Advanced Algorithms & Data Structures

Department of CSE

ADVANCED ALGORITHMS AND DATA STRUCTURES
23CS03HF

Topic:

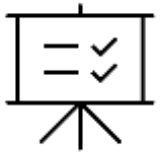
Edmonds-Karp Algoirthm

Session - 27

AIM OF THE SESSION

To familiarize students with the concept of the Network flow algorithms.

INSTRUCTIONAL OBJECTIVES



This Session is designed to:

1. Demonstrate :- Edmonds-Karp Algorithm.
2. Describe :- Network flow using Edmonds-karp algorithm

LEARNING OUTCOMES

At the end of this session, you should be able to:

1. Define :- Edmonds-Karp Algorithm.
2. Describe :- Network flow using Edmonds-karp algorithm
3. Summarize:- Finding maximum flow in a network.

Edmonds-Karp Algorithm

- The Edmonds-Karp algorithm solves the maximum flow problem for a directed graph.
- The flow comes from a source vertex (ss) and ends up in a sink vertex (tt), and each edge in the graph allows a flow, limited by a capacity.
- The Edmonds-Karp algorithm is very similar to the Ford-Fulkerson algorithm, except the Edmonds-Karp algorithm uses Breadth First Search (BFS) to find augmented paths to increase flow.

- The Edmonds-Karp algorithm works by using Breadth-First Search (BFS) to find a path with available capacity from the source to the sink (called an *augmented path*), and then sends as much flow as possible through that path.

How it works:

1.Start with zero flow on all edges.

2.Use BFS to find an *augmented path* where more flow can be sent.

3.Do a *bottleneck calculation* to find out how much flow can be sent through that augmented path.

4.Increase the flow found from the bottleneck calculation for each edge in the augmented path.

5.Repeat steps 2-4 until max flow is found. This happens when a new augmented path can no longer be found.

Residual Network in Edmonds-Karp

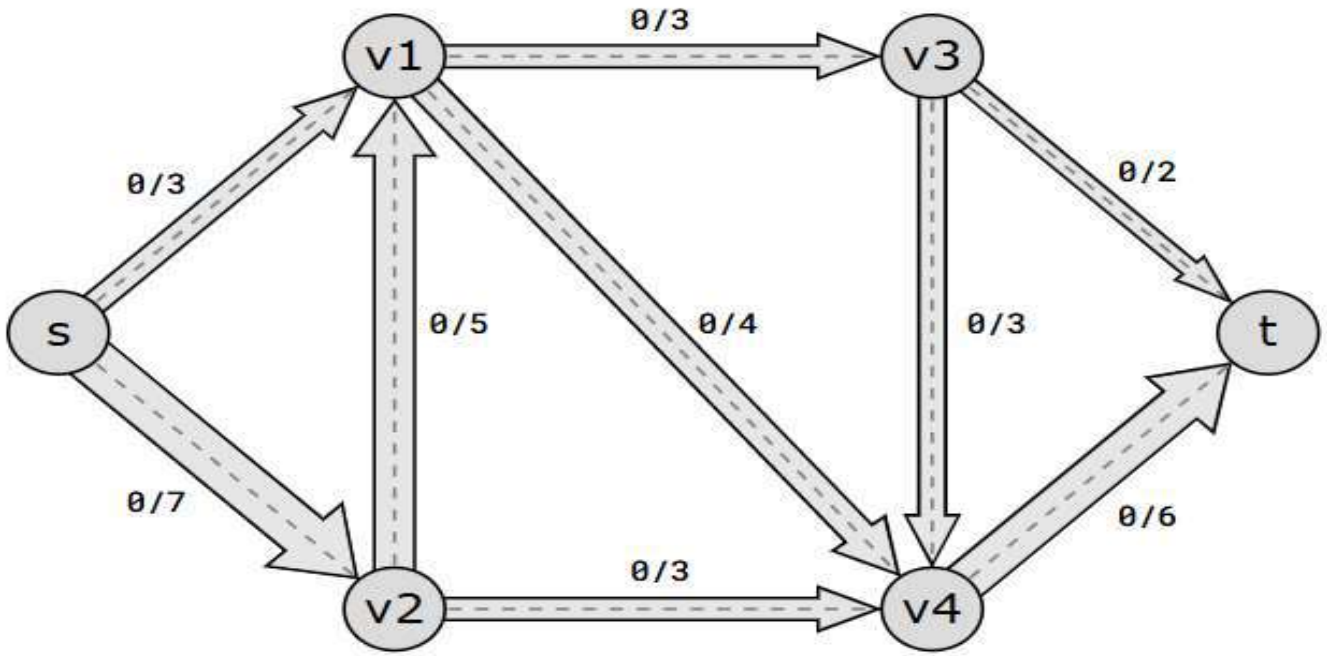
- The Edmonds-Karp algorithm works by creating and using something called a *residual network*, which is a representation of the original graph.

- In the residual network, every edge has a *residual capacity*, which is the original capacity of the edge, minus the the flow in that edge. The residual capacity can be seen as the leftover capacity in an edge with some flow.

- For example, if there is a flow of 2 in the v3→v4v3→v4 edge, and the capacity is 3, the residual flow is 1 in that edge, because there is room for sending 1 more unit of flow through that edge.

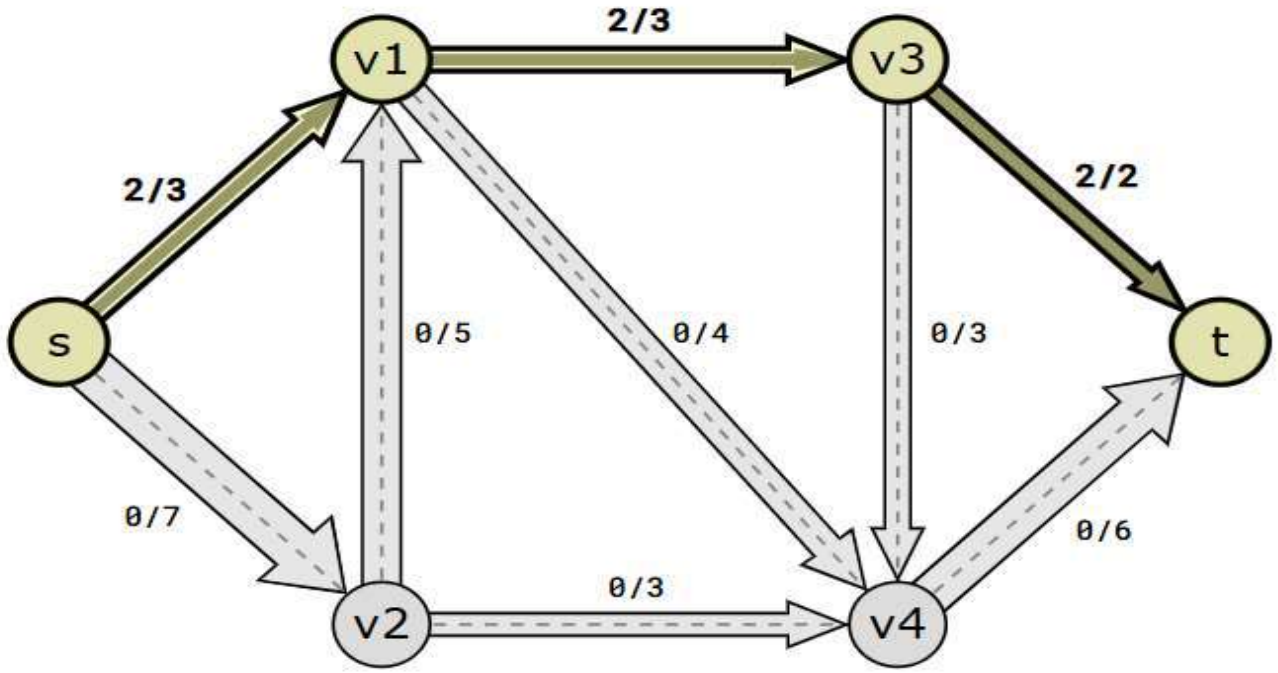
Reversed Edges in Edmonds-Karp

- The Edmonds-Karp algorithm also uses something called *reversed edges* to send flow back. This is useful to increase the total flow.
- To send flow back, in the opposite direction of the edge, a reverse edge is created for each original edge in the network. The Edmonds-Karp algorithm can then use these reverse edges to send flow in the reverse direction.
- A reversed edge has no flow or capacity, just residual capacity. The residual capacity for a reversed edge is always the same as the flow in the corresponding original edge.
- In our example, the edge v1→v3 has a flow of 2, which means there is a residual capacity of 2 on the corresponding reversed edge v3→v1.

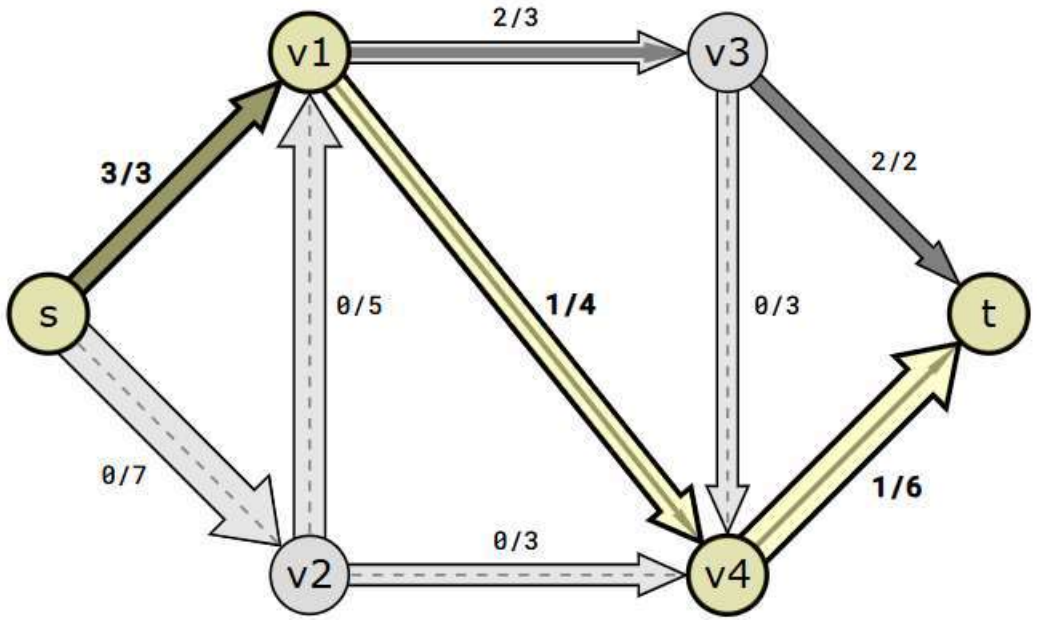


Max flow: 0

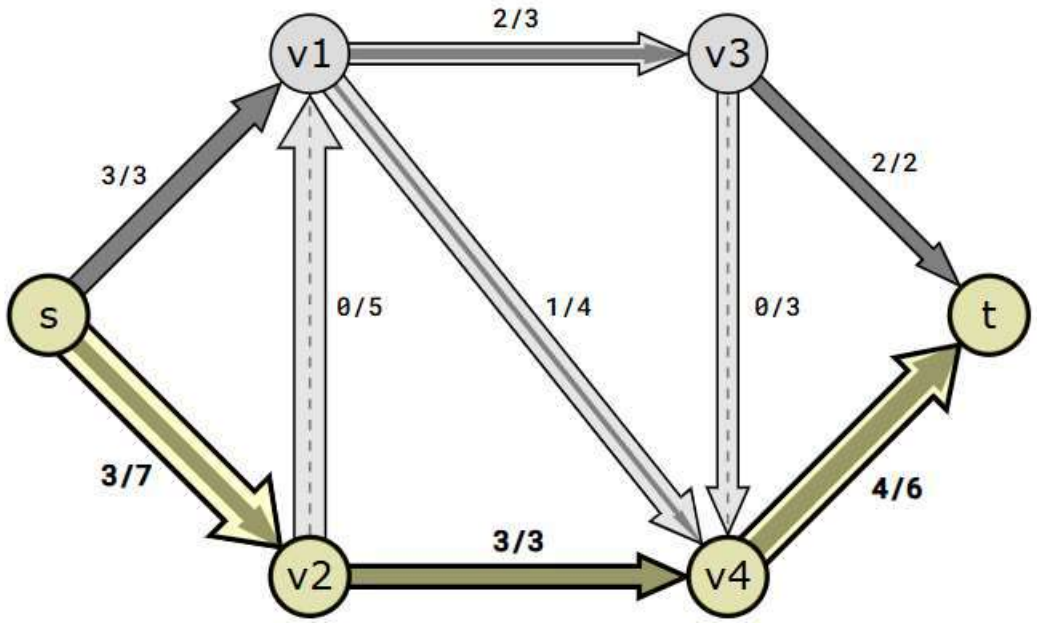
- The Edmonds-Karp algorithm starts with using Breadth-First Search to find an augmented path where flow can be increased, which is s→v1→v3→t.
- After finding the augmented path, a bottleneck calculation is done to find how much flow can be sent through that path, and that flow is: 2.
- So a flow of 2 is sent over each edge in the augmented path.



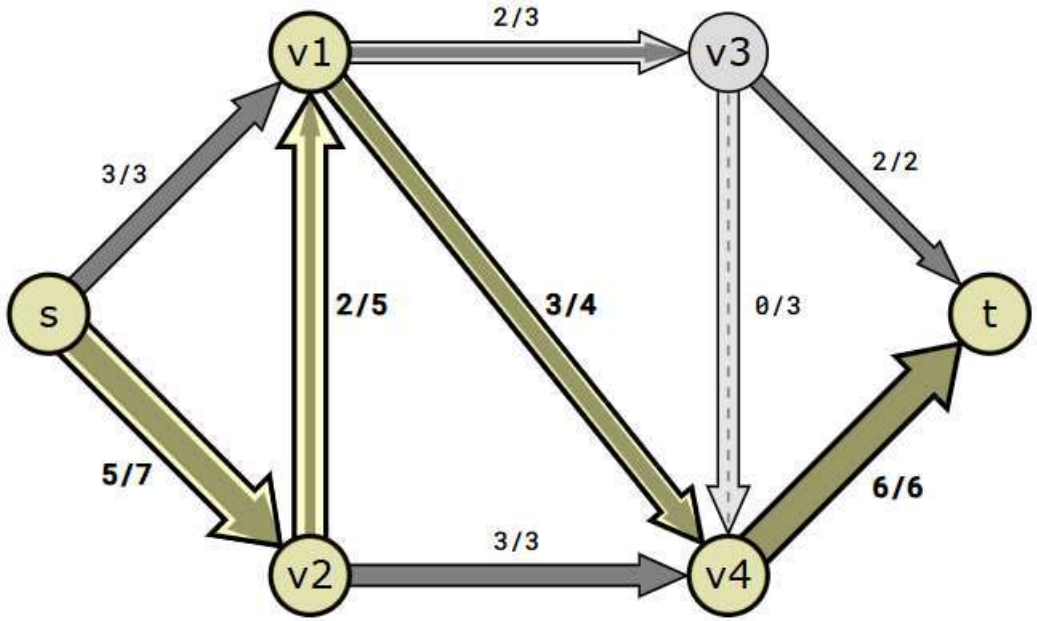
- The next augmented path is found to be s→v1→v4→t.
- The flow can only be increased by 1 in this path because there is only room for one more unit of flow in the s→v1 edge.



- The next augmented path is found to be s→v2→v4→t.
- The flow can be increased by 3 in this path. The bottleneck (limiting edge) is v2→v4 because the capacity is 3.



- The last augmented path found is s→v2→v1→v4→t.
- The flow can only be increased by 2 in this path because of edge v4→t being the bottleneck in this path with only space for 2 more units of flow (capacity−flow=1).



- At this point, a new augmenting path cannot be found (it is not possible to find a path where more flow can be sent through from s to t), which means the max flow has been found, and the Edmonds-Karp algorithm is finished.
- The maximum flow is 8. As you can see in the image above, the flow (8) is the same going out of the source vertex ss, as the flow going into the sink vertex tt.

SUMMARY

- The Edmonds-Karp algorithm computes the maximum flow in a network by using the Ford-Fulkerson method with BFS to find the shortest augmenting paths. It updates the flow and residual graph iteratively, augmenting flow along the path with the smallest residual capacity. The algorithm has a time complexity of $O(VE^2)$

SELF-ASSESSMENT QUESTIONS

What happens when no augmenting paths are found in the residual graph?

A. The algorithm continues searching for alternative paths.

B. The algorithm terminates, and the maximum flow is achieved.

C. The algorithm retries with a different source.

D. The algorithm reduces the flow value and retries.

What does the Edmonds-Karp algorithm use to find augmenting paths?

A. Depth-First Search (DFS)

B. Breadth-First Search (BFS)

C. Dijkstra's Algorithm

D. Prim's Algorithm

TERMINAL QUESTIONS

1. Compare the Edmonds-Karp algorithm with the basic Ford-Fulkerson method.
2. Discuss the time complexity of the Edmonds-Karp algorithm.
3. What are the real-world applications of the Edmonds-Karp algorithm?

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books :

1Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein., 3rd, 2009, The MIT Press.

2 Algorithm Design Manual, Steven S. Skiena., 2nd, 2008, Springer.

3 Data Structures and Algorithms in Python, Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser., 2nd, 2013, Wiley.

4 The Art of Computer Programming, Donald E. Knuth, 3rd, 1997, Addison-Wesley Professiona.

MOOCS :

1. https://www.coursera.org/specializations/algorithms?=

2.https://www.coursera.org/learn/dynamic-programming-greedy-algorithms#modules

THANK YOU

THANK YOU