TAD



|  |
| --- |
| Graph A (Adjacency Matrix) |
| G = (V,E)  A = [aij]  m = The number of rows  n = The number of columns  Non-multiple graphs :  1, if {vi, vj} is an edge of G  aij = { 0, otherwise  Weighted graphs :  The weight of the edge, if {vi, vj} is an edge of G  aij = { otherwise  Multiple graphs :  The number of adjacent graphs, if {vi, vj} is an edge of G  aij = { 0, otherwise |
| {inv:   * m and n are constant integer numbers. * The adjacency matrix always has integers (except for infinity that is used to represent that there is no connection between one vertex and another when using weighted graphs) * All kinds of undirected graphs form a symmetric matrix } |
|  |
| |  |  |  |  | | --- | --- | --- | --- | | Name | Operation Type | Input | Output | | GraphA | Constructor | m: The number of rows  n: The number of columns | New graph | | addEdge | Modifier | vi, vj | New edge | | BFS | Analyzer | v: The start vertex | BFS Tree | | Dijkstra | Analyzer | v: The start vertex | Array of vertices | |

|  |
| --- |
| GraphA(m,n):  “Creates a new adjacency matrix (graph representation) with m rows and n columns”  {Pre: TRUE}  {Post: GraphA = new vertex[m][n] } |

|  |
| --- |
| addEdge(vi, vj):  “Creates a new edge between vi and vj ”  {Pre: Graph must be initialized}  {Post: Now there is an edge between vi and vj} |

|  |
| --- |
| BFS(v) :  “Shows all vertices at a distance K from an initial vertex”  {Pre: Graph must be initialized}  {Post: Generate a BFS tree that contains all vertices at a distance K from an initial vertex} |

|  |
| --- |
| Dijkstra(v):  “Find the best path (lowest possible weight) between a chosen vertex and the rest of them.”  {Pre: Graph must be initialized}  {Post: An array of distances and an array of vertices that allow finding the path with the least weight.} |

|  |
| --- |
| Graph B (Adjacency list) |
| G = (V,E)  S = The number of vertices of the graph |
| {inv: S is constant} |
|  |
| |  |  |  |  | | --- | --- | --- | --- | | Name | Operation Type | Input | Output | | GraphB | Constructor | S: The number of vertices | New graph | | addEdge | Modifier | vi, vj | New edge | | BFS | Analyzer | v: The start vertex | BFS Tree | | Dijkstra | Analyzer | v: The start vertex | Array of vertex | |

|  |
| --- |
| GraphB(S):  “Creates a new adjacency list (graph representation) with s vertices”  {Pre: TRUE}  {Post: GraphB = new adjacencyList(S) } |

|  |
| --- |
| addEdge(vi, vj):  “Creates a new edge between vi and vj ”  {Pre: Graph must be initialized}  {Post: Now there is an edge between vi and vj} |

|  |
| --- |
| BFS(v) :  “Shows all vertices at a distance K from an initial vertex”  {Pre: Graph must be initialized}  {Post: Generate a BFS tree that contains all vertices at a distance K from an initial vertex} |

|  |
| --- |
| Dijkstra(v):  “Find the best path (lowest possible weight) between a chosen vertex and the rest of them.”  {Pre: Graph must be initialized}  {Post: An array of distances and an array of vertices that allow finding the path with the least weight.} |