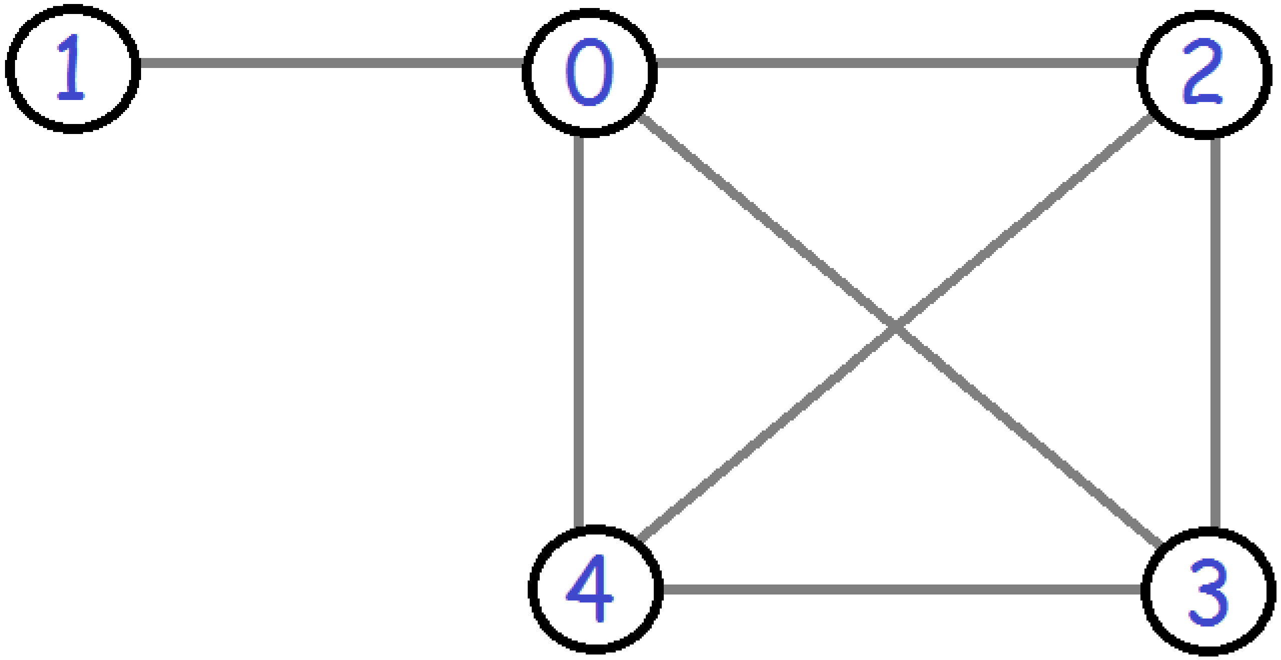
Apply DFS to this graph starting at vertex 0. Traverse the neighbours of each vertex in numerical order.



Show the stack at every step.
What is in the parent, level and Depth-first index (DFI) arrays when the DFS terminates?

Final exam tutorial time:
Thursday Dec. 5: from 11am until all questions are answered. Room TBA.
Programming Assignment #4B: Upload to connex by Tues. Nov. 26, 11:55pm.
I gave you a program for generating random MST problems for testing.
Written Assignment #5: due Wed. Dec. 4, beginning of class.
Please take the time to give some feedback to your lab instructor.

TODAY: Sponsored by WECS

3pm in ECS 660:
Becoming an Exec in the Tech Field
Experienced professionals will talk about how they got to the top:
- Sherry Smeaton, Associate Partner, IBM,
- Blaine Bey, Sierra Systems
- Nelson Lah, CGI Victoria.

4pm in ECS 668:
Chocolate-themed Open House

ALL students are invited to attend!

Combinatorial Potlatch this Saturday in Clearihue A 127:
11:00 am: Richard Hoshino, Quest University Canada

Applying Combinatorics to Inspire Change

Over the past seven years, I've learned that a combinatorialist can make a difference, and I'll share some ways in which I've applied simple ideas in discrete mathematics to inspire real change: reducing wait times at Canadian airports; implementing a roommate-matching algorithm for students at my university; and helping a billion-dollar professional baseball league design a regular-season schedule to cut costs and reduce greenhouse gas emissions.

Everyone welcome, coffee at 10am coffee and muffins, no registration fee.

Minimum weight spanning tree:

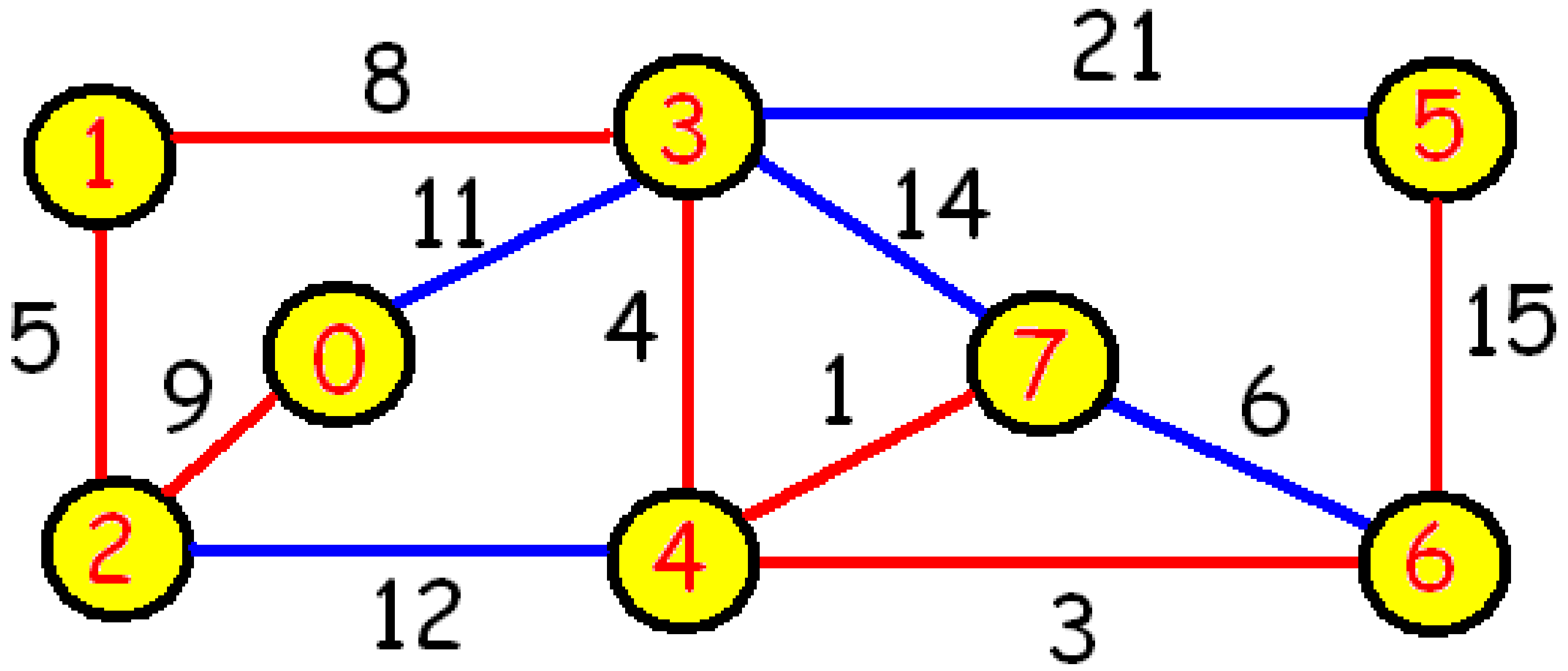
Put a non-negative weight on each edge.

Weight of a tree: sum of weights of its edges.

Problem: find a spanning tree of graph G with minimum weight.

Kruskal's algorithm: sort edges by weight.

Then for each edge: add it to the tree if its endpoints are in different components.



Edge weights:

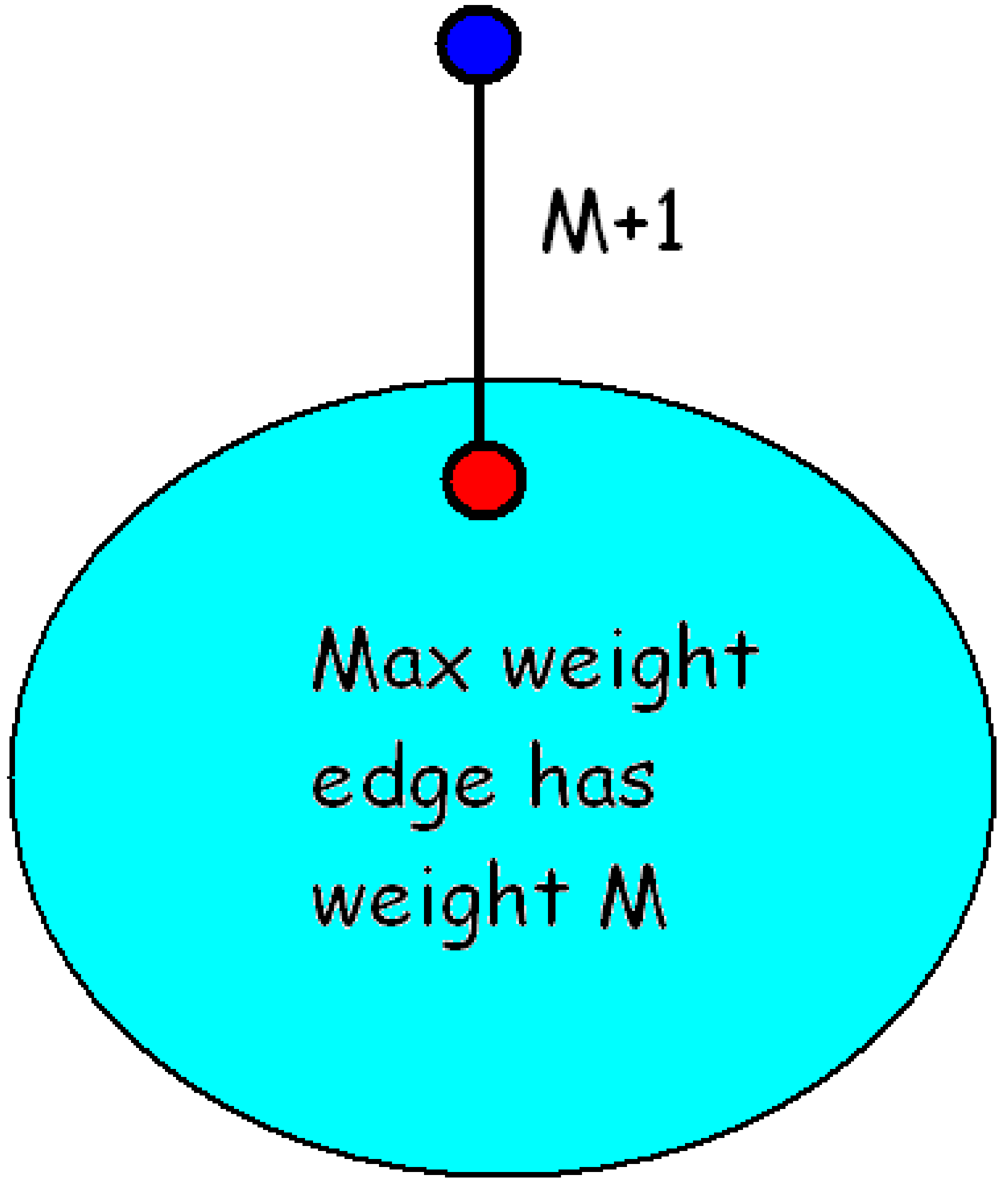
1, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 21

Analysis:

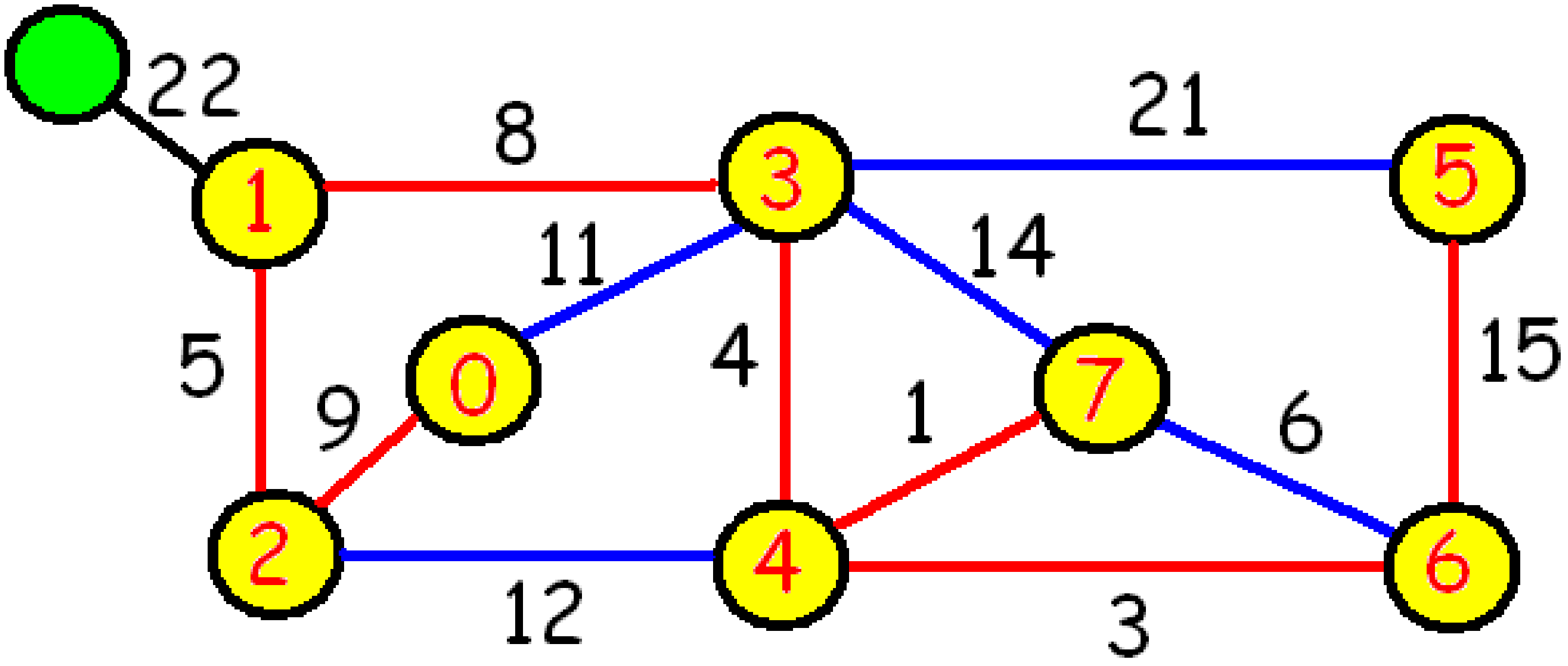
Which data structures/algorithms can we use to implement Kruskal's algorithm.

Tasks:

1. Repeatedly choose a min weight edge.
2. If the endpoints of the edge are in different components, add the edge to the spanning tree.


M+1
Max weight edge has weight M

The max weight edge can be in a min weight spanning tree:



Data structures for the graph:

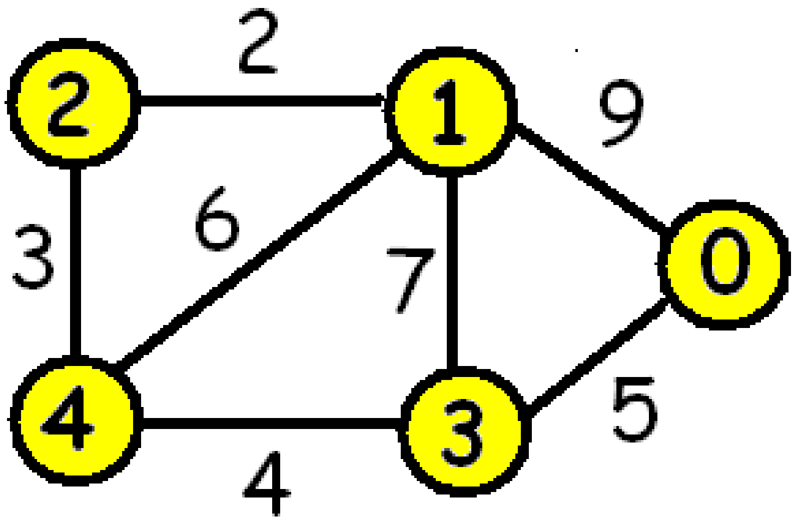
1. Adjacency matrix.

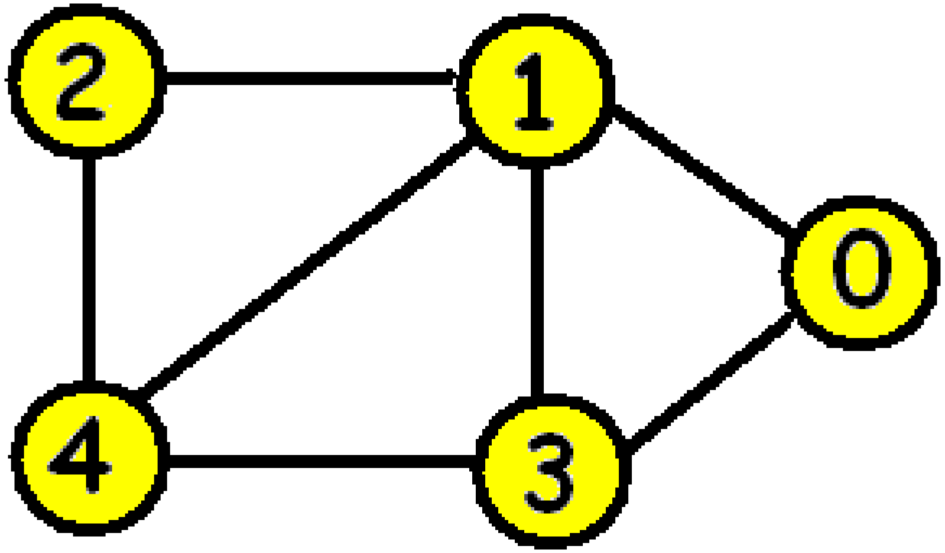
2. Adjacency list.

3. List of edges in the graph stored in an array or in a linked list.

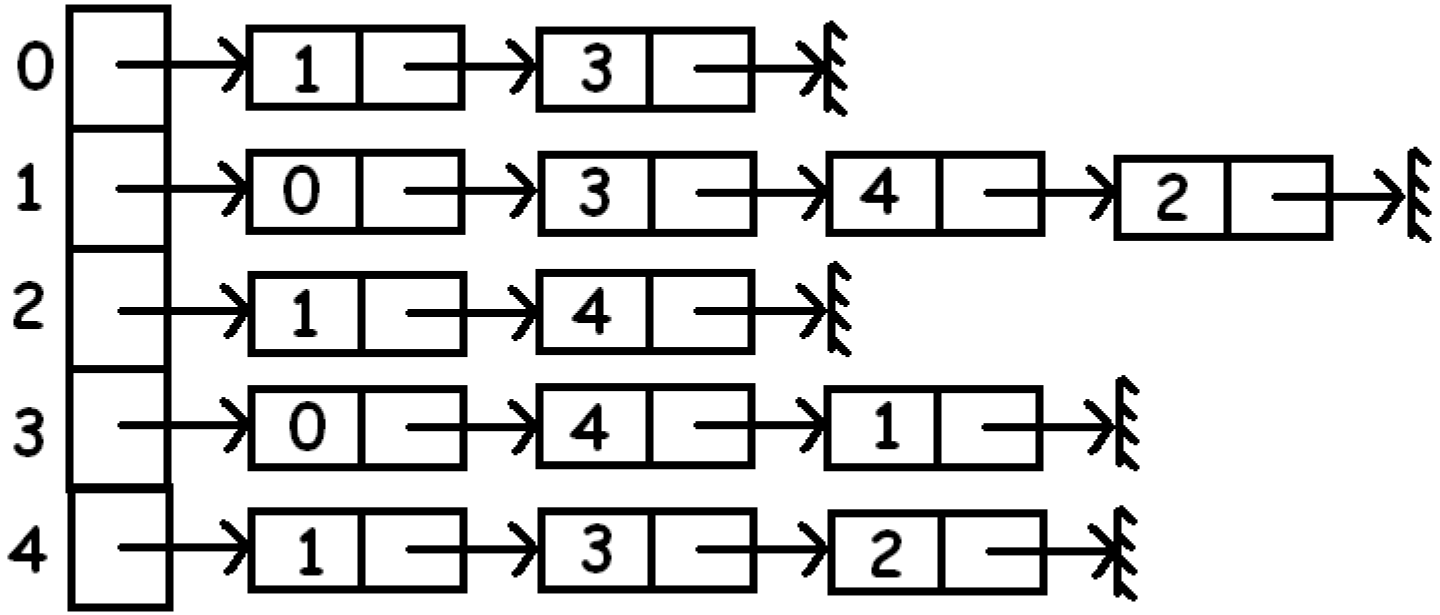
Output: we could just print the edges in the spanning tree.

	0	1	2	3	4
0	-1	9	-1	5	-1
1	9	-1	2	7	6
2	-1	2	-1	-1	3
3	5	7	-1	-1	4
4	-1	6	3	4	-1





Adjacency list:



Union/find:

(a) Flat union/find.

(b) Weighted union/non-collapsing find.

(c) Weighted union/collapsing find.

How many finds occur in the worst case?

How many unions?

The approach to choose will depend on:

1. If code is to be reused many times or is throwaway code.
2. If code is research-based or part of a consumer product.
3. The number of vertices/edges in the typical graphs that will be tested- for time and space issues.
4. If the weights are viable for a radix sort.
5. Importance of a fast running time.
6. Importance of correctness.
7. Ease of debugging and code maintenance.
8. Data structures and algorithms already available.

A throwaway approach for a researcher:

Presort edges by weight using a system sort before using them for input to the program.

To help ensure correctness- use a flat union/find.

Worst case time: $O(m \log m + n^2)$.

Recall that we do up to 2m find's and for the flat scheme these take O(1) time and n-1 unions each taking O(n) time.

What is worst case complexity of Kruskal’s alg. if
(a) The adjacency matrix is scanned at each step to find a minimum weight edge. The flat union/find data structure is applied.
(b) The edges are stored in a min-heap and deleteMin is used to remove an edge from the heap at each step. The union/find data structure is a weighted union with a non-collapsing find.
(c) Radix sort is used to first sort the edges by weight. Weighted union is used with a collapsing find.

The column *E* represents the total amount of work that the algorithm does in order to pick a minimum weight edge summed over all the iterations.
The column *U* represents the total amount of work that is done for all the union/find operations.
The column *K* represents the time complexity of Kruskal's algorithm.

Algorithm	E	U	K
(a)			
(b)			
(c)			

Assuming G has Ω(n edges):

Under the comparison model, Kruskal's algorithm is $\Theta(m \log_2 m)$ in the worst case

Use: adjacency lists or edge lists, mergesort, and weighted union with a collapsing or a non-collapsing find.

Thought question:
When is the Dijkstra/Prim algorithm which is $\Theta(n^2)$ faster than Kruskal's algorithm which is $\Theta(m \log m)$?

Course evaluations: I greatly appreciate any constructive comments you have on how to improve CSC 225.

Regarding the labs:
We have 7 this term:
Would it be better to have more or less?
Bear in mind that more labs means less marking hours.
Beginning of term: more structure.
End of term: questions on assignments.
Is this the right balance?
Please sort the sheets into 2 piles.