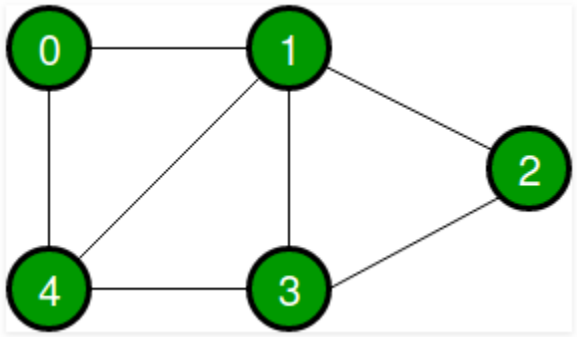
Pawan Chandra

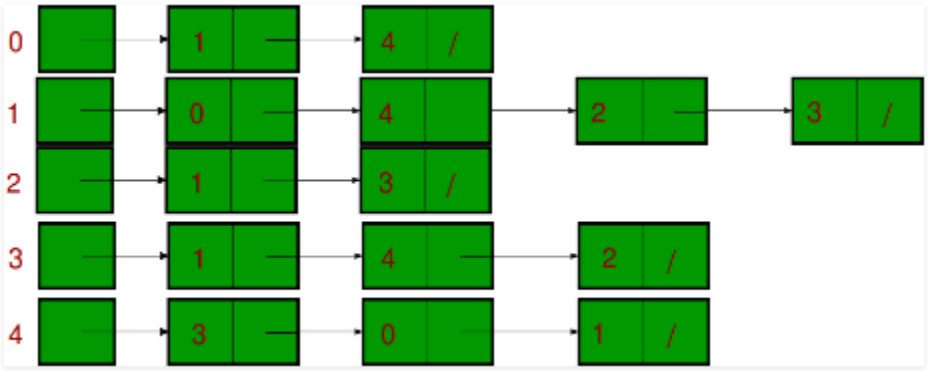
1) Graph Representation

a) Using an adjacency-matrix representation would lead to a large amount of empty space. Since there are a million nodes, and most nodes only have a few edges, the amount of "unconnected" nodes are significant and a large amount of memory is allocated for it.

b) Nodes that have hundreds of thousands of edges would be slow to search and insert to. This means that the linked lists with the hundreds of thousands of edges would take linear time to search through.

c) Because millions of nodes have few edges, the efficient way to do this is with Linked List to save space.





d) A flee market would be one example of an unbalanced graph. The large majority of nodes with only a few edges would be representative of small vendors with only a few things to sell whereas the few nodes with a lot of edges would represent places that attract more people, like the foot stands or vendors with large inventory with more items to sell.

2) How to Graduate As Soon As Possible

a) Best way to approach this problem is by using a Directed Acyclic Graph - a breadth first search. Since courses have pre-requisites and all courses must be taken, each node in every index of the graph must be visited before moving on to the next index.

b) Every time a level in the graph is completed and a node or (class) is enqueued, each of the classes dequeued can be store in separate arrays. These arrays will effectively be all the classes required in a specific semester.

c) The asymptotic running time of this algorithm would be O(V+E) where V is the amount of vertices and E is the amount of edges.

3) Dijkstra's Algorithm

a) We find adjacent edges from A and update the distance and the previous vertices whose distance is less than the current value.

Names	s	A	B	C	D	E	F	G	H	I	T
Vertex	X	X	X	X	X	X	X	X		X	X
Previous	S	S	A	B	S	A	E	S	G	E	C
Cost	0	1	3	5	4	3	6	6	12	6	9

b) The algorithm will not necessarily choose the path with the fewest edges, but the one with the least total coast. You could modify the algorithm to have a counter with each vertex visited. At the end, if there are multiple paths with the same cost, the algorithm with choose the path with the lower counter variable.

c) An example where Dijkstra's algorithm gives the wrong answer is in a negative cycle. Each time we traverse the loop the weights decreases and therefore it can run infinite times and each time weight is different.

d) It is not possible to add weights on each edge because shortest path might get changed. If there (a) - > (b) ->(c) and ab = 6 and bc =-3 by adding 3 we would get ab = 9 and bc = 0 which will mean the shortest path is ac but that is not true.