Week 10: Intro to Graphs and Graph Algorithms

Day 16 (M 3/23): Intro to Graphs

- **Video (18 min):** Watch the following video introducing graphs.
  `https://youtu.be/OMRjXgKSXGo`

- **Exercise (15 min):** A graph is an abstract data type, which means that the implementation or representation is not specified. One way to represent a graph $G = (V, E)$ is with an *adjacency matrix* M. In this representation, we assume that the nodes of the graph are labelled $V = \{1, 2, \ldots, n\}$ and M is an $n \times n$ matrix where

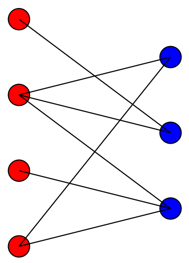
$$M_{i,j} = \begin{cases} 1 & (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

1. Draw the graph $G = (V, E)$, where $V = \{1, 2, 3, 4\}$ and $E = \{(1, 2), (3, 2), (4, 1)\}$. Then write its corresponding adjacency matrix.

2. Find the runtimes for the following operations using an adjacency matrix implementation of a graph. Let $n = |V|$ and $m = |E|$.
   - Iterate through all graph nodes.
   - Iterate through all graph edges.
   - Determine whether there is an edge between node i and j in G.
   - Find a list of a node i's neighbors.

3. Can you think of an alternative way to implement a graph? **Hint:** Hash maps may be useful to allow arbitrary node labels.

- **Video (19 min):** Watch the following video talking about breadth-first and depth-first search. `https://youtu.be/cyNCwxtxFww`
- **Exercise (10 min):** A special kind of graph is a *bipartite graph*. In a bipartite graph, the nodes can be partitioned into two disjoint sets X, Y such that every edge is between a vertex in X and a vertex in Y. An example is given below. Design an algorithm to determine whether a graph is a bipartite graph. **Hint:** You may want to base your algorithm on one of the graph traversal algorithms you just learned about.



- **Post any questions about this section to the corresponding Canvas discussion thread prior to the live discussion.**

Day 17 (R 3/26): Cycles and Trees

- **Video (16 min)**: Watch the following video introducing trees and cycles. `https://youtu.be/meIpu2qCsnw`
- **Exercise (10 min)**: Use induction to prove that a tree on n vertices has $n-1$ edges.

This implies that a connected undirected graph has a cycle if and only if it has $> n-1$ edges. Using this fact, expalin how you could test whether a graph $G = (V, E)$ has a cycle.

- **Video (20 min):** Watch the following video about minimum spanning trees.
  `https://youtu.be/JG_nVMrnvrs`
- **Exercise (15 min)**: Kruskal's algorithm for finding a minimum spanning tree is also a greedy algorithm. In Kruskal's algorithm, we start with each vertex in its own tree with no edges. Then, we continuously find the minimum edge connecting any two trees, add that edge to the spanning tree, and merge those trees into one. Pseudocode for this algorithm is below (note that this is an application of a union-find data structure). Following a similar argument to in the video, explain why Kruskal's algorithm finds a minimum spanning tree.

  Hint: Think about the first edge e_i added to the solution that is not in an optimal solution $T*$ and then add that edge to T^*.

```
Kruskal(G)
MST = []
for each v in V:
    initialize a set with v
for each (u,v) in E in order of increasing weight:
    if find-set(u) != find-set(v):
        union u and v's sets
        append (u,v) to MST
return MST
```

- **Post any questions about this section to the corresponding Canvas discussion thread prior to the live discussion.**

Extra Resources for the Week

- Intro to Graphs:
  `https://medium.com/basecs/a-gentle-introduction-to-graph-theory-77969829ead8`
- Breadth-First Search Visualization: `https://www.cs.usfca.edu/~galles/visualization/BFS.html`
- Depth-First Search Visualization: `https://www.cs.usfca.edu/~galles/visualization/DFS.html`