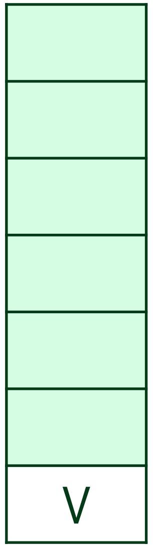
Adjacency List

for a directed, weighted graph

Three classes

- DirectedGraph<T>
  - List<Vertex<T>> V
- Vertex<T>
  - T Name
  - List<Edge<T>> E
  - bool Visited (used for depth-first and breadth-first searches)
- Edge<T>
  - Vertex<T> AdjVertex
  - int Cost

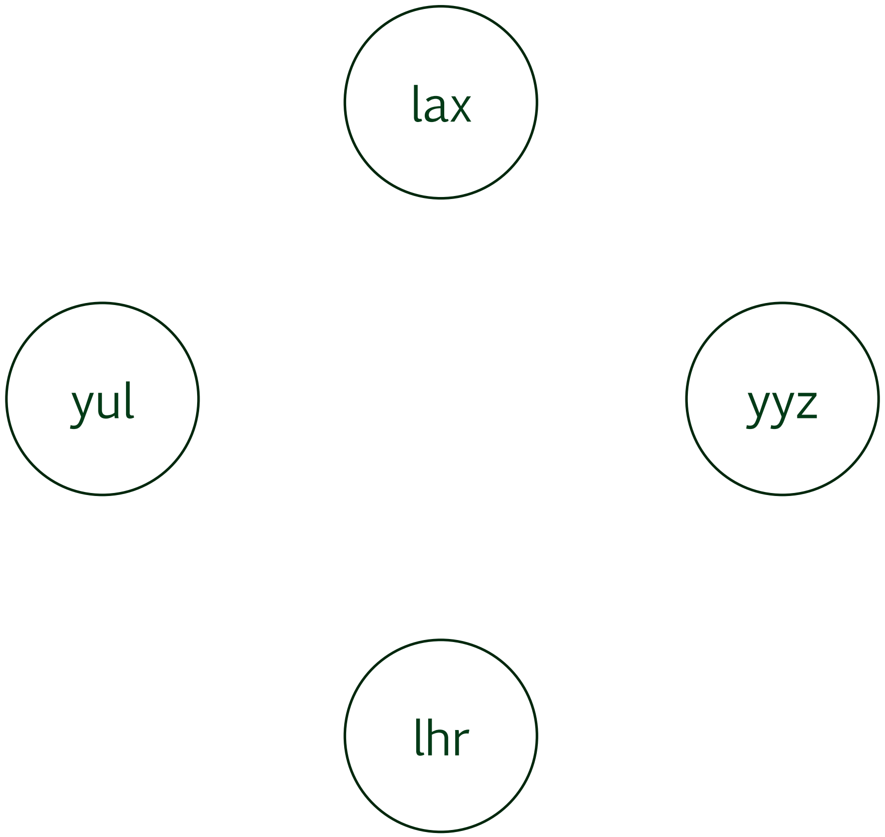
Initially

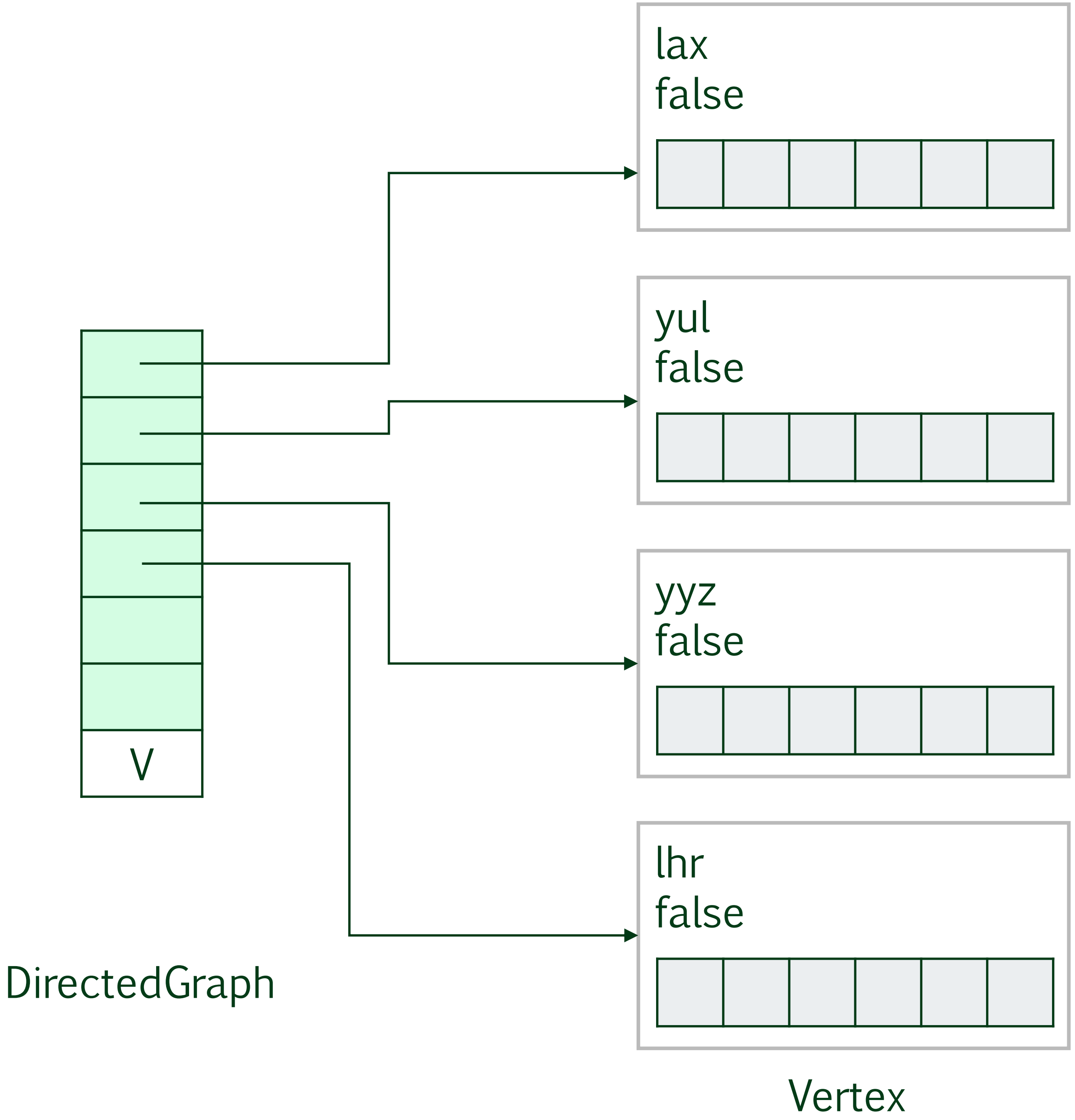


DirectedGraph

Let’s add four vertices using AddVertex

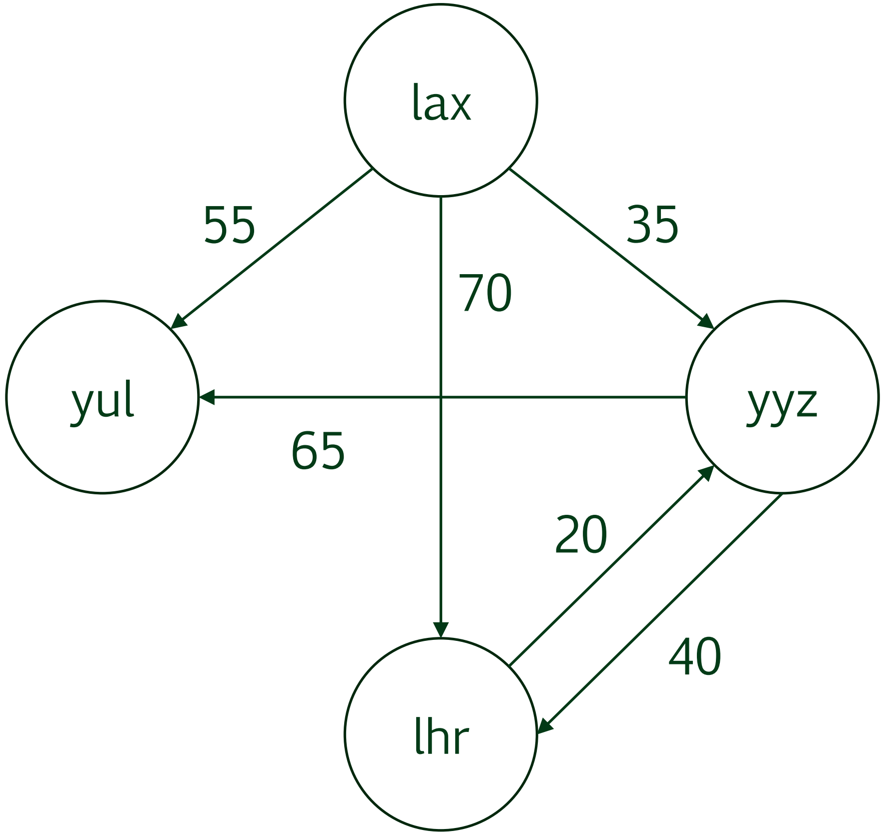
- “lax”
- “yul”
- “yyz”
- “lhr”

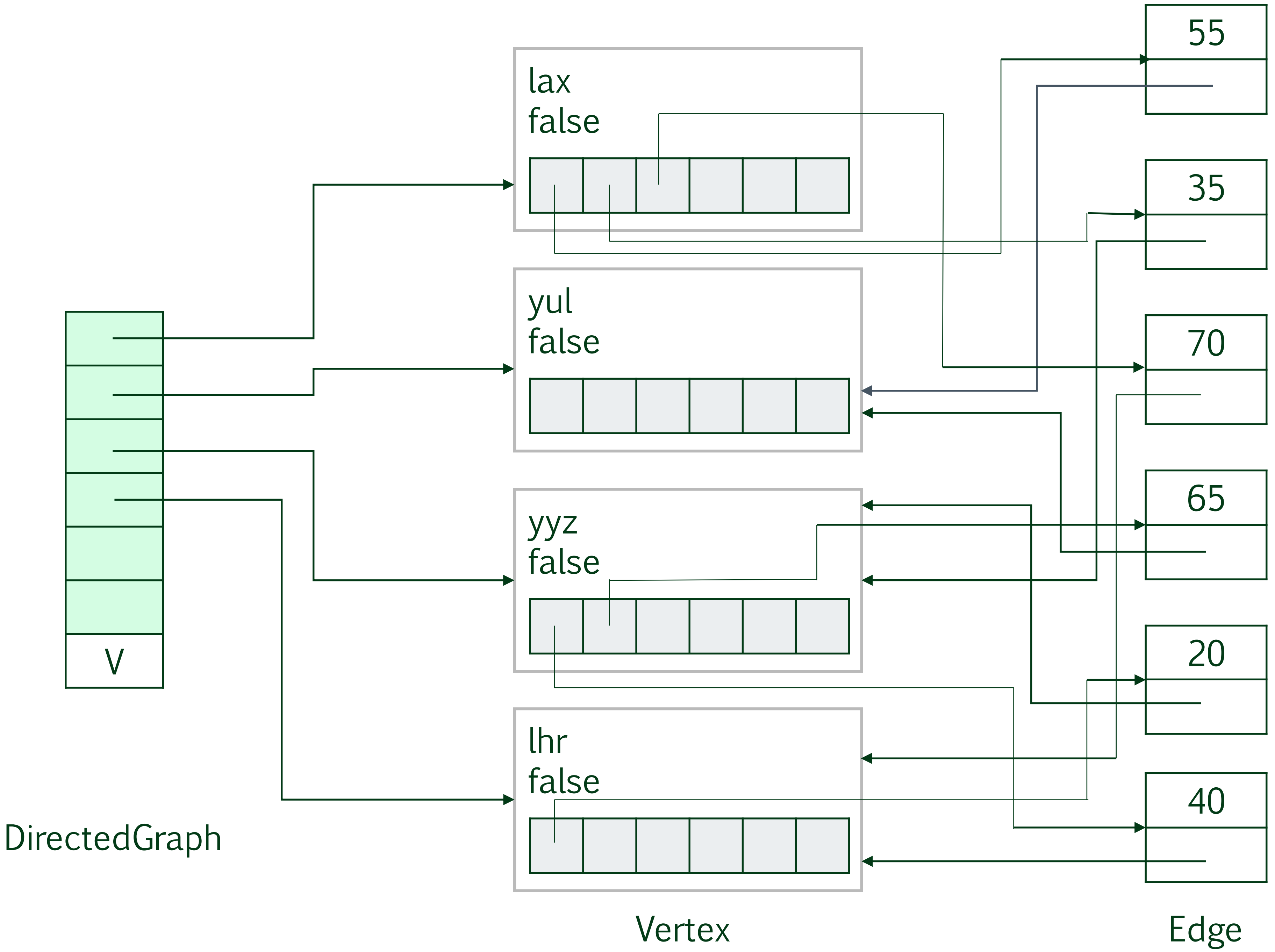



lax
false
yul
false
yyz
false
lhr
false
V
DirectedGraph
Vertex

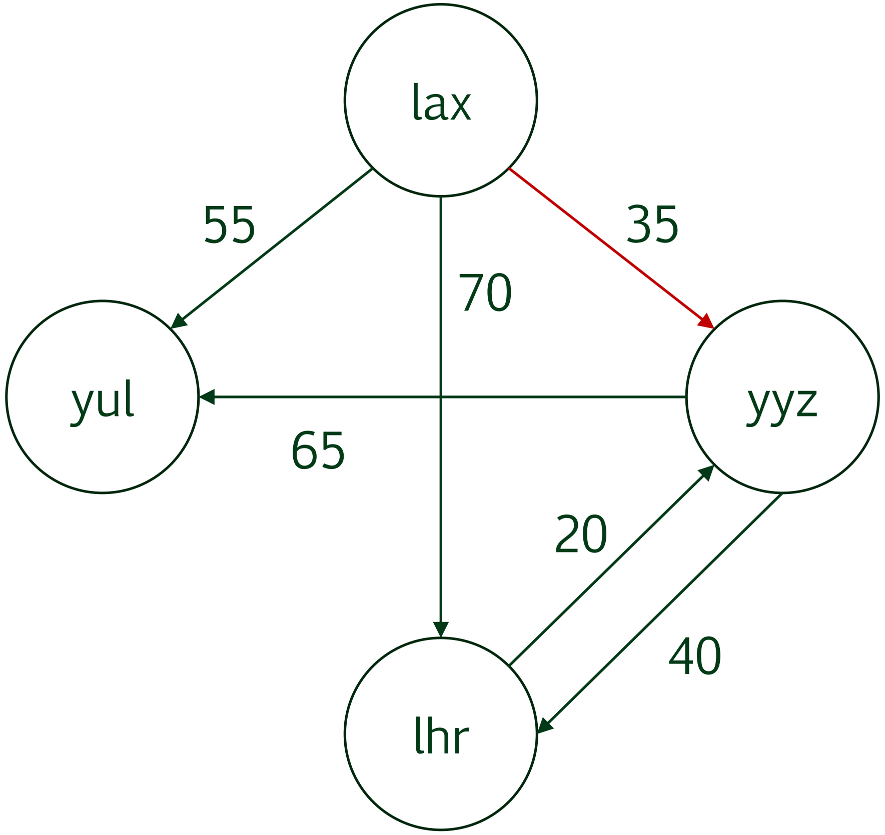
Now let’s add six edges using AddEdge

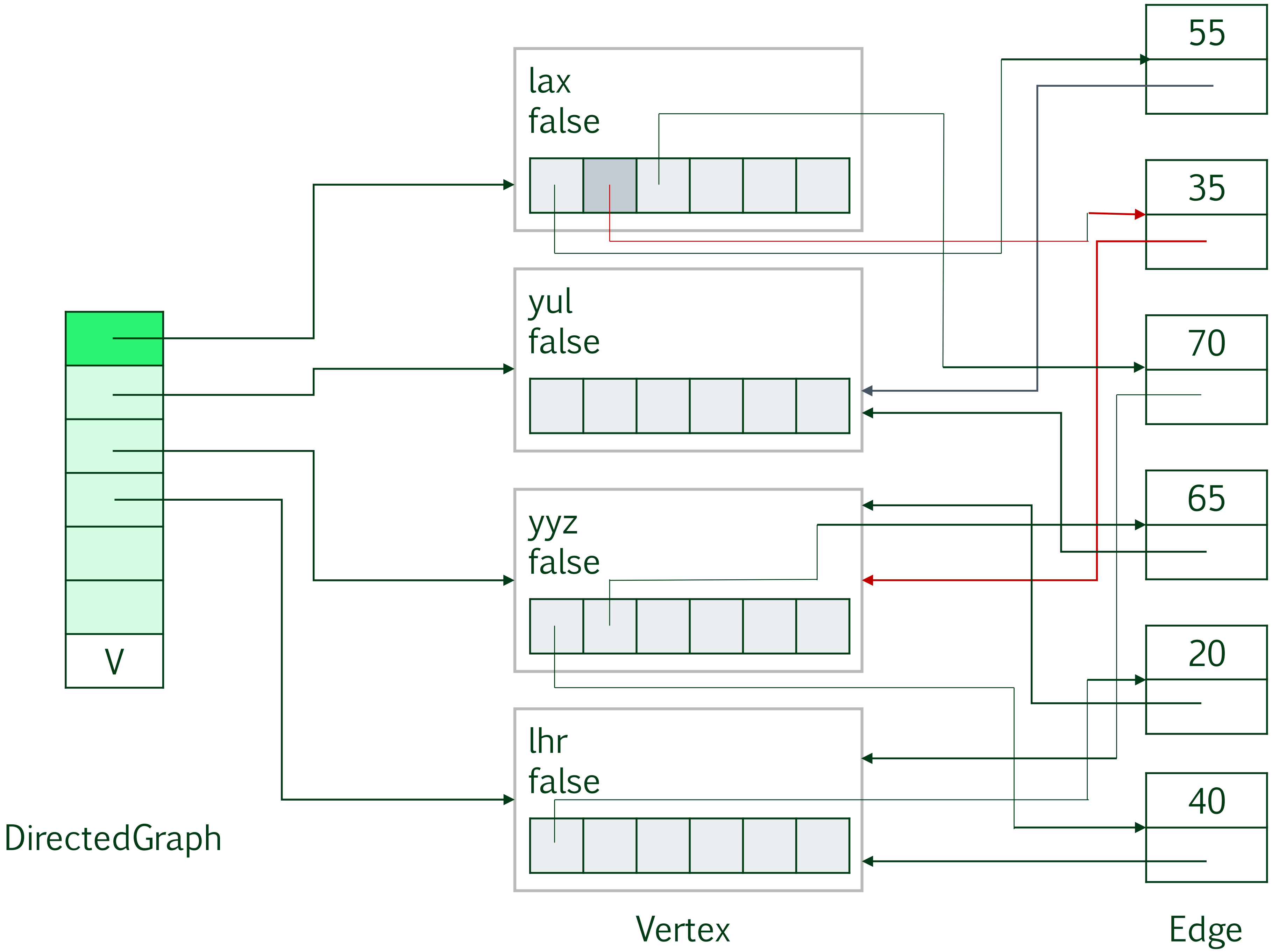
- (“lax”, “yul”, 55)
- (“yyz”, “lhr”, 40)
- (“lax”, “yyz”, 35)
- (“lhr”, “yyz”, 20)
- (“yyz”, “yul”, 65)
- (“lax”, “lhr”, 70)

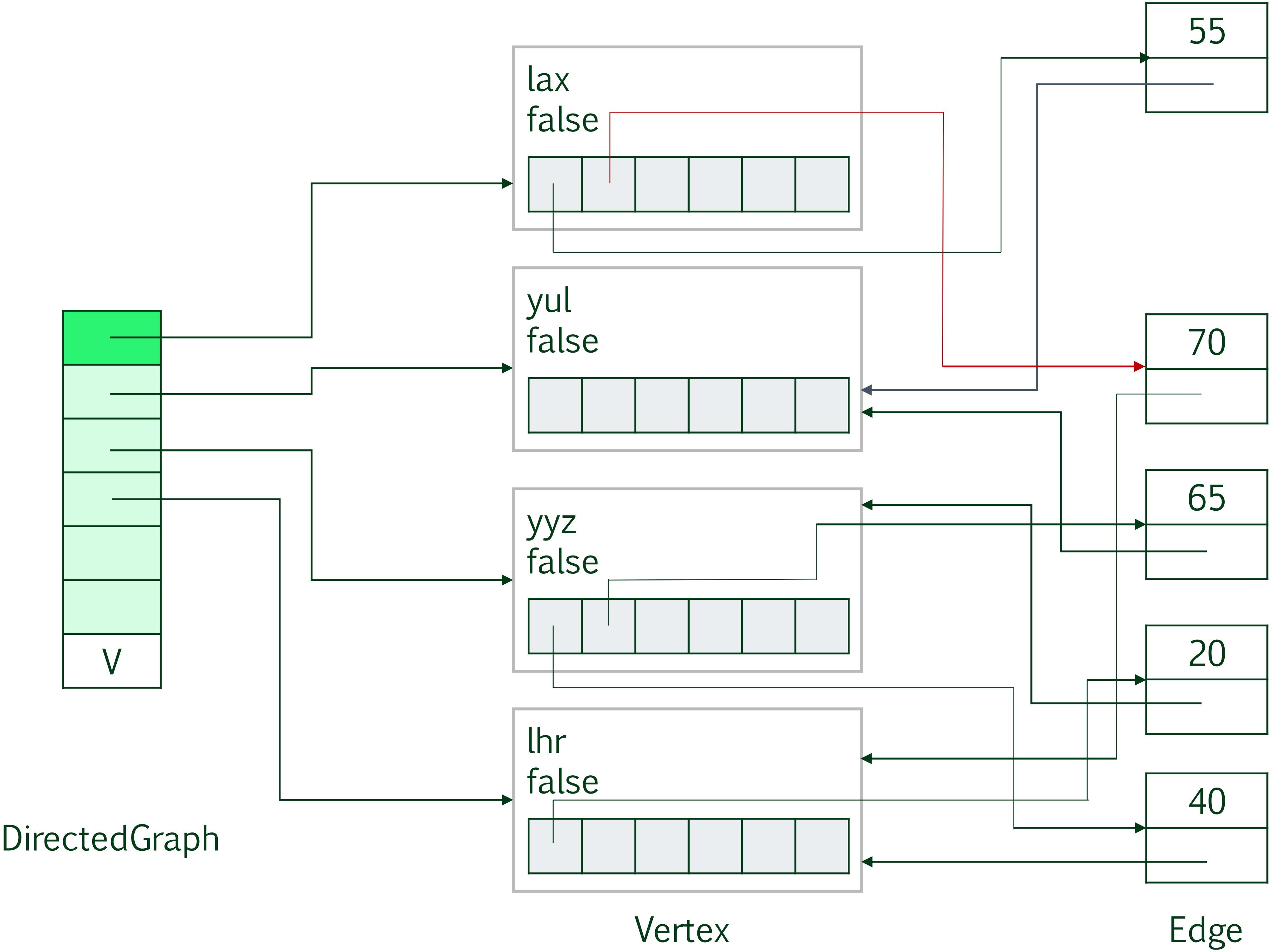



55
35
70
65
20
40
lax
false
yul
false
yyz
false
lhr
false
V
DirectedGraph
Vertex
Edge

RemoveEdge(“lax”, “yyz”)

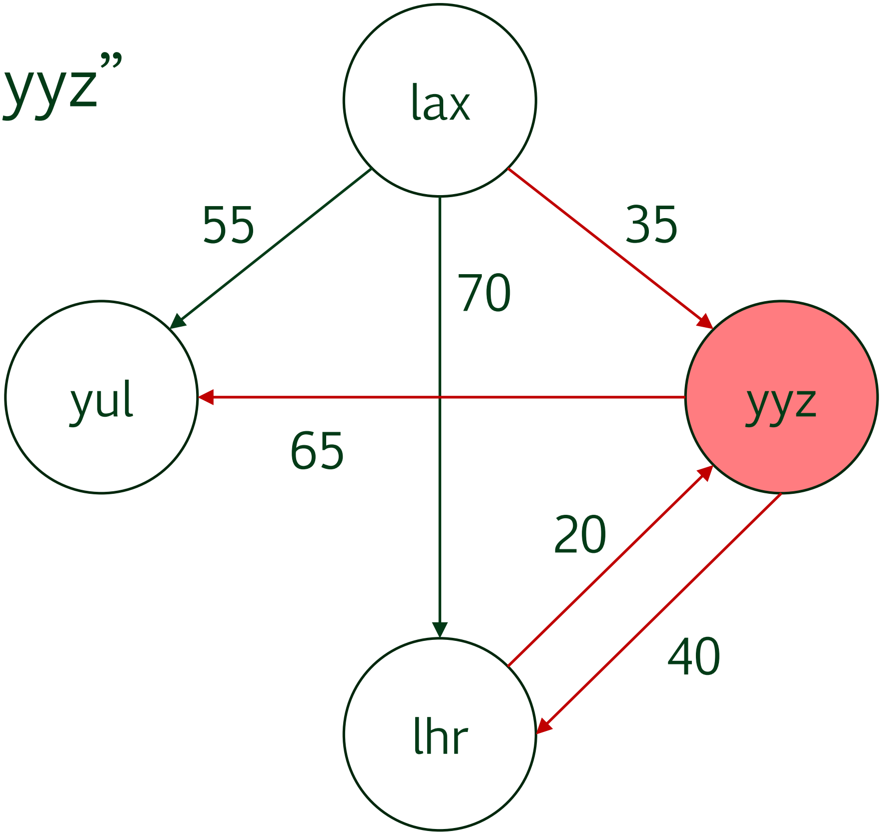


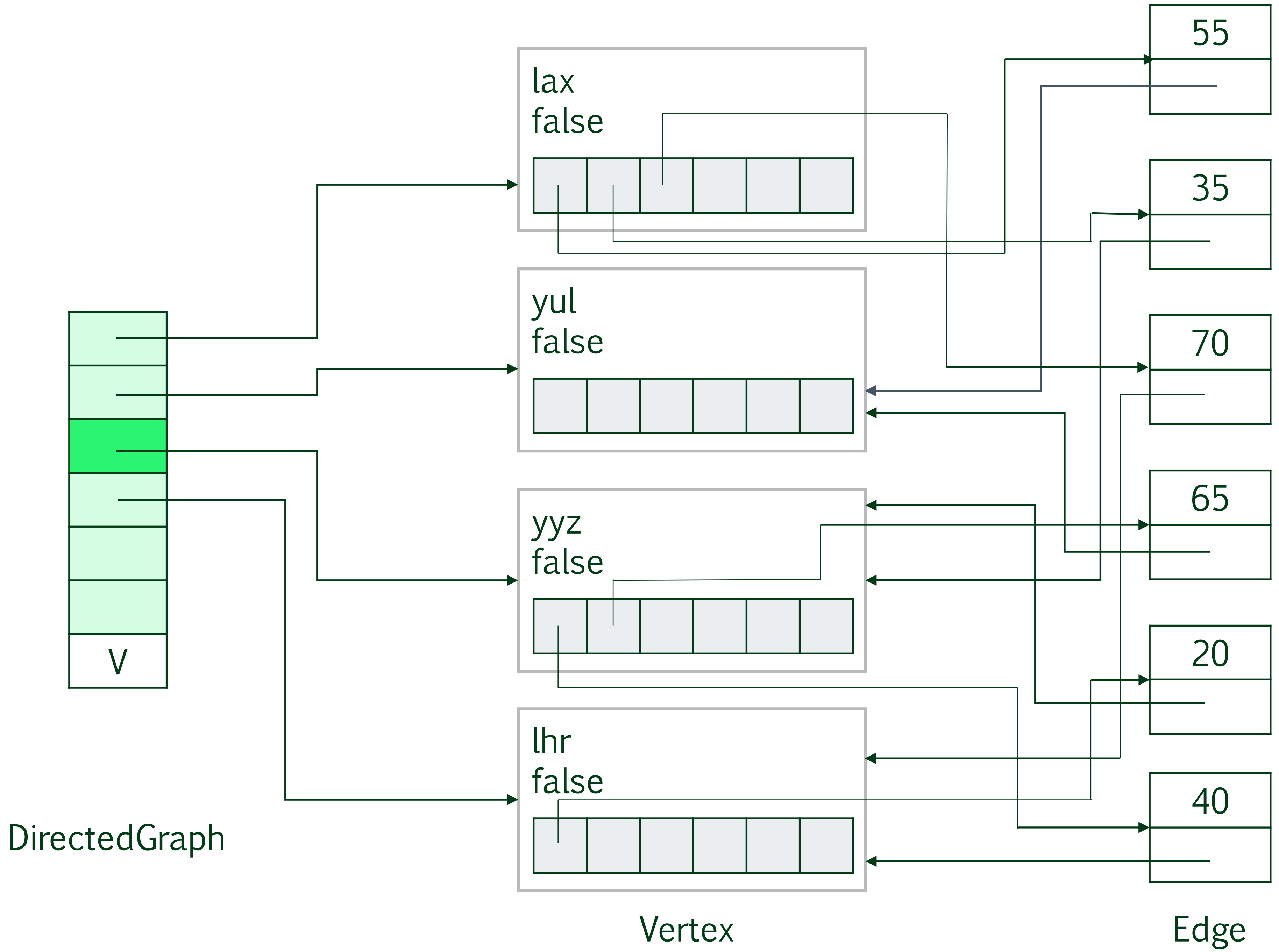

lax
false
yul
false
yyz
false
lhr
false
V
DirectedGraph
55
35
70
65
20
40
Vertex
Edge

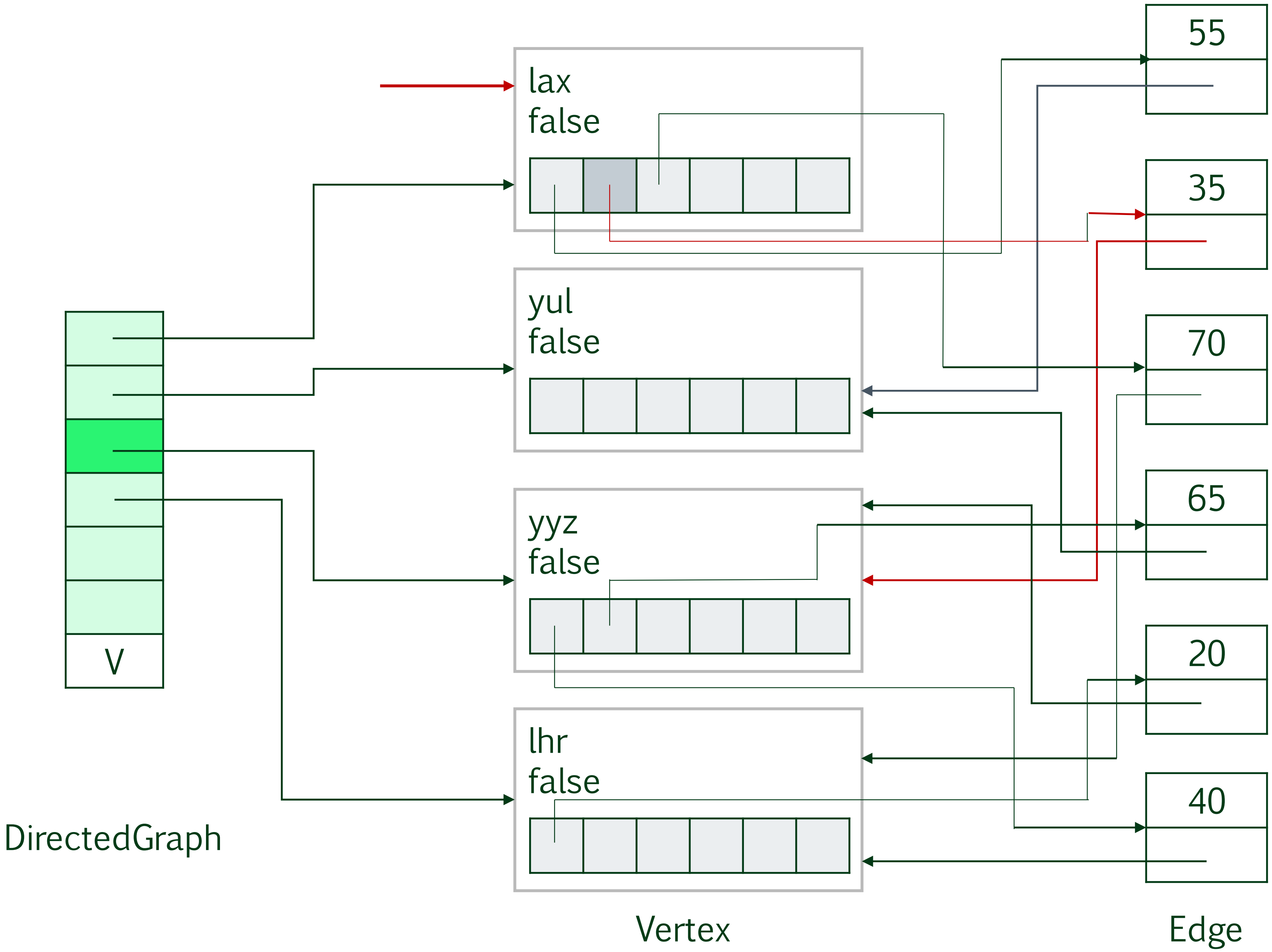

55
lax
false
yul
false
70
65
yyz
false
V
20
lhr
false
40
DirectedGraph
Vertex
Edge

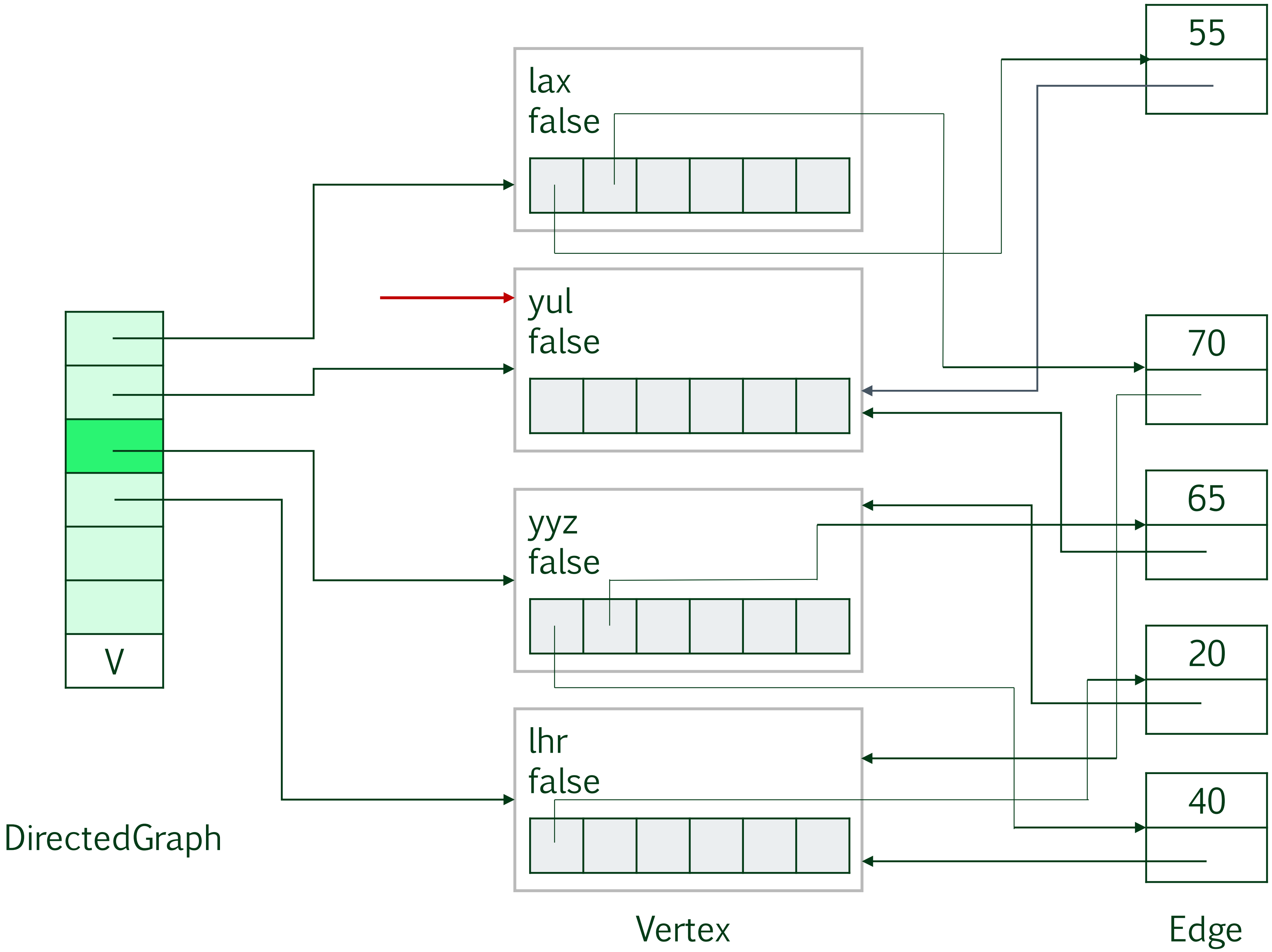
RemoveVertex(“yyz”)

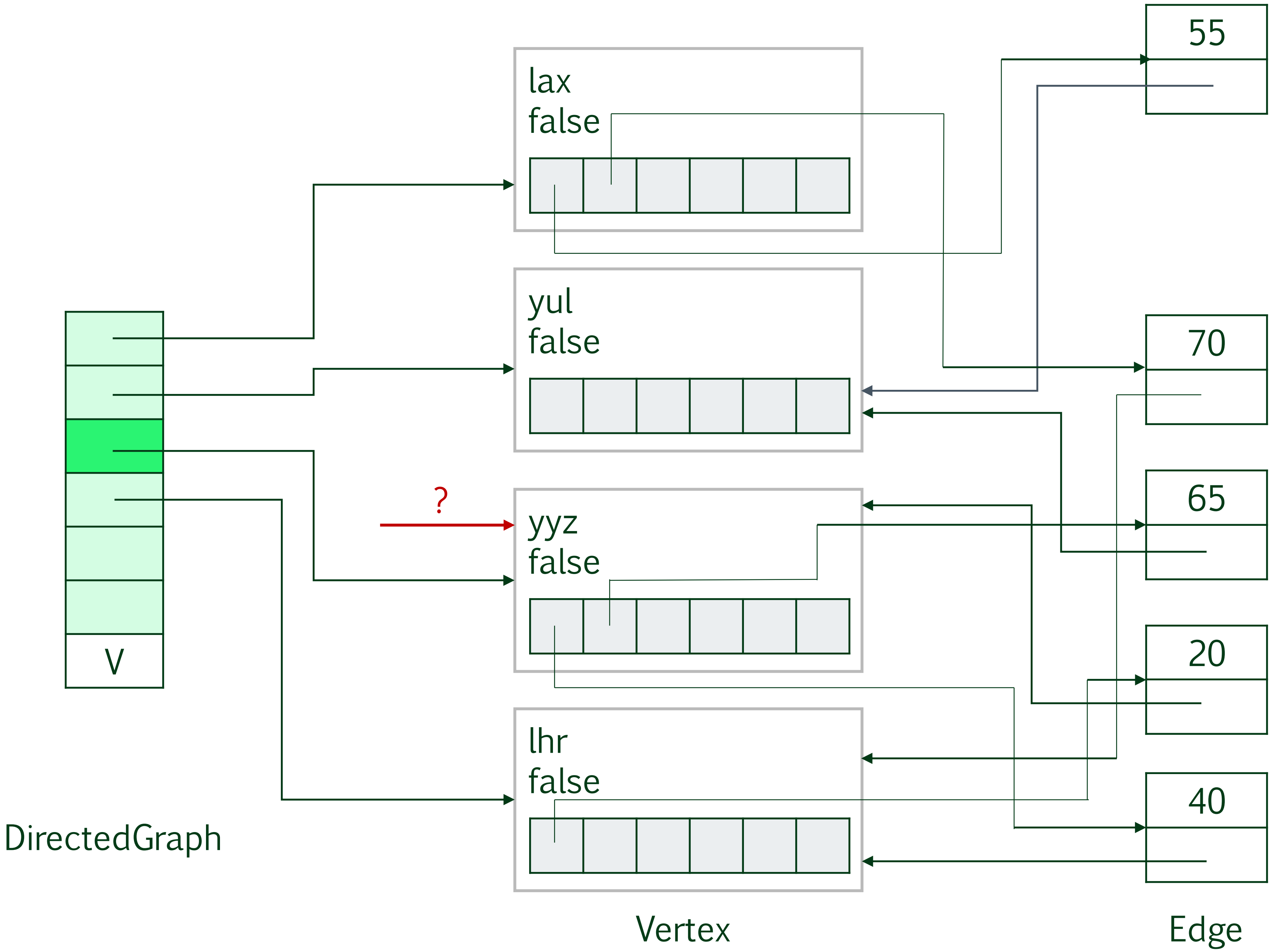
- Two steps
  - Remove edges leading to “yyz”
  - Remove vertex “yyz”

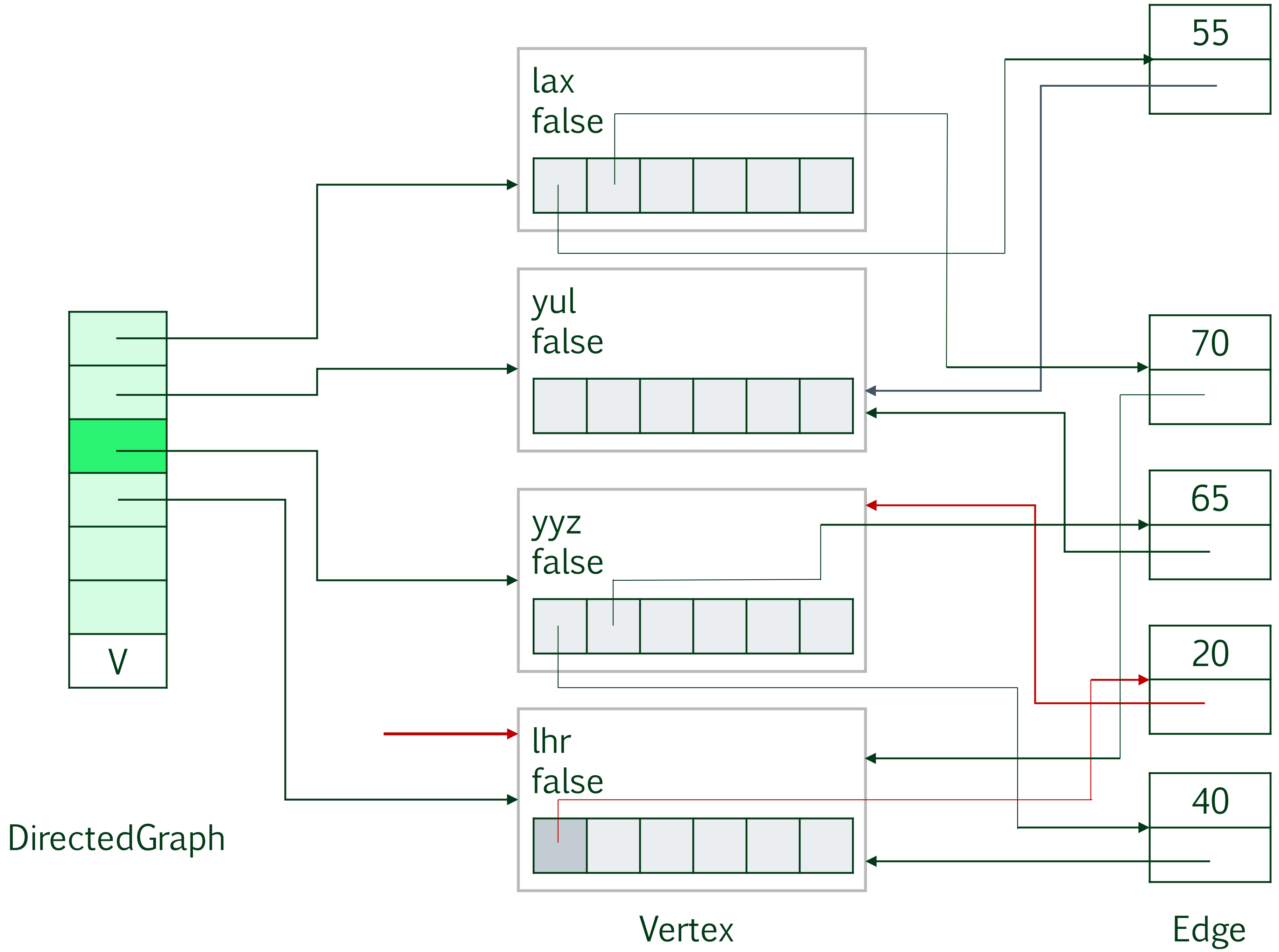


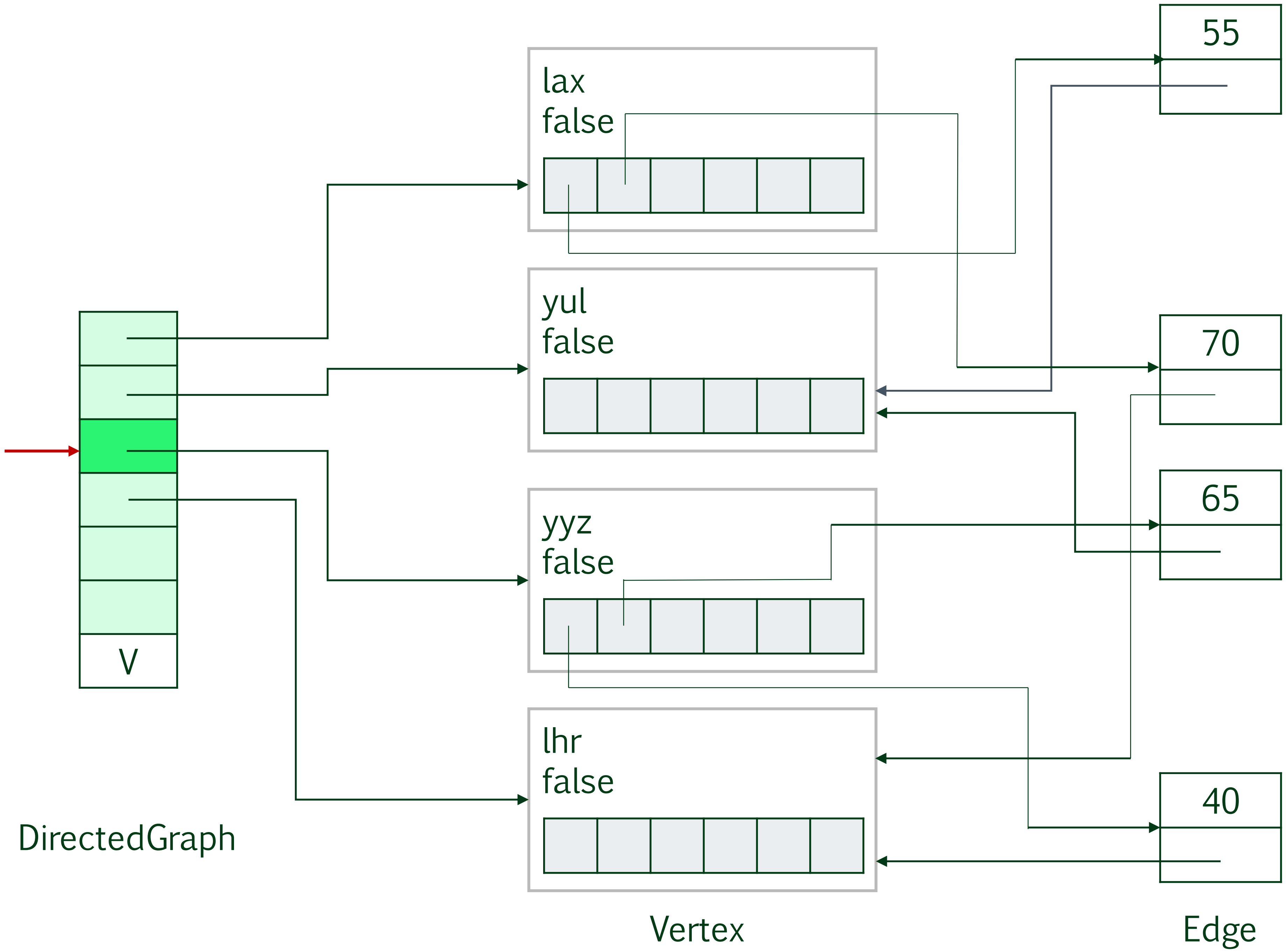

lax
false
yul
false
yyz
false
lhr
false
V
DirectedGraph
Vertex
Edge
55
35
70
65
20
40

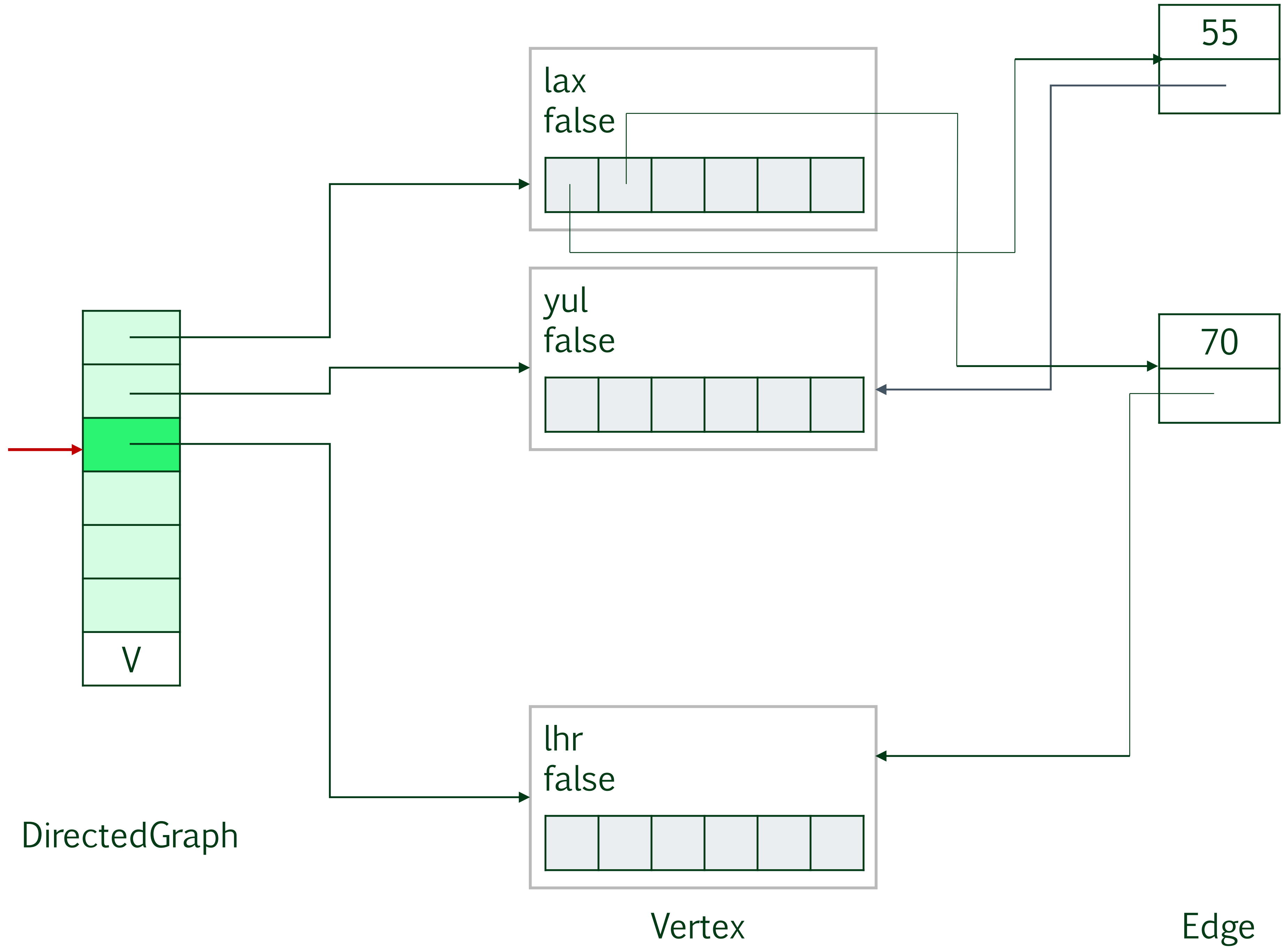

lax
false
yul
false
yyz
false
lhr
false
55
35
70
65
20
40
V
DirectedGraph
Vertex
Edge


55
lax
false
yul
false
70
65
yyz
false
V
20
lhr
false
40
DirectedGraph
Vertex
Edge

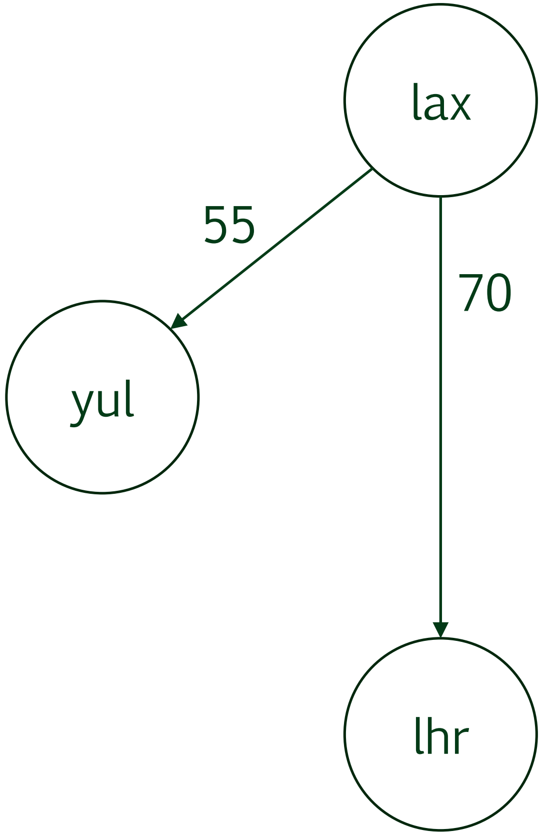

lax
false
yul
false
?
yyz
false
lhr
false
V
DirectedGraph
Vertex
55
70
65
20
40
Edge


55
lax
false
yul
false
70
65
yyz
false
20
V
lhr
false
40
DirectedGraph
Vertex
Edge


lax
false
yul
false
yyz
false
lhr
false
V
DirectedGraph
Vertex
55
70
65
40
Edge


55
lax
false
yul
false
70
V
lhr
false
DirectedGraph
Vertex
Edge

After “yyz” has been removed



Two supporting methods

- int FindVertex(T name)   // of DirectedGraph class
  - Returns the index of the given vertex in V (if found); otherwise returns -1

- int FindEdge(T name)   // of Vertex class
  - Returns the index of the given (adjacent) vertex in E (if found); otherwise returns -1

Comparison

ADJACENCY MATRIX

- Better for dense graphs
- Time complexity of the four basic methods is not dependent on the number of edges

ADJACENCY LIST

- Better for sparse graphs
- Time complexity of RemoveVertex is dependent on the number of edges

Exercises

- Implement the same additional methods that you did for the Adjacency Matrix implementation.

- Argue why the time complexity of RemoveVertex is O(max(n,m)) where n is the number of vertices and m is the number of edges in the Graph. What assumption(s) do you make about the List method RemoveAt?

- Modify the Adjacency List implementation of the class DirectedGraph if the graph is unweighted.

- Modify the Adjacency List implementation of the class DirectedGraph to allow for parallel edges (i.e., edges that have the same endpoints and orientation). Differentiate between parallel edges based on cost.