Graphs

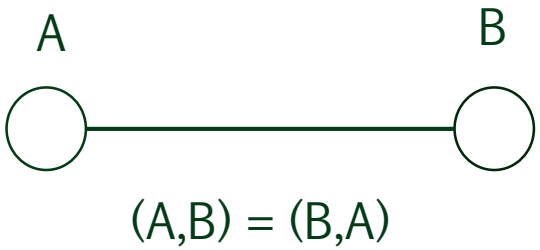
Sections 22.1-22.3 and Appendix B.4

Definition

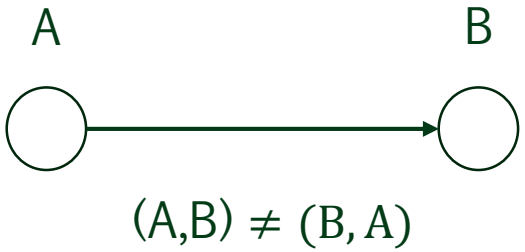
- A graph G is composed of two sets:
  - A set of vertices V
  - A set of edges E whose endpoints are drawn from V

Types of Graphs

- Undirected
  - Each edge connects two vertices but has no orientation



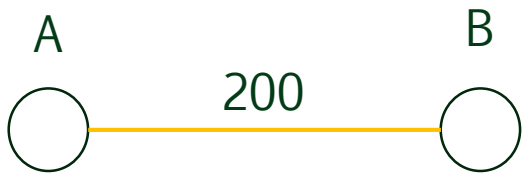
- Directed
  - Each edge connects two vertices and is oriented from one vertex to the other



Types of Graphs

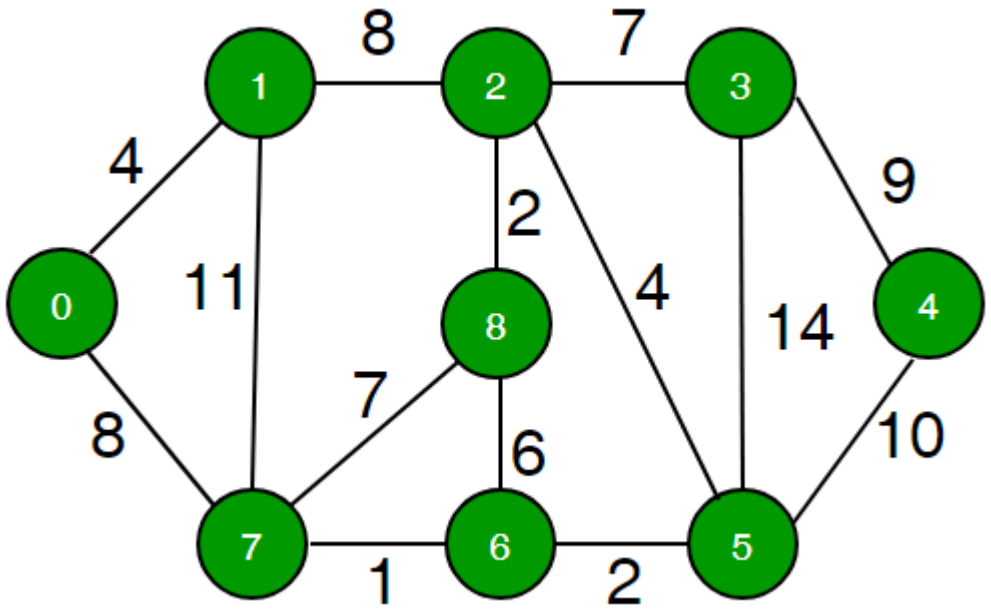
- Unweighted
  - Each edge has no additional attributes

- Weighted
  - Each edge has one or more additional attributes



Undirected, weighted graph

V = { 0, 1, 2, ..., 8 }
E = { (0,1), (0,7), (1,2), (1,7), (2,3), (2,5), (2,8), ... }



Four primary methods

- void AddVertex (T name)
- void RemoveVertex (T name)
- void AddEdge (T name1, T name2, int cost)
- void RemoveEdge (T name1, T name2)

- Which is likely to be the most difficult to implement?

Two important additional methods

- void DepthFirstSearch( )
- void BreadthFirstSearch( )

- Used to traverse the graph (compare with the preorder, inorder, and postorder traversals of a binary tree)

Adjacency Matrix

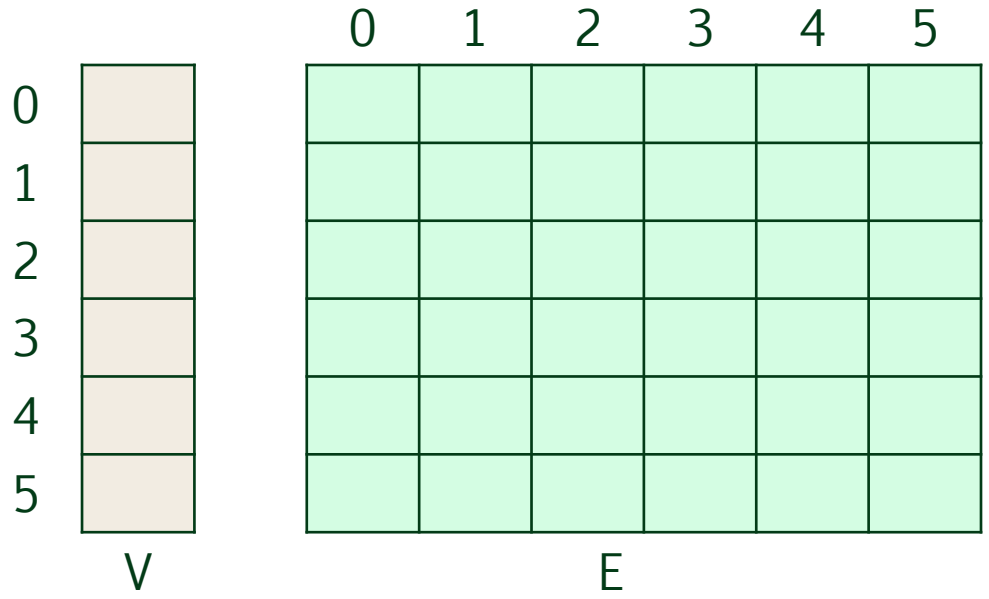
for a directed, weighted graph

Data structure

- public T[] V { set; get; } // Vertex list
- public int[,] E { set; get; } // Adjacency matrix
- public int NumVertices { set; get; }
- public int MaxNumVertices { set; get; }

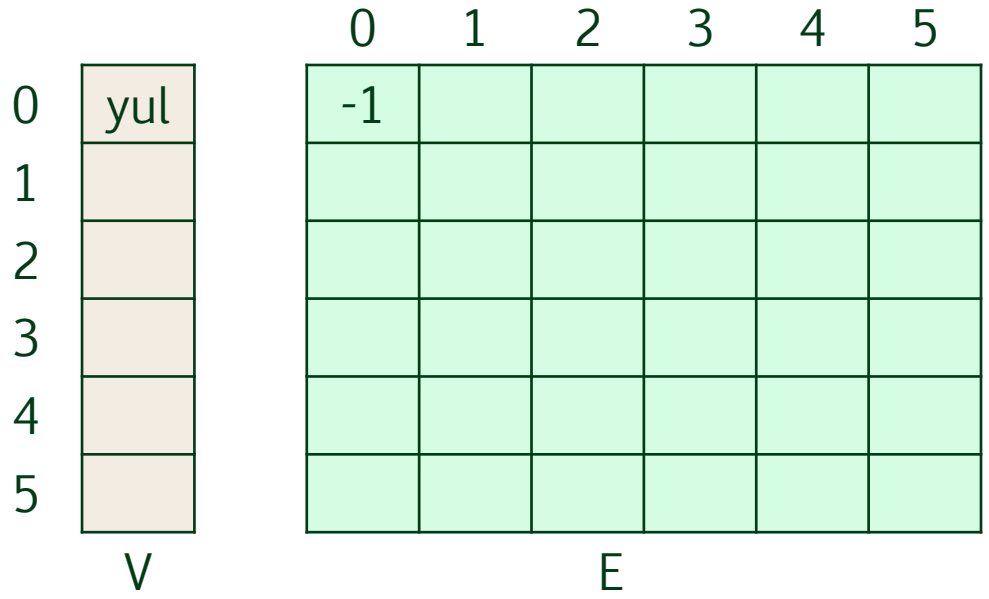
Initially

- NumVertices 0
- MaxNumVertices 6



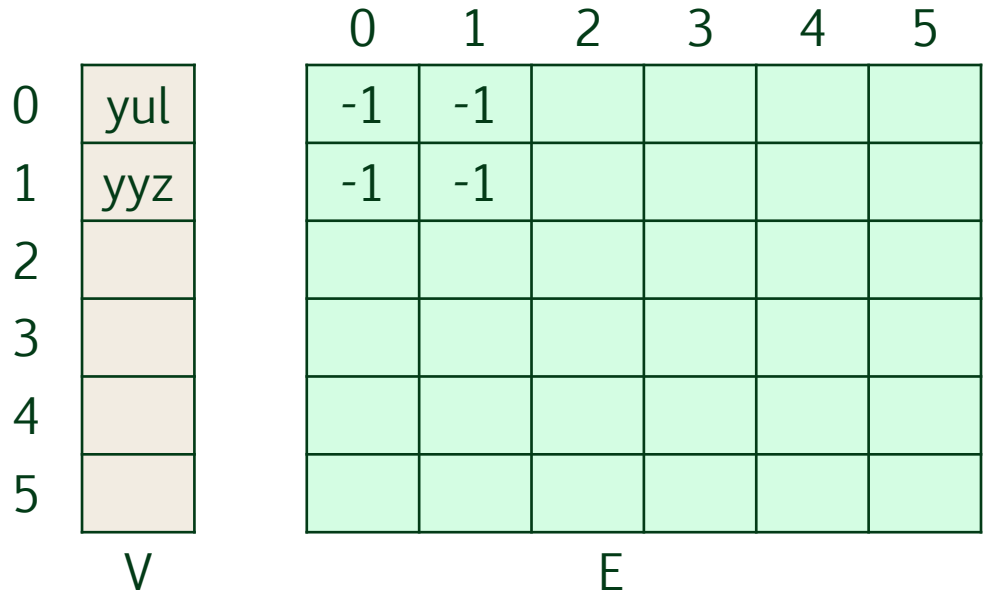
AddVertex(“yul”)

- NumVertices 1
- MaxNumVertices 6



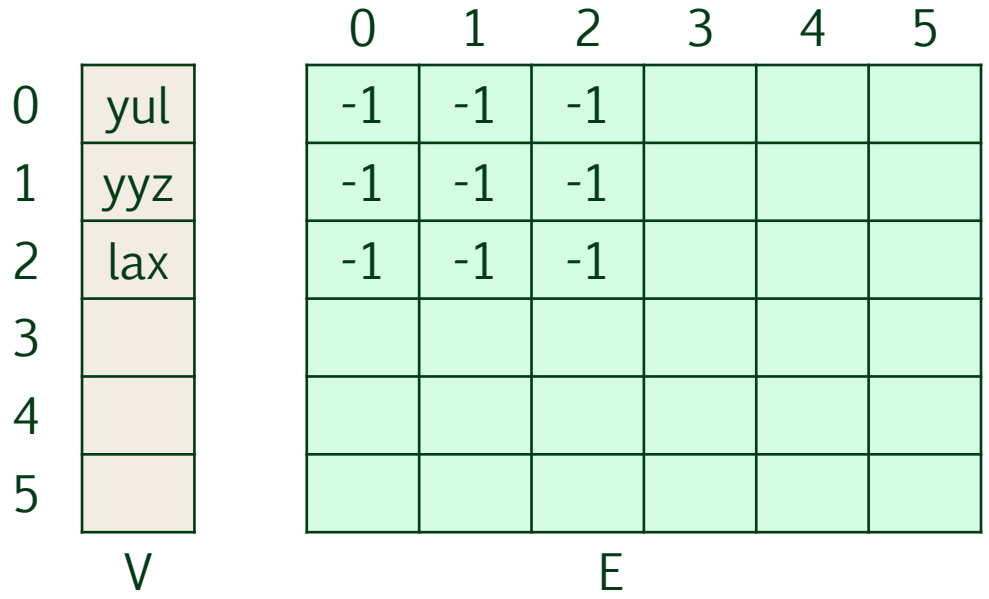
AddVertex(“yyz”)

- NumVertices 2
- MaxNumVertices 6



AddVertex("lax")

- NumVertices 3
- MaxNumVertices 6



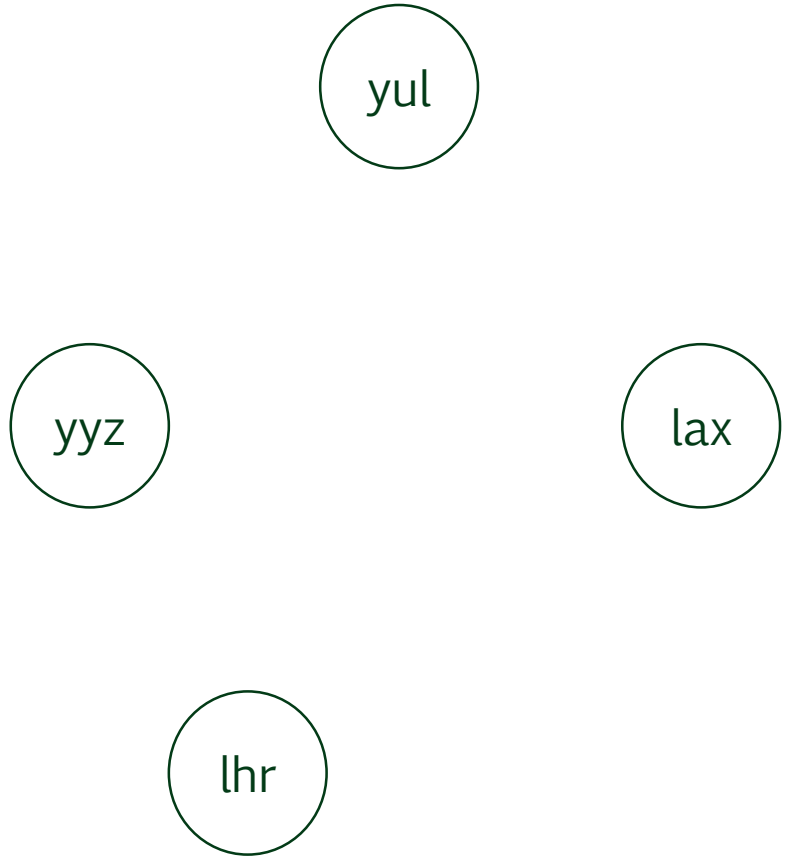
AddVertex("lhr")

- NumVertices 4
- MaxNumVertices 6

	V
0	yul
1	yyz
2	lax
3	lhr
4	
5	

	0	1	2	3	4	5
0	-1	-1	-1	-1		
1	-1	-1	-1	-1		
2	-1	-1	-1	-1		
3	-1	-1	-1	-1		
4						
5						

E



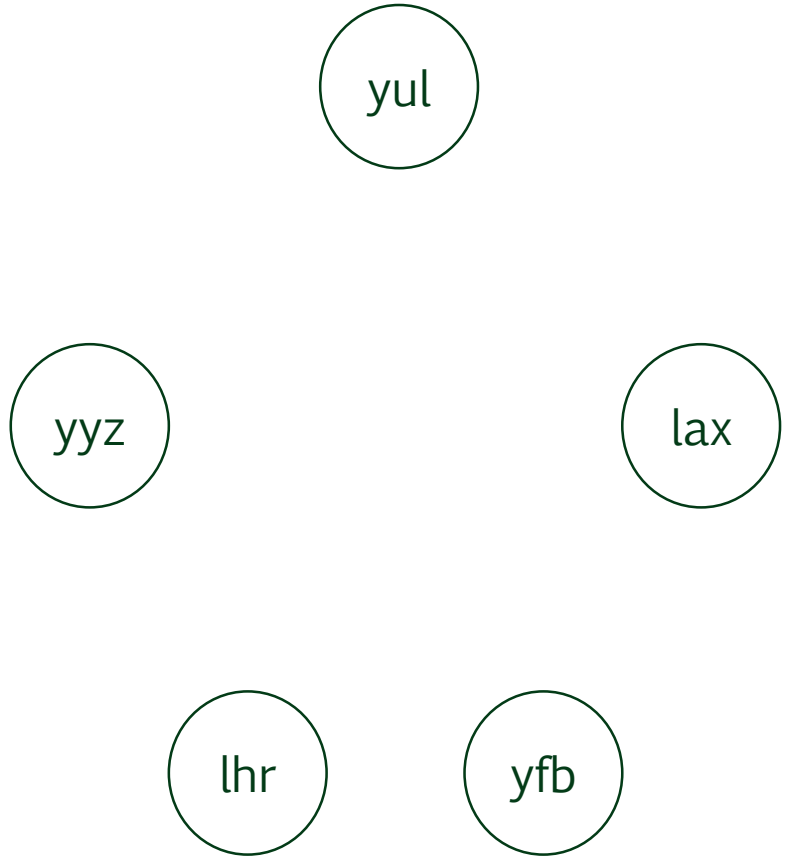
AddVertex("yfb")

- NumVertices 5
- MaxNumVertices 6

	V
0	yul
1	yyz
2	lax
3	lhr
4	yfb
5	

E

	0	1	2	3	4	5
0	-1	-1	-1	-1	-1	
1	-1	-1	-1	-1	-1	
2	-1	-1	-1	-1	-1	
3	-1	-1	-1	-1	-1	
4	-1	-1	-1	-1	-1	
5						



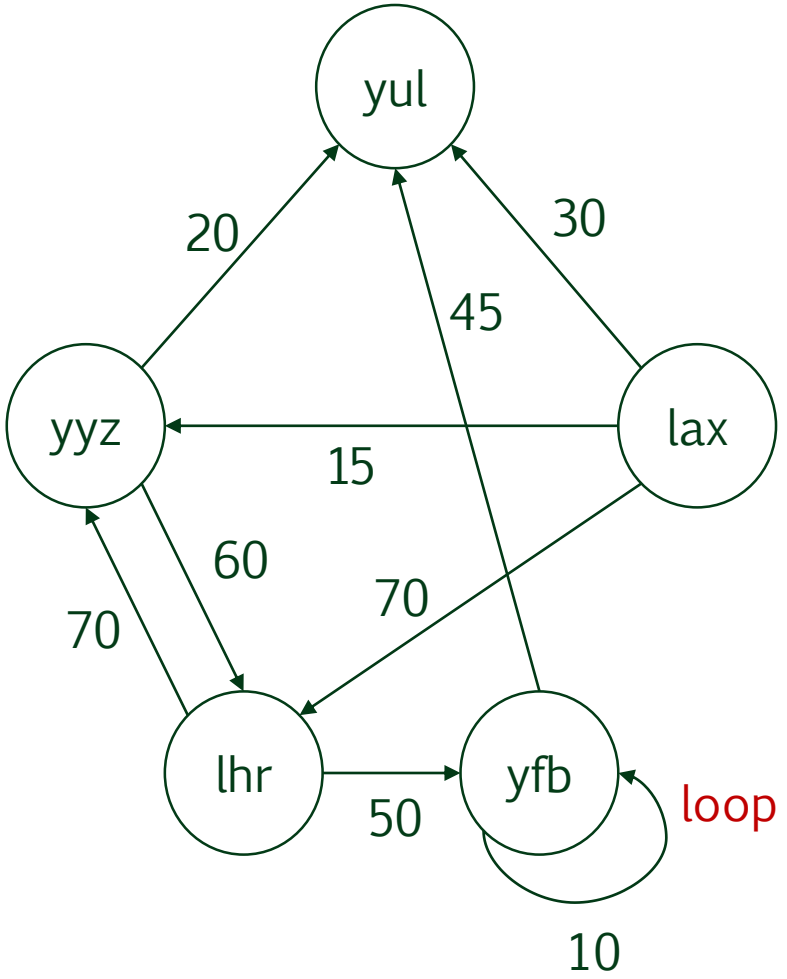
AddEdges

- NumVertices 5
- MaxNumVertices 6

	V
0	yul
1	yyz
2	lax
3	lhr
4	yfb
5	

	0	1	2	3	4	5
0	-1	-1	-1	-1	-1	
1	-1	-1	-1	-1	-1	
2	-1	-1	-1	-1	-1	
3	-1	-1	-1	-1	-1	
4	-1	-1	-1	-1	-1	
5						

E



AddEdges

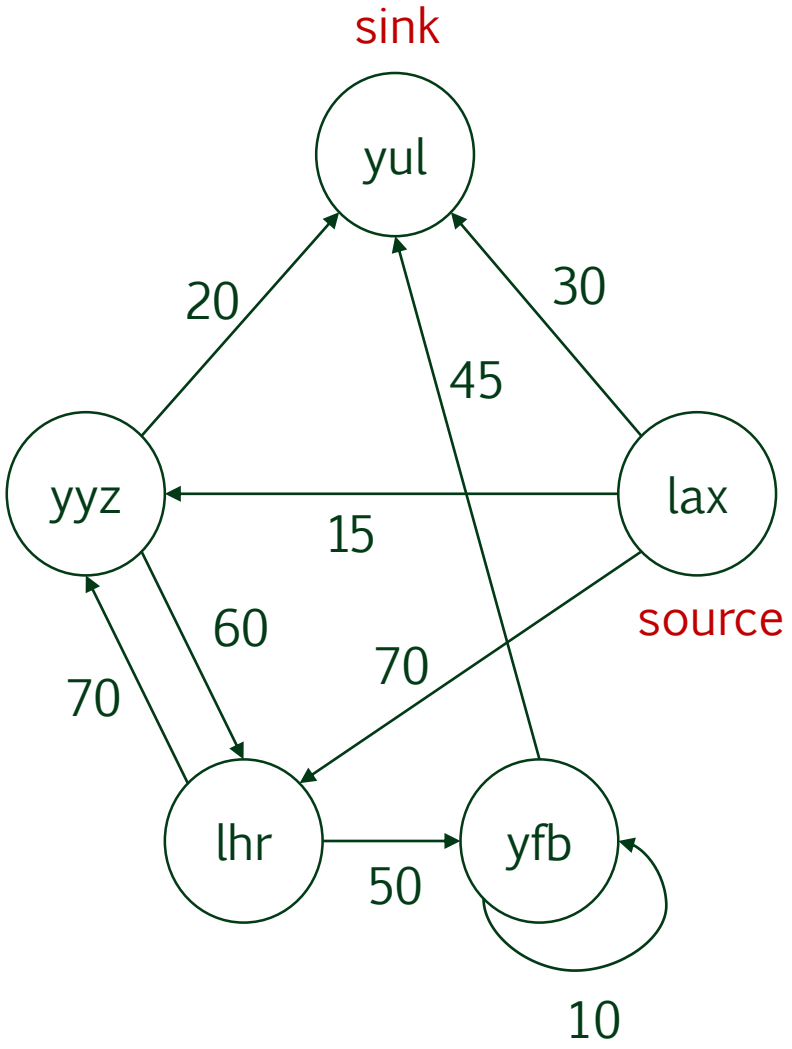
- NumVertices 5
- MaxNumVertices 6

V

0	yul
1	yyz
2	lax
3	lhr
4	yfb
5	

E

	0	1	2	3	4	5
0	-1	-1	-1	-1	-1	
1	20	-1	-1	60	-1	
2	30	15	-1	70	-1	
3	-1	70	-1	-1	50	
4	45	-1	-1	-1	10	
5						



RemoveEdge("lax", "yyz")

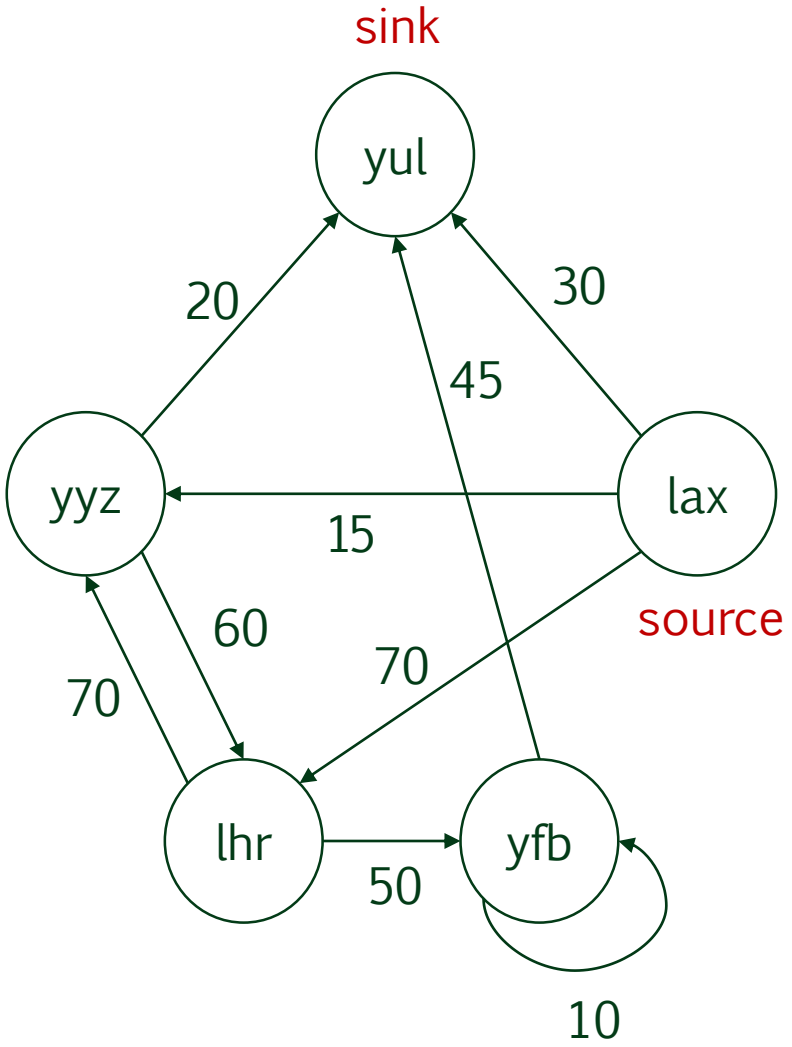
- NumVertices 5
- MaxNumVertices 6

V

0	yul
1	yyz
2	lax
3	lhr
4	yfb
5	

E

	0	1	2	3	4	5
0	-1	-1	-1	-1	-1	
1	20	-1	-1	60	-1	
2	30	15	-1	70	-1	
3	-1	70	-1	-1	50	
4	45	-1	-1	-1	10	
5						



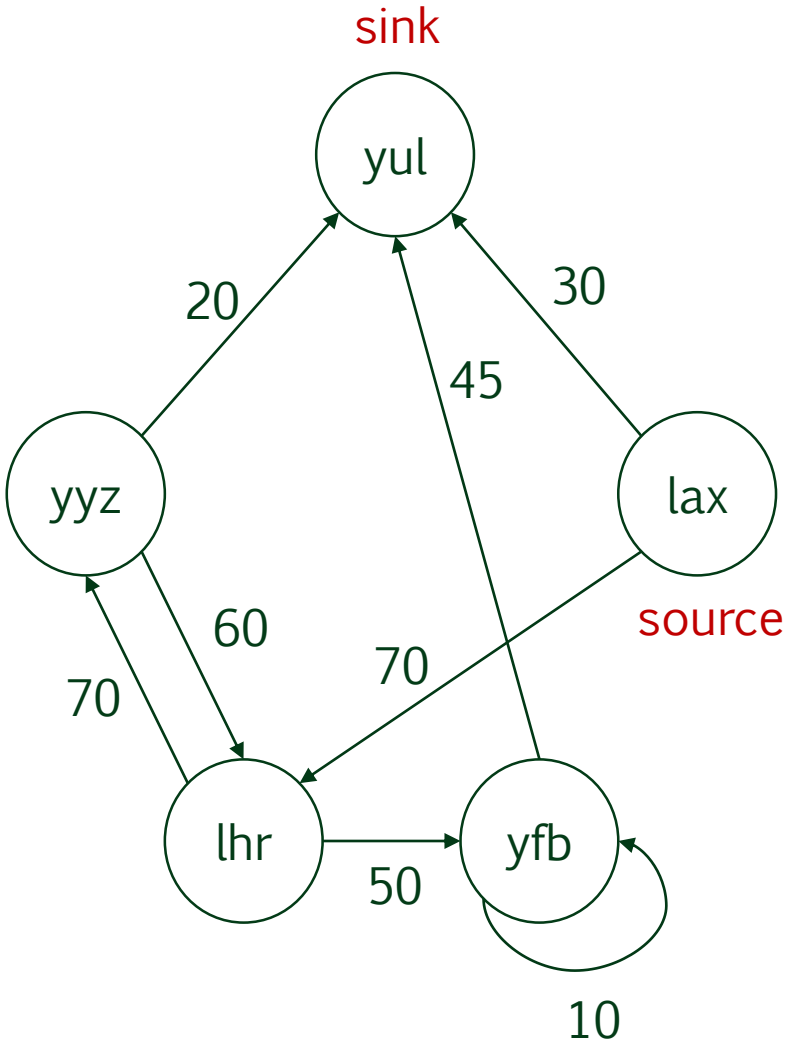
RemoveEdge(“lax”, “yyz”)

- NumVertices 5
- MaxNumVertices 6

	V
0	yul
1	yyz
2	lax
3	lhr
4	yfb
5	

E

	0	1	2	3	4	5
0	-1	-1	-1	-1	-1	
1	20	-1	-1	60	-1	
2	30	-1	-1	70	-1	
3	-1	70	-1	-1	50	
4	45	-1	-1	-1	10	
5						



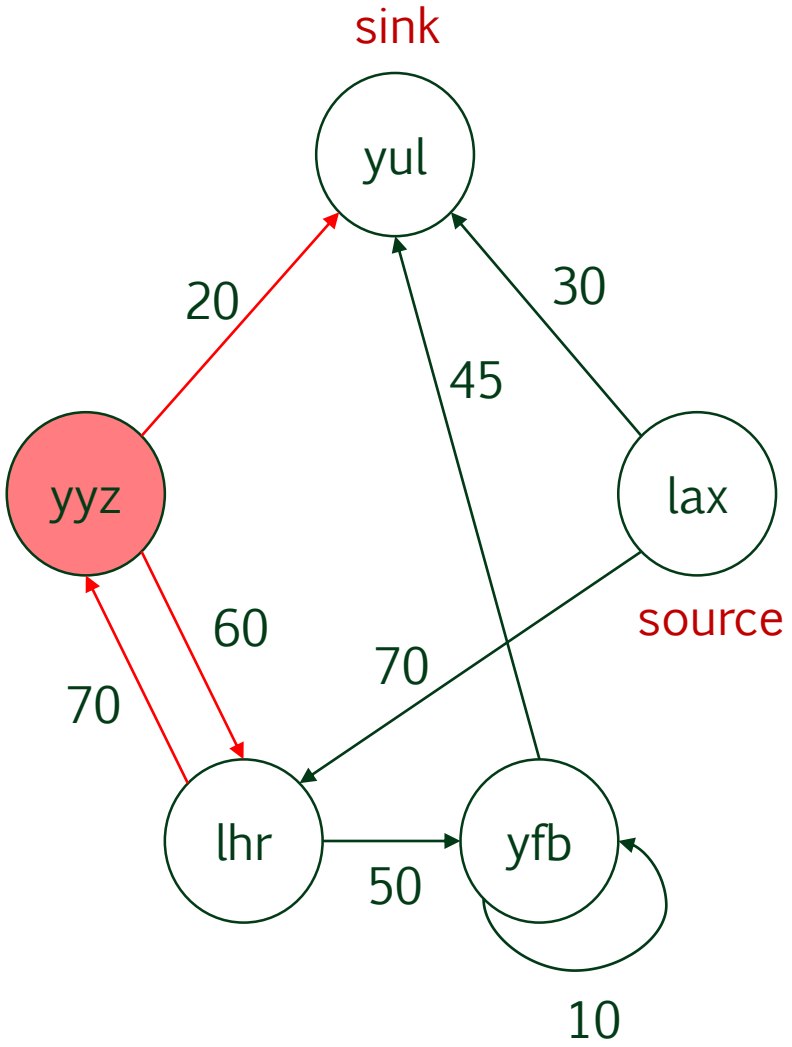
RemoveVertex("yyz")

- NumVertices 5
- MaxNumVertices 6

	V
0	yul
1	yyz
2	lax
3	lhr
4	yfb
5	

	0	1	2	3	4	5
0	-1	-1	-1	-1	-1	
1	20	-1	-1	60	-1	
2	30	-1	-1	70	-1	
3	-1	70	-1	-1	50	
4	45	-1	-1	-1	10	
5						

E



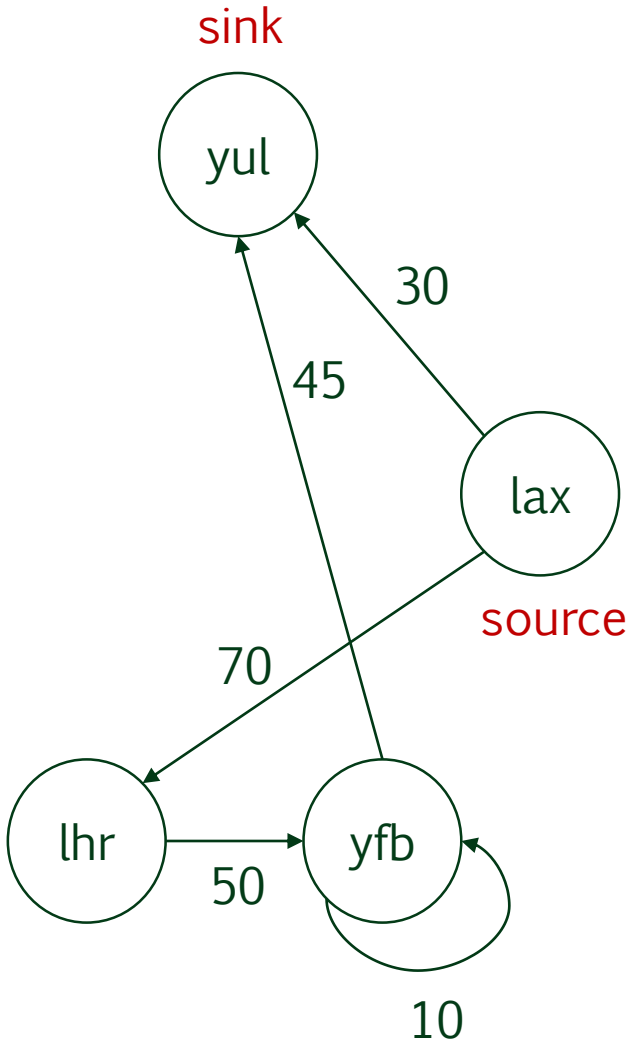
First attempt

- NumVertices 4
- MaxNumVertices 6

	V
0	yul
1	---
2	lax
3	lhr
4	yfb
5	

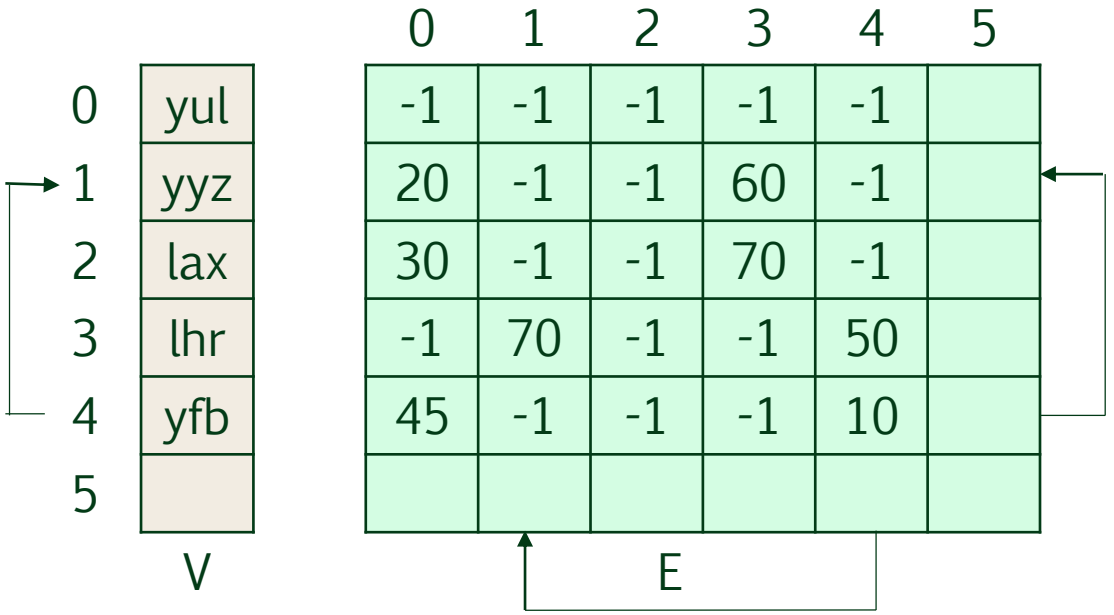
E

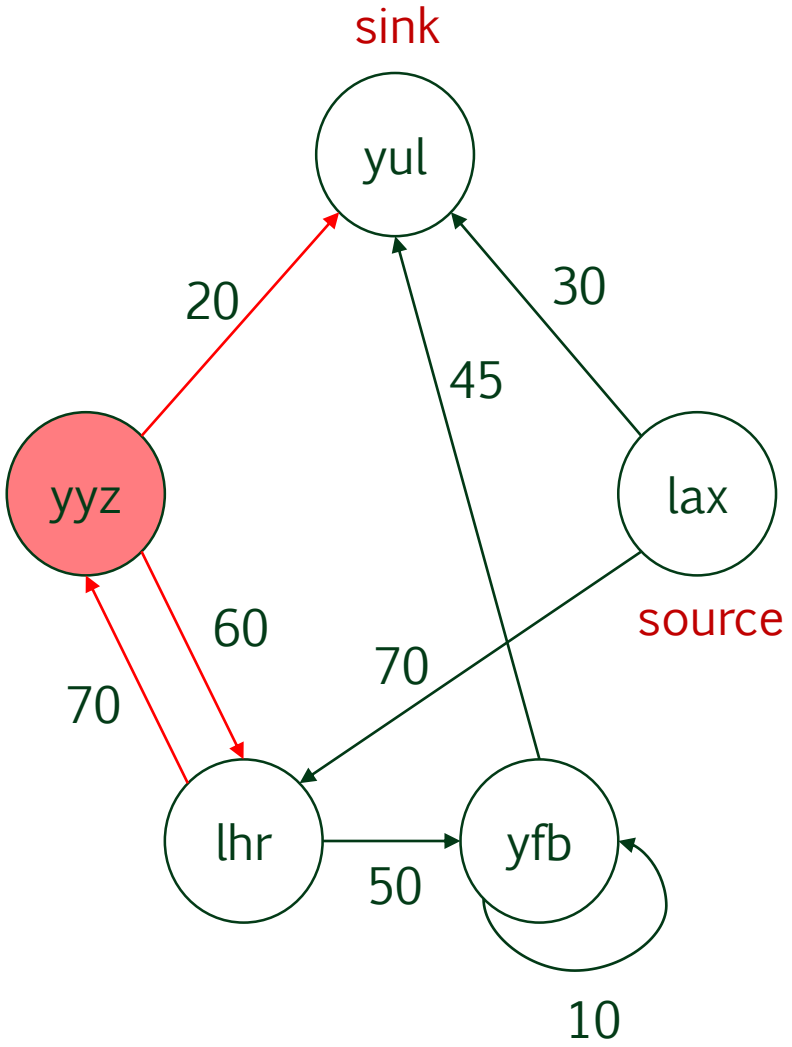
	0	1	2	3	4	5
0	-1	-1	-1	-1	-1	
1	-1	-1	-1	-1	-1	
2	30	-1	-1	70	-1	
3	-1	-1	-1	-1	50	
4	45	-1	-1	-1	10	
5						



Second (better) attempt

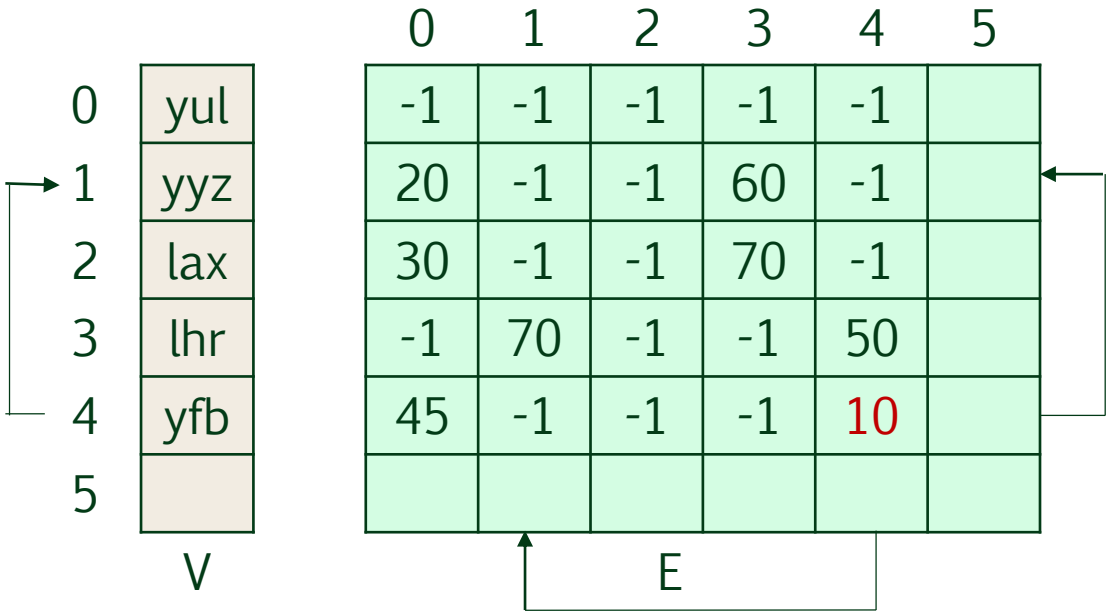
- NumVertices 4
- MaxNumVertices 6

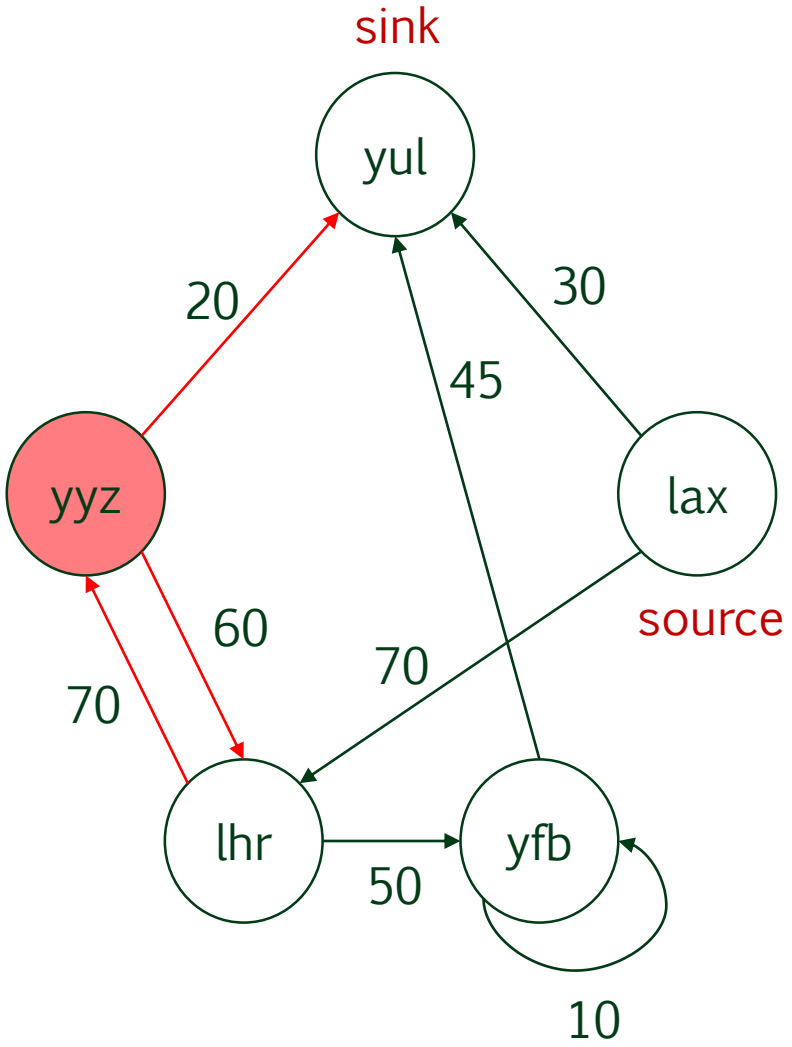




Second (better) attempt

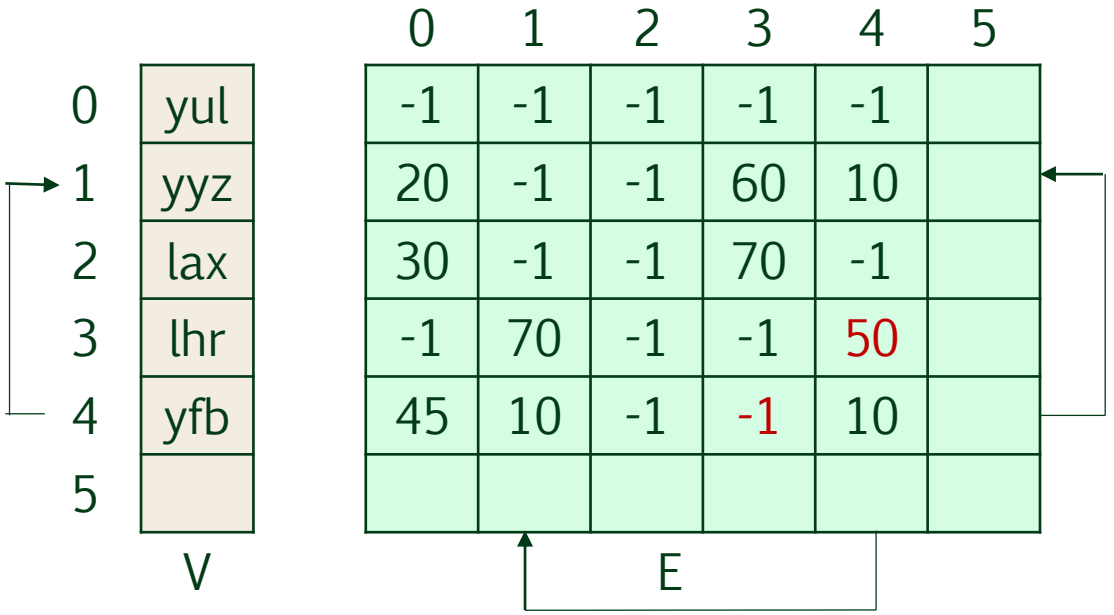
- NumVertices 4
- MaxNumVertices 6

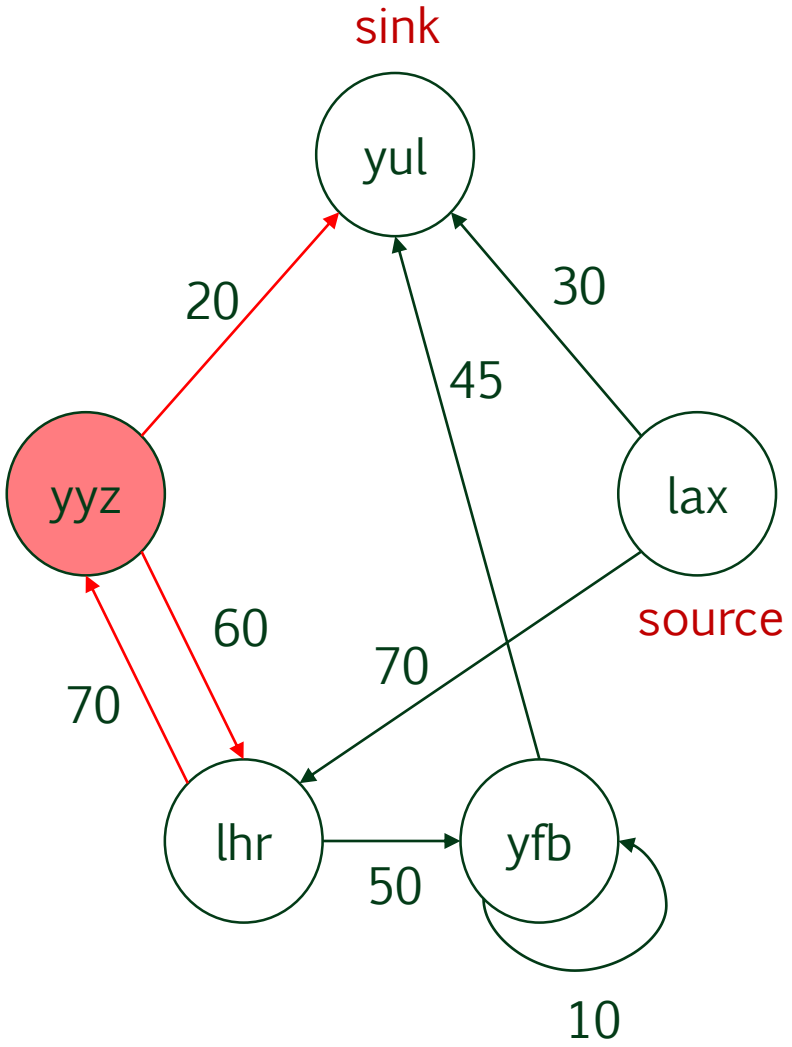




Second (better) attempt

- NumVertices 4
- MaxNumVertices 6





Second (better) attempt

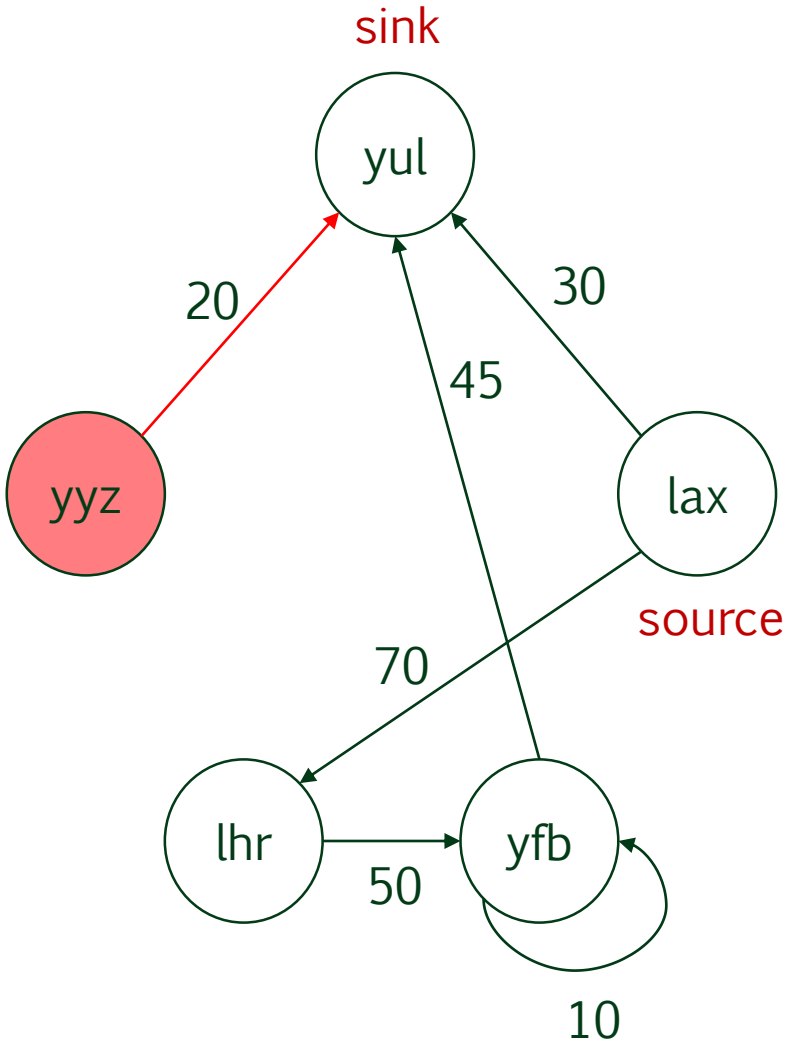
› NumVertices 4

› MaxNumVertices 6

	V
0	yul
1	yyz
2	lax
3	lhr
4	yfb
5	

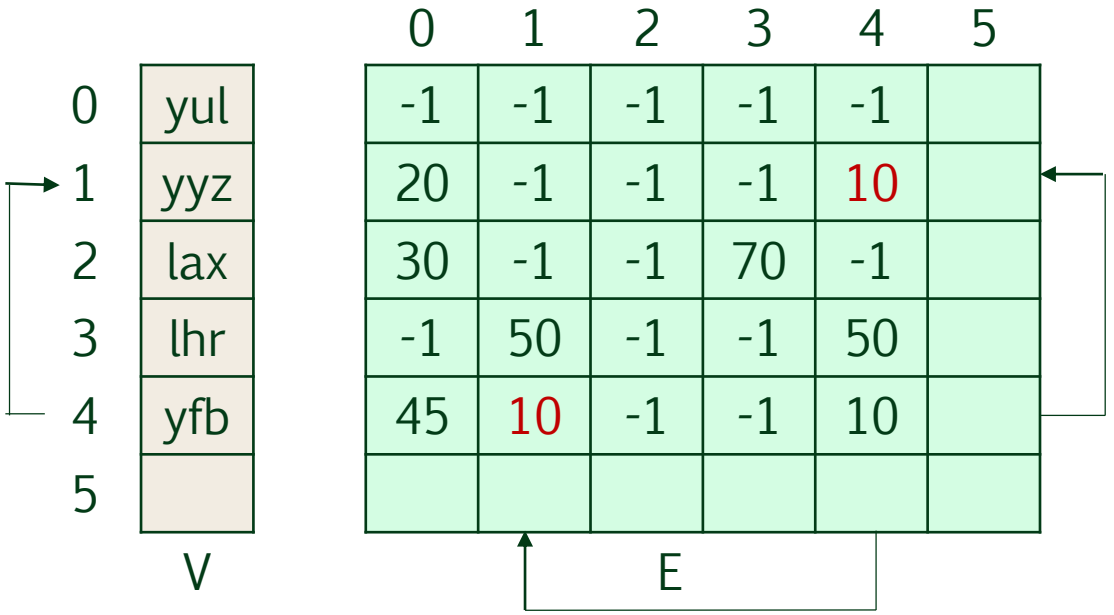
E

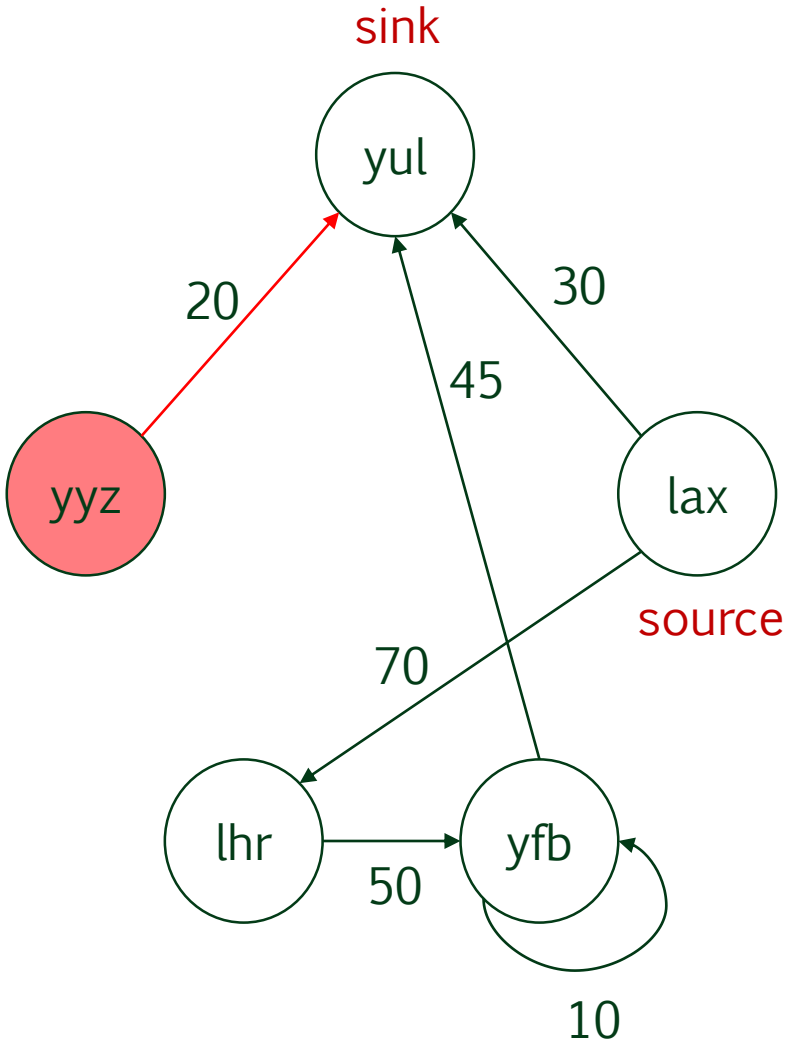
	0	1	2	3	4	5
0	-1	-1	-1	-1	-1	
1	20	-1	-1	-1	10	
2	30	-1	-1	70	-1	
3	-1	50	-1	-1	50	
4	45	10	-1	-1	10	
5						



Second (better) attempt

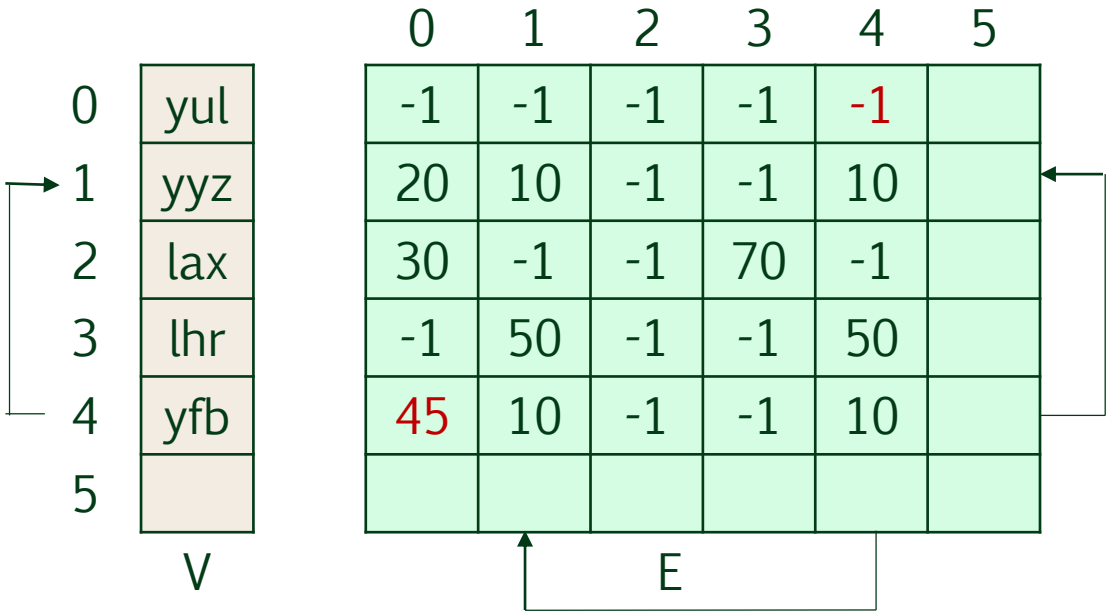
- NumVertices 4
- MaxNumVertices 6

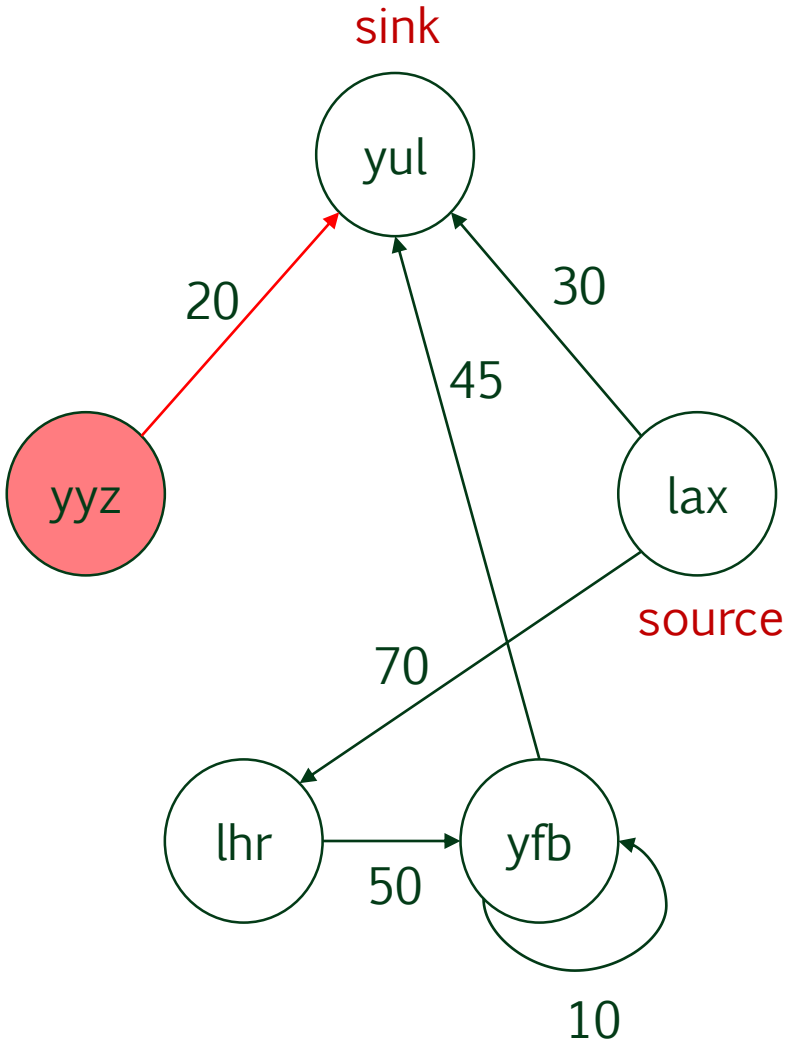




Second (better) attempt

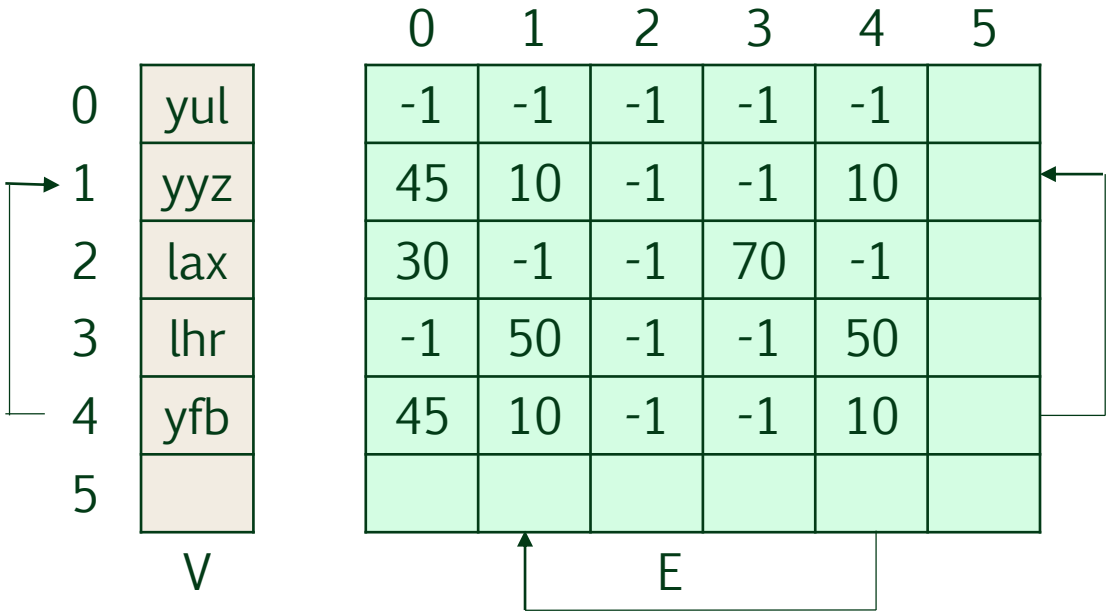
- NumVertices 4
- MaxNumVertices 6

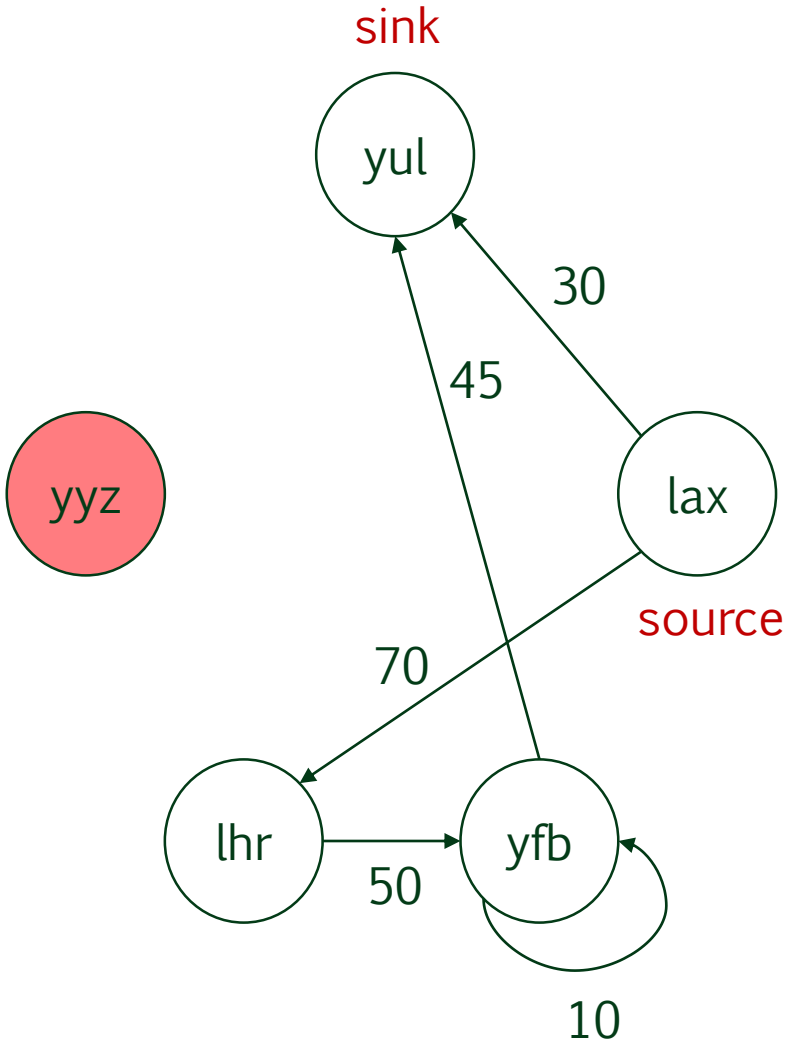




Second (better) attempt

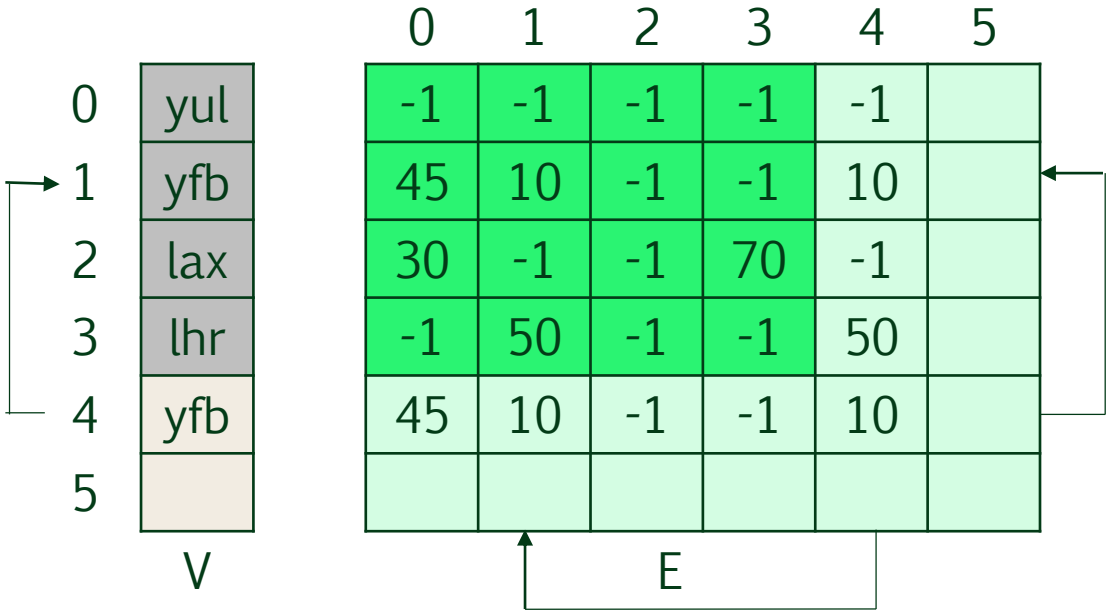
- NumVertices 4
- MaxNumVertices 6

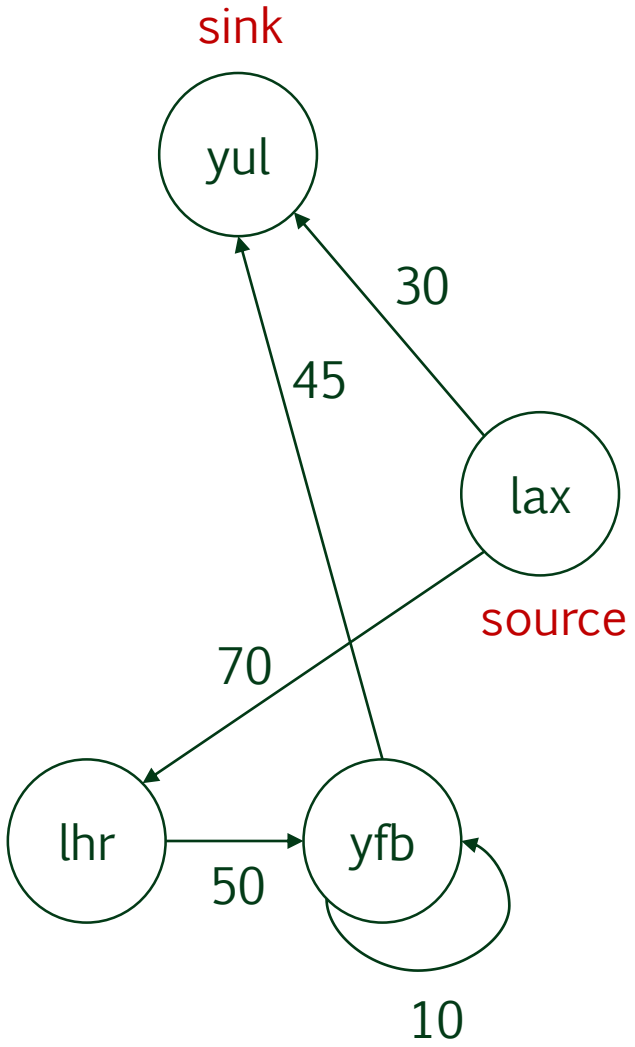




Second (better) attempt

- NumVertices 4
- MaxNumVertices 6





A few additional points

- A supporting method called FindVertex(T name) is needed to map the vertex name to its index. The value -1 is returned if the vertex is not found. The running time of FindVertex is O(n) in the worst case.
- Adjacency matrices can be used to represent undirected and unweighted graphs as well. What changes need to be made?

Exercises

- Let G be weighted, directed graph.
- Write a method that:
  - Returns the number of edges in G
  - Returns true if a given vertex is a sink (or source); false otherwise
  - Returns the in/outdegree of a vertex
  - Reverses the orientation of all edges
  - Returns true if there are no loops; false otherwise
- Show why the loop in RemoveVertex cannot be reversed to:

```
for (j=0; j<=NumVertices; j++) { ... }
```