Elementen:

Een graaf bestaat uit knopen en takken. Toegang tot de informatie in een knoop loopt via een unieke integer sleutel (eventueel via een hash tabel op te zetten). Een tak verbindt een paar specifieke knopen; takken kunnen gericht of ongericht zijn; een tak mag geen parallel gerichte tak hebben.

Structuur:

Er zijn N knopen:voor elk mogelijk paar sleutels i.j met i,j ε [1,N] is er plaats voor takinformatie (Boolean voor wel/niet tak).

Een tak is een unieke 1-op-1 relatie tussen knoop(i) en knoop(j) i,j ε [1,N]; knopen kunnen [0..N] in- en/of uitgaande takken hebben. Lussen zijn toegestaan.

Domein:

Het aantal elementen in een graaf is begrensd: als aantal knopen is N, dan max aantal takken is N^2.

Type: Graaf

Operaties: er zijn 5 operaties gedefinieerd:

Procedure **Create**(var G:Graaf;max:Integer:var created:Boolean);

Postconditie: als created=Y, dan is G een graaf met plaats voor max^2 takken; takken staan op uit.

Procedure **Delete**(var G:Graaf);

Postconditie: graaf G bestaat niet (meer).

Procedure **InsertEdge**(var G:Graaf; key1,key2:Integer;var inserted:Boolean);

Preconditie: key1,key2 ε [1..max]

Postconditie: inserted=Y dan is gerichte tak key1 -> key2 op 1 gezet, anders inserted=N

Procedure **DeleteEdge**(var G:Graaf; key1,key2:Integer;var deleted:Boolean);

Preconditie: key1,key2 ε [1..max]

Postconditie: deleted=Y dan is gerichte tak key1 -> key2 op 0 gezet, anders deleted=N

Function **EdgePresent**(var G:Graaf; key1,key2:Integer):Boolean;

Preconditie: key1,key2 ε [1..max]

Postconditie: EdgePresent=Y als tak key1->key2=1, anders N

Elementen:

Een graaf bestaat uit knopen en takken. Toegang tot de informatie in een knoop loopt via een unieke integer sleutel. Een tak verbindt een paar specifieke knopen; takken kunnen gericht of ongericht zijn; een tak mag geen parallel gerichte tak hebben.

Structuur:

Er zijn maximaal N knopen:voor elke knoop in de lijst is er plaats voor een linked list van knopen waarmee een gerichte verbinding bestaat. Een tak is een unieke 1-op-1 relatie tussen knoop(i) en knoop(j) i,j ε [1,N]; knopen kunnen [0..N] in- en/of uitgaande takken hebben. Lussen zijn toegestaan.

Domein:

Het aantal elementen in een graaf is begrensd: het maximum aantal knopen is N, evenals het maximum aantal knopen waarmee het verbonden is; voor de graaf als geheel zijn er dan max N^2 takken.

Type: Graaf

Operaties: Er zijn 10 operaties gedefinieerd:

Procedure **Create**(var G:Graaf;max:Integer:var created:Boolean);

Postconditie: als created=Y, dan is G een graaf met plaats voor max knopen met elk maximaal max eindknopen.

Procedure **Delete**(var G:Graaf);

Postconditie: graaf G bestaat niet (meer).

Function **FullNodes**(var G:Graaf):Boolean;

Postconditie: FullNodes=Y als de Graaf maximum aantal toegestane knopen heeft, anders N.

Function **FullNeighbors**(var G:Graaf;var key:Integer):Boolean;

Preconditie: key moet voorkomen

Postconditie: FullNeighbors=Y als de Graaf voor knoop=key het maximum aantal toegestane buurknopen heeft, anders N.

Procedure **InsertEdge**(var G:Graaf; key1,key2:Integer;var inserted:Boolean);

Preconditie: key1,key2 ε [1..max]

Postconditie: inserted=Y dan is key2 in linked-list van key1 gezet, anders inserted=N

Procedure **DeleteEdge**(var G:Graaf; key1,key2:Integer;var deleted:Boolean);

Preconditie: key1,key2 ε [1..max]

Postconditie: deleted=Y dan is key2 uit linked-list van key1 gehaald, anders deleted=N

Function **EdgePresent**(var G:Graaf; key1,key2:Integer):Boolean;

Preconditie: key1,key2 ε [1..max]

Postconditie: present=Y als key2 in lijst vertex met key1, anders N

Procedure **InsertVertex**(var G:Graaf; key:Integer;var inserted:Boolean);

Preconditie: key ε [1..max]; key nog niet in lijst

Postconditie: inserted=Y dan is key toegevoegd, anders inserted=N

Procedure **DeleteVertex**(var G:Graaf; key:Integer;var deleted:Boolean);

Preconditie: key ε [1..max]; key in lijst aanwezig

Postconditie: deleted=Y dan is vertex met key inclusief edge list verwijderd, anders deleted=N

Function **VertexPresent**(var G:Graaf; key:Integer):Boolean;

Preconditie: key ε [1..max]

Postconditie: present=Y als key in lijst, anders N