CS 180A : Discussion

Week 10: Graphs

Zexi Huang

Outlines:

1. Algorithm Design
   - Bipartite Graph
2. Minimal Spanning Trees
   - Prim's Algorithm
   - Kruskal's Algorithm
3. Union-Find Structures

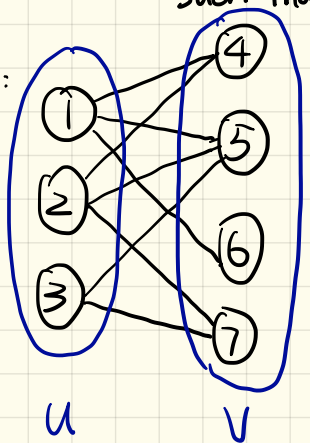
ESCI TA Evaluation

- Please log in to esci.id.ucsb.edu to complete TA evaluation for Sean and me, if you have not done so!
- Your feedback is crucial to helping us improve our teaching quality!

Algorithm Design: Bipartite Graph

Bipartite Graph: graph whose nodes can be divided into two sets U and V such that every edge of the graph connects a node in U to one in V.

Example:



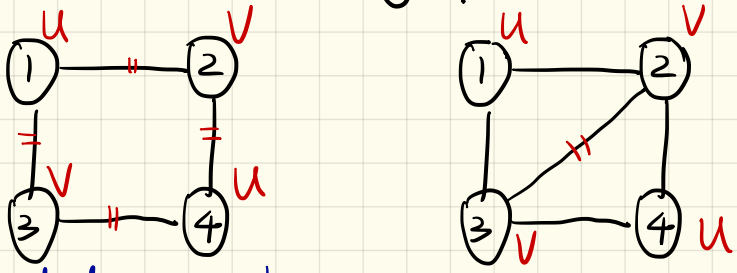
Question: Given an undirected graph, design an algorithm to find whether it is bipartite.

Algorithm Design: Bipartite Graph

Bipartite Graph: graph whose nodes can be divided into two sets U and V such that every edge of the graph connects a node in U to one in V.

Question: Given an undirected graph, find whether it is bipartite.

Example:



for each connected components: $O(m+n)$

(1) Start from a random node s and put it into U.

(2) Put neighbours t of s into V.

Try to put neighbours of t into U.

...

(3) return true if no conflicts, otherwise return false

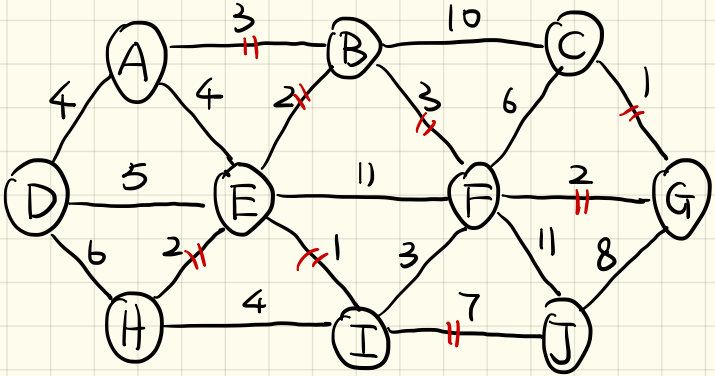
$\Rightarrow$ (2) BFS/DFS from s.

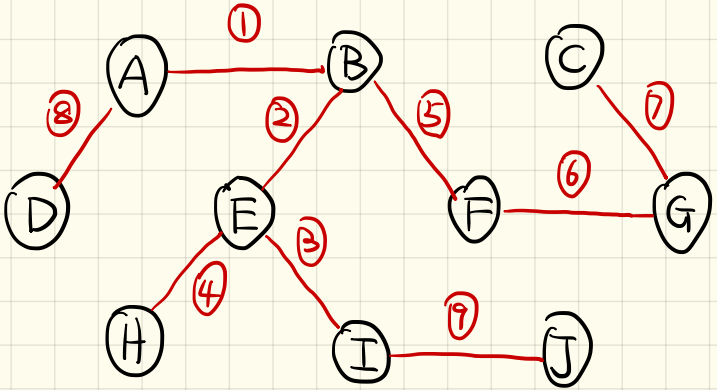
DFS: { put neighbors into U/V if node in V/U.

DFS(neighbours) }

Minimal Spanning Trees

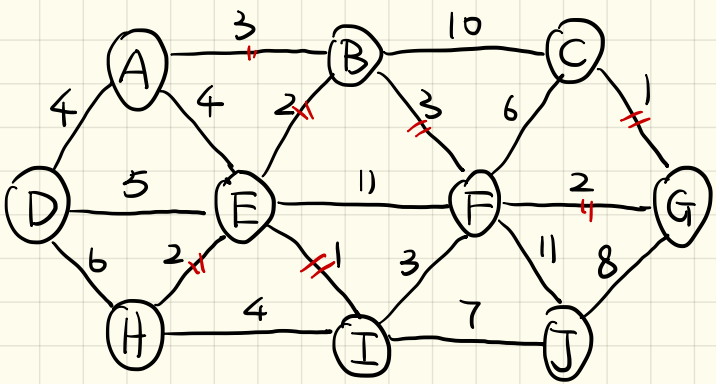
Prim's Algorithm: Iteratively add nodes with the smallest connecting edges

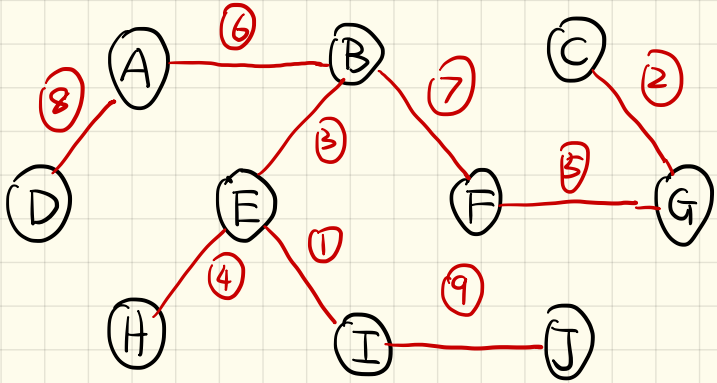




Minimal Spanning Trees

Kruskal's Algorithm: Iteratively add the smallest inter-cluster edges.





Union-Find Structures

Structures: Trees with parent pointers.

Operations:

① Find(i): find the root of node i.

② Union(i,j): (1) find the root of node i and the root of node j.

(2) Make one root become the parent of the other.

Heuristics:

① Union-by-Size: when unioning, make the root of the larger tree become the parent of the root of the smaller tree.

② Path compression: when finding, reset the parent of each node visited to be the root.

Union-Find Structures

a. Union(1,2)
b. m = Find(1)
c. Union(3,4)
d. n = Find(4)
e. Union(m,n) ☆
f. m = Find(1)
g. Union(m,5) ☆
h. Union(6,7)
i. m = Find(6)
j. Union(m,8) ☆
k. Union(9,10)
l. m = Find(10)
m. n = Find(4)
n. n = Find(2)
o. Union(m,n) ☆
p. m = Find(10)
q. n = Find(8)
r. Union(m,n) ☆

m = 1
n = 6