Innovative Clinical Collaboration, Not Software

www.3DnetMedical.com

Brief Overview of Vessel Module (Design and Implementation)

Biotronics 3D

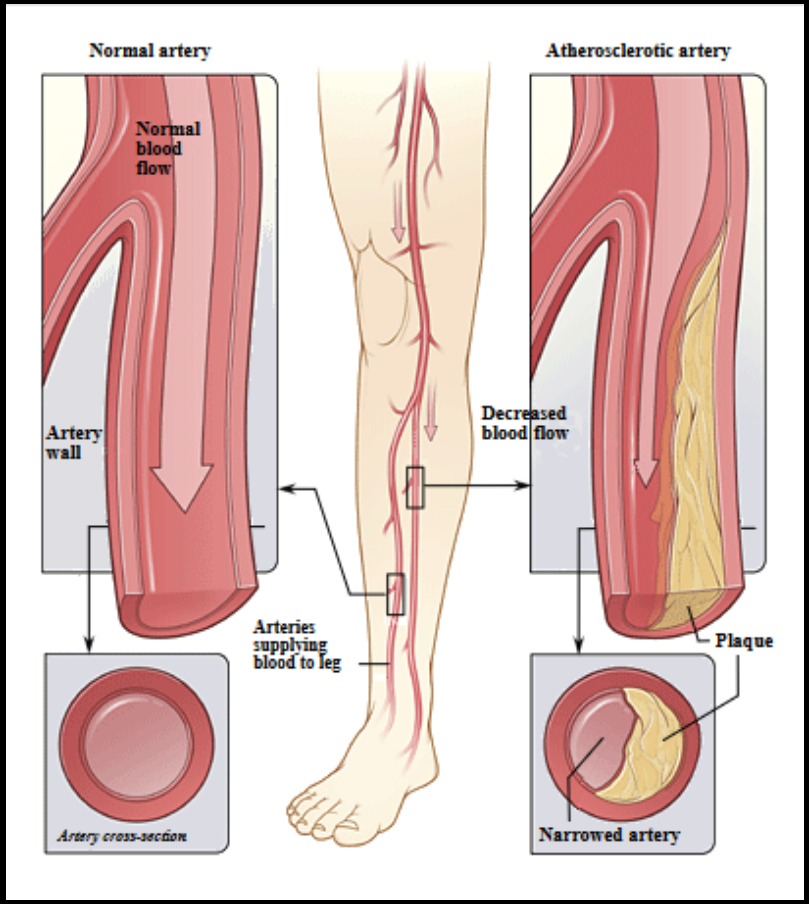
Analyze · Collaborate · Discover

Overview

- Use case
  - Efficient use of Computed Tomography (CT) angiography in Peripheral Artery Disease (PAD)
- Design of the Vessel Module
- Implementation
  - Vessel Centerline Extraction
- Testing and Reusability

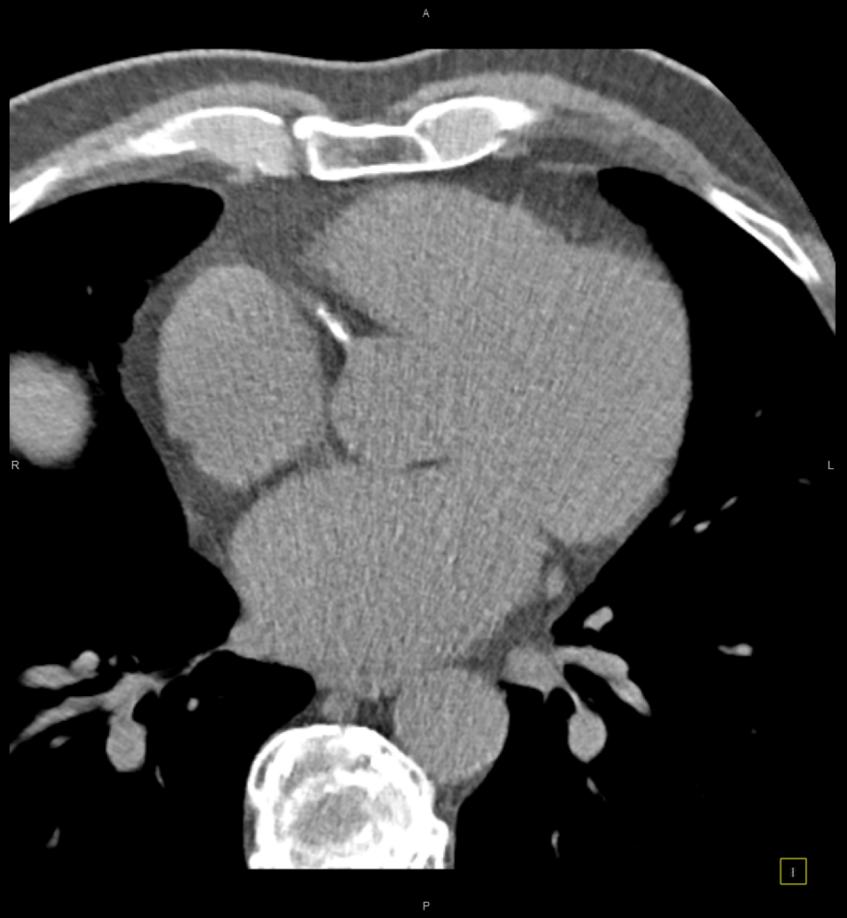
What is Peripheral Artery Disease (PAD)?

- Pain and severe cramping when walking or exercising in the calf muscles of the affected leg.
- Severe PAD complications include tissue death.
- Risk factors includes smoking, high blood sugar and high blood pressure.



Use of Computed Tomography (CT) angiography in Peripheral Artery Disease (PAD)

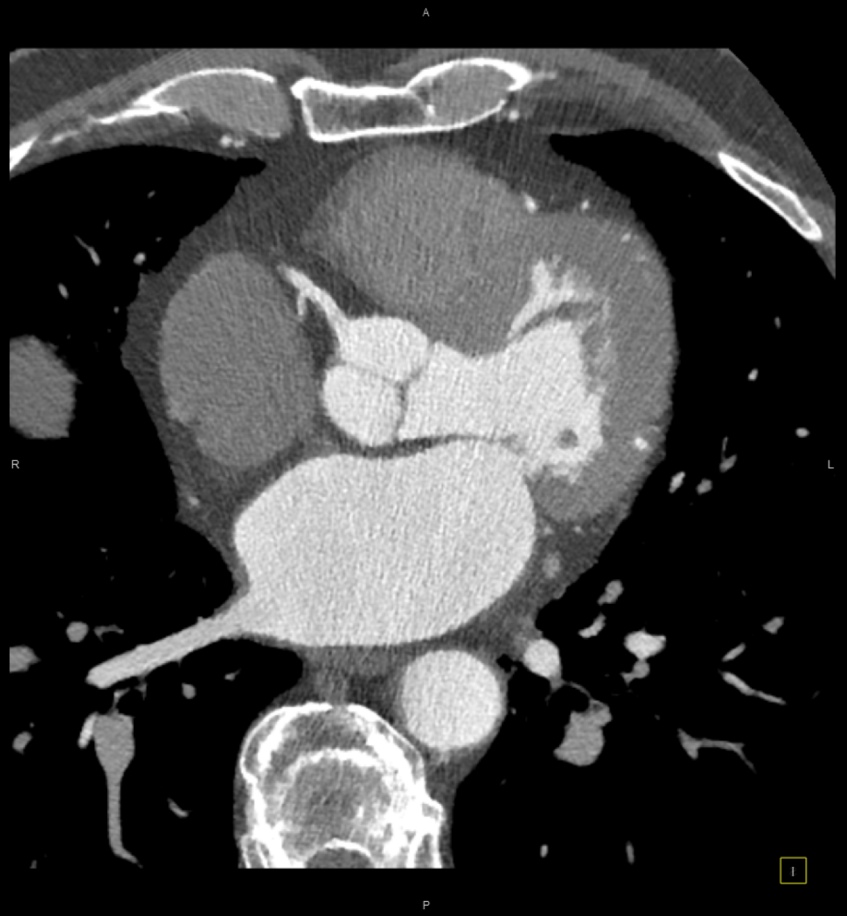
- CT angiography is a contrast CT where images are taken with a certain delay after injection of radiocontrast material (e.g. iodine)



Before injection

Use of Computed Tomography (CT) angiography in Peripheral Artery Disease (PAD)

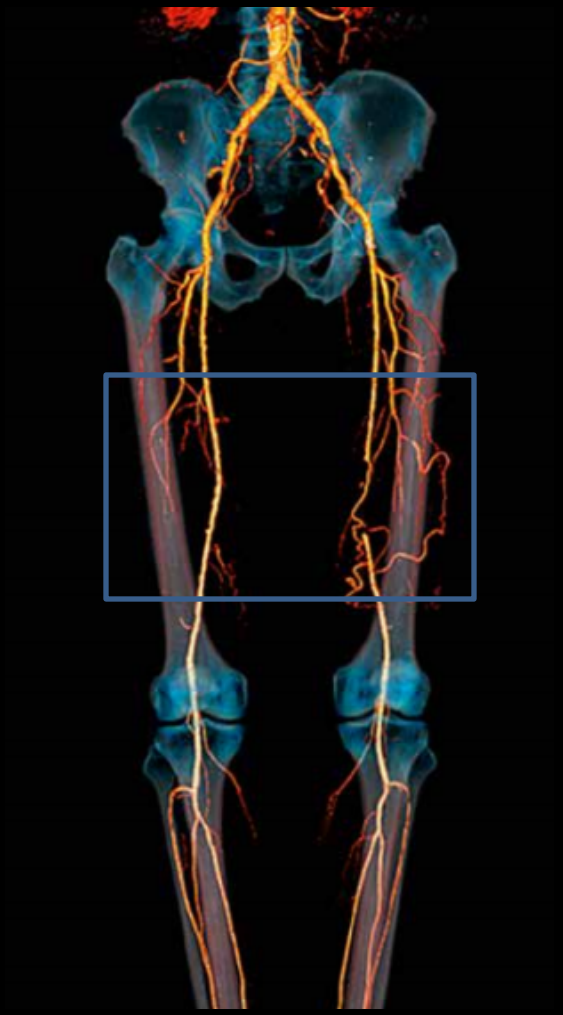
- CT angiography is a contrast CT where images are taken with a certain delay after injection of radiocontrast material (e.g. iodine)



After injection

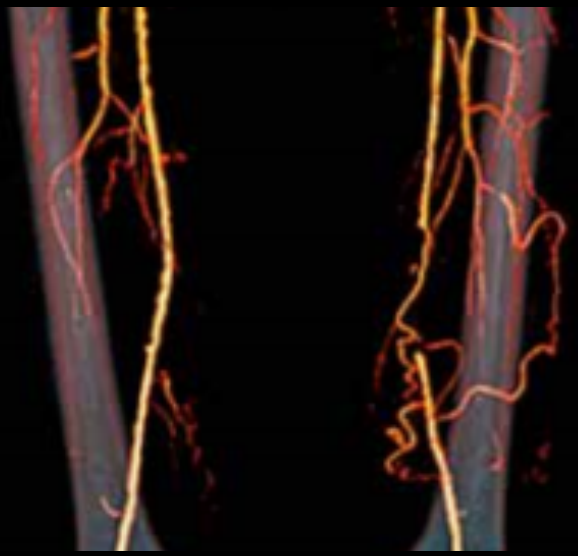
Use of Computed Tomography (CT) angiography in Peripheral Artery Disease (PAD)

- A complete thrombotic occlusion in the left superficial femoral artery (SFA), with reformation in the adductor canal through collaterals.



Use of Computed Tomography (CT) angiography in Peripheral Artery Disease (PAD)

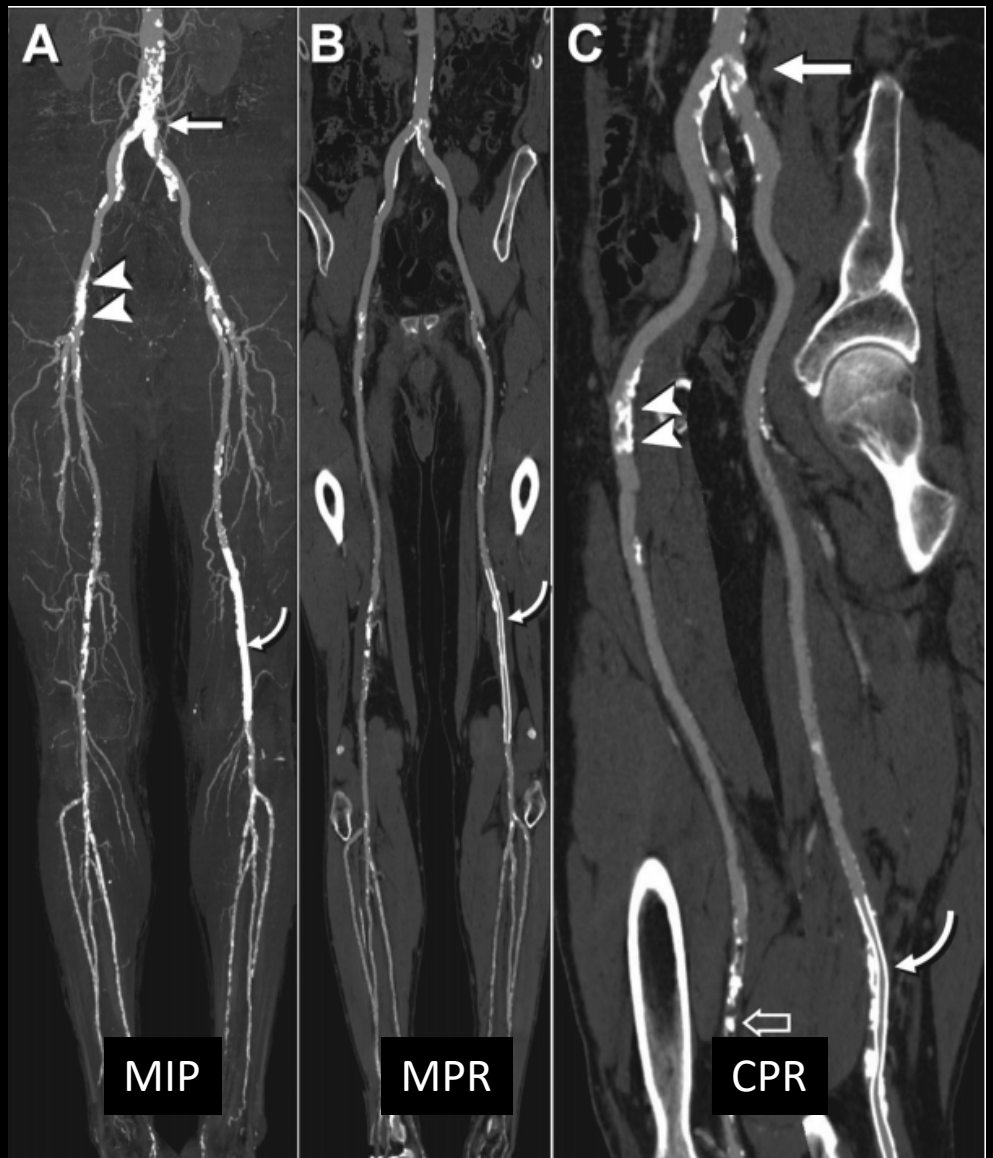
- A complete thrombotic occlusion in the left superficial femoral artery (SFA), with reformation in the adductor canal through collaterals.



Use of Computed Tomography (CT) angiography in Peripheral Artery Disease (PAD)

- More subtle narrowing of vessels (stenosis).

Use of Computed Tomography (CT) angiography in Peripheral Artery Disease (PAD)



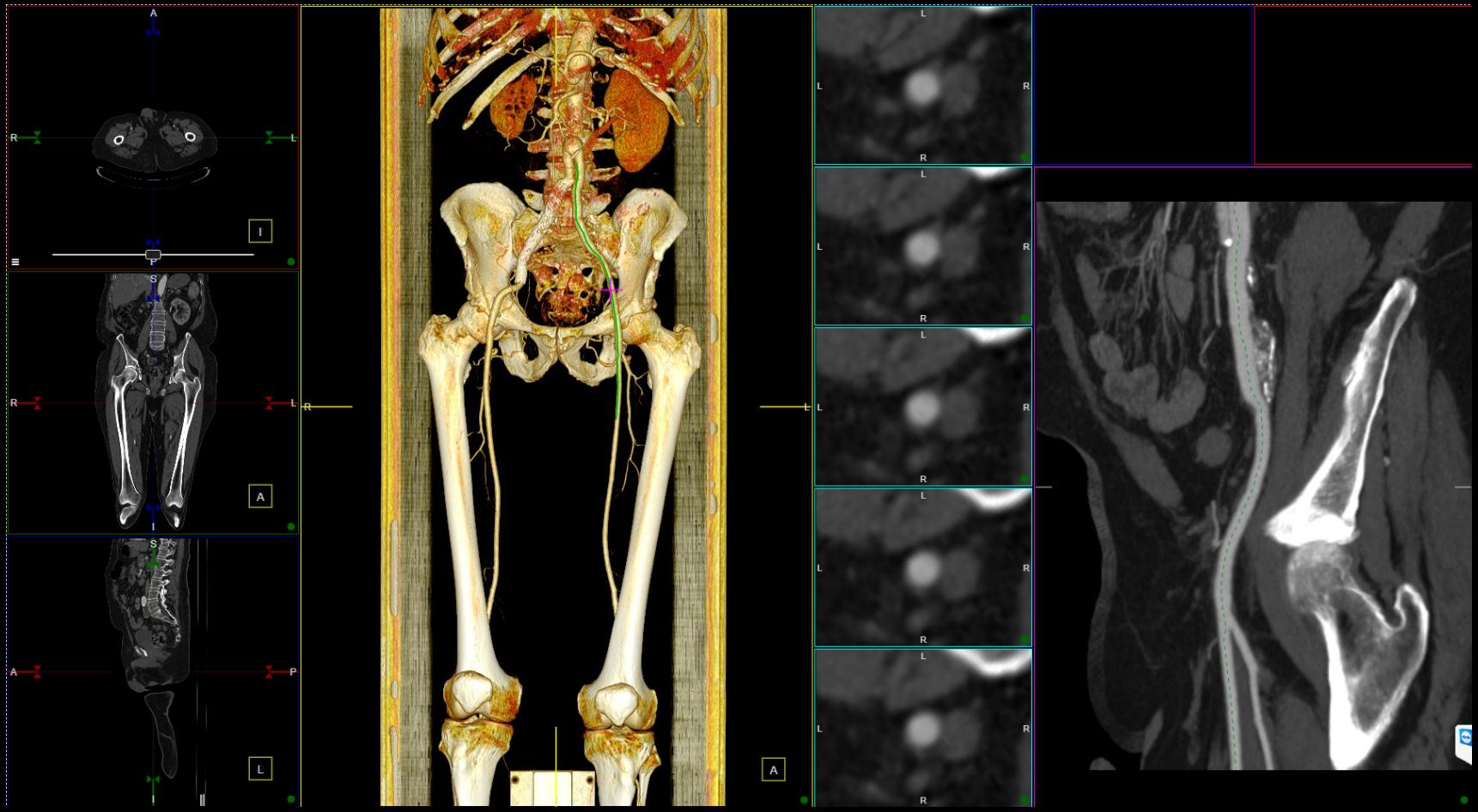
Use of Computed Tomography (CT) angiography in Peripheral Artery Disease (PAD)

- Big challenges:
  - Difference between quick read vs. painful (literally) scrolling through images.
  - Axial (transverse) images inadequate.
  - Volumetric review of volumetric datasets.

- Need cross sections (MPR/CPR)
  - Try to delegate (3D-Lab, trained technologist) if routinely performing runoff CTAs

Design of the Vessel Module

- Demo



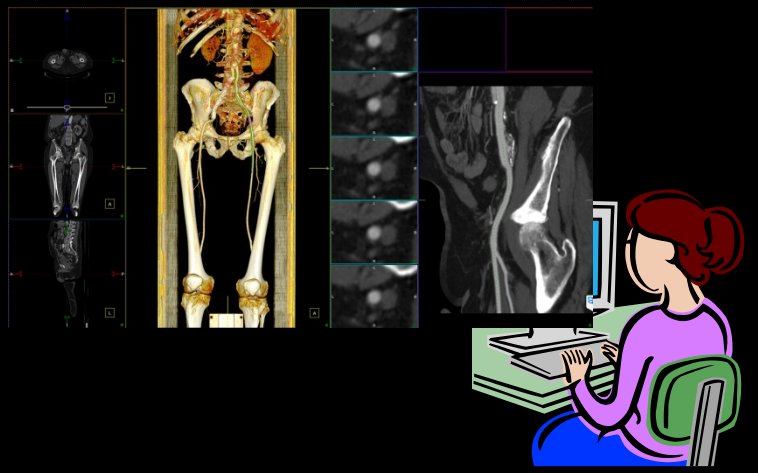
Overview

- Use case
  - Efficient use of Computed Tomography (CT) angiography in Peripheral Artery Disease (PAD)
- Design of the Vessel Module
- Implementation
  - Vessel Centerline Extraction
- Testing and Reusability

Design of the Vessel Module

Server

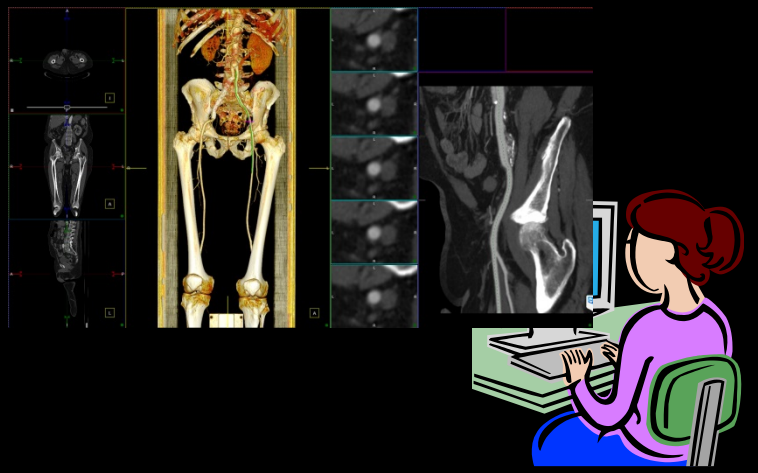
User Interface (web client)



Design of the Vessel Module

Server
- Load data
- Process MPR, VR and CPR
- Process vessel extraction
- Talk to client

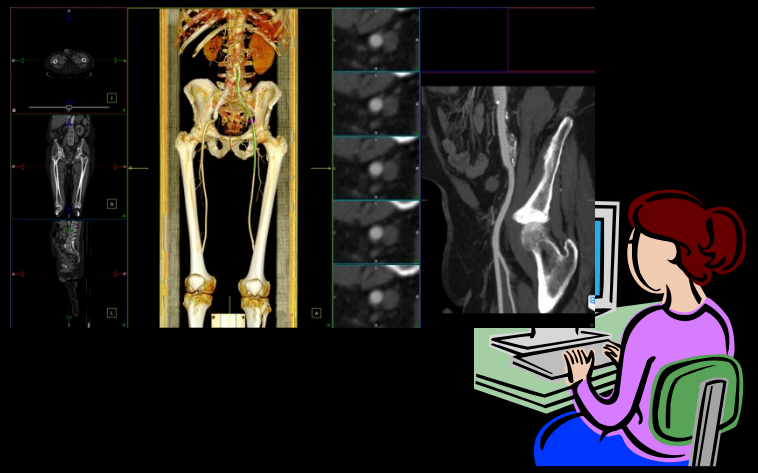
User Interface (web client)
- User inputs
- Display MPR, VR and CPR
- Talk to server



Design of the Vessel Module

Server
- Load data
- Process MPR, VR and CPR
- Process vessel extraction
- Talk to client

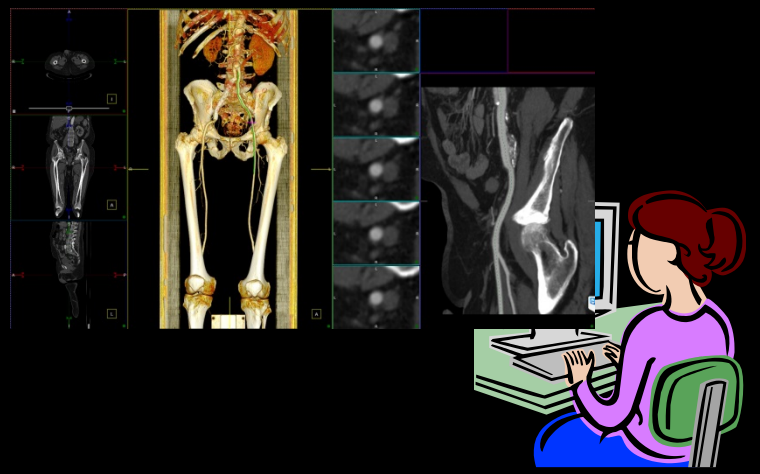
User Interface (web client)
- User inputs
  - TrackedVesselMouseHandler
  - StenosisMouseHandler
- Display MPR, VR and CPR
- Talk to server
  - ITrackedVesselValue
  - IStenosisValue



Design of the Vessel Module

Server
- Load data
- Process MPR, VR and CPR
- Process vessel extraction
  - ModuleProcessingQueue
  - CenterlineExtractionAlgorithm
- Talk to client
  - SegmentationVesselTrackResponse
  - SegmentationStenosisResponse

User Interface (web client)
- User inputs
  - TrackedVesselMouseHandler
  - StenosisMouseHandler
- Display MPR, VR and CPR
- Talk to server
  - ITrackedVesselValue
  - IStenosisValue



Design of the Vessel Module (UI)

(… mouse messages handling …)

 mouse start dragging

 mouse end dragging

User Interface (web client)

- User inputs
  - TrackedVesselMouseHandler
  - StenosisMouseHandler
- Display MPR, VR and CPR
- Talk to server
  - ITrackedVesselValue
  - IStenosisValue

Design of the Vessel Module (UI)

Base class InternalMouseDragHandler
- OnMouseDragStart
- OnMouseDragEnd
- Polymorphic behaviour when dragging

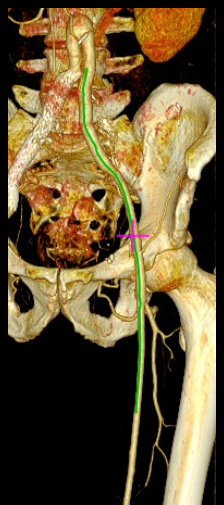
TrackedVesselMouseHandler
- OnMouseDragStart
- OnMouseDragEnd
- Create ITrackedVesselValue

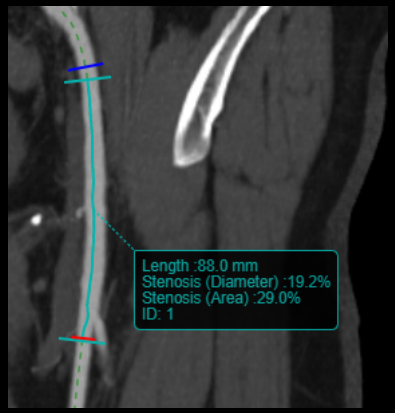
StenosisMouseHandler
- OnMouseDragStart
- OnMouseDragEnd
- Create IStenosisValue

(… mouse messages handling …)
InternalMouseDragHandler handler;

 calls handler.OnMouseDragStart();

 calls handler.OnMouseDragEnd();





User Interface (web client)
- User inputs
  - TrackedVesselMouseHandler
  - StenosisMouseHandler
- Display MPR, VR and CPR
- Talk to server
  - ITrackedVesselValue
  - IStenosisValue

Overview

- Use case
  - Efficient use of Computed Tomography (CT) angiography in Peripheral Artery Disease (PAD)
- Design of the Vessel Module
- Implementation
  - Vessel Centerline Extraction
- Testing and Reusability

Implementation of the Vessel Module

- Vessel centerline extraction algorithm for CPR

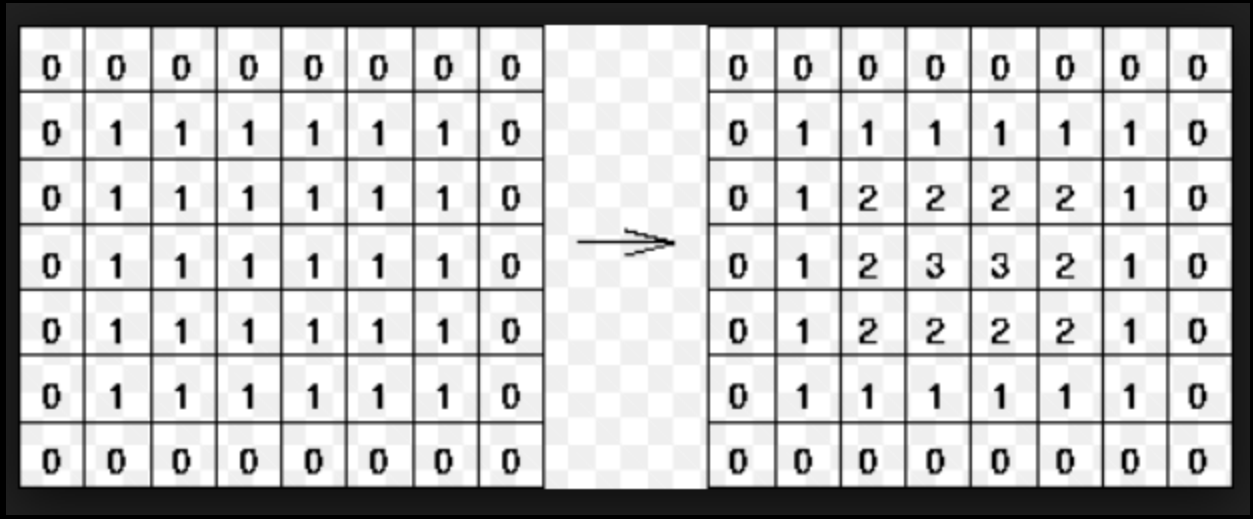
Inputs:
- Two points
- Thresholds of vessel intensity

Implementation of the Vessel Module

- Vessel centerline extraction algorithm for CPR
  - Distance transform
    - Labels each pixel of the image with the distance to the boundary pixel.
  - Shortest path
    - between the start and end points in the distance transform.

Implementation of the Vessel Module

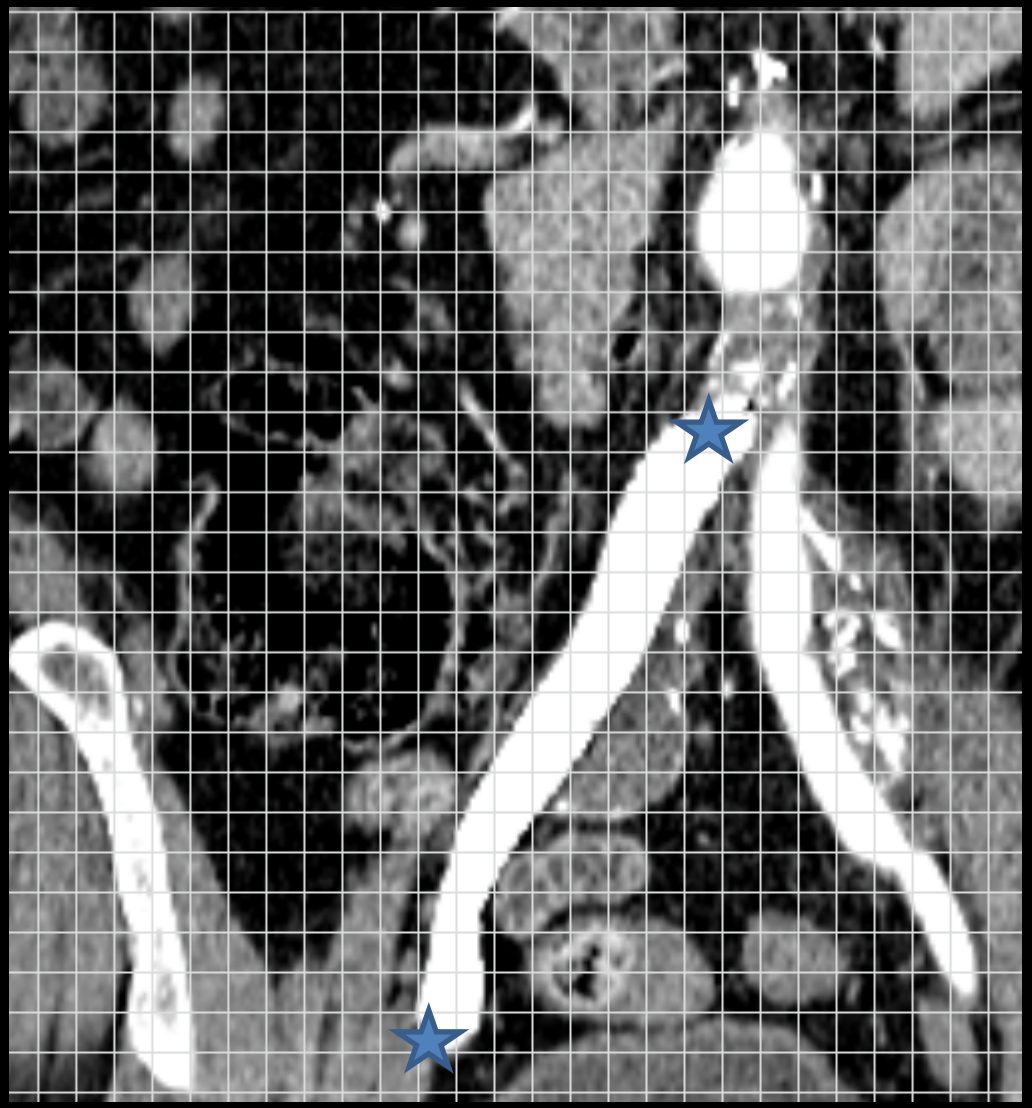
- Vessel centerline extraction algorithm for CPR
  - Distance transform
    - Labels each pixel of the image with the distance to the boundary pixel.



  - Shortest path
    - between the start and end points in the distance transform.

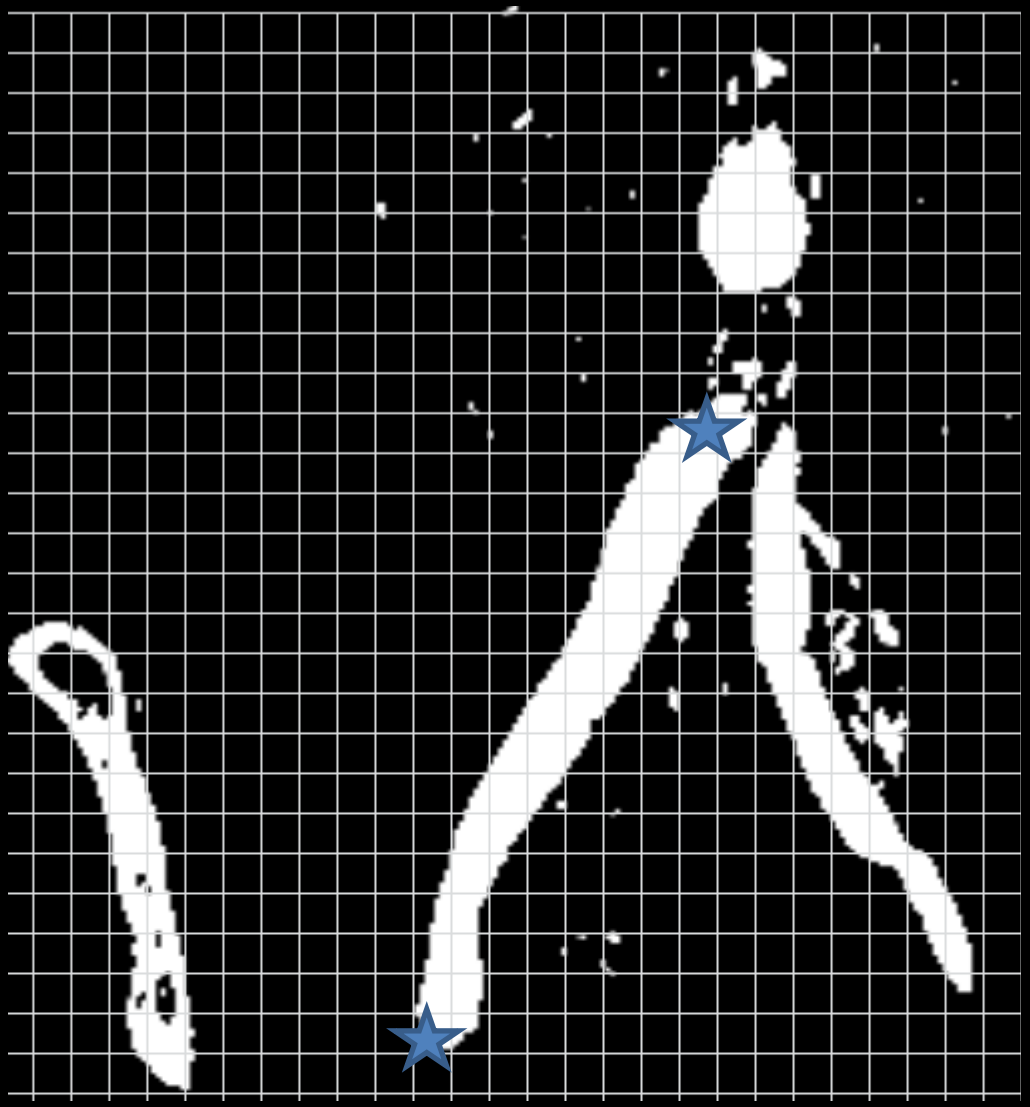
Implementation of the Vessel Module

- Distance transform
  - Labels each pixel of the image with the distance to the boundary pixel.
- Shortest path
  - between the start and end points in the distance transform.



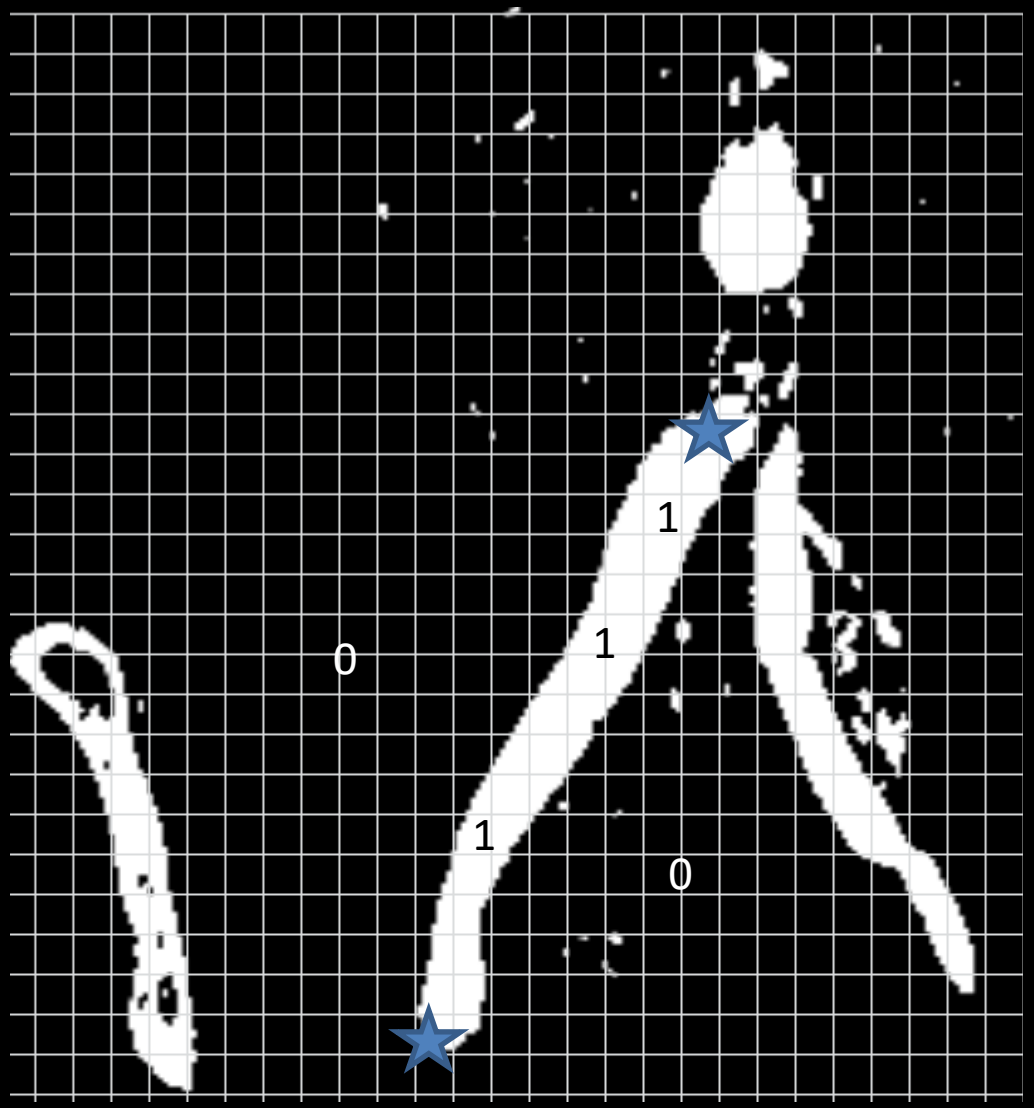
Implementation of the Vessel Module

- Distance transform
  - Labels each pixel of the image with the distance to the boundary pixel.
- Shortest path
  - between the start and end points in the distance transform.



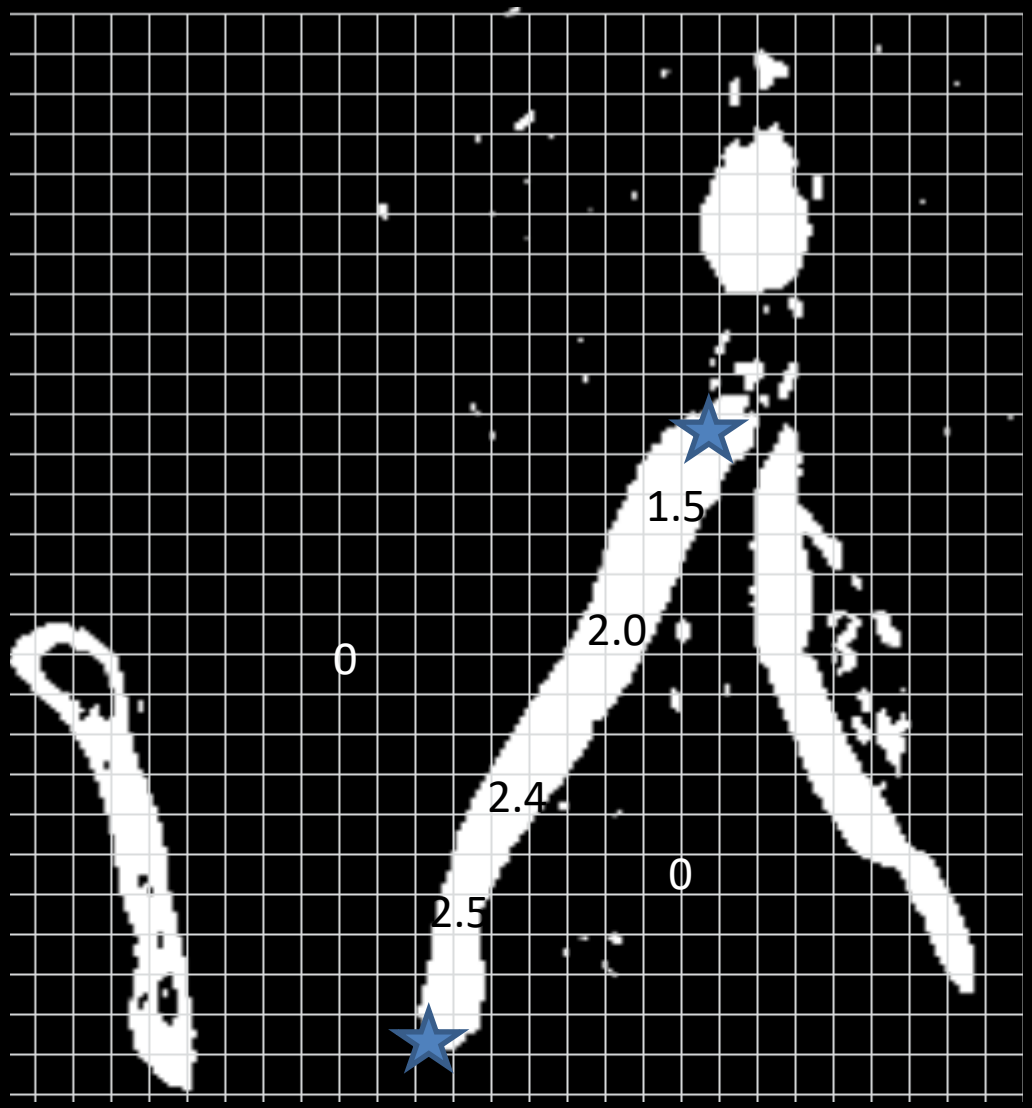
Implementation of the Vessel Module

- Distance transform
  - Labels each pixel of the image with the distance to the boundary pixel.
- Shortest path
  - between the start and end points in the distance transform.



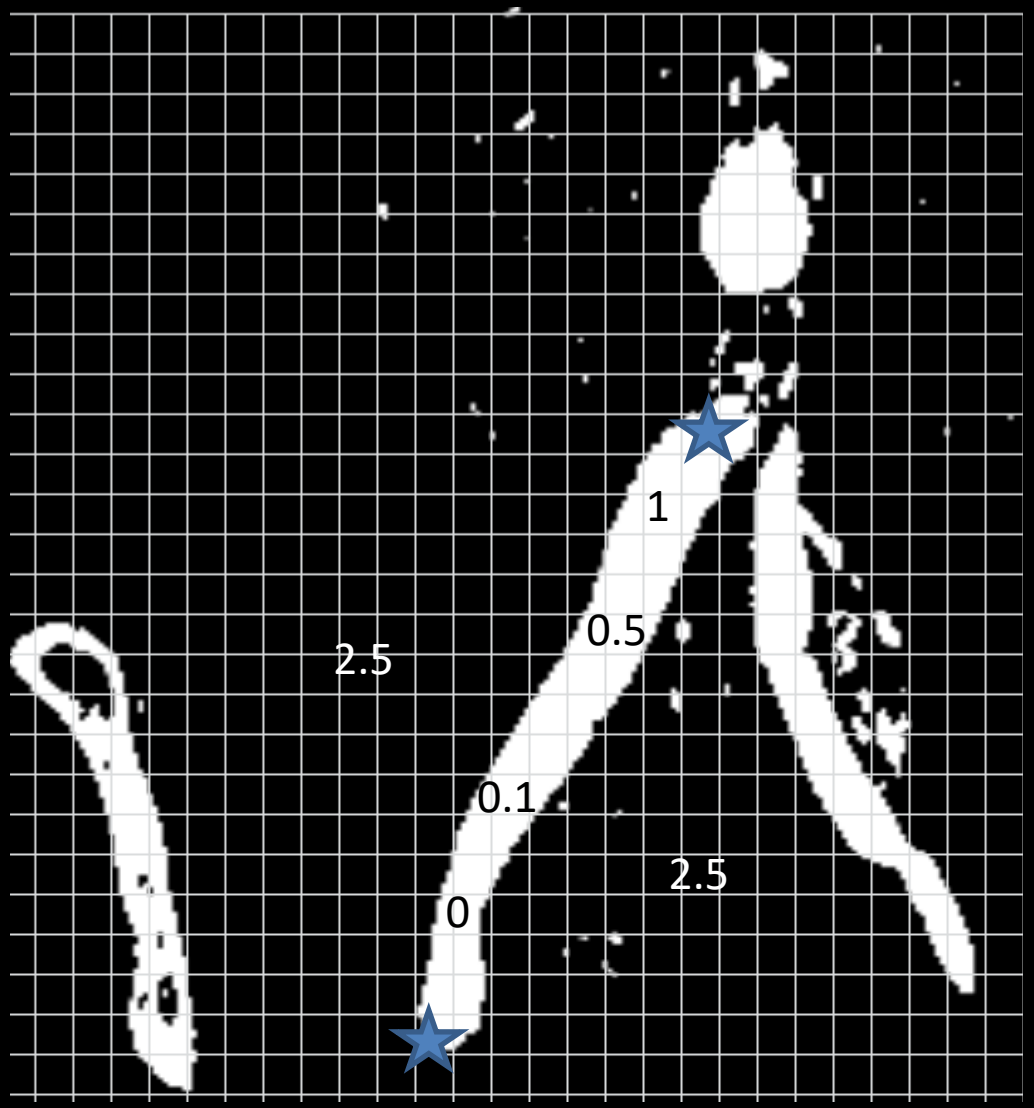
Implementation of the Vessel Module

- Distance transform
  - Labels each pixel of the image with the distance to the boundary pixel.
- Shortest path
  - between the start and end points in the distance transform.



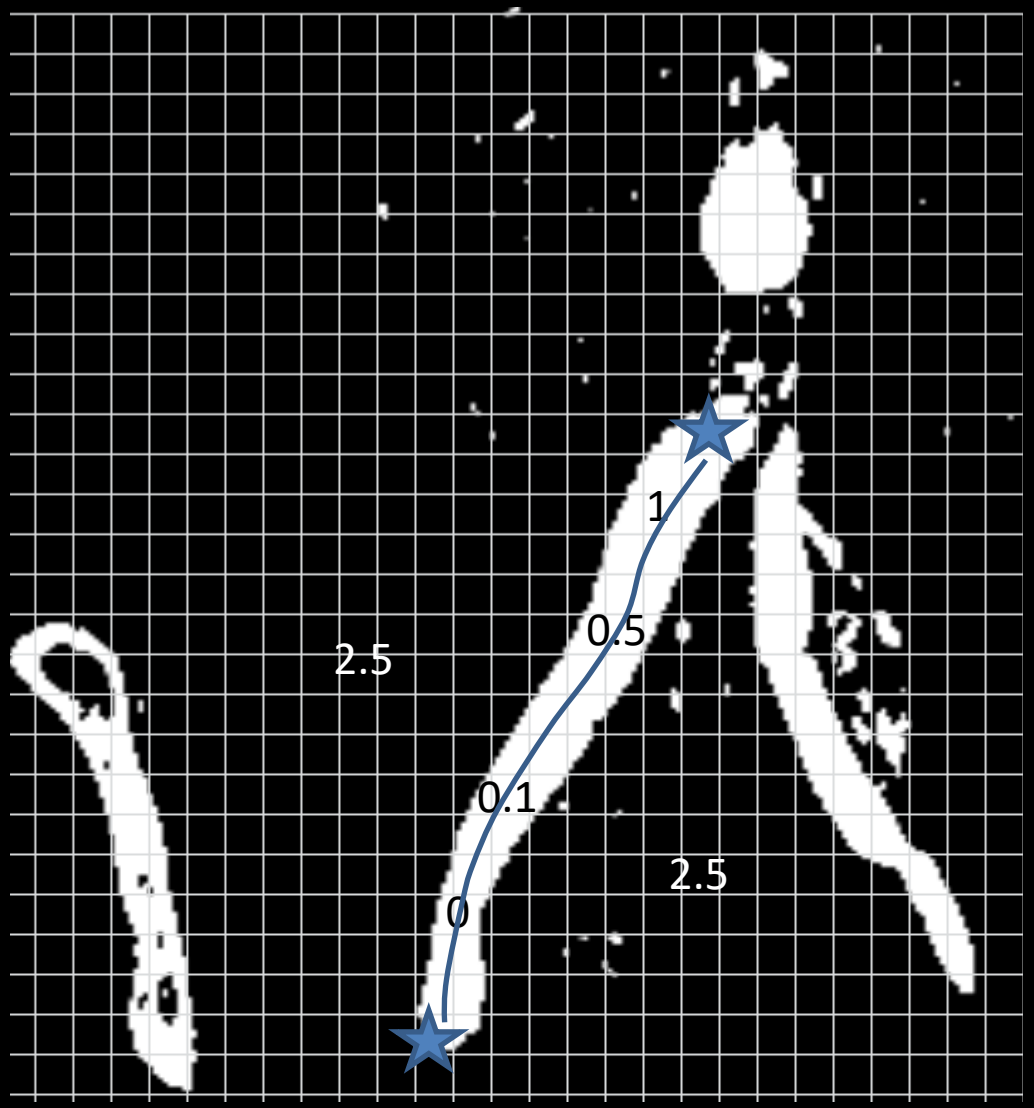
Implementation of the Vessel Module

- Distance transform
  - Labels each pixel of the image with the distance to the boundary pixel.
- Shortest path
  - between the start and end points in the distance transform.



Implementation of the Vessel Module

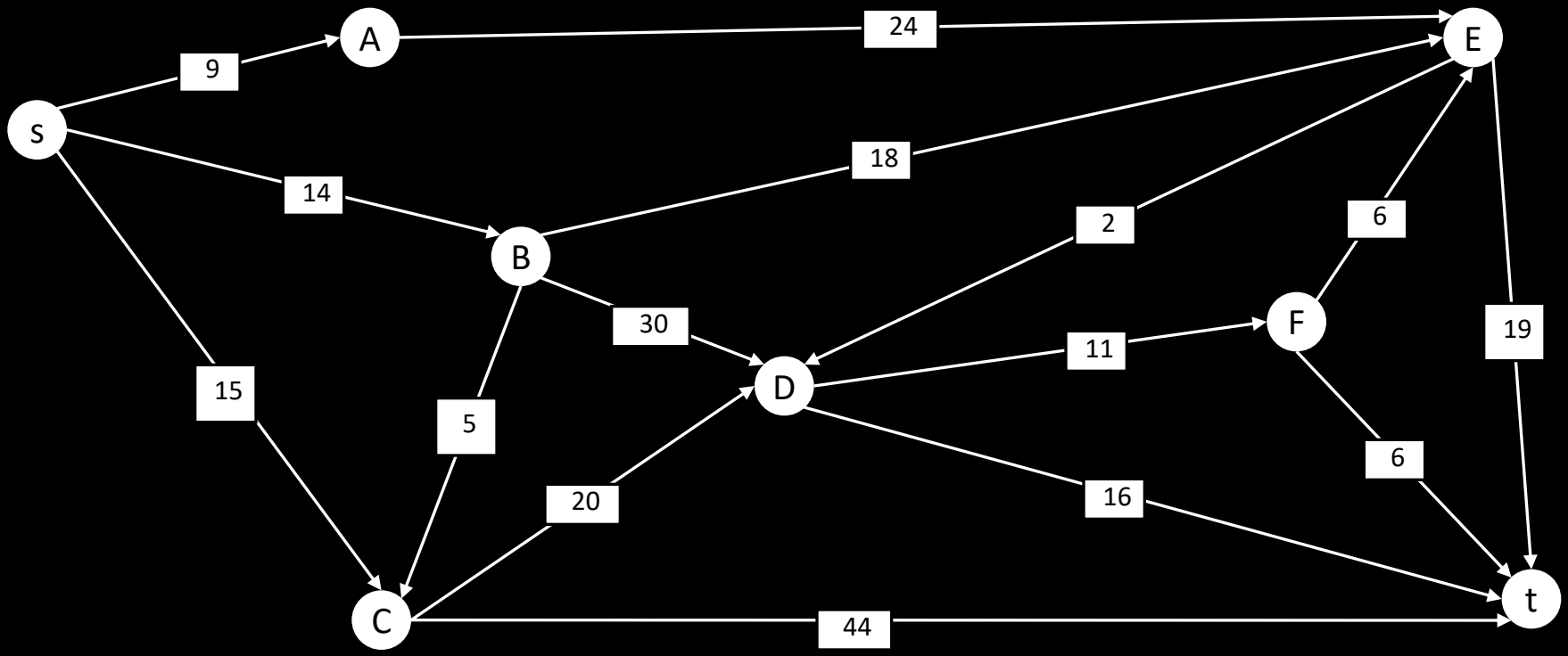
- Distance transform
  - Labels each pixel of the image with the distance to the boundary pixel.
- Shortest path
  - between the start and end points in the distance transform.



Dijkstra's Shortest Path Algorithm

- Find shortest path from s to t – edges with non-negative weights.

- It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbour, and updates the neighbour's distance if smaller.

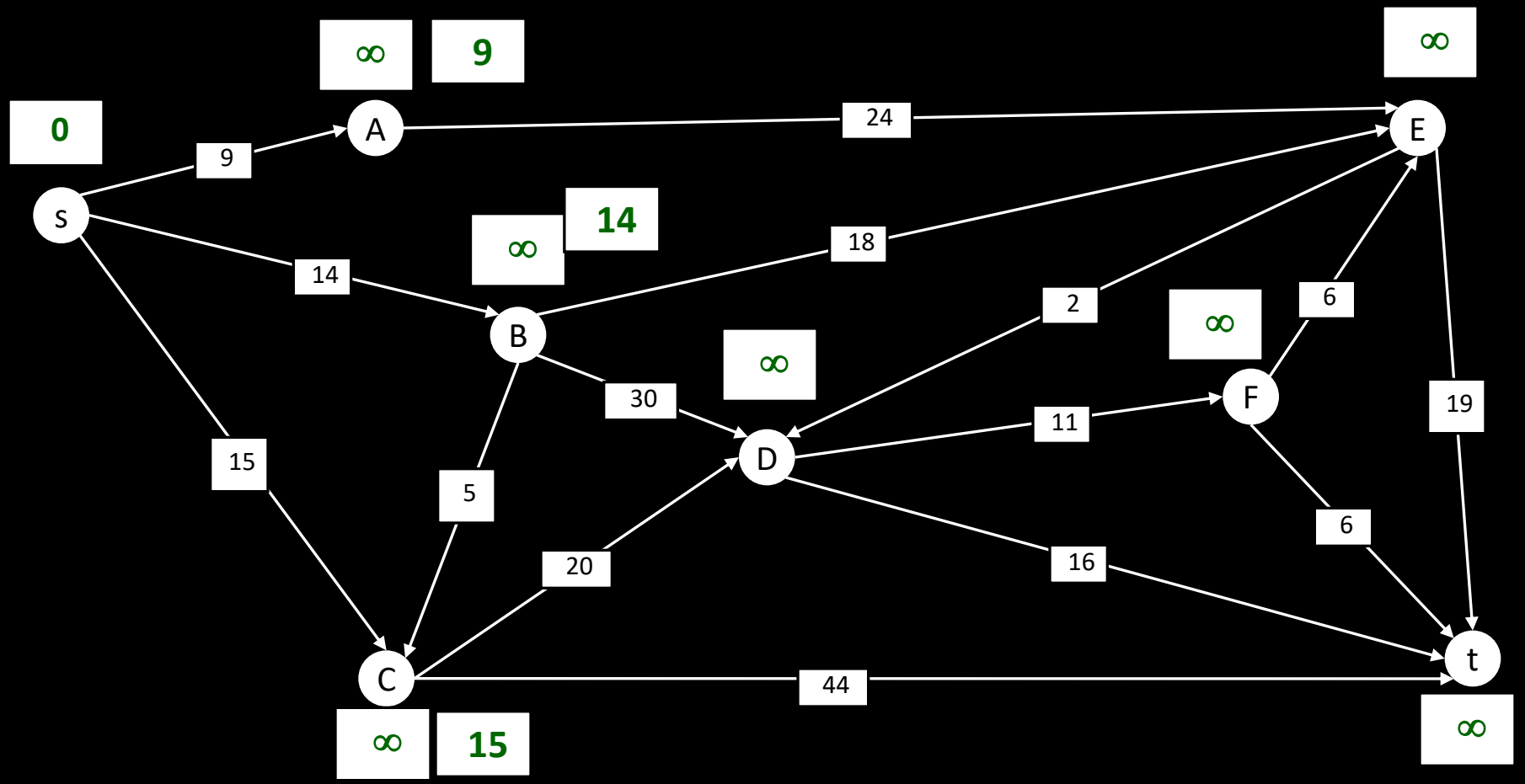


Dijkstra's Shortest Path Algorithm

It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbour, and updates the neighbour's distance if smaller.

Nodes = { s, A, B, C, D, E, F, t }

Pick s

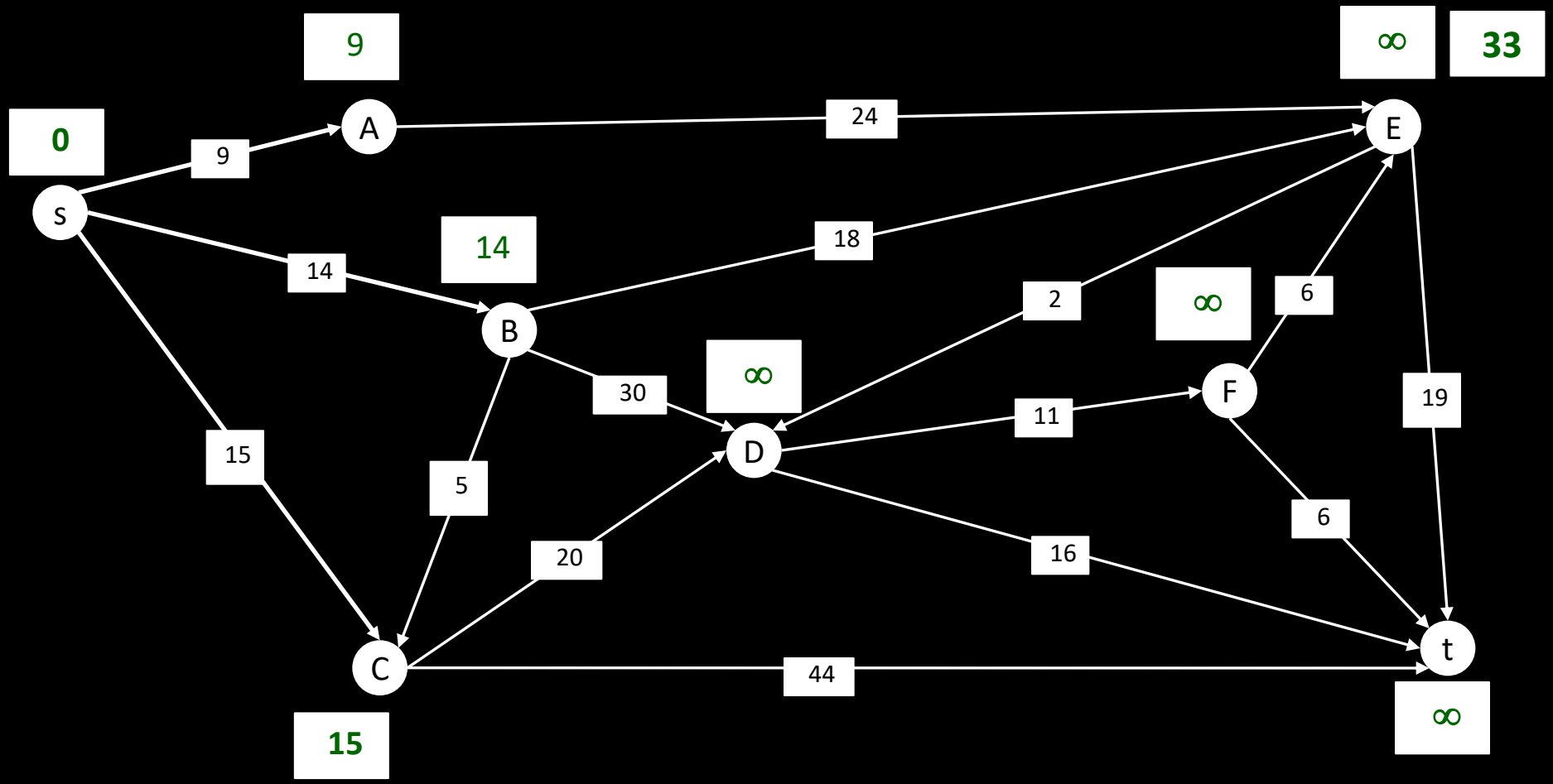


Dijkstra's Shortest Path Algorithm

It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbour, and updates the neighbour's distance if smaller.

Nodes = { A, B, C, D, E, F, t }

Pick A

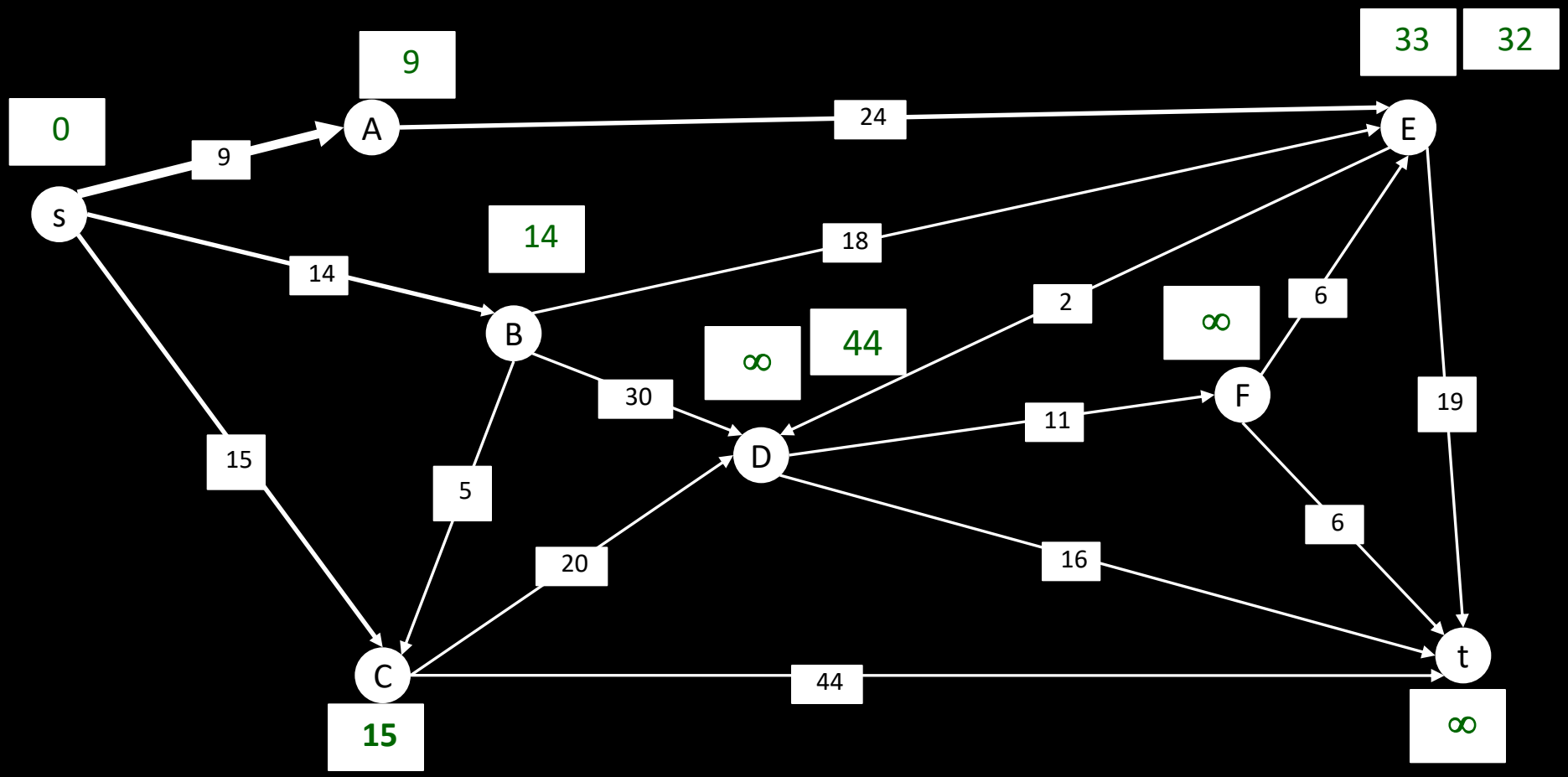


Dijkstra's Shortest Path Algorithm

It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbour, and updates the neighbour's distance if smaller.

Nodes = { B, C, D, E, F, t }

Pick B

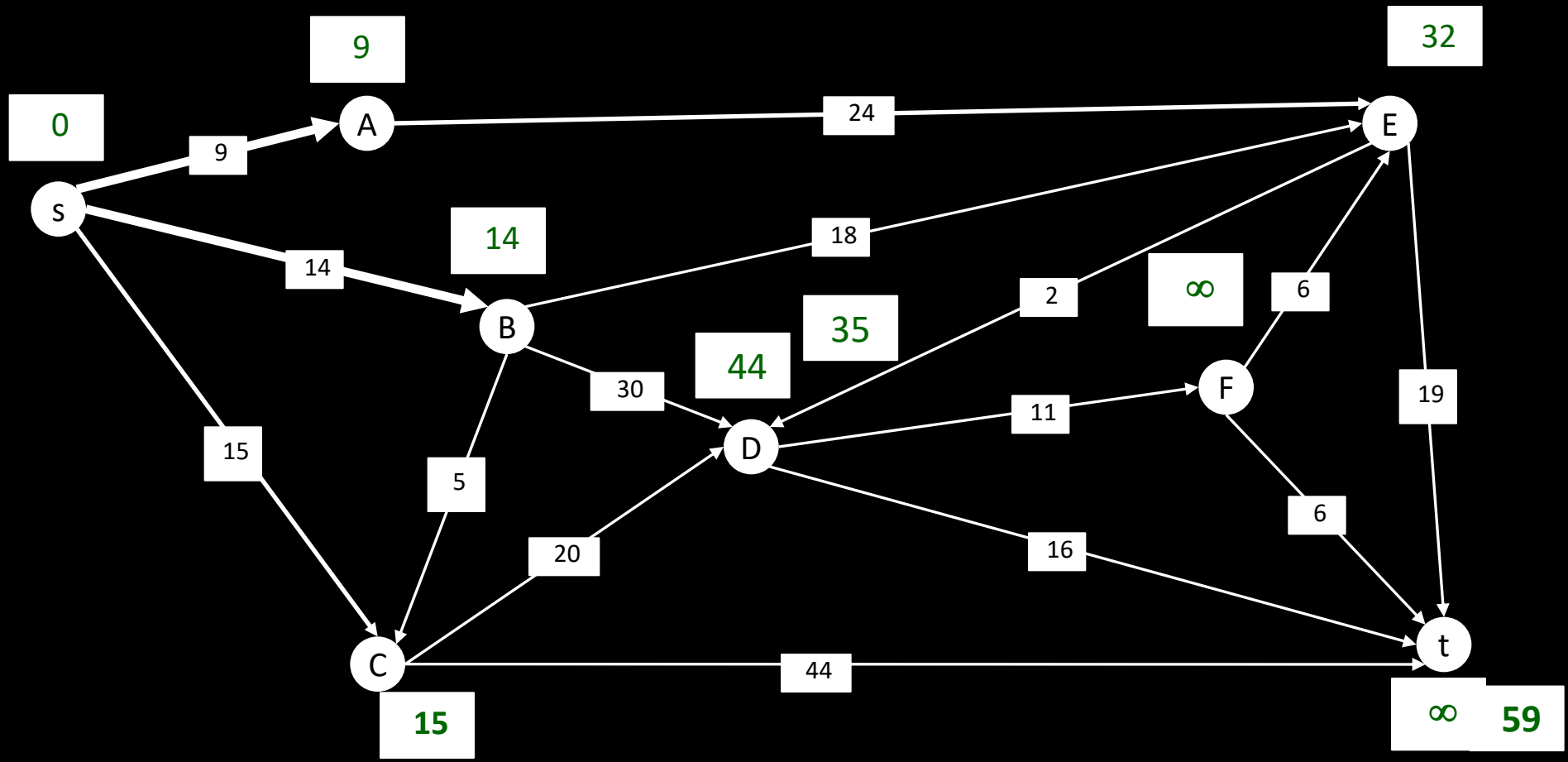


Dijkstra's Shortest Path Algorithm

It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbour, and updates the neighbour's distance if smaller.

Nodes = { C, D, E, F, t }

Pick C

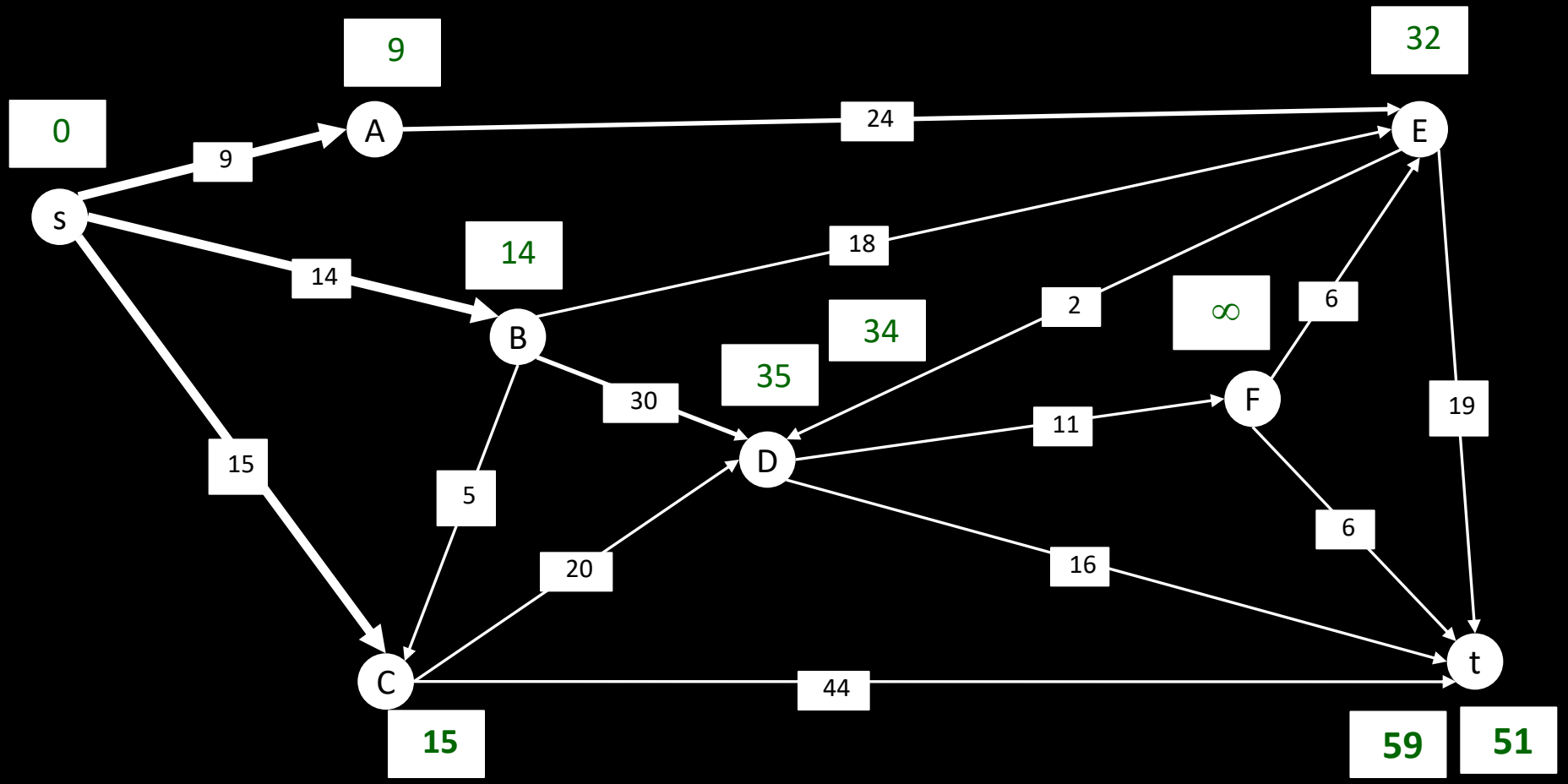


Dijkstra's Shortest Path Algorithm

It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbour, and updates the neighbour's distance if smaller.

Nodes = { D, E, F, t }

Pick E

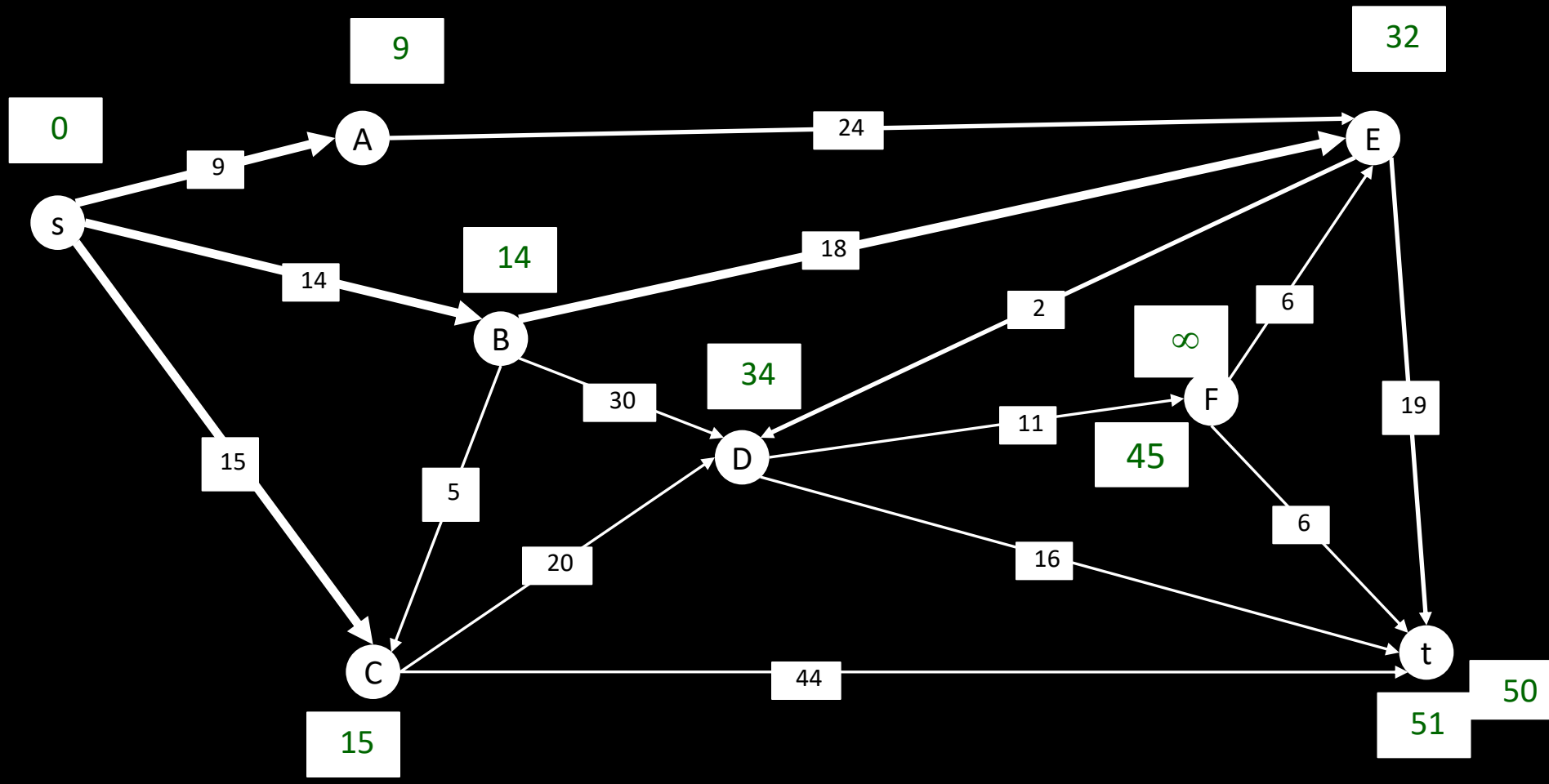


Dijkstra's Shortest Path Algorithm

It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbour, and updates the neighbour's distance if smaller.

Nodes = { D, F, t }

Pick D

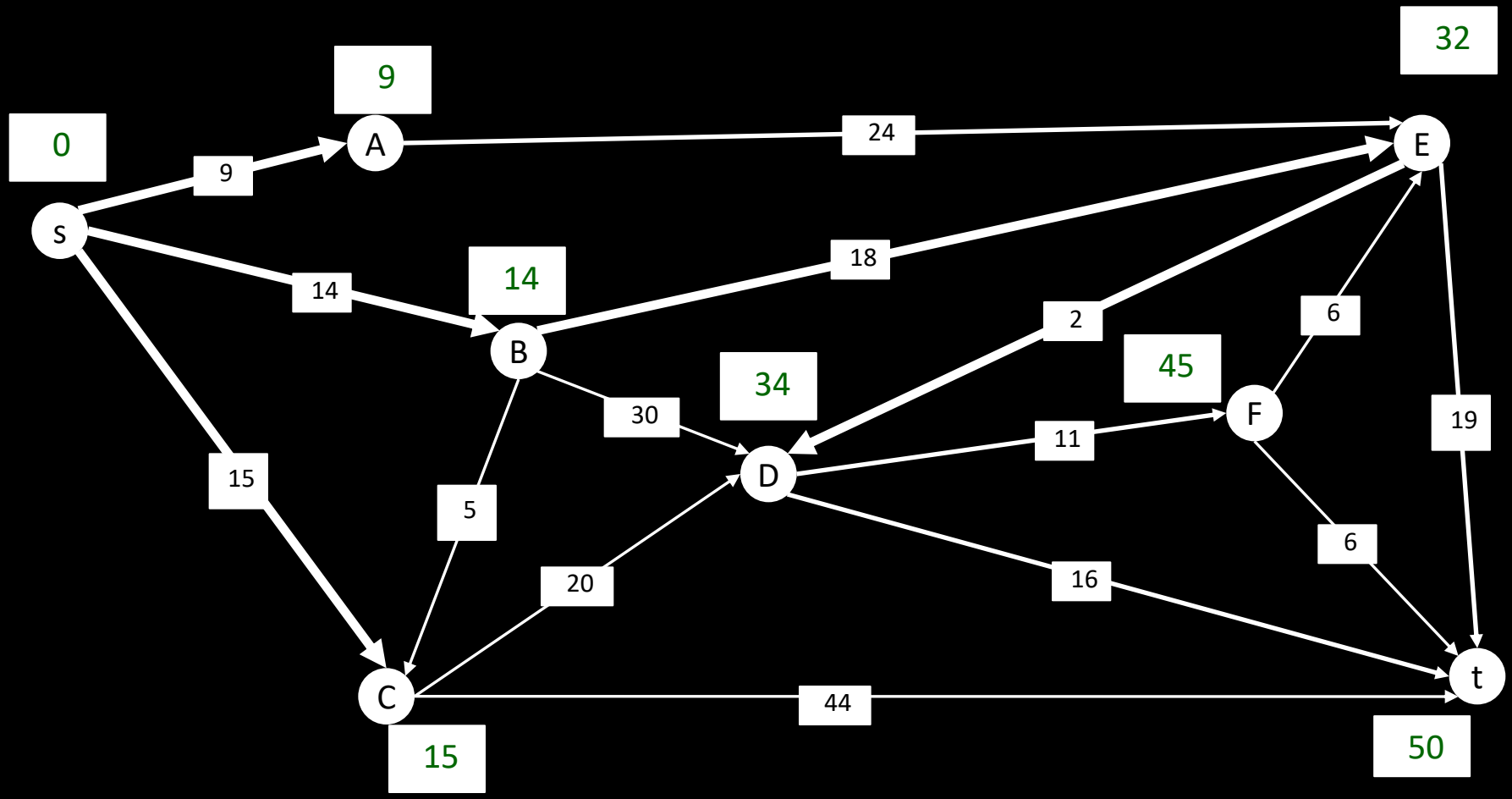


Dijkstra's Shortest Path Algorithm

It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbour, and updates the neighbour's distance if smaller.

Nodes = { F, t }

Pick F

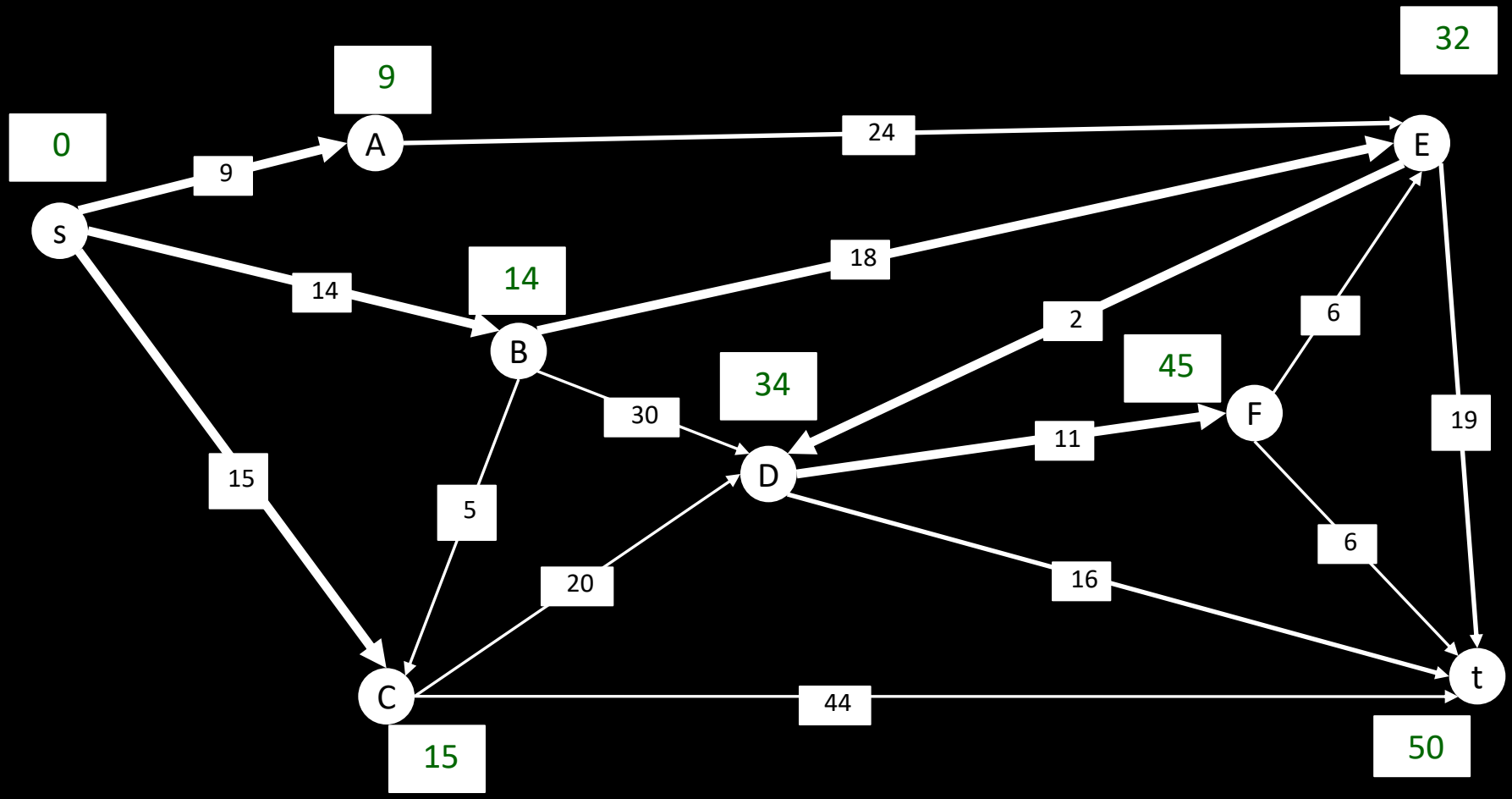


Dijkstra's Shortest Path Algorithm

It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbour, and updates the neighbour's distance if smaller.

Nodes = { t }

Pick t

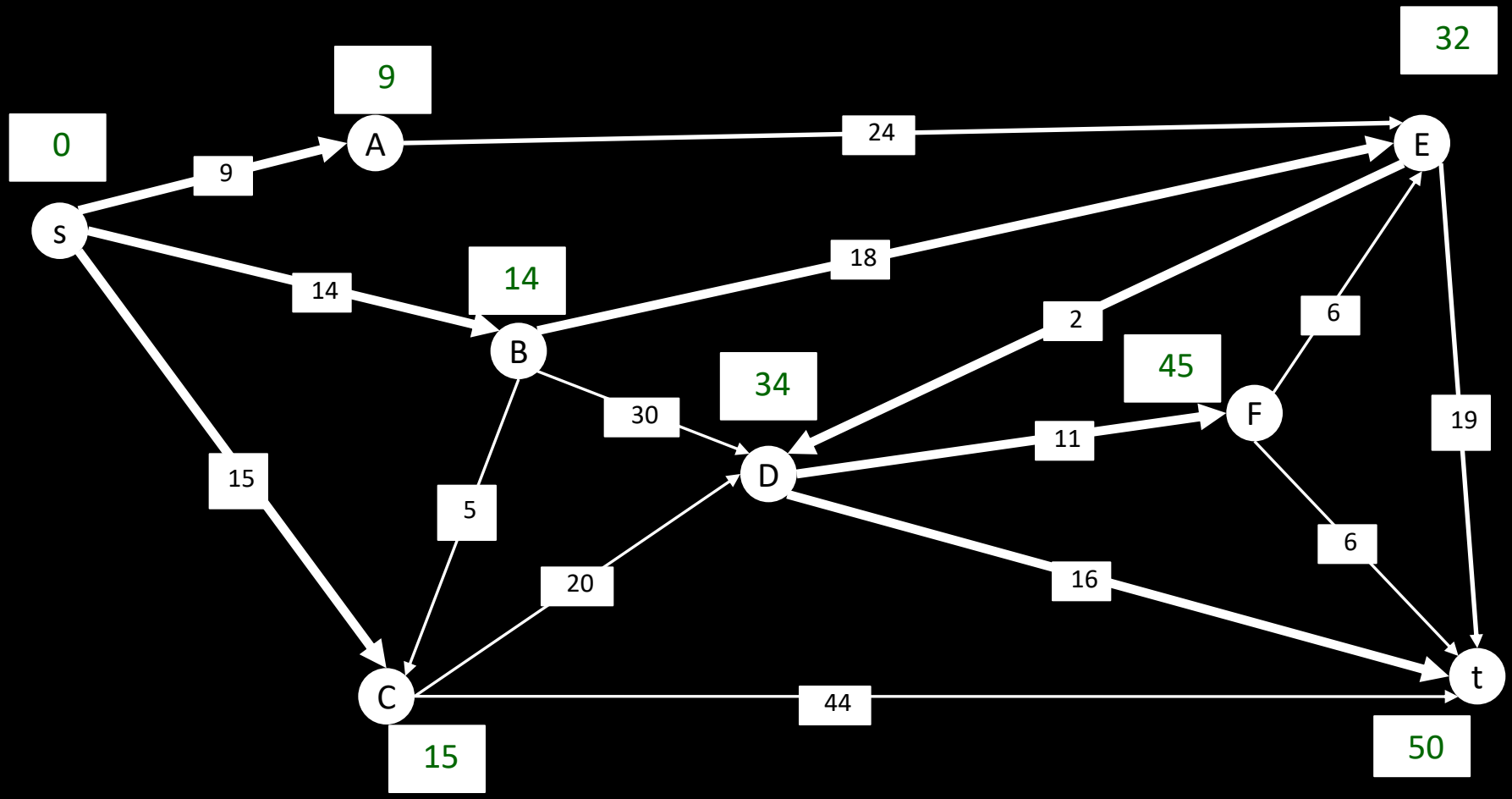


Dijkstra's Shortest Path Algorithm

It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbour, and updates the neighbour's distance if smaller.

Nodes = { t }

Pick t

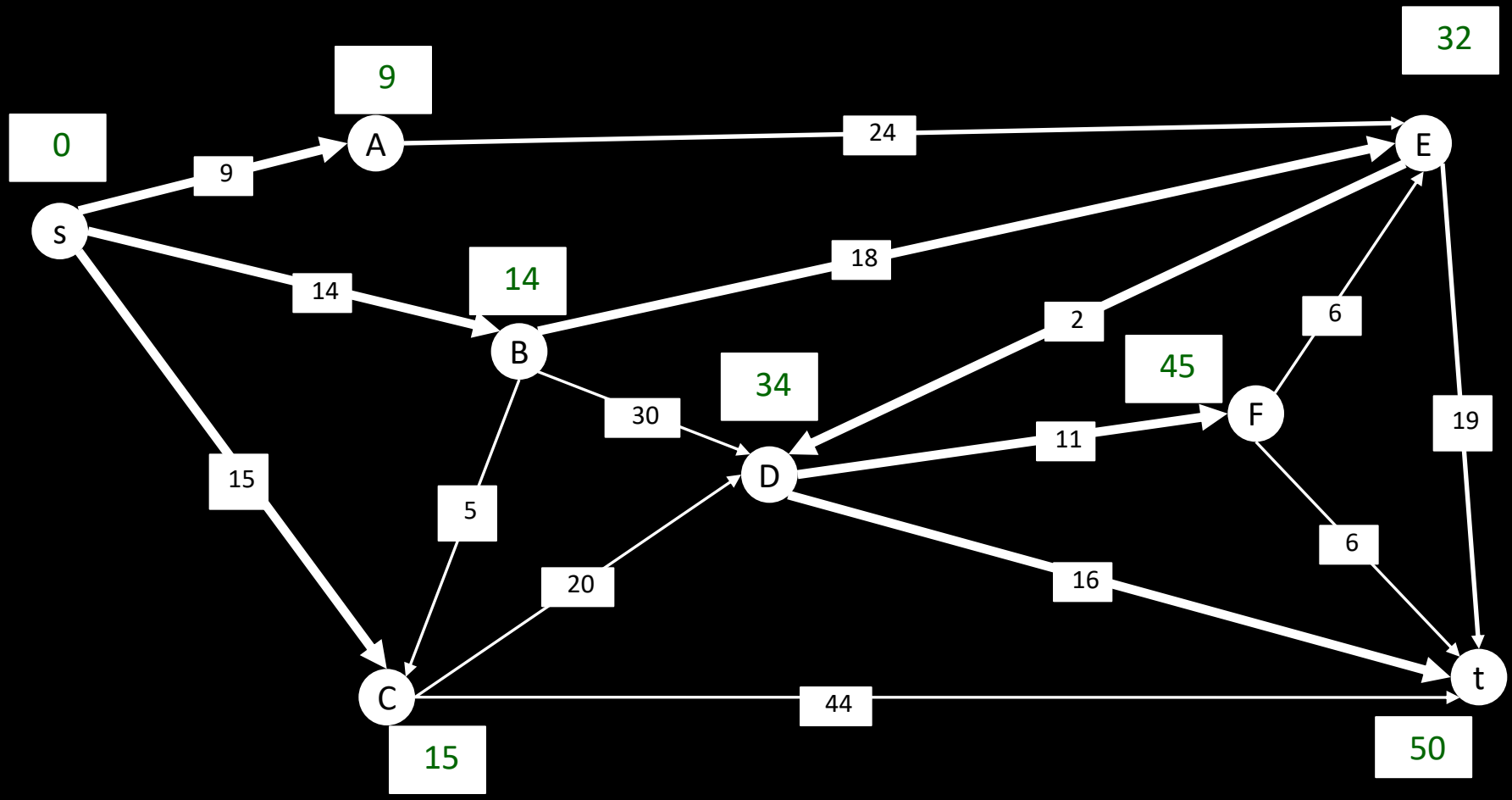


Dijkstra's Shortest Path Algorithm

It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbour, and updates the neighbour's distance if smaller.

Work backward to find the shortest path between s and t….

t -> D -> E -> B -> s



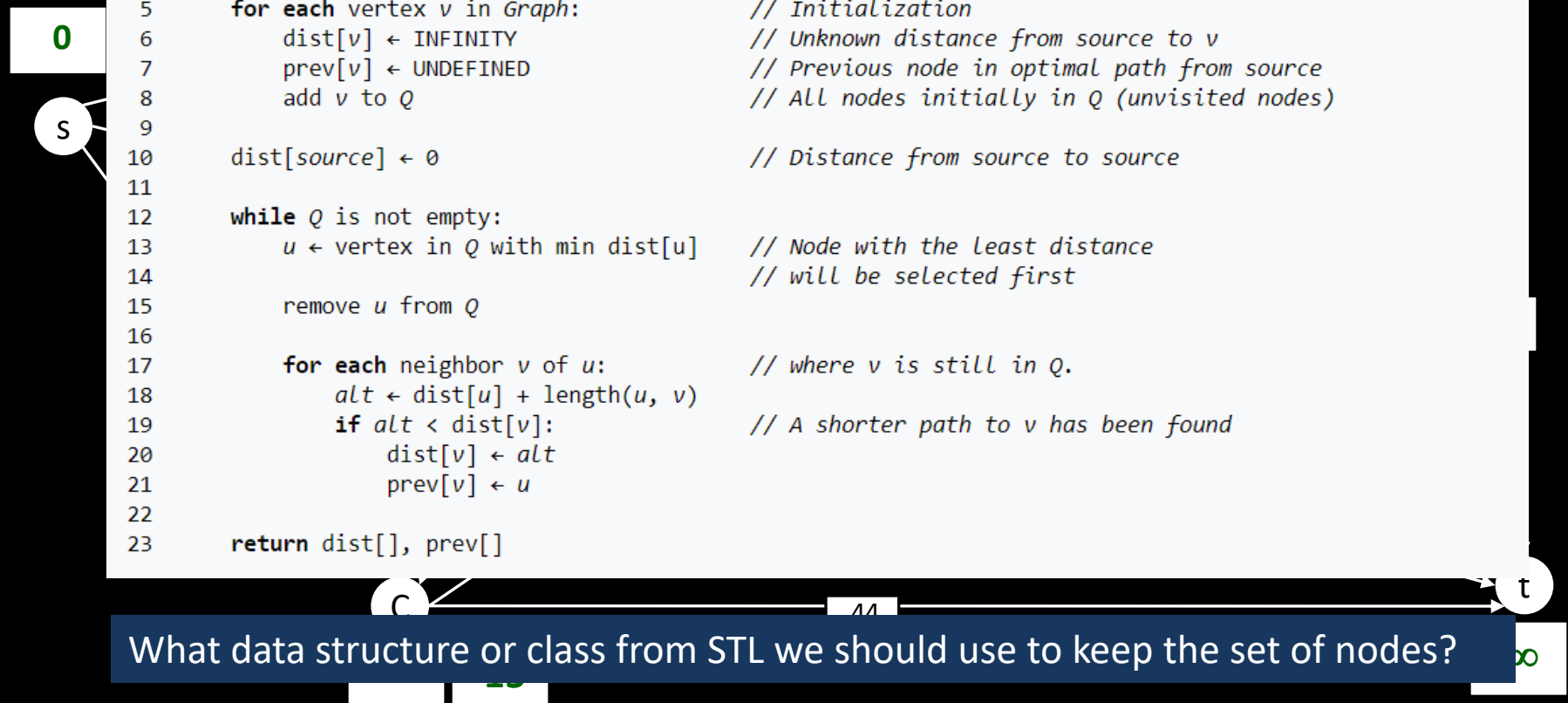
Dijkstra's Shortest Path Algorithm

It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbour, and updates the neighbour's distance if smaller.

Nodes = { s, A, B, C, D, E, F, t }

Pick s

```
1  function Dijkstra(Graph, source):
2
3      create vertex set Q
4
5      for each vertex v in Graph:            // Initialization
6          dist[v] ← INFINITY                 // Unknown distance from source to v
7          prev[v] ← UNDEFINED                // Previous node in optimal path from source
8          add v to Q                         // All nodes initially in Q (unvisited nodes)
9
10     dist[source] ← 0                       // Distance from source to source
11
12     while Q is not empty:
13         u ← vertex in Q with min dist[u]   // Node with the least distance
14                                            // will be selected first
15         remove u from Q
16
17         for each neighbor v of u:          // where v is still in Q.
18             alt ← dist[u] + length(u, v)
19             if alt < dist[v]:              // A shorter path to v has been found
20                 dist[v] ← alt
21                 prev[v] ← u
22
23     return dist[], prev[]
```



What data structure or class from STL we should use to keep the set of nodes?

Implementation of the Vessel Module

- Running time of the algorithm depends on the time used to
  - Pick the node with the minimal cost and remove it (Tp)
  - Update the cost (Tu)

- Total run time ~ |E| * Tu + |N| * Tp

- What data structure?
  - std::vector ?
  - std::map ?
  - ???

Implementation of the Vessel Module

Nodes = { s, A, B, C, D, E, F, t }

Pick s

- Total run time ~ |E| * Tu + |N| * Tp
- std::vector
- Tu = ?
  - Constant update time O(1)
- Tp = ?
  - Find the minimal cost by going through the list of nodes and remove it.
  - O(N)
- O(|E| + |N| * |N|)

Implementation of the Vessel Module

Nodes = { s, A, B, C, D, E, F, t }

Pick s

- Total run time ~ |E| * Tu + |N| * Tp
- std::map
  - Key as the cost, Value as the pointer to the node.

- Tu = ?
  - Doesn’t directly support updating the key
  - By adding and removing the element
  - O(log(N))

- Tp = ?
  - Find the minimal cost by getting the first element (already sorted) and remove it.
  - O(log(N))

- O(|E| * log |N| + |N| * log |N|) ~ O(|E| * log |N|)

Implementation of the Vessel Module

Nodes = { s, A, B, C, D, E, F, t }

Pick s

- Total run time ~ |E| * Tu + |N| * Tp
- std::priority_queue
- Store the cost and node in one class and define a function to compare between two objects of the new class.
- Same as std::map

Implementation of the Vessel Module

Nodes = { s, A, B, C, D, E, F, t }

Pick s

- Total run time ~ |E| * Tu + |N| * Tp
- Fibonacci heap
- Tu = ?
  - Amortized constant O(1)
- Tp = ?
  - Amortized O(log(N))
- |E| + |N| * log|N|

Implementation of the Vessel Module

- Total run time ~ |E| * Tu + |N| * Tp
- std::vector
  - |E| + |N| * |N|
- std::map
  - |E| * log|N|
- std::priority_queue (same as std::map)
  - |E| * log|N|
- Fibonacci Heap
  - |E| + |N| * log|N|

Implementation of the Vessel Module

```
typedef std::pair<float4, PositionPair> CostPositionPair;

// Set of the nodes .
 std::priority_queue<CostPositionPair, std::vector<CostPositionPair>, CostPositionPairComparitor>
mapBuckets;
// Push the source node.
mapBuckets.push(start_point);

while (mapBuckets.size() > 0)
{
  const auto& minNode = mapBuckets.top();
  // remove position from the buckets
  mapBuckets.pop();
  // Break if the minNode is the destination.
  if (minNode  == end_point)   break;

  // update costs of the 26 neighbours
  for (int i = 0; i < 26; ++i)
  {
    // …… if the new cost is smaller than the existing cost, update the cost.
   if (fTmp < vNeighValsCost[iIndex])
       mapBuckets.push(CostPositionPair(fTmp, PositionPair(uCellOff, uValOff)));
  }
}
```

Overview

- Use case
  - Efficient use of Computed Tomography (CT) angiography in Peripheral Artery Disease (PAD)
- Design of the Vessel Module
- Implementation
  - Vessel Centerline Extraction
- Testing and Reusability

Testing

- Cannot emphasis enough the importance of testing.
- Definitely not the last thing you do in the development.
- Different levels of testing
  - Unit testing
  - Functional testing
  - Acceptance testing
- On phantom data
- On different data

Testing

- Unit testing
  - Smallest testable part of an application, e.g. a function in some math class.
  - Lost of different frameworks, CTest, Cppunit (e.g. see https://stackoverflow.com/questions/91384/unit-testing-for-c-code-tools-and-methodology)

- Functional testing
  - The system is tested against the functional requirements or specifications, e.g. is the ruler measurement correct?
  - Use of phantom and different data.

- Acceptance testing
  - The system is tested for compliance with the business requirements and assess whether it is acceptable for delivery.
  - E.g. does it actually solve the problem of the end users?

Reusability

- Different levels of reusability
- Reusable functions and classes
- Reusable “software components”
- Needs some more domain knowledge.
- For example, other modules may use similar components…..

Reusability

- Different levels of reusability
- Reusable functions and classes
- Look at the bigger picture – reusable “software components”
- Needs some more domain knowledge.
- For example, other modules may use similar components…..

Reusability

- Horizontal CPR in dental application.
- The module and the whole CPR components should be reusable to allow to “plug together” another module.

Take Home Messages …

- Know the use cases and requirements
- Design patterns are used to help to build a solid foundation.
- STL provides a ranges of data structures and algorithms for efficient software development …
- … but, carefully pick the correct data structures for the job.
- Testing … Testing … Testing …

Questions?

P.9: Don't waste time or space

Example, bad

```
void lower(zstring s)
{
    for (int i = 0; i < strlen(s); ++i) s[i] = tolower(s[i]);
}
```