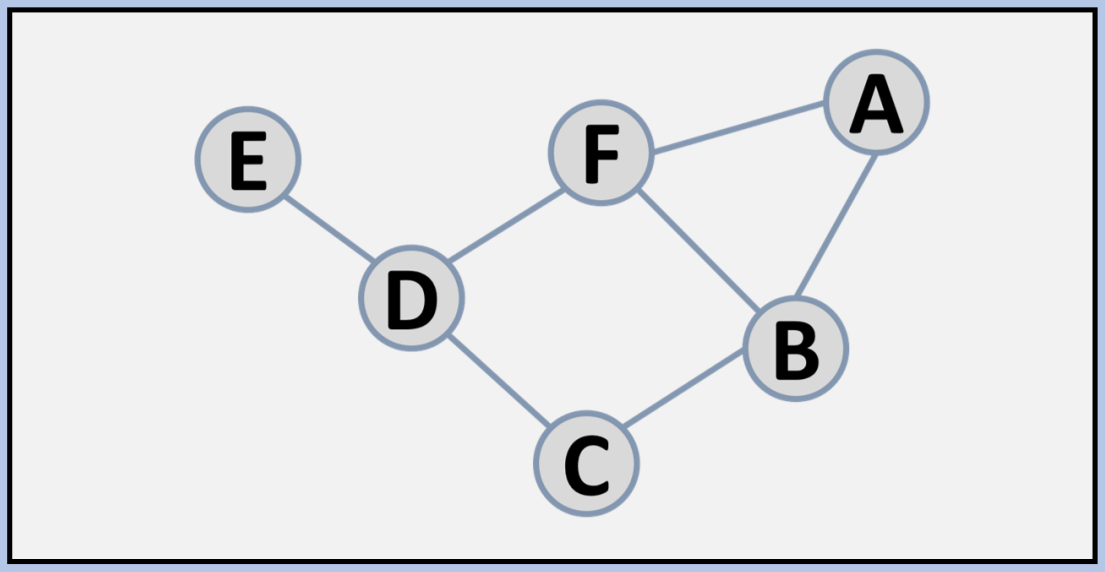
ADT AdjacencyList Graph



$G = (V, E), where\ V = \{v_1, v_2, \dots, v_n\}\ and\ E = \{e_1, e_2, \dots, e_n\}$

$\{\ inv: \mathbf{V} \neq \emptyset\}$

▪ Graph:	----	→	Graph
▪ addVertex:	Graph x Vertex	→	Graph
▪ addEdge:	Graph x Vertex x Vertex	→	Graph
▪ addEdge:	Graph x Vertex x Vertex x Number	→	Graph
▪ removeVertex:	Graph x Vertex	→	Graph
▪ removeEdge:	Graph x Vertex x Vertex	→	Graph
▪ hasVertex:	Graph x Vertex	→	Boolean
▪ hasEdge:	Graph x Vertex x Vertex	→	Boolean
▪ size:	Graph	→	Integer
▪ clear:	Graph	→	Graph
▪ BFS:	Graph x Vertex	→	Graph
▪ DFS:	Graph	→	Graph

GraphConstuctor()

Description: “Builds an empty graph.”

{ *pre*: — }

{ *post*: graph $\mathrm{g} = (\mathrm{V}, \mathrm{E}) = (\emptyset, \emptyset)$ }

addVertex(graph, v)

Description: “Adds the vertex v to the set of vertices V.”

{ *pre*: graph.vertices $\mathrm{V} = \emptyset$ or $\mathrm{V} = (v_1, v_2, \dots, v_n)$}

{ *post*: graph.vertices $\mathrm{V} = (v_1, v_2, \dots, v_n, v)$ }

addEdge(graph, v_1, v_2)

Description: “Creates an edge e from v_1 to v_2 with a weight of 0 and adds it to the set of edges E.”

{ *pre*: graph. edges $\mathrm{E} = \emptyset$ or $\mathrm{E} = (e_1, e_2, \dots, e_n)$}

{ *post*: graph. edges $\mathrm{E} = (e_1, e_2, \dots, e_n, e)$}

addEdge(graph, v_1, v_2, w**)**

Description: "Creates an edge e from v_1 to v_2 with a weight w and adds it to the set of edges E."

{ *pre*: graph.edges $\mathrm{E} = \emptyset$ or $\mathrm{E} = (e_1, e_2, \ldots, e_n)$}

{ *post*: graph.edges $\mathrm{E} = (e_1, e_2, \ldots, e_n, e)$}

removeVertex(graph, v**)**

Description: "Removes a vertex v from the set of vertices V."

{ *pre*: graph.vertices $\mathrm{V} = (v_1, v_2, \ldots, v_n, v)$}

{ *post*: graph.vertices $\mathrm{V} = (v_1, v_2, \ldots, v_n)$}

removeEdge(graph, e**)**

Description: "Removes a edge e from the set of edges E."

{ *pre*: graph.edges $\mathrm{E} = (e_1, e_2, \ldots, e_n, e)$}

{ *post*: graph.edges $\mathrm{E} = (e_1, e_2, \ldots, e_n)$}

hasVertex(graph, v**)**

Description: "Checks if the vertex v is in the set of vertices V."

{ *pre*: graph.vertices $\mathrm{V} = \emptyset$ or $\mathrm{V} = (v_1, v_2, \dots, v_n)$ *or* $\mathrm{V} = (v_1, v_2, \dots, v_n, v)$}
{ post: true or false }

hasEdge(graph, e**)**

Description: "Checks if the edge e is in the set of edges E."

{ *pre*: graph.edges $\mathrm{E} = \emptyset$ or $\mathrm{E} = (e_1, e_2, \dots, e_n)$ or $\mathrm{E} = (e_1, e_2, \dots, e_n, e)$}
{ *post*: true or false }

size(graph**)**

Description: "Returns the size of the set of vertices V."

{ *pre*: graph.vertices $\mathrm{V} = \emptyset$ or $\mathrm{V} = (v_1, v_2, \dots, v_n)$}
{ *post*: Integer }

clear(graph**)**

Description: "Clears the graph."

{ *pre*: graph $\mathrm{g} = (\emptyset, \emptyset)$ *or* $\mathrm{g} = ((v_1, v_2, \dots, v_n), (e_1, e_2, \dots, e_n))$ }
{ *post*: graph $\mathrm{g} = (\emptyset, \emptyset)$}

BFS(graph, v)

Description: "Explore all neighbors of vertex v, and for each of its neighbors, explore their respective adjacent neighbors until traversing the entire graph."

{ *pre*: graph g $= \big((v_1, v_2, \dots, v_n, v), (e_1, e_2, \dots, e_n)\big)$ $where\ v_n =$ $(father = null, distance = 0, visited = false)$}

{ *post*: graph. vertices V $= (v_1, v_2, \dots, v_n, v)$ $where\ v_n =$ $(father = v_n, distance = integer, visited = boolean)$ }

DFS(graph)

Description: "Perform a depth-first search, traversing all nodes of the graph in an ordered but non-uniform manner.."

{ *pre*: graph g $= (V, E)$ $where\ V \neq \emptyset\ and\ v_n =$ $(father = null, visited = false, iTime = 0, fTime = 0)$}

{ *post*: graph.vertices V $= (v_1, v_2, \dots, v_n)$ $where\ v_n =$ $(father = v_n, visited = true, iTime = integer, fTime = integer)$}