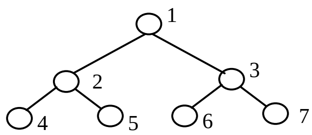
Week 12 Tutorial

1. Give an adjacency-list representation for a complete binary tree T on 7 vertices, as shown on the figure. Give also an equivalent adjacency –matrix representation.



Adj[1]={2,3}
Adj[2]={4,5}
Adj[3]={6,7}
Adj[4]= ∅
Adj[5]= ∅
Adj[6]= ∅
Adj[7]= ∅

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

2. The transpose $G^T = (V, E^T)$ of a directed graph $G = (V, E)$ is the graph such that $(u,v) \in E^T$ just in case $(v, u) \in E$. Thus, G^T is the graph with all edges reversed. Describe efficient algorithms for computing G^T from G first for adjacency lists and then adjacency-matrix representations.

Adjacency list: For each vertex v go through its adjacency set Adj[v] adding v to the adjacency set of every member u in Adj[v].

Adjacency matrix: Simply transpose the matrix.

3. A graph $G = (V, E)$ is ***bipartite*** if the set of vertices V can be split in two disjoint sets A and B such that every edge in E has one vertex in A and the other in B. Design an efficient algorithm which tests if a given graph is bipartite.

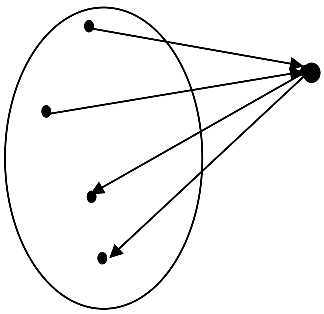
Simply do breath first search of the graph, colouring all discovered and of even degree $d[v]$ vertices v red and all discovered of odd degree $d[v]$ vertices v blue. If in the process we never encounter an edge with both ends of same colour, the graph is bipartite.

4. ***Euler tour*** of a connected, directed graph $G = (V, E)$ is a cycle that traverses each edge exactly once (vertices can be visited several times). *in-degree* (v) of a vertex v is the number of edges coming into v, and *out-degree(v)* is the number of edges leaving v.

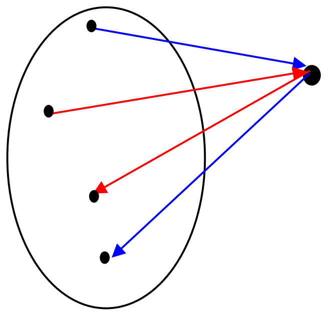
   a. Show that $G = (V, E)$ has an Euler tour if and only if *in-degree* (v) = *out-degree(v)* for every $v \in V$.

   b. Describe an $O(E)$ algorithm to find an Euler tour if one exists.

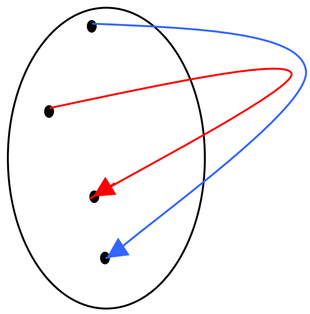
a. Clearly, if there exists an Euler tour, each time we visit an edge we must also leave it; thus *in-degree* (v) = *out-degree(v).* Opposite, if for every v we have *in-degree* (v) = *out-degree(v)* we can proceed by induction on the number of vertices. Assume the statement is true for all graphs with *n* vertices and consider a graph with *n+1* vertices. Remove any vertex, merging pairs of outgoing and incoming edges into a single edge. Find an Euler's tour for such graph with *n* vertices. Get the corresponding tour for the original graph by splitting back the edges:



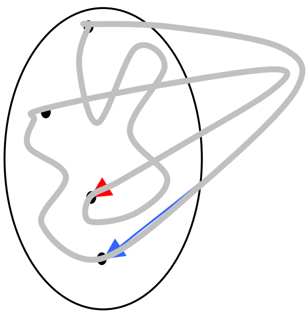
Graph with *n+1* vertices



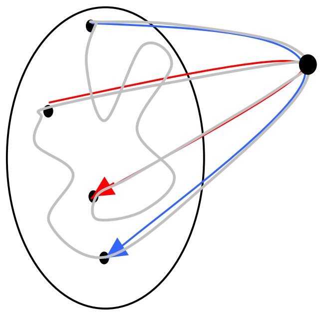
in v and *out-of v* edges paired



vertex removed and pairs of edges joined



Euler tour found for *n* vertex graph



Euler tour for *n+1* vertex graph found by splitting back edges and adding *n+1*st *vertex.*