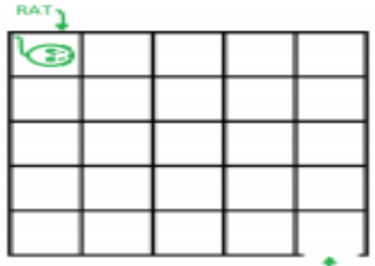
Application of Data Structures

Applications of stack:

- Expression evaluation and syntax parsing
- Balancing of symbols, paranthesis
- Infix to Postfix /Prefix conversion
- Reverse a String using Stack
- Redo-undo features at many places like editors, photoshop.
- Forward and backward feature in web browsers
- Used in many algorithms like Tower of Hanoi, tree traversals, stock span problem, histogram problem.

Applications of stack:

- **Backtracking**
  - Backtracking is an algorithmic-technique for
    - solving problems recursively by trying to build a solution incrementally, one piece at a time,
    - removing those solutions that fail to satisfy the constraints of the problem at any point of time
  - Other applications can be Backtracking, Knight tour problem, rat in a maze, N queen problem and sudoku solver



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

MAZE MATRIX

- **Backtracking-F**inding the correct path in a maze.

    - There are a series of points, from the starting point to the destination.

    - We start from one point. To reach the final destination, there are several paths.

    - Suppose we choose a random path. After following a certain path, we realise that the path we have chosen is wrong. So we need to find a way by which we can return to the beginning of that path.

- **Backtracking-F**inding the correct path in a maze.
    - This can be done with the use of stacks. With the help of stacks, we remember the point where we have reached.
    - This is done by pushing that point into the stack.
    - In case we end up on the wrong path, we can pop the top point from the stack and thus return to the last point and continue our quest to find the right path.

Applications of stack:

- **Depth-first search**
  - The prototypical example of a backtracking algorithm is depth-first search, which finds all vertices of a graph that can be reached from a specified starting vertex.

- In Graph Algorithms like Topological Sorting and Strongly Connected Components

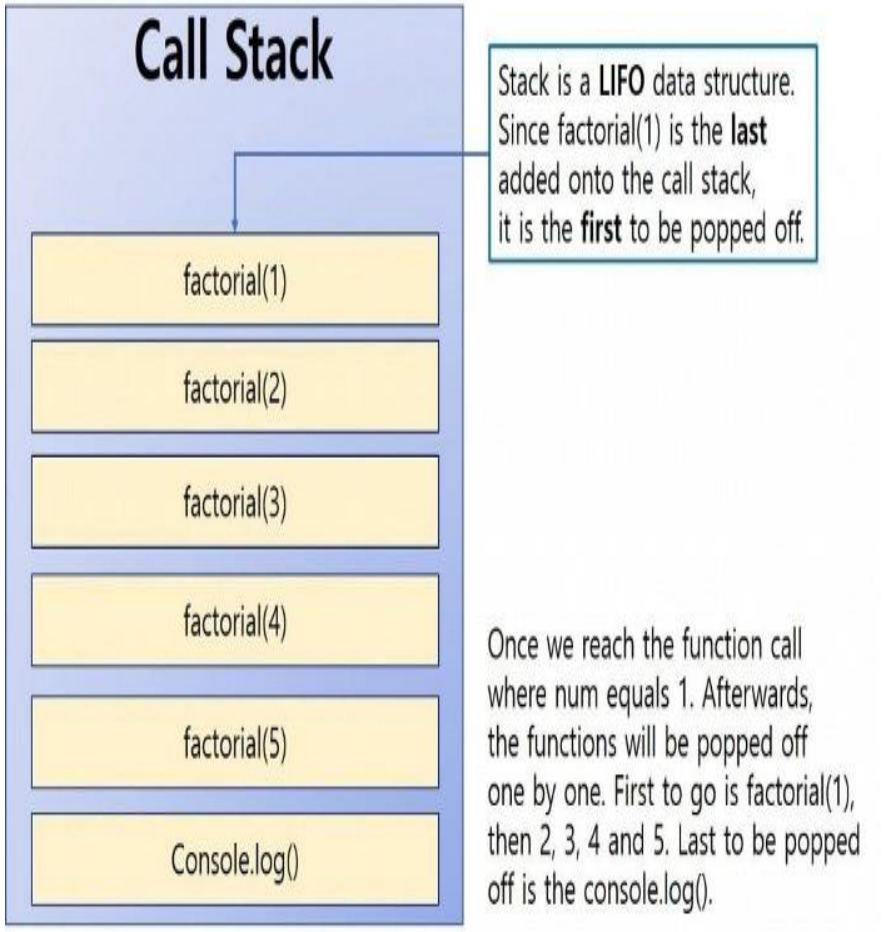
Applications of stack:

- **Compile time memory management**
- Almost all calling conventions

Function calls using stack

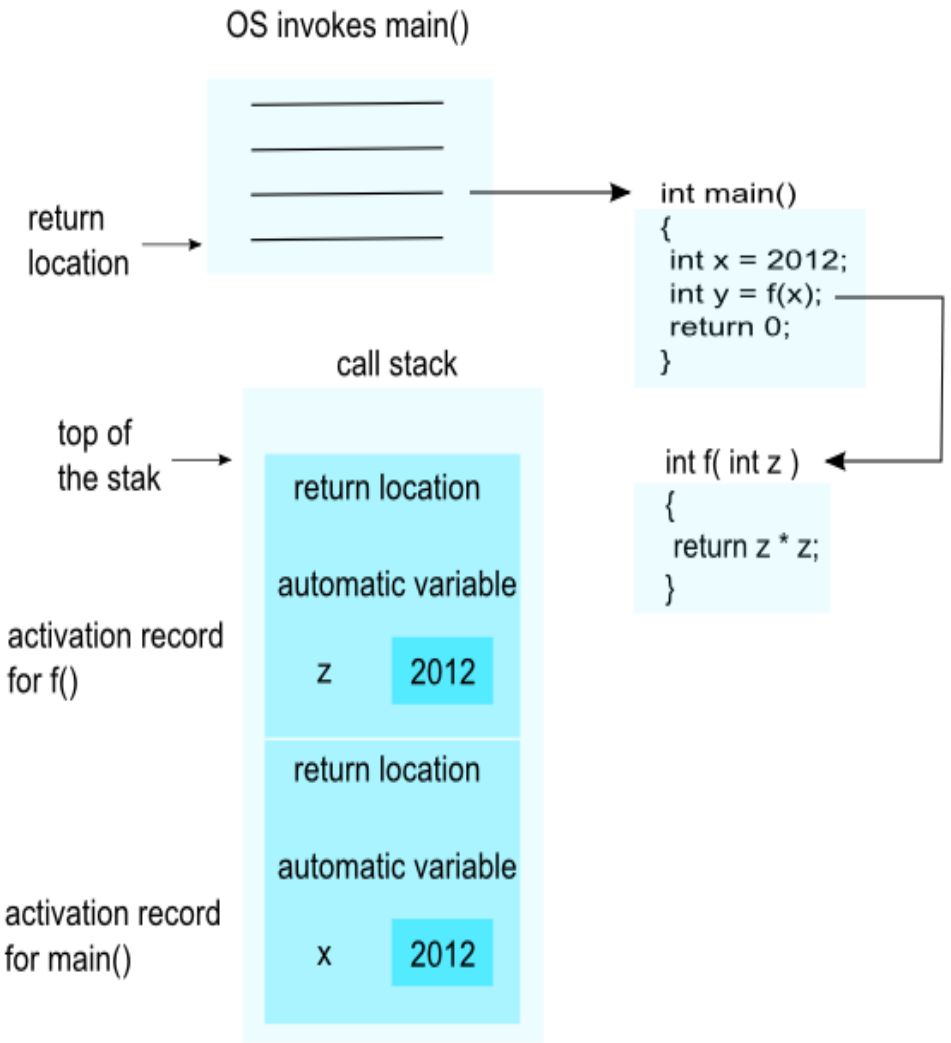
- **Compile time memory management**
  - A number of programming languages are stack-oriented,
    - meaning they define most basic operations as taking their **arguments from the stack, and placing any return values back on the stack.**
    - Eg- adding two numbers, printing a character

Function calls using stack



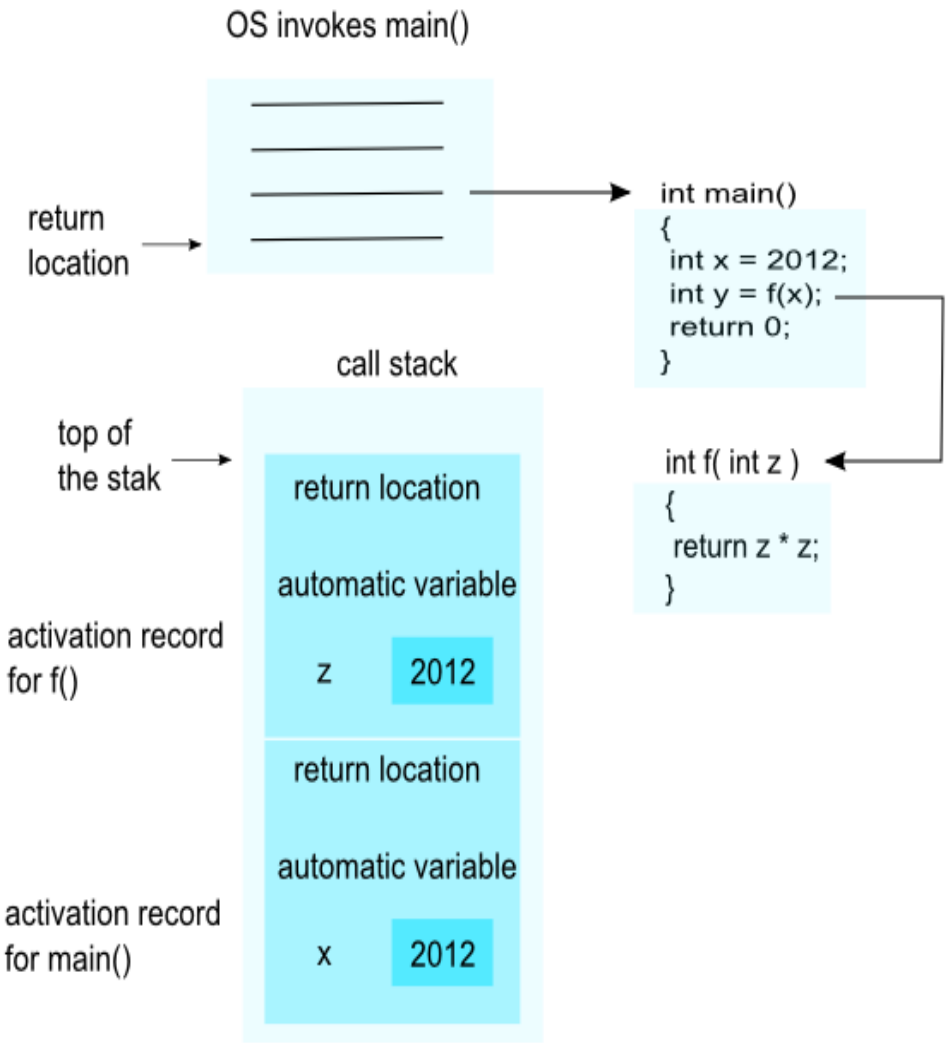
- **Stacks are an important way of supporting nested or recursive function calls.**

Function calls using stack



- The ways in which subroutines receive their parameters and return results—use a special stack (the "call stack")

Function calls using stack



- Call stack holds information
- about procedure/function calling and nesting
- in order to switch to the context of the called function and
- restore to the caller function when the calling finishes.

Queues

Real life Examples of Queue

- The people standing in a railway reservation row are an example of queue. Each new person comes and stands at the end of the row (rear end of queue) and persons getting their reservations confirmed , get out of the row from the front end.

Real life Examples of Queue

- Our software queues have counterparts in real world queues.
- We wait in queues
  - to buy pizza,
  - to enter movie theaters,
  - to drive on a turnpike, and
  - to ride on a roller coaster.

Applications of Queue:

- Queues are widely used as waiting lists for a single shared resource like printer, disk, CPU.

- **Disk Scheduling**

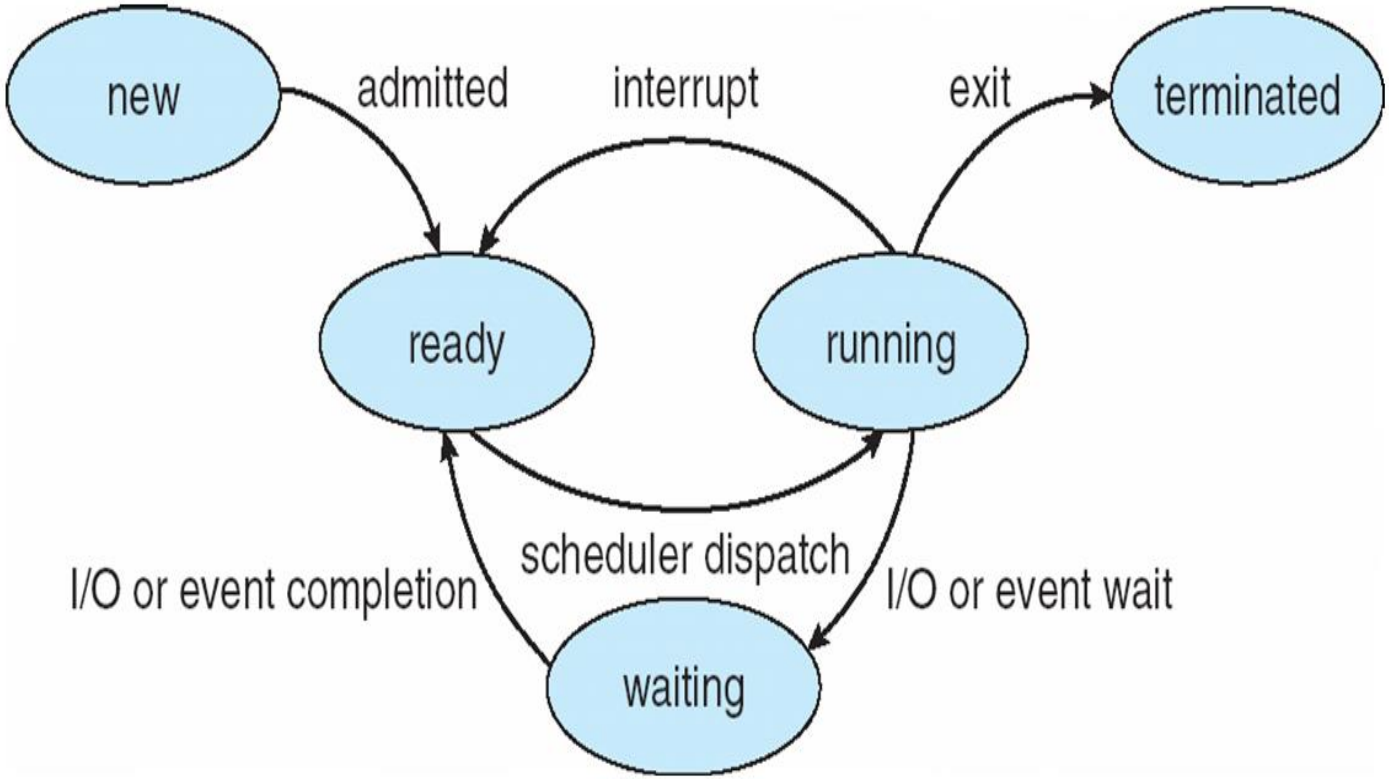
Applications of Queue:

- **CPU scheduling-**

When multiple processes require CPU at the same time, various CPU scheduling algorithms are used which are implemented using Queue data structure.

Applications of Queue:

- **CPU scheduling-**



Applications of Queue:

- When things don’t have to be processed immediately, but have to be processed in First In First Out order like **Breadth First Search**.

- When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes. Examples include **IO Buffers, pipes, file IO, etc.**

- Queue are used to maintain the play list in media players in order to add and remove the songs from the play-list.

Courtesy:https://www.javatpoint.com/data-structure-queue

Queue Example

Accessing printer in multiuser environment-

- If a printer is in process and more than one user wants to access the printer then
- it maintains the queue for user requesting access and serves in FIFO manner for giving access.

Applications of Tree Data Structure

- Store hierarchical data, like folder structure, organization structure, XML/HTML data.

- Binary Search Tree is a tree that allows fast search, insert, delete on a sorted data. It also allows finding closest item

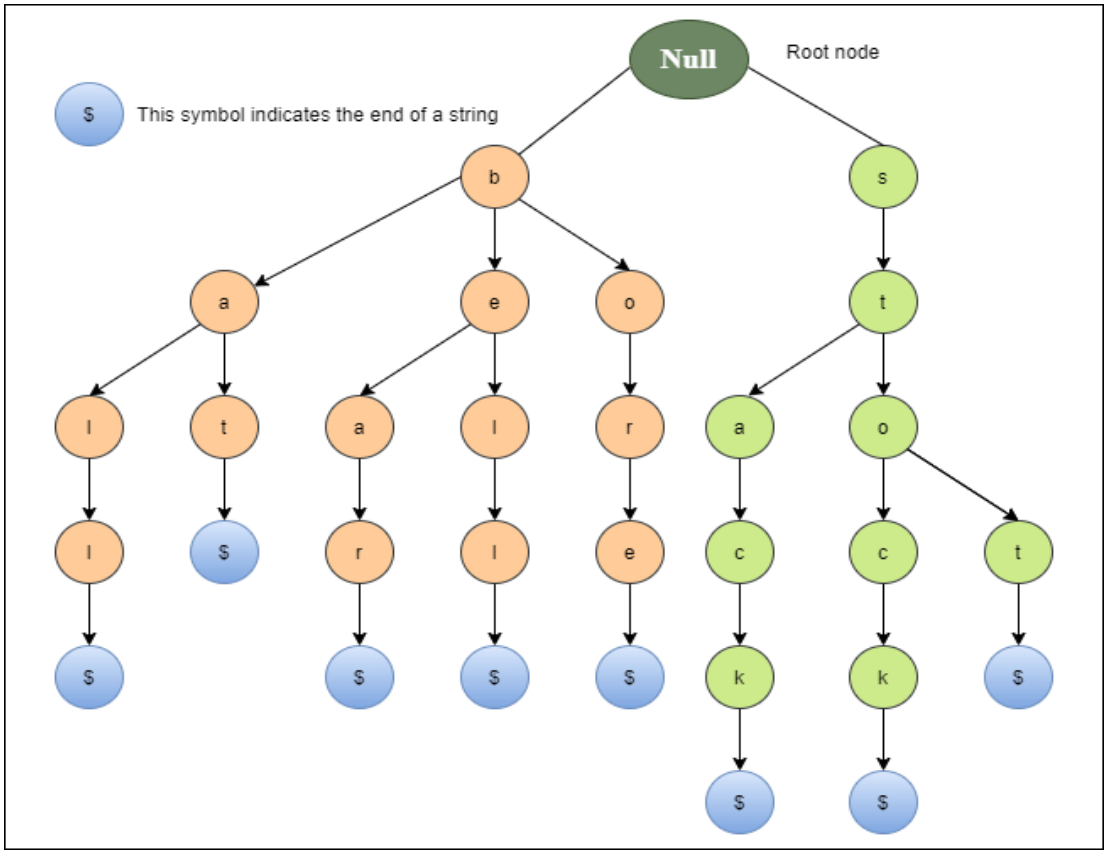
- Heap is a tree data structure which is implemented using arrays and used to implement priority queues.

Applications of Tree Data Structure

- B-Tree and B+ Tree : They are used to implement indexing in databases.
- Syntax Tree: Used in Compilers.
- Spanning Trees and shortest path trees are used in routers and bridges respectively in computer networks

Applications of Tree Data Structure

- Trie : Used to implement dictionaries with prefix lookup.



Applications of Graph Data Structure

- Google maps
- Facebook
- World Wide Web
- Operating System

Applications of Graph Data Structure

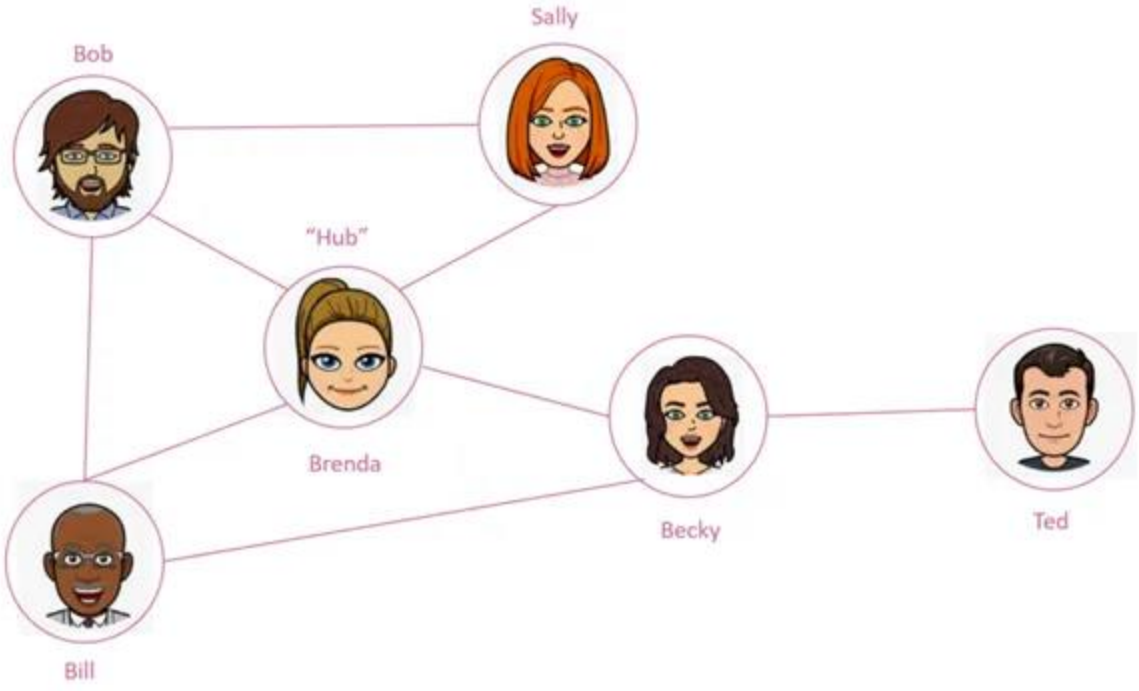
- In Computer science graphs are used to represent the flow of computation.
- Google maps –
  - uses graphs for building transportation systems,
  - where intersection of two(or more) roads are considered to be a vertex and
  - the road connecting two vertices is considered to be an edge,
  - thus their navigation system is based on the algorithm to calculate the shortest path between two vertices.

Social Network Analysis

- In Facebook,
  - users are considered to be the vertices and
  - if they are friends then there is an edge running between them.
  - Facebook's Friend suggestion algorithm uses graph theory.
  - Facebook is an example of undirected graph.

Social Network Analysis

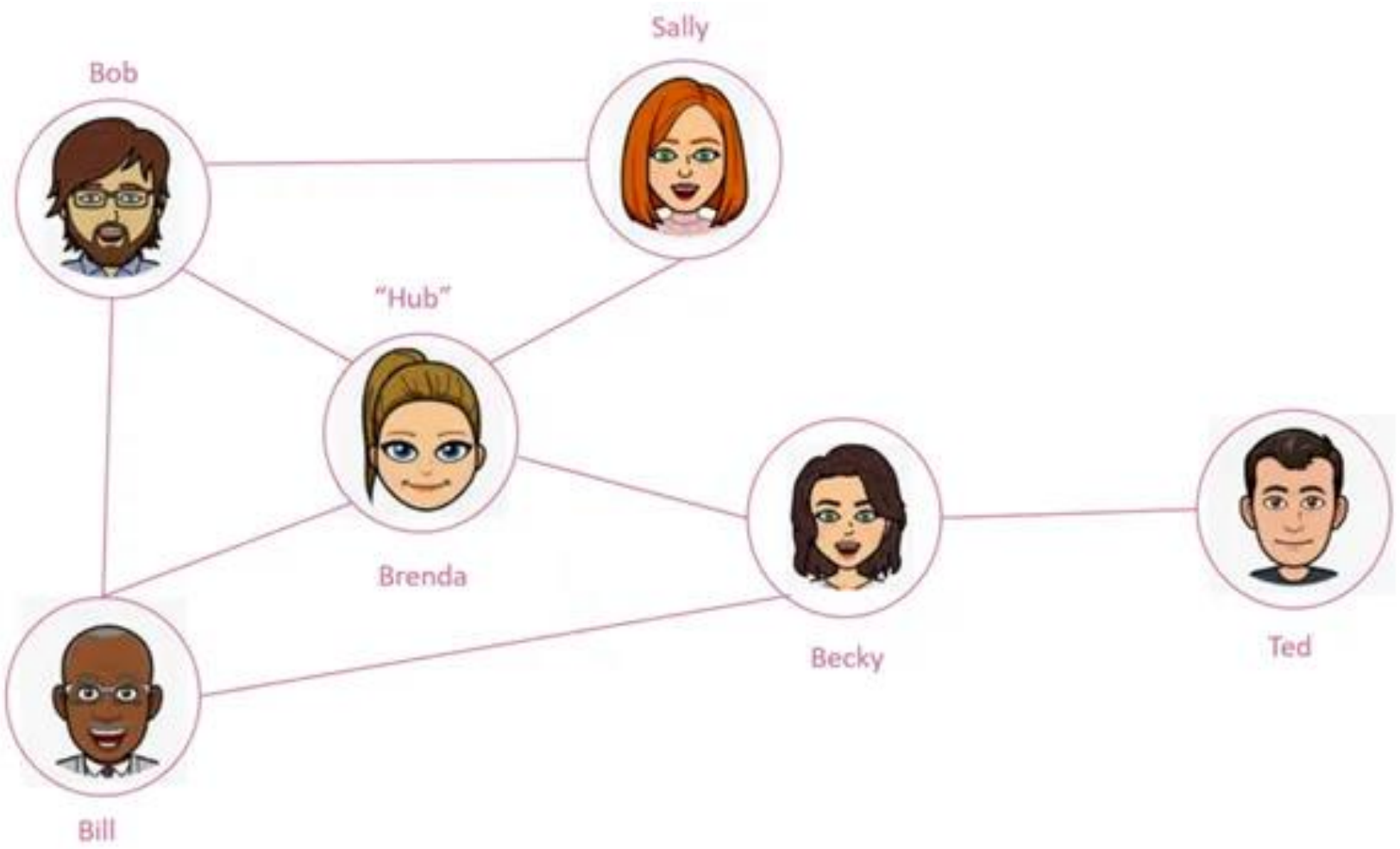
- So, for Facebook, we can think of a node as a friend and the edge as the relationship that links the friends together.
- Below is a basic illustration of some friends in a network:



Social Network Analysis

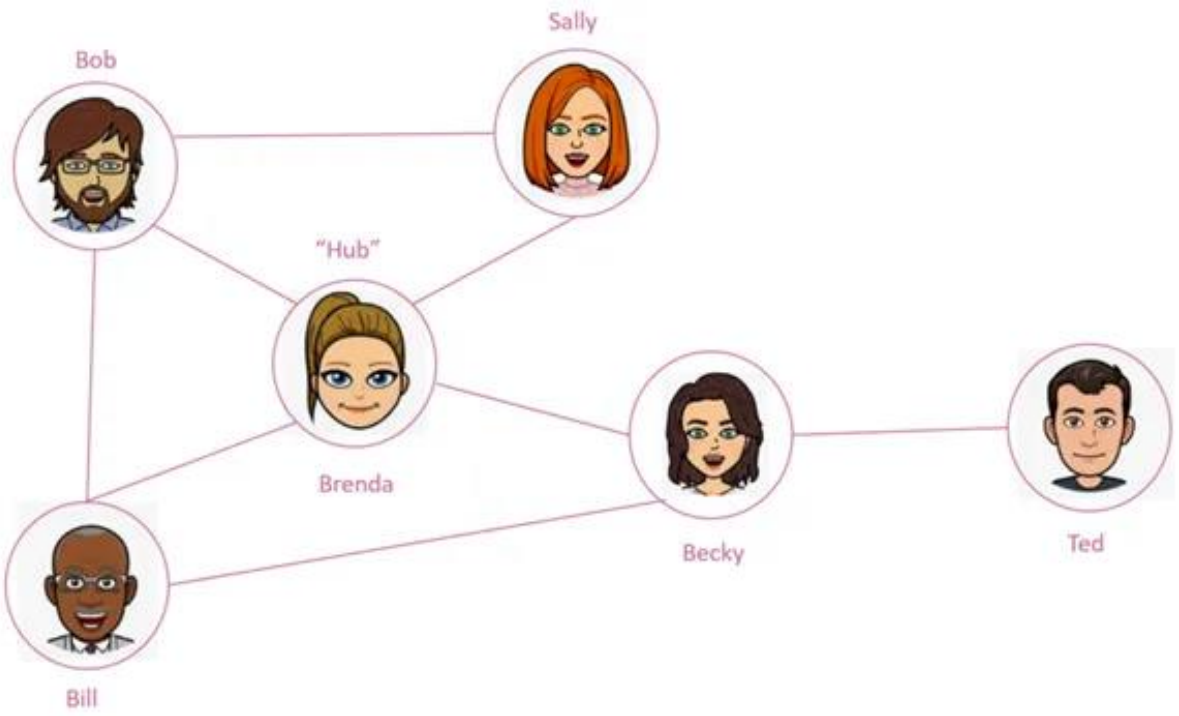
- From here we can start to measure centrality of the graph or the influence among friends.

Social Network Analysis



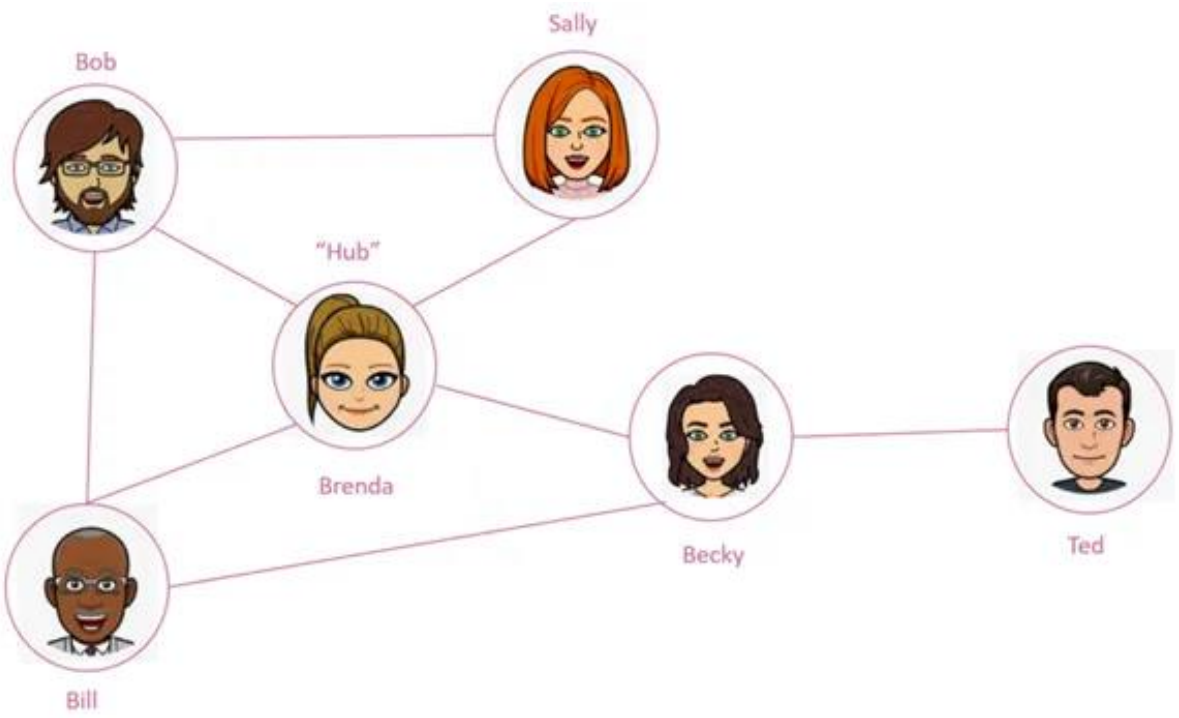
Social Network Analysis

- Degree centrality is a **calculation of how many direct friends that any individual friend in the network has.**
- Friends with a high degree of centrality are not necessary the most powerful or influential, but they act as hubs within the network.



Social Network Analysis

- Brenda is acting as the hub in this social network since she has the most connections or highest degree within the network.



Social Network Analysis

Closeness Centrality-

- Calculates the shortest paths between all nodes, then assigns each node a score based on its sum of shortest paths.
- **When to use it:** For finding the individuals who are best placed to influence the entire network most quickly.
- Closeness centrality can help find good ‘broadcasters’

Applications of Graph Data Structure

- In World Wide Web,
  - web pages are considered to be the vertices.
  - There is an edge from a page u to other page v if there is a link of page v on page u.
  - This is an example of Directed graph.
  - It was the basic idea behind Google Page Ranking Algorithm.

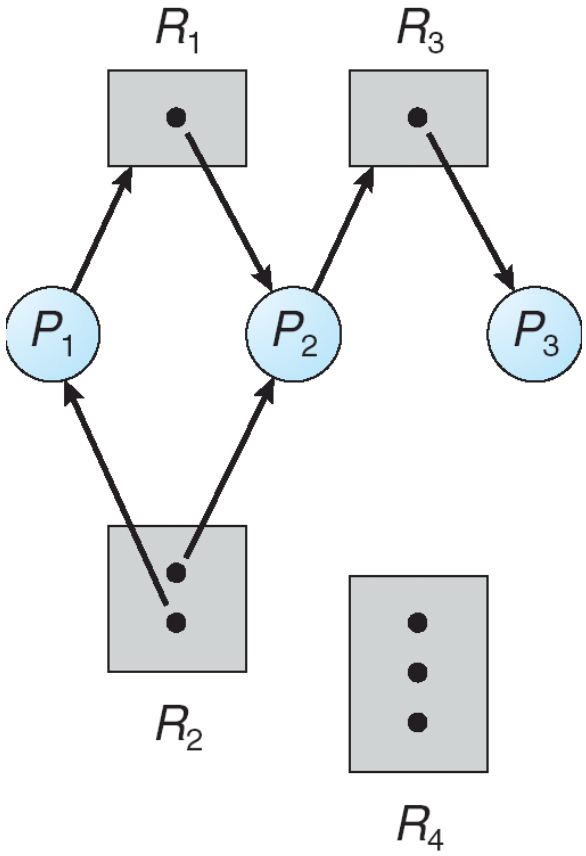
Applications of Graph Data Structure

- In Operating System,
  - we come across the **Resource Allocation Graph** where
    - **each process and resources are considered to be vertices.**

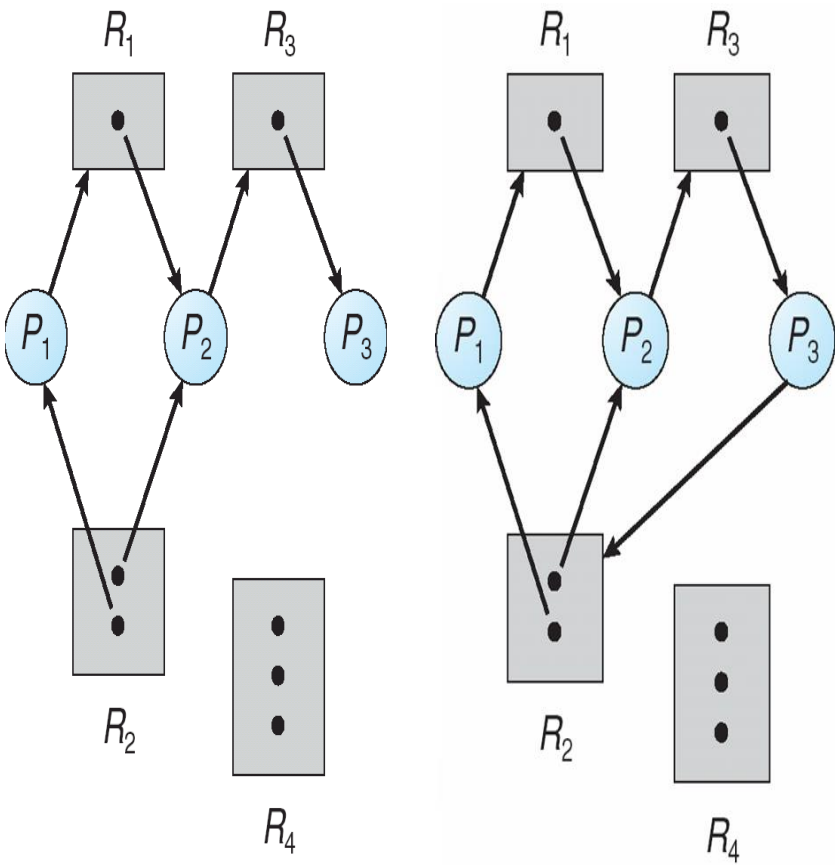
Applications of Graph Data Structure

- Edges are drawn from
  - resources to the **allocated** process, or
  - from **requesting** process to the requested resource.

- If this leads to any **formation of a cycle then a deadlock may occur.**



Resource Allocation Graph With A Deadlock



- Suppose P3 requests for R2,
- Since no resource instance is free, a request edge P3->R2 is added
- So, Now Two cycles exist –

P1->R1->P2->R3->P3->R2->P1

P2->R3->P3->R2->P2

- P1,P2,P3 are deadlocked