Graph Traversal Algorithms

- What you know:
  - How to traverse a graph
- What you will learn:
  - How to traverse a graph in an optimal fashion

Categories Of Shortest Path Algorithms

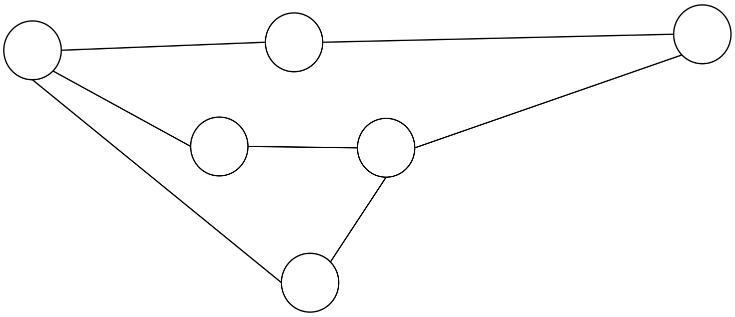
- Unweighted shortest paths
- Weighted shortest paths

Unweighted Graphs

	A	B	C	D	E	F	G	H	I
A		1		1	1				
B					1				
C		1							
D							1		
E						1		1	
F			1					1	
G								1	
H									1
I						1			

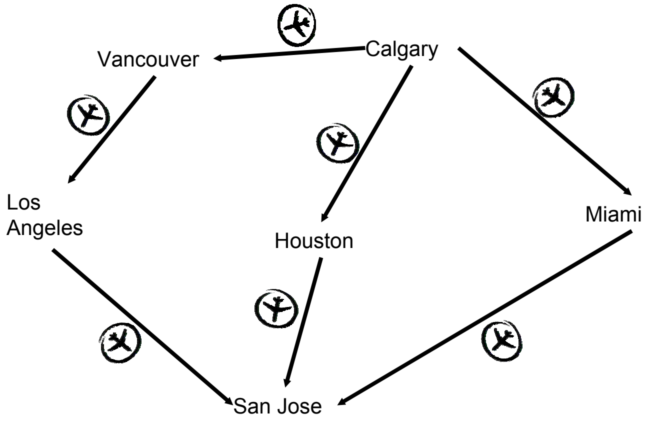
Unweighted Shortest Paths

- There is an equal cost incurred from traveling from one node to another (traversing an arc)
- To find the shortest path, traverse the series of nodes which requires the fewest arcs to be traversed.

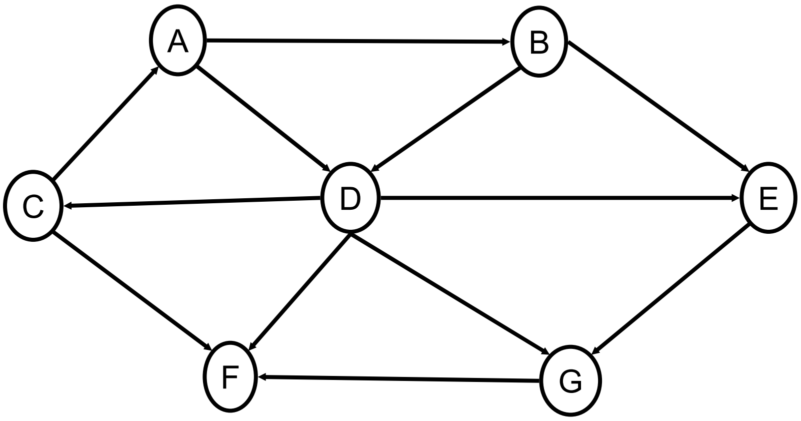


An Application Of Finding The Shortest Path

- Airplane flights: Minimizing the number of cities to pass through (stop-overs)



Example Of An Unweighted Shortest Path Traversal: A -> F



Definition Of The Node Class

```
public class Node
{
    private boolean visited;
    private int pathLength;
    private Node predecessor;
    private List neighbors;
       :        :          :
}
```

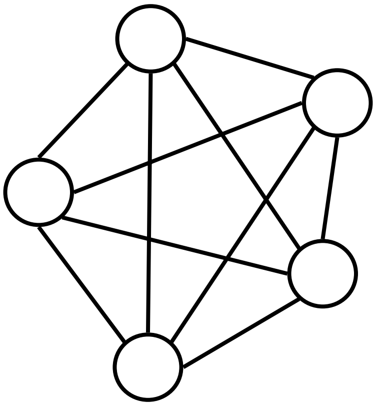
Unweighted Shortest Path Traversal: Algorithm

```
unweightedTraversal (Node start, Node end)
{
    boolean done = false;
    Queue nodeQueue = new Queue ();
    Node temp;
    Node nextNeighbor;
    start.setVisited (true);
    start.setPathLength (0);
    nodeQueue.enqueue (start);
```

Unweighted Shortest Path Traversal: Algorithm (2)

```
    while ((done == false) && (nodeQueue.isEmpty() == false))
    {
        temp = nodeQueue.dequeue();
        while ((done == false) && (temp.hasUnvisitedNeighbor())
        {
                nextNeighbor = temp.getNextUnvistedNeighbor();
                nextNeighbor.setVisited(true);
                nextNeighbor.setPathLength (temp.getPathLength() + 1);
                nextNeighbor.setPredecessor (temp);
                nodeQueue.enqueue (nextNeighbor);
                if (end.equals (nextNeighbor) == true)
                    done = true;
        }
    }
}
```

The Worse Possible Case For Determining The Shortest Path?



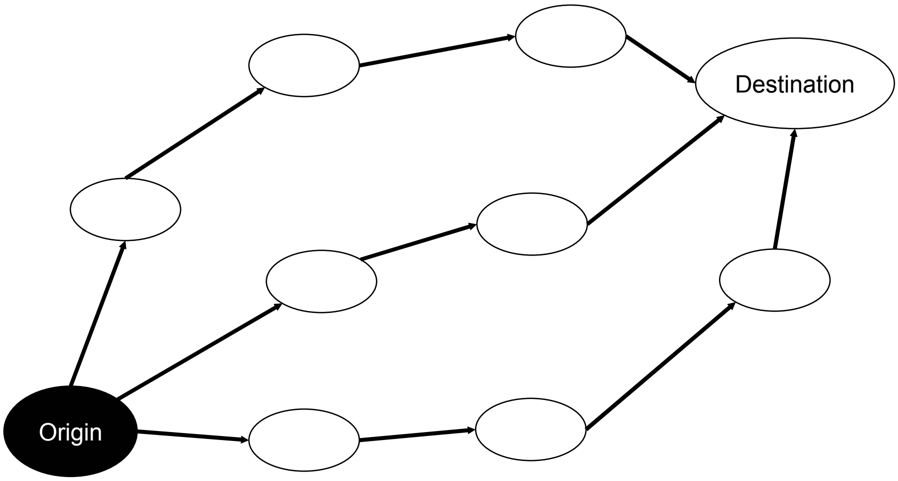
Efficiency Of The Shortest Path Algorithm: No Path Weights?

- *O* (n^2)

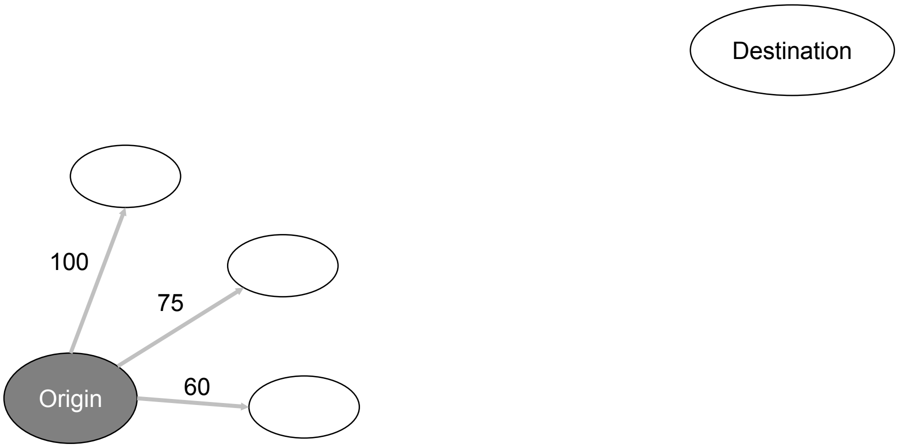
```
while ((done == false) && (nodeQueue.isEmpty() == false))
{
            :        :        :        :        :

    while ((done == false) && (temp.hasUnvisitedNeighbor())
    {
            :        :        :        :        :
    }
}
```

Finding The Shortest Weighted Path

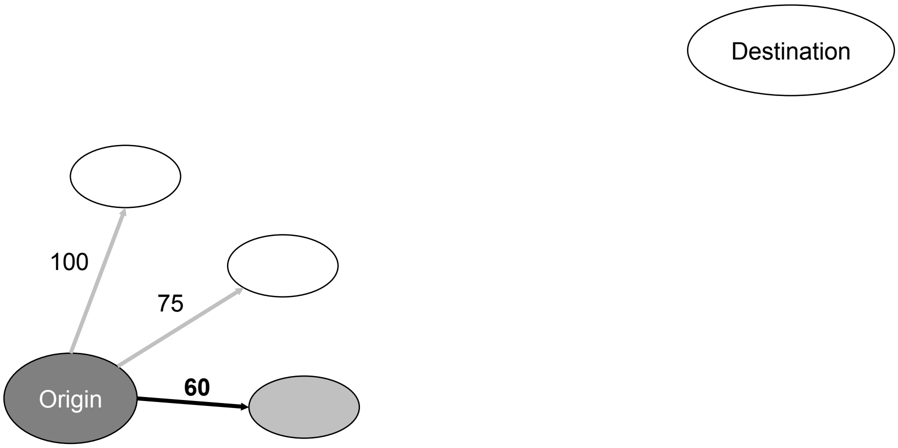


Finding The Shortest Weighted Path

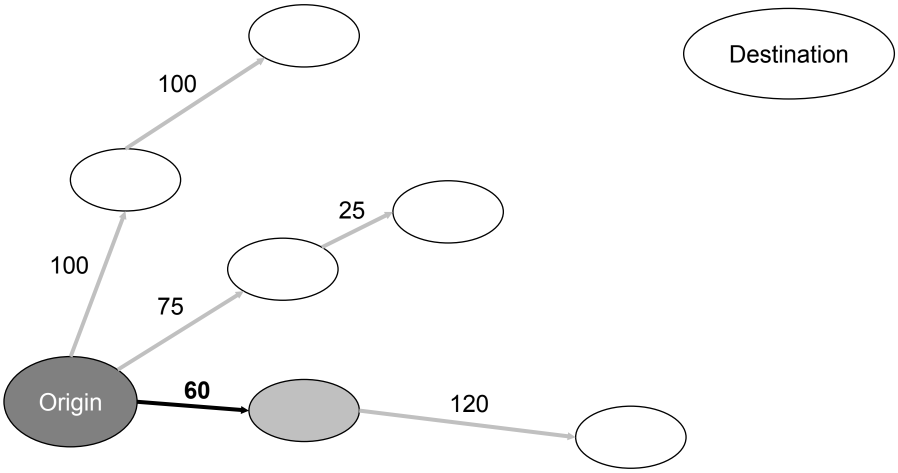


Finding The Shortest Weighted Path

Pick the choice that appears to be the best choice at a particular stage

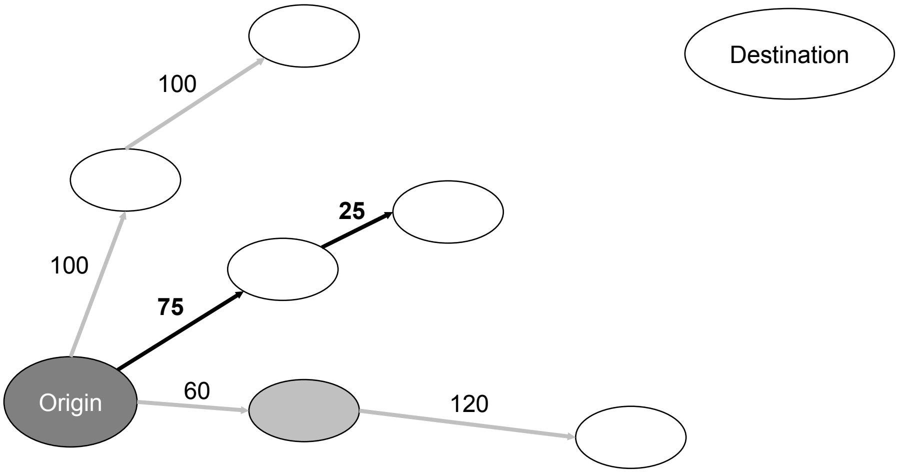


Finding The Shortest Weighted Path

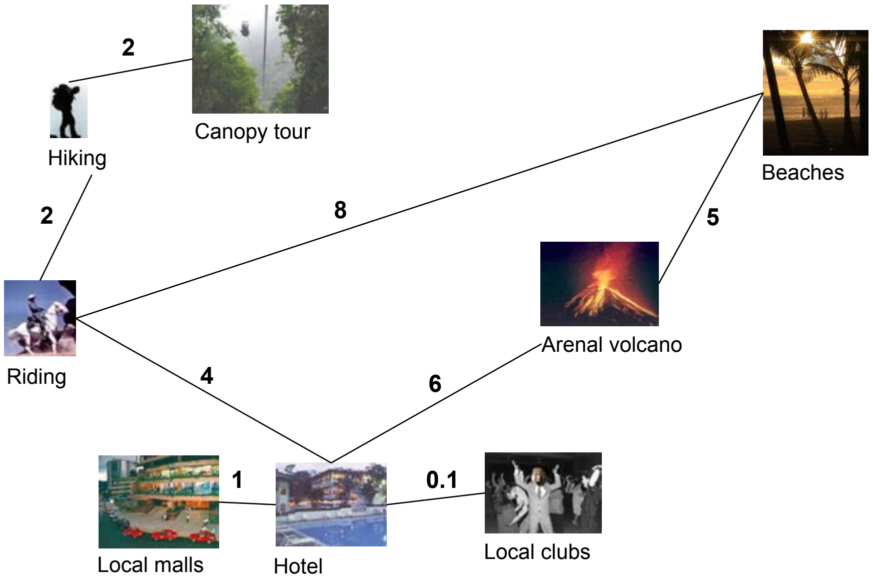


Finding The Shortest Weighted Path

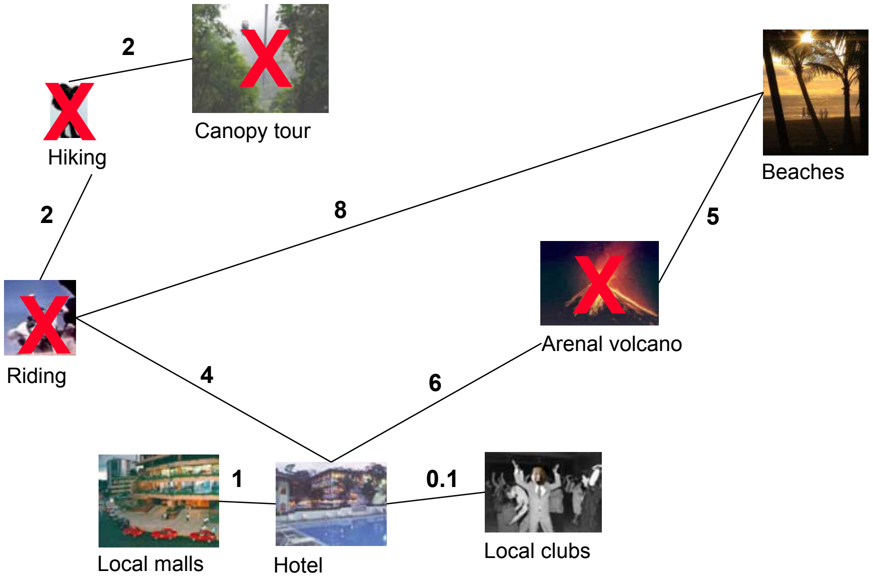
Based on the information received at the next stage: pick the choice that is currently the best option



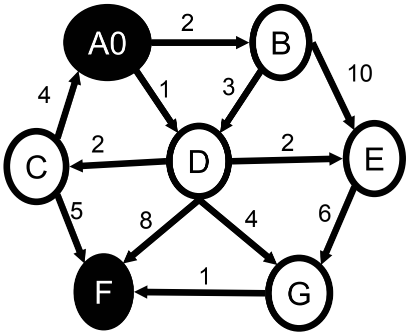
An Application Of Finding The Shortest Weighted Path



An Application Of Finding The Shortest Weighted Path

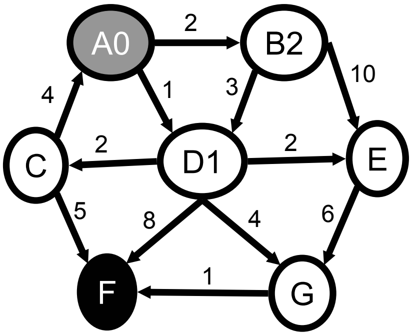


An Example Of Finding The Shortest Weighted Path: Start



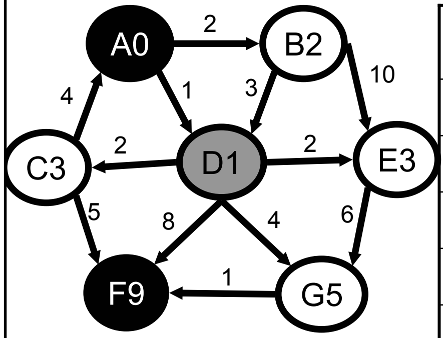
Node	Known	Distance	Predecessor
A	F	0	0
B	F	∞	0
C	F	∞	0
D	F	∞	0
E	F	∞	0
F	F	∞	0
G	F	∞	0

An Example Of Finding The Shortest Weighted Path: Declare “A” As Known



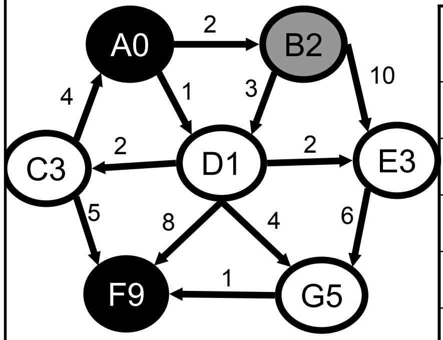
Node	Known	Distance (from A)	Predecessor
A	*T*	0	0
B	F	*2*	*A*
C	F	∞	0
D	F	*1*	*A*
E	F	∞	0
F	F	∞	0
G	F	∞	0

An Example Of Finding The Shortest Weighted Path: Declare D As Known



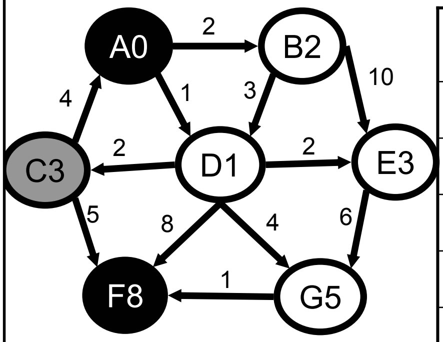
Node	Known	Distance (from A)	Predecessor
A	T	0	0
B	F	2	A
C	F	*3*	*D*
D	*T*	1	A
E	F	*3*	*D*
F	F	*9*	*D*
G	F	*5*	*D*

An Example Of Finding The Shortest Weighted Path: Declare B As Known



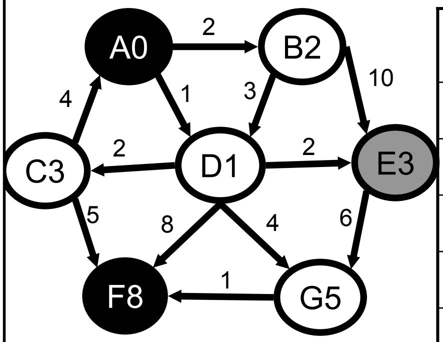
Node	Known	Distance (from A)	Predecessor
A	T	0	0
B	*T*	2	A
C	F	3	D
D	T	1	A
E	F	3	D
F	F	9	D
G	F	5	D

An Example Of Finding The Shortest Weighted Path: Declare C As Known



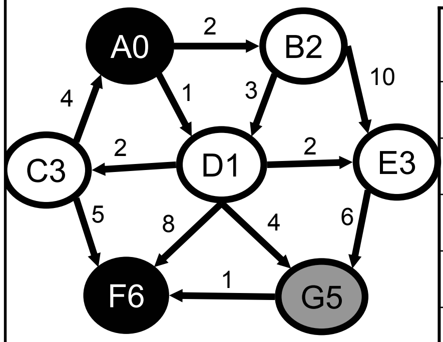
Node	Known	Distance (from A)	Predecessor
A	T	0	0
B	T	2	A
C	*T*	3	D
D	T	1	A
E	F	3	D
F	F	*8*	*C*
G	F	5	D

An Example Of Finding The Shortest Weighted Path: Declare E As Known



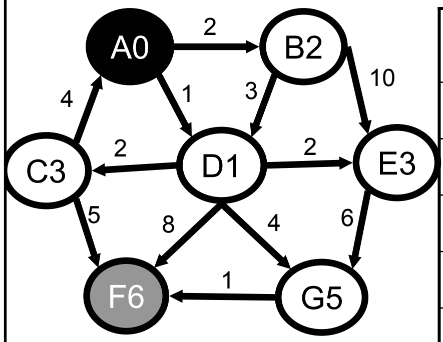
Node	Known	Distance (from A)	Predecessor
A	T	0	0
B	T	2	A
C	T	3	D
D	T	1	A
E	*T*	3	D
F	F	8	C
G	F	5	D

An Example Of Finding The Shortest Weighted Path: Declare G As Known



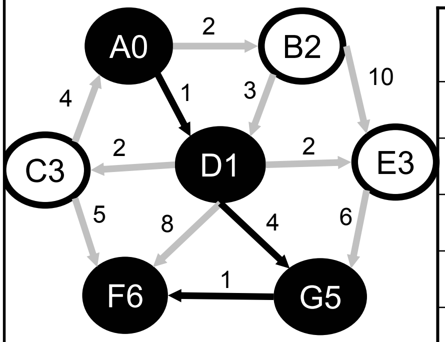
Node	Known	Distance (from A)	Predecessor
A	T	0	0
B	T	2	A
C	T	3	D
D	T	1	A
E	T	3	D
F	F	*6*	*G*
G	*T*	5	D

An Example Of Finding The Shortest Weighted Path: Declare F As Known



Node	Known	Distance (from A)	Predecessor
A	T	0	0
B	T	2	A
C	T	3	D
D	T	1	A
E	T	3	D
F	*T*	6	G
G	T	5	D

An Example Of Finding The Shortest Weighted Path: Algorithm Ends



Node	Known	Distance (from A)	Predecessor
A	T	0	0
B	T	2	A
C	T	3	D
D	T	1	A
E	T	3	D
F	T	6	G
G	T	5	D

Initializing The Starting Values

```
public void initializeTable (Table t, Graph g,
                             Node start)
{
   for (int i = 1; i < NO_NODES; i++)
   {
      t[i].setNode (g.getNextNode())
      t[i].distance (∞[1]);
      t[i].setPredecessor (null);
   }
   t [index of starting node].distance = 0;
}
```

Node	Distance	Predecessor
A	0	0
B	∞	0
C	∞	0
D	∞	0
E	∞	0
F	∞	0
G	∞	0

[1] Just pick a distance value that will be greater than any existing distance in the graph

Algorithm For Finding The Shortest Path

E.W. Dijkstra

```
public void dijkstraShortestPath (Graph g, Table t, Node start)
{
    List toBeChecked = new List ();
    Node temp;
    for (all nodes in the graph)
        toBeChecked.add (graph.traverse());
```

Algorithm For Finding The Shortest Path(2)

E.W. Dijkstra

```
while (toBeChecked.empty() == false)
{
       temp = toBeChecked.removeMinDistanceNode ();
       for (all nodes suc adjacent to temp which are contained in toBeChecked)
       {
             if (t [suc].getDistance () >
               (t [temp].getDistance () + distance from suc to temp))
             {
                 t [suc].setDistance (t[temp].getDistance () +
                                          distance from suc to temp));
                 t [suc].setPredecessor (temp);
             }
       }
}
```

Efficiency Of The Shortest Path Algorithm: Weighted

- *O* (n^2)

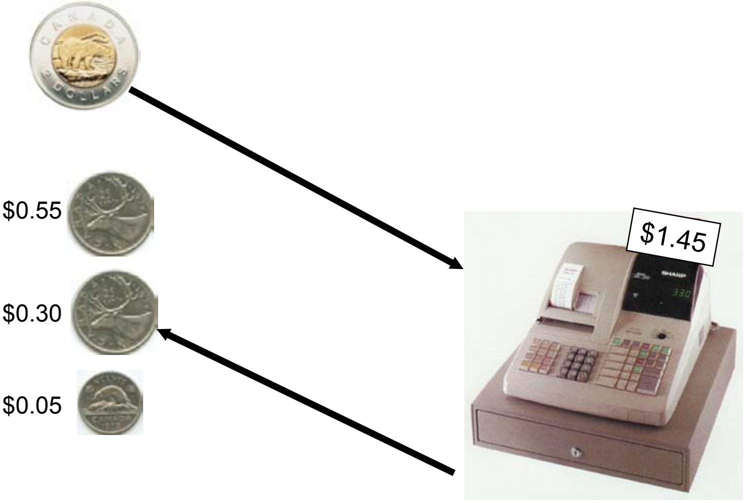
***O* (n)**

```
for (all nodes in the graph)
    toBeChecked.add (graph.traverse());
```

***O* (n^2)**

```
while (toBeChecked.empty() == false)
{
      temp = toBeChecked.removeMinDistanceNode ();
      for (all nodes suc adjacent to temp which are contained in toBeChecked)
      {
             :        :        :        :        :
      }
}
```

A Related Algorithm: Making Change



Ideal case: The number of coins returned is minimized

Similarities Of Both Algorithms

- At each stage the selection made is what appears to be best at that point in time:
  - Shortest path: based on the current paths traversed select the shortest combination of paths.
  - Making change: at each stage in the change making process, give back the largest possible denomination (without having the total amount of change given back exceed the original amount of change owed).

You Should Now Know

- How to determine the shortest path for traversing a graph when:
  - The graph is unweighted
  - The graph is weighted (Dijkstra’s algorithm)
- What is a greedy algorithm and how Dijkstra’s algorithm is an example of a greedy approach.

Sources Of Lecture Material

- *“Data Structures and Algorithm Analysis in C++”* by Mark Allen Weiss
- *“Data Structures and Algorithms with Java”* by Frank M. Carrano and Walter Savitch
- *"Data Structures and Algorithms in Java“* by Adam Drozdek
- CPSC 331 course notes by Marina L. Gavrilova http://pages.cpsc.ucalgary.ca/~marina/331/