* VB Data Structures ArrayList
* Queue, stack, and hashtable

GEOG 4/590: GIS Programming

1

Data Structures - Collections

System
- **Array** (index by ordinal values) – not a collection

System.Collections
- **Arraylist**, **Queue**, and **Stack** (indexed by ordinal values, allows insertion and deletion)
- **Hashtable** (indexed by key values, allows insertion and deletion)

System.Collections.Generic
- **List** (indexed by ordinal values, allows insertion and deletion)

2

List(Of T)

- List of a specified type
  `List(Of String)`

3

List Example – Declaration, capacity, count

```
'declaration
Dim dinosaurs As New List(Of String)
'report capacity and count
Console.WriteLine("Dim Capacity: {0}, Count: {1}", _
        dinosaurs.Capacity, dinosaurs.Count)

dinosaurs.Add("Tyrannosaurus")
dinosaurs.Add("Amargasaurus")
dinosaurs.Add("Mamenchisaurus")
dinosaurs.Add("Deinonychus")
dinosaurs.Add("Compsognathus")

Console.WriteLine("Add Capacity: {0}, Count: {1}", _
        dinosaurs.Capacity, dinosaurs.Count)
'trimexcess
dinosaurs.TrimExcess()
Console.WriteLine("TrimExcess Capacity: {0}, Count: {1}", _
        dinosaurs.Capacity, dinosaurs.Count)
'clear
dinosaurs.Clear()
Console.WriteLine("Clear Capacity: {0}, Count: {1}", _
        dinosaurs.Capacity, dinosaurs.Count)
```

4

List Example – Get, search, insert, remove items

```
'get list items
For i As Integer = 0 To dinosaurs.Count - 1
    Console.WriteLine(dinosaurs(i))
Next

'Contains
Console.WriteLine("Contains(""Deinonychus""): {0}", _
    dinosaurs.Contains("Deinonychus"))

'Insert (0-based index)
Console.WriteLine("Insert(2, ""Compsognathus"")")
dinosaurs.Insert(2, "Compsognathus")

For Each dinosaur As String In dinosaurs
    Console.WriteLine(dinosaur)
Next

'Remove - they die again!
Console.WriteLine(vbLf & "Remove(""Compsognathus"")")
dinosaurs.Remove("Compsognathus")
dinosaurs.RemoveAt(3)
dinosaurs.RemoveRange(0, dinosaurs.Count)
```

5

When to use an arraylist or a list?

- Sophisticated versions of an array
- Easier to use than an array
  - The size of an Array is fixed (you need to resize it), whereas the capacity of an ArrayList or a List(Of T) is automatically expanded.
  - ArrayList and List(Of T) provide methods that add, insert, or remove a range of elements.

6

When to use an array?

- You can set the lower bound of an Array, but the lower bound of an ArrayList or a List(Of T) is always zero.
- An Array can have multiple dimensions, but an ArrayList or a List(Of T) always has exactly one dimension.
- An Array of a specific type provides better performance than an ArrayList.

7

Queue Class

```
Public Sub ListqQueue()
    Dim numbers As New Queue(Of String)
    'add items to queue
    numbers.Enqueue("one")
    numbers.Enqueue("two")
    numbers.Enqueue("three")

    ' A queue can be enumerated without disturbing its contents.
    For Each number As String In numbers
        Console.WriteLine(number)
    Next

    'Dequeuing... read the itme first inline in the queue and remove it
    Console.WriteLine("Dequeuing '{0}'", numbers.Dequeue())
    'Peek the queue to see what's coming up next, without removing it
    Console.WriteLine("Peek next item to dequeue: '{0}'", _
        numbers.Peek())
    'Dequeuing... once more
    Console.WriteLine("Dequeuing '{0}'", numbers.Dequeue())

    ' Create a copy of the queue, using the ToArray method
    Dim queueCopy As New Queue(Of String)(numbers.ToArray())
    'Now you can dequeue from the copy
    Console.WriteLine("Dequeuing '{0}'", queueCopy.Dequeue())
    'Done with the copy
    queueCopy.Clear()
    Console.WriteLine("queueCopy.Count = {0}", queueCopy.Count)

    'trouble maker
    queueCopy.Dequeue()
End Sub
```

8

Stack Class

```
Public Sub ListStack()
    Dim myStack As New Stack()
    ' add items to Stack.
    'add items to queue
    myStack.Push("one")
    myStack.Push("two")
    myStack.Push("three")

    ' A stack can be enumerated without disturbing its contents.
    For Each number As String In myStack
        Console.WriteLine(number)
    Next

    'Popping... read the itme first inline in the stack and remove it
    Console.WriteLine("Popping '{0}'", myStack.Pop())
    'Peek the queue to see what's coming up next, without removing it
    Console.WriteLine("Peek next item to pop: '{0}'", _
        myStack.Peek())
    'Popping... once more
    Console.WriteLine("Popping '{0}'", myStack.Pop())

    'Done with the stack
    myStack.Clear()
    Console.WriteLine("mystack.Count = {0}", myStack.Count)
End Sub
```

9

HashTable Class

- Represents a collection of key/value pairs that are organized based on the hash code of the key
- Allows users to look up values using keys
- The relationship between key and value could be:
  - 1 to 1
  - Many to 1 (many different keys point to the same value)

10

HashTable Example – Declaration, add, remove

```
' Create a new hash table.
Dim openWith As New Hashtable()

' Add some elements to the hash table. There are no
' duplicate keys, but some of the values are duplicates.
openWith.Add("txt", "notepad.exe")
openWith.Add("bmp", "paint.exe")
openWith.Add("dib", "paint.exe")
openWith.Add("rtf", "wordpad.exe")

' The Add method throws an exception if the new key is
' already in the hash table.
Try
    openWith.Add("txt", "winword.exe")
Catch
    Console.WriteLine("An element with Key = ""txt"" already exists.")
End Try

' Use the Remove method to remove a key/value pair.
Console.WriteLine(vbLf + "Remove(""bmp"")")
openWith.Remove("bmp")

If Not openWith.ContainsKey("bmp") Then
    Console.WriteLine("Key ""bmp"" is not found.")
End If
```

11

HashTable Example – Get values from HashTable

```
' The Item property is the default property, so you
' can omit its name when accessing elements.
Console.WriteLine("For key = ""rtf"", value = {0}.", _
    openWith("rtf"))

' The default Item property can be used to change the value
' associated with a key.
openWith("rtf") = "winword.exe"
Console.WriteLine("For key = ""rtf"", value = {0}.", _
    openWith("rtf"))

' If a key does not exist, setting the default Item property
' for that key adds a new key/value pair.
openWith("doc") = "winword.exe"
```

12

HashTable Example – ContainsKey and DictionaryEntry

```
' ContainsKey can be used to test keys before inserting
' them.
If Not openWith.ContainsKey("ht") Then
    openWith.Add("ht", "hypertrm.exe")
    Console.WriteLine("Value added for key = ""ht"": {0}", _
        openWith("ht"))
End If

' When you use foreach to enumerate hash table elements,
' the elements are retrieved as KeyValuePair objects.
Console.WriteLine()
For Each de As DictionaryEntry In openWith
    Console.WriteLine("Key = {0}, Value = {1}", _
        de.Key, de.Value)
Next de
```

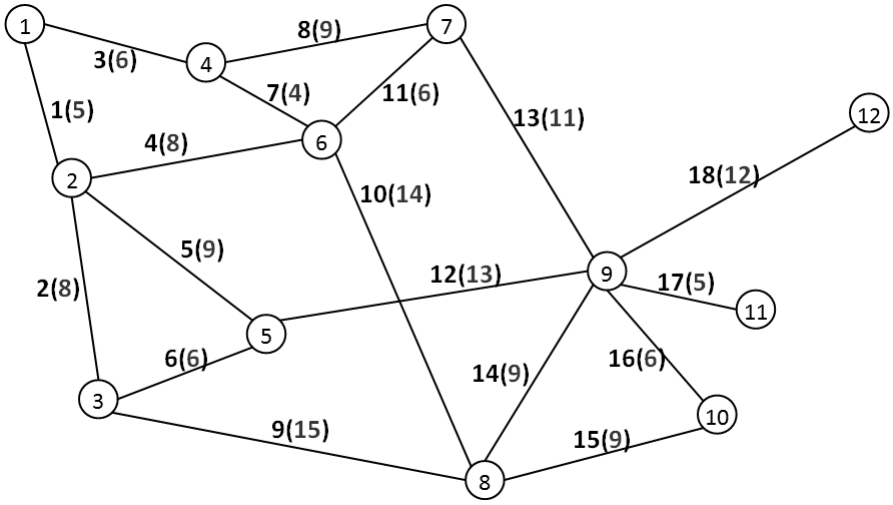
13

HashTable Example – Keys and Values

```
' To get the values alone, use the Values property.
Dim valueColl As ICollection = openWith.Values

' The elements of the ValueCollection are strongly typed
' with the type that was specified for hash table values.
Console.WriteLine()
For Each s As String In valueColl
    Console.WriteLine("Value = {0}", s)
Next s

' To get the keys alone, use the Keys property.
Dim keyColl As ICollection = openWith.Keys

' The elements of the KeyCollection are strongly typed
' with the type that was specified for hash table keys.
Console.WriteLine()
For Each s As String In keyColl
    Console.WriteLine("Key = {0}", s)
Next s
```

14

Lab 3. Dijkstra Shortest Path



15

Network Elements (defined outside class)

- Edge

```
Public Structure Edge
    Public ID As Integer
    Public FromNode As Integer
    Public ToNode As Integer
    Public Length As Double
End Structure
```

- Neighbor

```
Public Structure Neighbor
    Public NeighborID As Integer
    Public Distance As Double
End Structure
```

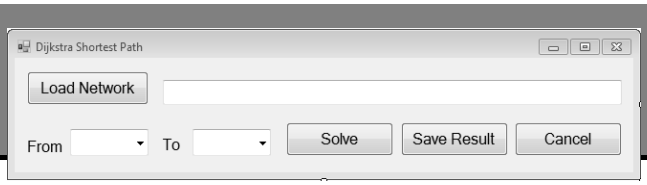
16

Data Structures for Lab 3

- Network
  - networkEdges (bi-directional) – List (Of Edge) (Class level)
  - networkNodes – array (Of Integer) (Class level)
- Solution
  - DistanceTab – HashTable (Class level)
    - Key: Node ID - Integer
    - Value: Distance from origin node - Double
  - RouteTab – HashTable (Class level)
    - Key: toNode - Integer
    - Value: fromNode – Integer
      (from each toNode, you know where it's from)
- Cursor/tracker
  - unvisitedNodes – List (Of Integer) (procedure level)
  - Neighbors – List (Of Neighbor) (procedure level)

17

GUI



- Load network
  - Read network from file
  - Set the values of the edge and node lists
  - Add nodes to the from/to comboboxes
- Textbox – display the file name
- From/to comboboxes – let users set the from and to nodes of the network
- Solve button – solve the shortest path problem
- Save Result button
- Cancel button - `Me.Close()`

18

Input network file

- Use notepad.exe to create a file containing the edge information. It looks something like:

```
1,1,2,5
2,2,3,8
3,1,4,6
4,2,6,8
…
```

- Use the code on next slide to set these values to the networkEdge list. Also on the slide is the code to extra nodes from edges.

19

Prepare the network Edges and Nodes lists

```
'do some erro handling here to make sure the data are correctly read
Dim fs As New StreamReader(networkFile, FileMode.Open)
Dim line As String
Dim stringArr As String()
Dim anEdge As New Edge

line = fs.ReadLine
Do While line IsNot Nothing
    stringArr = line.Split(",")
    anEdge.ID = CInt(stringArr(0))
    anEdge.FromNode = CInt(stringArr(1))
    anEdge.ToNode = CInt(stringArr(2))
    anEdge.Length = CDbl(stringArr(3))
    networkEdges.Add(anEdge)

    'implement undirectioned network
    anEdge.FromNode = CInt(stringArr(2))
    anEdge.ToNode = CInt(stringArr(1))
    networkEdges.Add(anEdge)
    line = fs.ReadLine
Loop
fs.Close()
```

```
For Each edge As Edge In networkEdges
    If Not networkNodes.IndexOf(edge.FromNode) >= 0 Then networkNodes.Add(edge.FromNode)
    If Not networkNodes.IndexOf(edge.ToNode) >= 0 Then networkNodes.Add(edge.ToNode)
Next
```

20

You might need to write something like this to make sure your network is correctly defined

```
Private Sub DumpEdgeRecords()
    For Each edge As Edge In networkEdges
        Console.WriteLine("Edge {0} from node {1} to node {2}, length {3}", _
             [edge].ID, [edge].FromNode, [edge].ToNode, [edge].Length)
    Next
End Sub
```

- Also, you can use the Sort method to sort your node list so that users can easily find a node on the comboboxes.

21

Add list of nodes to the ComboBoxes

- Items property – an object representing the **collection** of the items contained in the combobox.

```
ComoBox.Items.Add(item, key)
ComoBox.Items.Remove(index or key) – 0-based
ComoBox.Items.Clear()
ComoBox.Items(index)
```

- SelectedItem

```
ComoBox.SelectedItem = 2
Dim s As String = ComoBox.SelectedItem.ToString)
```

22

Pseudo-code of the Solve button

- Initialize distance and route HashTables
- While there are unvisited nodes, do:
  - Find the node with the smallest distance value
  - Remove the node from the unvisited list
  - Find the neighboring unvisited nodes
  - For each neighbor:
    - Assign distance to it if the new distance is smaller than its original value, meanwhile, update the route if an assignment occurs
  - Repeat until all nodes are visited

23

Essential Subroutines – InitSolver

```
Private Sub InitSolver(ByVal startNode As Integer)
    'clear the hashtables
    DistanceTab.Clear()
    RouteTab.Clear()

    For Each i As Integer In networkNodes
        DistanceTab.Add(i, Double.MaxValue)
        RouteTab.Add(i, Nothing)
    Next

    DistanceTab(startNode) = 0
End Sub
```

24

Essential Subroutines – GetMinNode

```
'find the next unvisited node that has the shortest distance
Private Function GetMinNode(ByRef unvisitedList As List(Of Integer)) As Integer
    Dim minDistance As Double = Double.MaxValue
    Dim minNode As Integer = -1

    For Each i As Integer In unvisitedList
        If DistanceTab(i) < minDistance Then
            minDistance = DistanceTab(i)
            minNode = i
        End If
    Next

    Return minNode
End Function
```

25

Essential Subroutines – GetNeighbors

```
Private Function GetNeighbors(ByVal fromNode As Integer, _
    ByVal unvisitedList As List(Of Integer)) As List(Of Neighbor)
    Dim neighbors As New List(Of Neighbor)
    Dim aNeighbor As New Neighbor

    For Each netEdge As Edge In networkEdges
        If netEdge.FromNode = fromNode And _
        (unvisitedList.BinarySearch(netEdge.ToNode) >= 0) Then
            aNeighbor.NeighborID = netEdge.ToNode
            aNeighbor.Distance = netEdge.Length
            neighbors.Add(aNeighbor)

            'Console.WriteLine("neighbor node: {0}, distance: {1}", _
            '        aNeighbor.NeighborID, aNeighbor.Distance)
        End If
    Next

    Return neighbors
End Function
```

26

Essential Subroutines – AssignDistance

```
Private Sub AssignDistance(ByVal fromNode As Integer, _
    ByVal toNode As Integer, ByVal length As Double)

    Dim newDistance As Double = DistanceTab(fromNode) + length
    If DistanceTab(toNode) > newDistance Then
        DistanceTab(toNode) = newDistance
        RouteTab(toNode) = fromNode
    End If
End Sub
```

27

Now, putting them together… (under the Solve button click procedure)

- Get the startNode and endNode values from the comboboxes (see the combobox slide) and then… see the pseudo code to complete the procedure.

```
'create a copy of the networkNodes
Dim unvisitedNodes As List(Of Integer) = networkNodes.ToList()
Dim currentNode As Integer
Dim nodeNeighbors As List(Of Neighbor)

'initialize the nodeQueus and route
InitSolver(startNode)

While unvisitedNodes.Count > 0
    currentNode = GetMinNode(unvisitedNodes)
    unvisitedNodes.Remove(currentNode)
```

... not finished yet! You need to finish it.

28

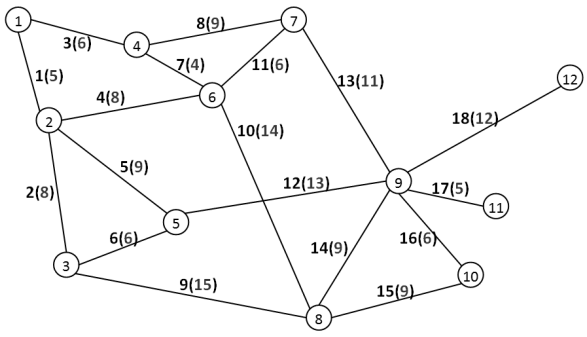
Where is the result?

- When the calculation is done, you can use a MsgBox to show the result.
- The DistanceTab HashTable contains the shortest distance values from the startNode to all the other nodes on the network. You can use the endNode as a key to get the value out of DistanceTab.
- The RouteTab contains the shortest routes (i.e., a list of node IDs) between the startNode and other nodes in reversed order. You have to back track the table to find the shortest route from startNode to endNode.

29

Here is what the RouteTab looks like… when the startNode is 1.

Key (toNode)	Value (fromNode)
12	9
11	9
10	9
9	7
8	6
7	4
6	4
5	2
4	1
3	2
2	1
1	Nothing



30

How to back track the RouteTab?

```
Private Function BackTrackRoute(ByVal startNode As Integer, _
                                ByVal endNode As Integer) As String
    Dim routeString As String = ""
    Dim tempString As String = ""
    Dim stringStack As New Stack()
    Dim currentNode As Integer = endNode
    Dim previousNode As Integer

    Do
        previousNode = currentNode
        currentNode = RouteTab(currentNode)
        tempString = currentNode & " to " & previousNode & vbCrLf
        stringStack.Push(tempString)
    Loop While currentNode <> startNode

    Do While stringStack.Count > 0
        routeString = routeString & stringStack.Pop()
    Loop

    Return routeString
End Function
```

31

Questions

1. How does the sample code implement an un-directed network (i.e., with bi-directional edges) with the networkEdges data structure? How does it relate to the implementation of the GetNeighbors function and the format of the input network file? (Hint: what would you do differently if there are one-way edges on the network?)

2. Complete the table below.

startNode	endNode	Length	Route (list the nodes)
1	12		
3	7		
10	1		
4	5		

32