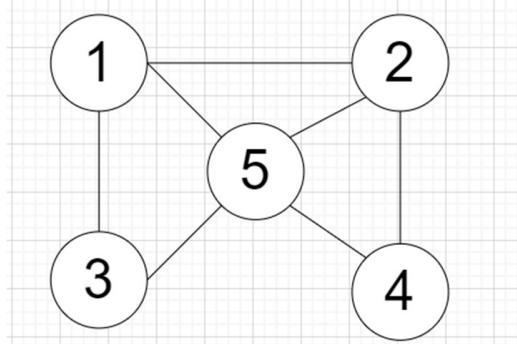


TAD Graph

Graph = $\{V = \{v_1, v_2, \dots, v_n\}, E = \{e_1 = (v_{i1}, v_{j1}, w_1), e_2 = (v_{i2}, v_{j2}, w_2), e_m = (v_{im}, v_{jm}, w_m)\}, \text{directed, weighted}\}$



{inv:

1. $\forall e_k \in E, v_{ik} \in V \wedge v_{jk} \in V, w_k > 0$
 2. $\text{directed} = \text{false} \Rightarrow (\forall (a, b) \in E \exists (b, a) \in E, a, b \in V)$
 3. $\text{weighted} = \text{false} \Rightarrow \forall e_k \in E, w_k = 1$
- }

Primitive Operations

- Graph $\langle \rangle \rightarrow \langle \text{Graph} \rangle$ Constructor
- insertVertex $\langle \text{Vertex} \rangle \rightarrow \langle \text{Graph} \rangle$ Modifier
- insertEdge $\langle \text{Vertex}, \text{Vertex} \rangle \rightarrow \langle \text{Graph} \rangle$ Modifier
- delete $\langle \text{Vertex} \rangle \rightarrow \langle \text{Graph} \rangle$ Modifier
- searchVertex $\langle \text{Graph} \rangle \rightarrow \text{List} \langle \text{Vertex} \rangle$ Analyzer
- dijkstra $\langle \text{Vertex} \rangle \rightarrow \text{Analyzer}$

Operations

Graph (boolean directed, boolean weighted, int n)
Create a new graph that may or may not be directed or weighted.
{pre: }
{post: Graph = {V={}, E={}, directed, weighted } }

insertVertex (Vertex v)
Insert a vertex in the graph.
{pre: $v \notin g.V$ }
{post: $v \in g.V$ }

insertEdge (Vertex v1, Vertex v2)
Add an edge of weight 1 that goes from v1 to v2. If the graph is not directed, it also adds it from v2 to v1.
{pre: $v1, v2 \in g.V$ }
{post: edge = (v1, v2, 1) \in g.E. If g.directed = false, edge = (v2, v1, 1) \in g.E }

insertEdge (Vertex v1, Vertex v2, double weight)
Add an edge of weight 1 that goes from v1 to v2. If the graph is not directed, it also adds it from v2 to v1.
{pre: v1, v2 ∈ g.V, g.weight = true, w > 0}
{post: edge = (v1, v2, weight) ∈ g.E. If g.directed = false, edge = (v2, v1, weight) ∈ g.E }

delete (Vertex v)
Eliminate v from the graph
{pre: v ∈ g.V }
{post: v ∉ g.V. All vertices that are incidents with v ∉ g. E }

dijkstra (Vertex v)
Carry out the Dijkstra algorithm, taking v as the initial vertex
{pre: v ∈ g.V, g }
{post: ∀v ∈ g.V, adds attributes v.pred and v.d, corresponding respectively to the predecessor and the distance added by Dijkstra's algorithm}

search(Vertex v)

Returns if there is a vertex with the given value in the graph.

{pre: }

{post: true if $\exists x \in g.V$: value