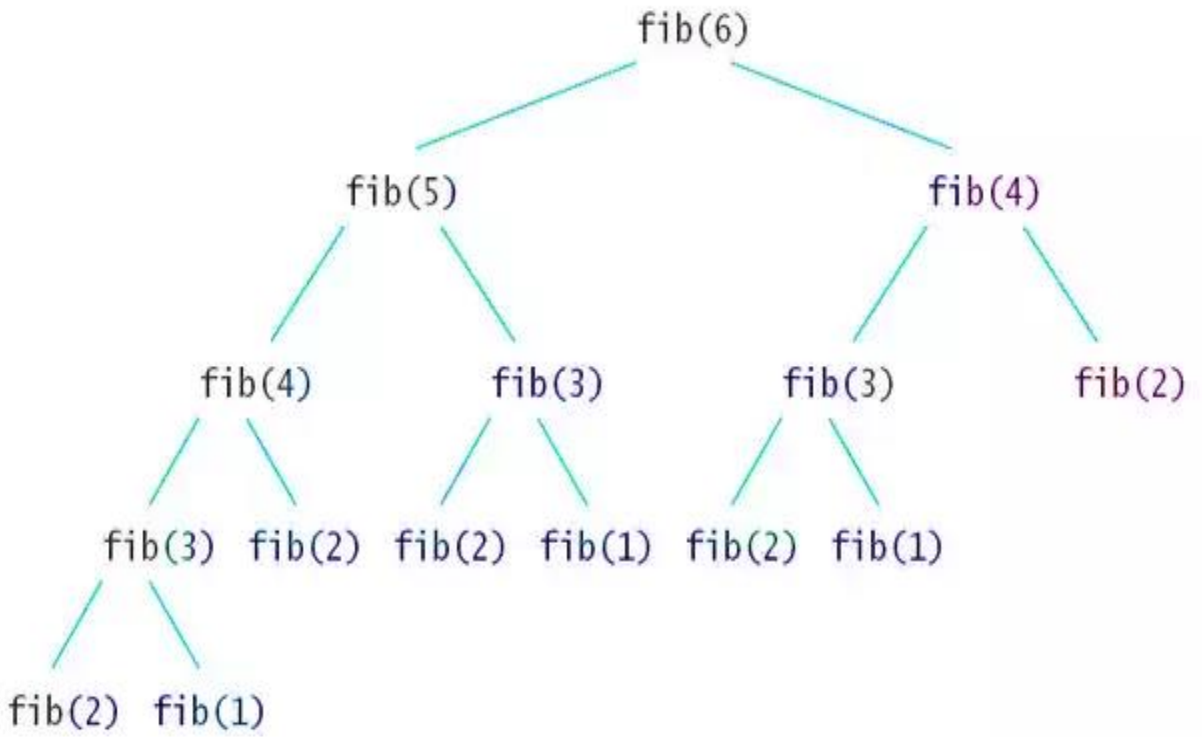
- A greedy algorithm is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage with the intent of finding a global optimum.
- We can make choice that seems best at the moment and then solve the sub-problems that arise later.
- The choice made by a greedy algorithm may depend on choices made so far, but not on future choices or all the solutions to the sub-problem.
- It iteratively makes one greedy choice after another, reducing each given problem into a smaller one.
- A greedy algorithm never reconsiders its choices.
- A greedy strategy may not always produce an optimal solution.



- Greedy algorithm decides minimum number of coins to give while making change.

Dynamic Programming

- Dynamic programming is another optimization over recursion.
- Typical DP problem give choices (to select from) and ask for optimal result (maximum or minimum).
- Technically it can be used for the problems having two properties
  - Overlapping sub-problems: To solve problem, we need to solve its sub-problems multiple times.
  - Optimal sub-structure: Optimal solution of problem can be obtained using optimal solutions of its sub-problems.
- DP solution is bottom-up approach.
- DP use 1-d array or 2-d array to save state.



0	1	2	3	4	5	6	7
	1	1	2	3	5	8	

Data Structures: Hash Table

- **HashTable:** Hash table is an **associative** data structure in which data is stored in **key-value pairs** so that for the given key value can be searched in minimal possible time. Ideal time complexity is **O(1)**. Internally it is an array (table) where values can accessed by the index (slot) calculated from the key.
- **Hash Function:** It is mathematical function of the key that yields slot of the hash table where key-value is stored. Simplest example is: f(k) = k % size.
- **Collision:** There is possibility that two keys result in same slot. This is called collision and must be handled using some **collision handling technique.** It is handled by Open addressing or Chaining.

Hashing Input										
insert values => 50, 700, 76, 85, 92, 73, 101										
Hash Function	**Key % 7**			**50%7=1**	**700%7=0**	**76%7=6**	**85%7=1**	**92%7=1**	**73%7=3**	**101%7=3**
	Hash Table with Capacity = 7									
	slot									
	0	**700**								
	1	**50**		**collision**						
	2									
	3	**73**		**collision**						
	4									
	5									
	6	**76**								

Data Structures: Hash Table

- **Open Addressing:**
  - All key-value pairs are stored in the hash table itself.
  - If key (to find) is not matching with the key in the slot calculated by hash function, it is probed in next possible slot using one of the following.
    - **Linear Probing:** In linear probing, if collision occurs next free slot will be searched/probed linearly.
    - **Quadratic Probing:** In quadratic probing, if collision occurs next free slot will be searched/probed quadratically.
- **Double Hashing:** In double hashing, if collision occurs next free slot will be searched/probed by using another hash function, so two hash functions can be use to find next/probe next free slot.

Hashing Input				Open Addressing				
insert values => 50, 700, 76, 85, 92, 73, 101								
				50%7=1	700%7=0	76%7=6	85%7=1	92%7=1
Hash Functi	Key % 7							
	Hash Table with Capacity = 7							
	slot							
	0	700						
	1	50			clustering			
	2	85						
	3	92						
	4							
	5							
	6	76						

Data Structures: Hash Table

- **Load Factor = n / m**
  - n = Number of key-value pairs to be inserted in the hash table
  - m = Number of slots in the hash table
  - If n < m, then load factor < 1
  - If n = m, then load factor = 1
  - If n > m, then load factor > 1
- **Limitations of Open Addressing**
  - Open addressing requires more computation.
  - Cannot be used if load factor is greater than 1 (i.e. number of pairs are more than number of slots in the table).

Data Structures: Hash Table

- **Chaining:**
  - Another collision handling technique.
  - Each slot of hash table holds a collection of key-values for which hash value of keys are same.
  - This collection in each slot is also referred as bucket.
  - Chaining is simple to implement, but requires additional memory outside the table.