CSCI 2270 Lecture Notes

04/01/2019

Graphs Recap

* A collection of vertices (nodes) connected by edges (pointers)
* each vertex contains a “key” and a list of edges (adjacent vertices)
* list of edges is stored in an adj. matrix or adj. list
* unlike a BST, there are no set relationships between vertices
  + all relationships have to be explicitly set

**ADT (abstract data type)**

* undirected (must be two-way relationship between vertices)
* weighted (strength of relationship in either direction)

private:

vertices

public:

constructor

insertVertex(value)

insertEdge(startVertex, endVertex, weight)

deleteVertex(value)

deleteEdge(startValue, endValue)

printGraph()

search(value)

struct vertex{

string key;

vector < adjVertex> adj;

}

struct adjVertex{

vertex \*v;

int weight;

}

Insert Vertex

* if using vectors
  + search to ensure no duplicate keys exist
  + append to the end (push\_back(key))

Adding Edge

* find the two given vertices and insert an edge to both adjacency lists (undirected graph)
  + loop across all vertices
  + if first vertex is found, loop across all vertices to find other vertex
  + add an entry to first vertex’s adjacency list with a pointer to second vertex
  + also add entry to second vertex’s adjacency list with a pointer to first vertex

Next Class

* displaying and searching graphs in various orders
  + breadth first
  + depth first

**STL Vectors**

* include vector library
* allows to index into elements like an array
* can append and grow length where memory allocation happens automatically
* can define container type to be either a primitive data type (int, double, float…) or object type (class, struct)