Edge-Weighted Graph API

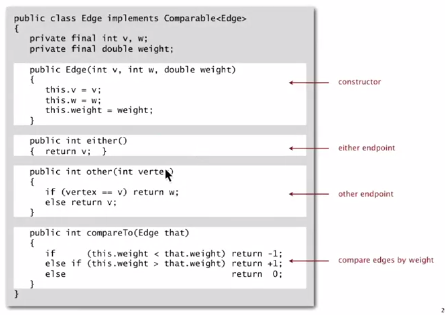
Edge abstraction is needed for weighted edges:

Public class Edge implements Comparable<Edge>  
Edge(int v, int w, double weight) : create a weighted edge v-w  
int either() : either endpoint  
int other() : the endpoint that’s not v  
int compareTo(Edge that) : compute this edge to that edge  
doubleweight() : the weight  
String toString(): string representation

A weighted edge is two vertices that connect and the double weight. We must be able to compare the weights and return -1, 0 or +1.

Idiom for processing an edge e: int v = e.either(), w = e.other(v);

Java implementation of Edge



Edge-weighted graph

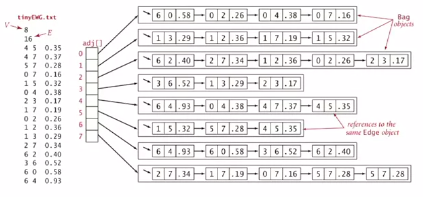
The graph API:

Public class EdgeWeightedGraph  
**EdgeWeightedGraph(int V) : create an empty graph with V vertices**  
EdgeWeightedGraph(In in) : create a graph from input stream  
**void addEdge(Edge e) : add weighted edge e to this graph  
Iterable<Edge> adj(int v) : edges incident to v**  
Iterable<Edge> edges() : all edges in this graph  
int V() : number of vertices  
int E() : number of edges  
String toString() : string representation

Conventions: allow self-loops and parallel edges

Adjacency-lists representation

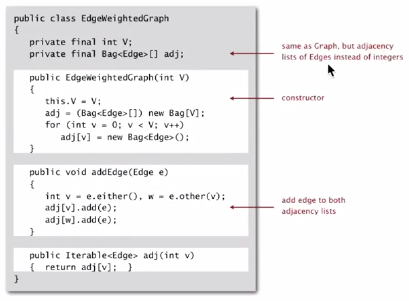
How? Maintain vertex-indexed array of Edge lists



List of edges connected to each vertex at each respective vertex’s index. Since it’s undirected, each edge appears twice

Vertex-indexed array of bags of edges (in Java)

Java Implementation

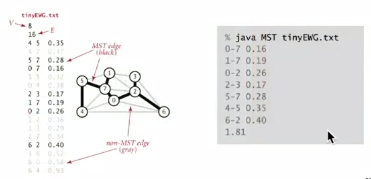


MST API

Usually client of an MST algorithm usually wants to have us compute MST for given edge weighted graph. How to represent the MST:

Public class MST  
MST(EdgeWeightedGraph G) : constructor  
Iterable<Edge> edges : edges in an MST  
double weight() : weight of MST

Constructor does the work and client can query about what happened



Sample test client implementation

