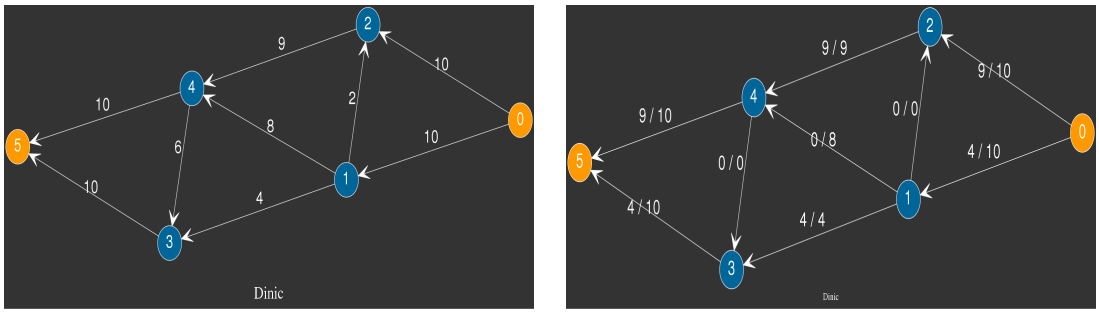
Visualization of Dinic Algorithm

Rahul Ramesh, Vignesh Manoharan, R.Gowrisankar

1 Problem Definition

The Dinic Algorithm aims to solve the commonly known maxflow problem. Maximum flow problem tries to find the largest flow from designated source to sink vertices in a flow network. A flow network is given by a directed graph, with the weights representing the maximum possible capacities along that edge.



(a) Graph showing capacities (b) Flow-Network

Figure 1.1: Example with Source-Sink flow = 13

A flow-network is a directed graph where the weights along each edge is sandwiched between zero and the maximum capacity. The value of this weight is known as the flow. The Dinic algorithm gives us a flow-network , such that the flow from the source to the sink is maximised. The same problem can be solved using many algorithms like the Ford-Fulkerson Algorithm, Edmonds-Karp Algorithm Push-Relabel Algorithm or the Dinic Algorithm. We will look at the Dinic Algorithm, and in particular the implementation that runs in $\mathcal{O}(V^2E)$ where V and E are the number of vertices and edges in the graph

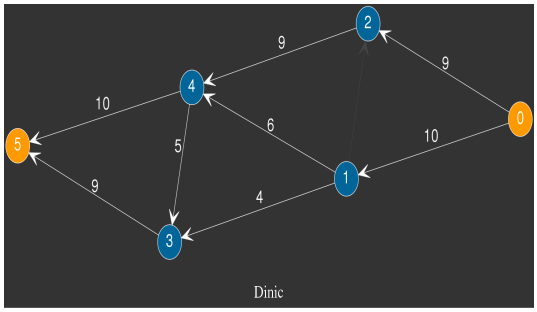


Figure 1.2: Flow-network with Max-flow value after Dinic

Before looking at the algorithm, we will define a few terms like Residual Graph, Level graph and Augmenting Path.

2 Definitions

2.1 Residual Graph

Consider two vertices V_1 and V_2. Let the flow from V_1 to V_2 to be F_{12} along the edge E_{12} and flow from V_2 to V_1 to be F_{21} along the edge E_{21}. If no edge exists from one vertex to another, then set the flow value to be 0. Also let C_{12} and C_{21} represent the maximum capacity of edges, E_{12} and E_{21}.
The Residual graph contains all the vertices present in the original graph. For each pair of vertices V_1' and V_2' in the residual graph, define the following:

$$\begin{aligned} F'_{12} &= (C_{12} - F_{12}) + C_{21} \\ F'_{21} &= (C_{21} - F_{21}) + C_{12} \end{aligned} \tag{2.1}$$

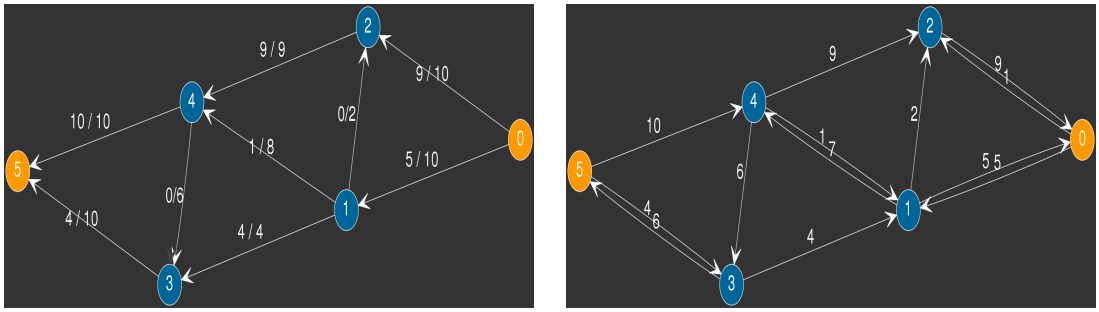
The residual graph in essence, builds a graph representing the maximum possible change in flow from V_1 to V_2 or, V_2 to V_1 constrained by the capacities. This graph is used as a tool to increase or decrease flow values along edges, in order to increase the overall source-sink flow. One can discard all edges with zero flow edges in the residual graph.

2.2 Level Graph

The Level graph is a subgraph of the residual graph. We first partition the vertices of the residual graph into various levels. A vertex is said to belong to level i if the shortest distance from the source vertex is i. After dividing the vertices into levels, retain only those edges, which go from a level i to level $i + 1$.

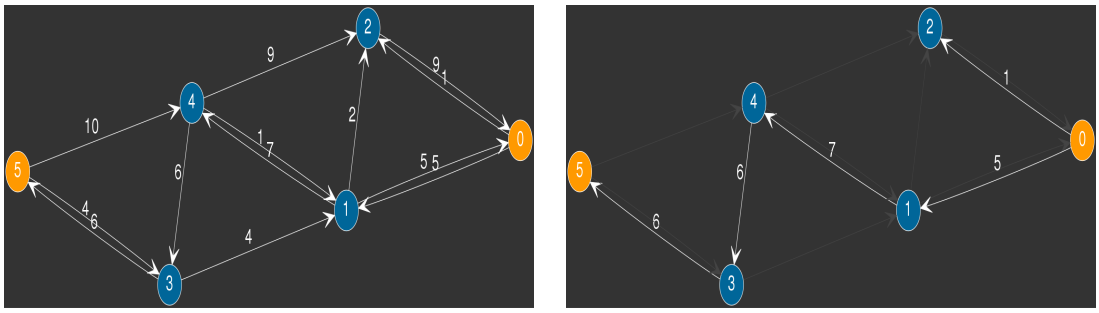
2.3 Augmenting path

Given a flow network and its corresponding maximum capacities, an augmenting path is a source-sink path, in which all the edges have flow strictly less than the maximum



(a) Flow-network (b) Residual graph for Flow-network

Figure 2.1: Example for Residual graph



(a) Residual Graph (b) Level-graph from Residual graph

Figure 2.2: Example for Level graph

capacity of the edge. Augmenting paths can be used to increase the source-sink flow. If we find an augmenting path, then we can increase the flow along that path by Min(C_{uv} - F_{uv}) for all edges E_{uv} such that E_{uv} belongs to the Augmenting Path. Note F_{uv} and C_{uv} denote the flow and capacity values. Figure *2.3* highlights an augmenting path, and the flow along the path can be increased by 4.

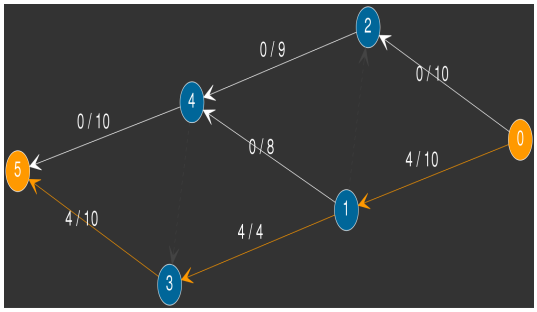


Figure 2.3: Augmenting Path denoted in Orange

2.4 Blocking Flow

A blocking flow is a flow network where all possible source-sink paths have atleast one saturated edge i.e. $C_{uv} = F_{uv}$ for atleast one edge E_{uv}, along every source-sink path. Alternatively, the graph has no augmenting path. A blocking flow need not be a maxflow, but the converse is true.

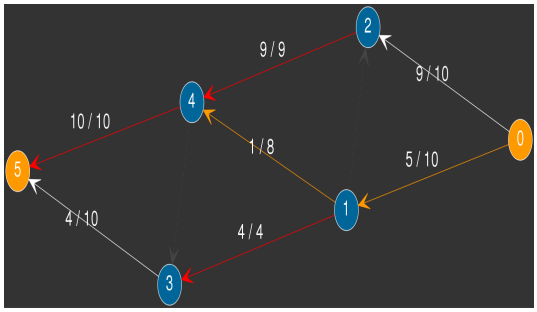


Figure 2.4: Blocking Flow with Red, denoting that the edge is saturated

3 Algorithm

3.1 Psuedo Code

Algorithm 1 Dinic Algorithm

Data: Graph capacities
Result: Maxlflow network and maxflow value
Initialization : Set the flow network values to be zero for all edges
for *(i=1 ; i<=V ; i++)* **do**
    G' = Level (Residual(G))
    G' = BlockingFlow (G')
    Increment edges in G with values from G'
end
return G

As mentioned earlier, the runtime of the algorithm is $\mathcal{O}(V^2E)$. Finding the level graph or residual graph takes $\mathcal{O}(E)$ since, it suffices to iterate over all edges in the graph. A blocking flow can be found in $\mathcal{O}(VE)$. Combining these two results, we get the required complexity of the algorithm.
The blocking flow subroutine calls a modified DFS E times.

$$T(BlockingFlow) = E \times T(DFSRoutine) \tag{3.1}$$

Each DFS routine takes $\mathcal{O}(V)$ time to find a source-sink path in each iteration. Additionally, over all the E iterations in the Blocking Flow routine, the DFS routine can delete atmost of E edges.

Note that :
T(DFS path finding) = $\mathcal{O}(V)$
T(DFS Deletion over E function calls) = $\mathcal{O}(E)$

$$T(BlockingFlow) = E\times \text{ T(DFS path finding) + T(DFS Deletion)} \tag{3.2}$$

$$\implies \text{T(Blocking Flow)} = \mathcal{O}(VE) \tag{3.3}$$

Algorithm 2 Blocking Flow

Data: Graph capacities
Result: Blocking Flow of the Graph
Init : Set the flow network values to be zero for all edges and Path_Exists = Yes
while *Path_Exists=Yes* **do**
  Path = Modified DFS(G)
  `/* Find Augmenting Path */`
  if *No Path found* **then**
    Path_Exists = No
  else
    Increase flow along Augmenting Path in G
  end
end
return G

Algorithm 3 Modified DFS

Data: Flow Network
Result: Augmenting Path
Do DFS from source
Stop DFS if sink is reached
Delete Vertex is no source-sink path possible from Vertex
if *Sink encountered* **then**
  Augmenting_Path = Path found from DFS
else
  Augmenting_path = None
  `/* If Sink is never encountered then no Source-Sink path exists */`
end
return Augmenting_path

3.2 Correctness

3.2.1 Termination

Firstly, we shall prove that the algorithm always terminates. Each iteration in the Dinic function always, increases the source-sink distance in the level Graph. If this is true,since, the source-sink distance is less than V, termination is ensured
Using proof by contradiction, assume that the source-sink distance does not change, after updating the graph with a blocking flow. Path present in the previous iteration cannot exist in current iteration since,

$\exists$ E_{uv}, such that $F_{uv} = C_{uv}$,

Hence this edge will be removed in the Residual graph and hence the path cannot be present in the next iteration. Hence new edge must be present in the path. Any new edge must be from Level 'i' to Level 'j' where 'i'>=j'. Note that the edges are with respect to the previous iteration. No edges can go to higher levels, by virtue of construction of the level graph. Hence since edges, are constrained by being from Level 'i' to Level 'j' where 'i'>=j', there can exist no source-sink path that is of same length as previous iteration. Hence we see that the algorithm terminates.

3.2.2 Correct Result

We shall first try to prove the following result. Let F be a flow-network. Also let G' = Level(Residual(G)). Then the following three statements are equivalent :

1. F is a maxflow
2. There is no augmenting path Graph G'
3. $\exists$ X,X' $\subset$ V where X $\cup$ X' = V , X $\cup$ X' = $\emptyset$ and |flow_val(X,X')| = capacity(X,X')

|Flow_val(X,X')| refers to the total flow from X to X'. With C_{uv} representing the capacity of the edge we define capacity(X,X') as :

$$\text{capacity(X,X')} = \sum_{\forall u,v, u \in X, v \in X'} C_{uv}$$

Proving 1 $\implies$ 2 is trivial, since if there existed an augmenting path, then it is possible to increase the flow along that path. Hence F will not be a maxflow.
Proving 3 $\implies$ 1 is also an easy task. We see that the |Flow_val(X,X')| $\leq$ Capacity(X,X'). Also |Flow_val(X,X')| is also equal to the source-sink flow. Hence, when both the quantities are equal, The source-sink flow must be a maximum and hence it is a maxflow.

To Prove 2 $\implies$ 3, we know that there exists no Augmenting path in G'. Hence there exists no source-sink path in the Graph G'. Let X be set of all vertices that can be

reached from the source and let the rest of the vertices belong to X'. Since X and X' have no edges between them in G', all edges in G from X to X' must be saturated. Hence |flow_val(X,X')| = capacity(X,X') for the selected X and X' proving that 2 $\implies$ 3.

Hence the above statements are proved. We already know that the algorithm terminates. When it terminates, the resultant graph has no source-sink path since , the Dinic routine loops V times and each iteration increases the Source-Sink distance by 1(Source-Sink distance $\leq$ V-1). Hence there exists no augmenting path on termination. Hence from the earlier stated theorem, the flow is a maxflow. Hence this concludes the algorithm correctness.