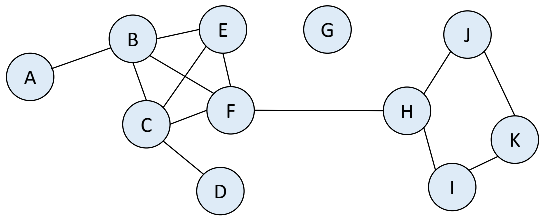
15-122: Principles of Imperative Computation

Recitation 12: Computing on the Edge — Andrew Benson

Graphs, Vertices, and Edges

A **graph** G is a set of vertices V and a set of edges E, where each edge is a pair of vertices.



For example, the graph above would be described by giving the vertex set and the edge set as in

$$V = \{A, B, C, D, E, F, G, H, I, J, K\}$$

$$E = \{(A, B), (B, C), (B, E), (B, F), (C, E), (C, F), (C, D), (F, H), (H, I), (H, J), (I, K), (J, K)\}$$

(For brevity, only edges in one direction are listed. This is okay because the graph above is **undirected**.)

The **subgraph** consisting of only vertices B, C, E, and F is called a **complete graph** because an edge exists between every two of those vertices.

Checkpoint 0

Let $T(n)$ be the number of edges in a complete graph with n vertices. Write a recursive formula for $T(n)$. You will need to write two expressions - one for the base case and one for the recursive case.

Checkpoint 1

Recall from lecture that we discussed two ways of representing graphs: adjacency matrices and adjacency lists. Draw the adjacency matrix and the adjacency list for the graph above.

Checkpoint 2

In terms of $|V|$ and $|E|$, give a big O bound for the `graph_hasedge` operation for both the adjacency matrix and adjacency list representations.

Checkpoint 3

We call a graph **sparse** if it has relatively few edges and **dense** if it has relatively many edges. Which representation might you want to use for a sparse graph? What about for a dense graph? Why?

Graph Search

We've discussed two algorithms for traversing a graph: depth-first search (DFS) and breadth-first search (BFS). DFS always visits the first neighbor of a vertex before visiting the rest of the neighbors, whereas BFS visits all neighbors of a vertex before continuing the search in a neighbor's neighbor.

Checkpoint 4

Assume you wish to search the graph from the previous page for vertex K starting at vertex A. List the vertices of the graph in the order visited by DFS and BFS. Assume that the algorithms consider neighbors in alphabetical order.

Now list the vertices in order of distance from A. Do you notice anything?

Checkpoint 5

In class we studied a recursive implementation of DFS. Finish the iterative implementation of DFS below.

```
bool dfs(graph_t G, vertex start, vertex target) {
  REQUIRES(start < graph_size(G) && target < graph_size(G));

  if (start == target) return true;

  bool mark[graph_size(G)];
  for (unsigned int i = 0; i < graph_size(G); i++)
    mark[i] = false;

  ______________________________________________________ // initialize data structure

  ____________________________________ // put start into our data structure
  mark[start] = true;

  while(____________________________________) { // unless our data structure is empty

    vertex v = ____________________________; // retrieve a vertex from our data structure
    if (v == target) {

      ____________________________________________ // free our data structure
      return true;
    }
    for (vertex w = 0; w < graph_size(G); w++) {
      if (graph_hasedge(G, v, w) && !mark[w]) {
        mark[w] = true;

        ______________________________________ // add the neighbor to our data structure
      }
    }
  }
  return false;
}
```