Course: CSC503

Instructor: prof. Massimo DiPierro

Student: Svetlin Stalinov

Parallel Solution for Single Source Shortest Path (Dijkstra’s algorithm )

1. **Definition**

Graph is set of vertices and collection of edges that each connects a pair of vertices.

A path in a graph is a sequence of vertices connected by edges. For non-weighted graphs length of the path is the number of edges. For weighted graph the length of the path is the sum of weights on the respective edges.

1. **Graph representations**

The two most widely used data structures for representing graphs are adjacency matrix and array of adjacency lists:

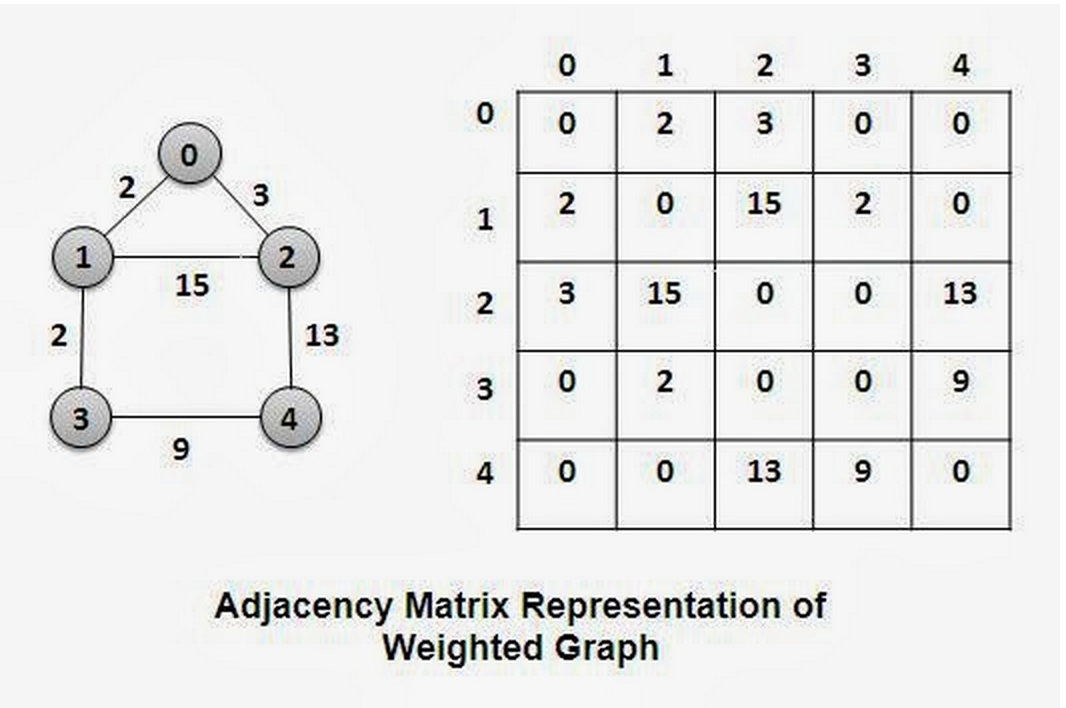
* 1. Adjacency matrix.

An adjacency matrix is NxN boolean matrix in which the value in the row “I” and column “j” is 1 if there is an edge between vertex “I” and vertex “j” otherwise 0. If graph is weighted the value of the weight could be used instead of 1.

This graph representation is better suitable for dense graphs where the number of edges is large factor of the number of vertices. The drawback is that it takes a lot of memory space. For a graph with N vertices it will require N2 memory space.

Some of the advantages are:

* Finding presence of an edge between vertices vi and vj will take constant time.
* Degree of a vertex can easily be calculated by counting all non-zero entries in the corresponding row of the adjacency matrix.
* Simple to implement



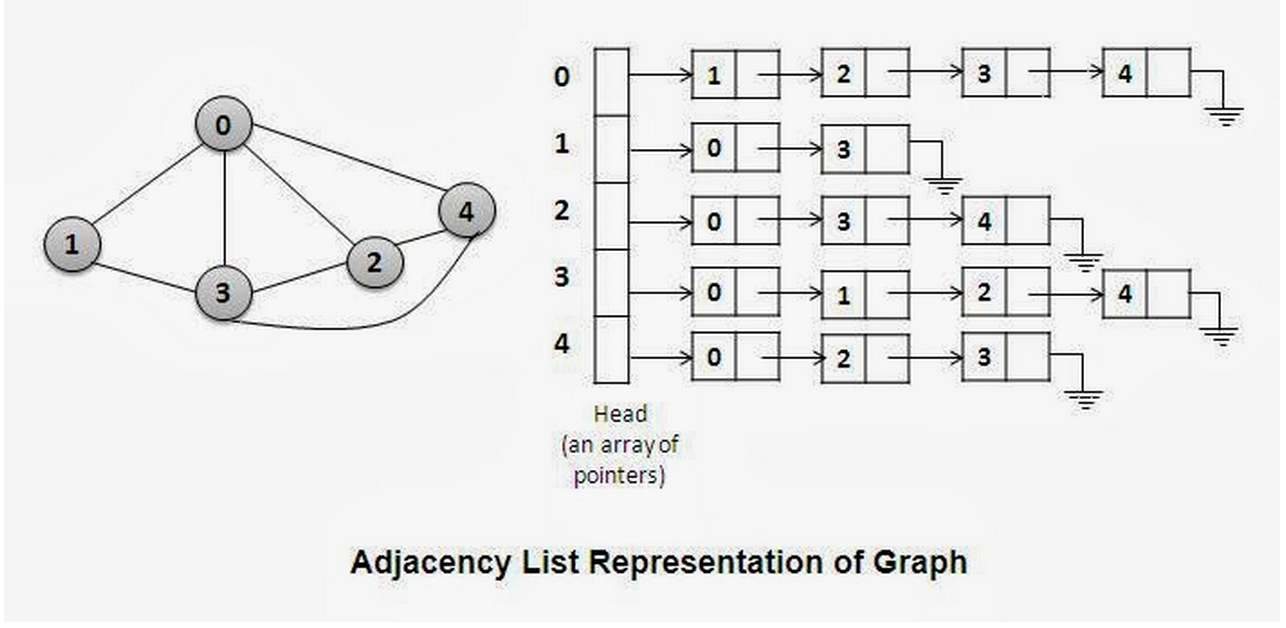
* 1. Array of adjacency lists.

Array of adjacency list is a vertex index array of linked lists. Each linked list has the neighbors of the index in the array.

This form of graph representation is more suitable for graphs that are not dense. It is very memory efficient because the graph has number of edges close small factor of number of vertices.

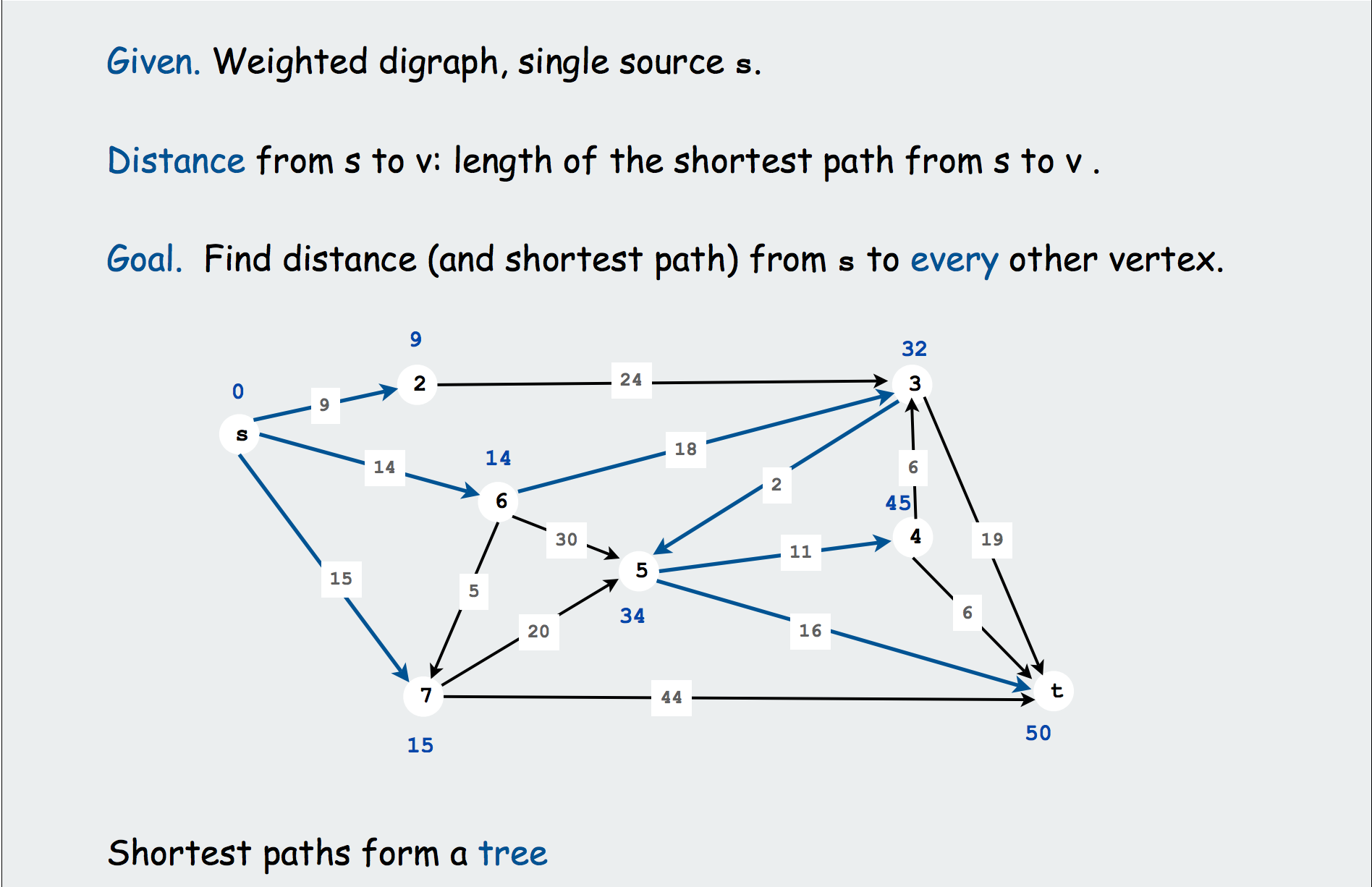
The advantages are:

* Space usage proportional to V+E
* Constant time to add an edge
* Time proportional to degree of vertex v to iterate through vertices adjacent to v, where the degree is the length of the linked list.



1. **Single source shortest path problem**

For a weighted graph G(V,E, w) the single source shortest problem is to find the shortest paths from ­a given vertex *v*  to all other vertices in V that are part of the graph.



The shortest path will be the one with minimum-weight. The result of the computation form is a tree, known as shortest-path tree (SPT). Depending on the application the weight can be representation of different things – such as distance, money, penalty, loss etc.

Properties of the shortest paths worth mentioning:

* Paths are directed.
* The weights are not necessarily distances.
* Not all vertices need to be reachable.
* Negatives weights introduce complications
* Shortest path is normally simple – we ignore 0 weights as they form cycle.
* Shortest path is not necessarily unique.

Finding shortest-path is a broadly used problem-solving model. It finds applications in maps, texture mapping, robot navigation, urban traffic planning, routing telecommunication messaging and many more.

Dijkstra’s algorithm for finding single shortest path from source vertex ***s*** works by incrementally finding the shortest path from *s* to other vertices in G. It is a greedy algorithm in sense that always chose edge to a vertex that appears closest.

1. **Sequential solution.**
   1. Implementation

Having graph below the single source shortest path is going to be computed in number of steps starting from source node S=0. In each step another node gets added to the set of resolved nodes such as the cost to reach that node from source is maintained minimum. Nodes that are considered in each step are the ones adjacent to already resolved nodes.

* Step 1:

We initialized source node with parent being None, and Cost = 0.

2

16

8

5

6

7

4

1

3

* Step 2:

Adding node 3 to set of resolved node, because has minim Cost=4 among nodes 3, 4, 1, and setting parent to 0.

2

16

8

5

6

7

4

1

3

* Step3:

Adding node 4 to set of resolved with Cost = 7 and parent= 3.

2

16

8

5

6

7

4

1

3

* Step4:

Adding node 5 with Cost = 8 and parent =4.

2

16

8

5

6

7

4

1

3

* Step5:

Adding node 1 with Cost =13, and parent = 5.

2

16

8

5

6

7

4

1

3

* Step6;

Adding node 2 with Cost=14 and parent= 5.

2

16

8

5

6

7

4

1

3

By end of all iterations we have formed shortest path tree (SPT).

In appendix-A method s\_SSP of the Graph class is a sequential solution for a Dijkstra’s single source shortest path. At each iteration next solved vertex V is added to an array of all solved vertices (solved\_v). Method print\_all\_shortest prints all shortest paths from the source with their respective costs. Below is output for the example graph from the above figures.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SEQUENTIAL \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Working with:

[[None, 16, None, 4, 8, None], [16, None, 2, None, 7, 5], [None, 2, None, None, None, 6], [4, None, None, None, 3, None], [8, 7, None, 3, None, 1], [None, 5, 6, None, 1, None]]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Path: 0, Cost: 0

Path: 0-3-4-5-1, Cost: 13

Path: 0-3-4-5-2, Cost: 14

Path: 0-3, Cost: 4

Path: 0-3-4, Cost: 7

Path: 0-3-4-5, Cost: 8

* 1. Running time

1. **Parallel solution**