**Mangesh Raut**

**Professor Mark Boady**

**CS 502 Data Structures and Algorithms**

**March 3rd, 2022**

**Edmonds-Karp Algorithm**

**Introduction**

Edmonds-Karp algorithm is just an implementation of the Ford-Fulkerson method that uses BFS (Breadth-first search) for finding augmenting paths. Breadth-first search is an algorithm for searching a tree data structure for a node that satisfies a given property. It starts at the tree root and explores all nodes at the present depth prior to moving on to the nodes at the next depth level. the maximum amount of flow that the network would allow to flow from source to sink. Multiple algorithms exist in solving the maximum flow problem. For example, a company might want to ship packages from Los Angeles to New York City using trucks to transport between intermediate cities. If there is only one truck for the route connecting a pair of cities and each truck has a maximum load, then the graph describing the transportation options will be a flow network. The complexity can be given independently of the maximal flow. Finding the maximum flow for a network was first solved by the Ford-Fulkerson algorithm. A network is often abstractly defined as a graph, G, that has a set of vertices, V, connected by a set of edges, E. There is a source, s, and a sink, t, which represent where the flow is coming from and where it is going to. Finding the maximum flow through a network was solved via the max-flow min-cut theorem, which then was used to create the Ford-Fulkerson algorithm.

Edmonds-Karp is identical to Ford-Fulkerson except for one very important trait. The search order of augmenting paths is well defined. As a refresher from the Ford-Fulkerson wiki, augmenting paths, along with residual graphs, are the two important concepts to understand when finding the max flow of a network. Augmenting paths are simply any path from the source to the sink that can currently take more flow. Over the course of the algorithm, flow is monotonically increased. So, there are times when a path from the source to the sink can take on more flow, and that is an augmenting path.

**History**

In computer science, the Edmonds–Karp algorithm is an implementation of the Ford–Fulkerson method for computing the maximum flow in a flow network in O(|V||E|2) time. Whereas V = number of vertexes and E = number of edges. The algorithm was first published by Yefim Dinitz in 1970 and independently published by Jack Edmonds and Richard Karp in 1972. Jack R. Edmonds (born April 5, 1934) is an American-born and educated computer scientist and mathematician who lived and worked in Canada for much of his life. He has made fundamental contributions to the fields of combinatorial optimization, polyhedral combinatorics, discrete mathematics, and the theory of computing. He was the recipient of the 1985 John von Neumann Theory Prize. Edmonds is well known for his theorems on max-weight branching algorithms and packing edge-disjoint branching’s and his work with Richard Karp on faster flow algorithms. The Edmonds–Gallai decomposition theorem describes finite graphs from the point of view of matchings. He introduced polymatroids, submodular flows with Richard Giles, and the terms clutter and blocker in the study of hypergraphs.

In 1971 Richard M. Karp co-developed with Jack Edmonds the Edmonds–Karp algorithm for solving the maximum flow problem on networks, and in 1972 he published a landmark paper in complexity theory, "Reducibility Among Combinatorial Problems", in which he proved 21 problems to be NP-complete.

**Why Edmonds-Karp Algorithm Invented:**

Edmonds-Karp algorithm is just an implementation of the Ford-Fulkerson method that uses BFS for finding augmenting paths. The Ford-Fulkerson algorithm is used to detect maximum flow from start vertex to sink vertex in a given graph. In this graph, every edge has the capacity. Two vertices are provided named Source and Sink. The source vertex has all outward edge, no inward edge, and the sink will have all inward edge no outward edge.

**Algorithm**

The algorithm is identical to the Ford–Fulkerson algorithm, except that the search order when finding the augmenting path is defined. The path found must be the shortest path that has available capacity. This can be found by a breadth-first search, where we apply a weight of 1 to each edge. The running time of O(|V| |E|2) is found by showing that each augmenting path can be found in O(|E|) time, that every time at least one of the E edges becomes saturated (an edge which has the maximum possible flow), that the distance from the saturated edge to the source along the augmenting path must be longer than last time it was saturated, and that the length is at most |V|. Another property of this algorithm is that the length of the shortest augmenting path increases monotonically. There is an accessible proof in Introduction to Algorithms.

The Edmonds-Karp Algorithm is a specific implementation of the Ford-Fulkerson method. In particular, Edmonds-Karp algorithm implements the searching for an augmenting path using the Breadth First Search (BFS) algorithm. Other implementations of the Ford-Fulkerson method use the Depth First Search (DFS) algorithm to find augmenting paths. This algorithm code starts with the maximum flow initially set to 0. The while loop executes until there are no more augmenting paths. Within the while loop, we call BFS to find the shortest path from source to sink and the minimum residual capacity along that path, min. We then walk the augmenting path from target to source. Using the minimum residual capacity, we reduce all residual capacities on the augmenting path by min and increase the residual capacities on the reverse edges (representing the flow).

**Pseudocode**

inputs

C[n x n] : Capacity Matrix

E[n x n] : Adjacency Matrix

s : source

t : sink

output

f : maximum flow

Edmonds-Karp:

f = 0 // Flow is initially 0

F = [n x n] // residual capacity array

while true:

m, P = Breadth-First-Search(C, E, s, t, F)

if m = 0:

break

f = f + m

v = t

while v != s:

u = P[v]

F[u, v] = F[u, v] - m //This is reducing the residual capacity of the augmenting path

F[v, u] = F[v, u] + m //This is increasing the residual capacity of the reverse edges

v = u

return f

In this pseudo-code, the flow is initially zero and the initial residual capacity array is all zeroes. Then, the outer loop executes until there are no more paths from the source to the sink in the residual graph.

Inside this loop, we are performing breadth-first search to find the shortest path from the source to the sink that has available capacity. This small change to BFS is quite easy because we are using the residual capacity matrix F which basically tells us whether there is available capacity on a given edge.

Once we have found a path with residual capacity, m, we add that capacity to our current maximum flow. Then, the code backtracks through the network, starting with the sink t. As it backtracks, it finds the parent of the current vertex which is defined as u. Then, the code updates the residual flow matrix F to reflect the newly found augmenting path capacity m. v is then set to be the parent, and the inner loop is repeated until it reaches the source vertex.

**Maximum flow - Ford-Fulkerson and Edmonds-Karp**

The Edmonds-Karp algorithm is an implementation of the Ford-Fulkerson method for computing a maximal flow in a flow network.

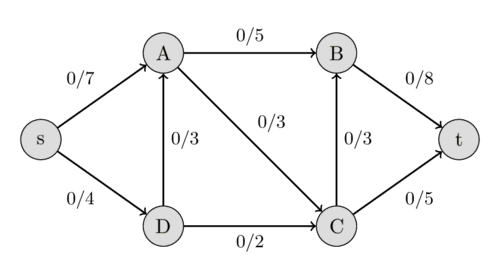
**Flow network**

First let's define what a flow network, a flow, and a maximum flow is. A network is a directed graph G with vertices V and edges E combined with a function c, which assigns each edge e ∈ E a non-negative integer value, the capacity of e. Such a network is called a flow network, if we additionally label two vertices, one as source and one as sink.

We represent edges as water pipes, the capacity of an edge is the maximal amount of water that can flow through the pipe per second, and the flow of an edge is the amount of water that currently flows through the pipe per second. This motivates the first flow condition.

There cannot flow more water through a pipe than its capacity. The vertices act as junctions, where water comes out of some pipes, and distributes it in some way to other pipes. This also motivates the second flow condition. In each junction all the incoming water has to be distributed to the other pipes. It cannot magically disappear or appear. The source s is origin of all the water, and the water can only drain in the sink t.

The following image shows a flow network. The first value of each edge represents the flow, which is initially 0, and the second value represents the capacity.

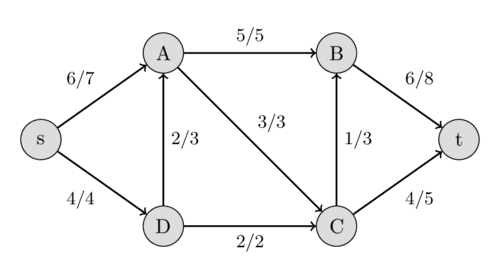


Maximum flow - Ford-Fulkerson and Edmonds-Karp

The value of a flow of a network is the sum of all flows that gets produced in source s, or equivalently of the flows that are consumed in the sink t. A maximal flow is a flow with the maximal possible value. Finding this maximal flow of a flow network is the problem that we want to solve.

In the visualization with water pipes, the problem can be formulated in the following way: how much water can we push through the pipes from the source to the sink.

The following image shows the maximal flow in the flow network.



Maximum flow - Ford-Fulkerson and Edmonds-Karp

**Example**:

Here is an example to demonstrate the method. We use the same flow network as above. Initially we start with a flow of 0. Given a graph which represents a flow network where every edge has a capacity. Also given two vertices source ‘s’ and sink ‘t’ in the graph, find the maximum possible flow from s to t with following constraints:

a) Flow on an edge doesn’t exceed the given capacity of the edge.

b) Incoming flow is equal to outgoing flow for every vertex except s and t.

Ford-Fulkerson Algorithm is called Edmonds-Karp Algorithm.

The following is simple idea of Ford-Fulkerson algorithm:

1) Start with initial flow as 0.

2) While there is an augmenting path from source to sink.

Add this path-flow to flow.

3) Return flow.

Diagram, engineering drawing

Description automatically generated

Now it is impossible to find an augmenting path between s and t, therefore the maximum possible flow in the above graph is 10. We have found the maximal flow.

It should be noted that the Ford-Fulkerson method doesn't specify a method of finding the augmenting path. Possible approaches are using DFS or BFS which both work in O(E).

If all capacities of the network are integers, then for each augmenting path the flow of the network increases by at least 1 (for more details see Integral flow theorem). Therefore, the complexity of Ford-Fulkerson is O(EF), where F is the maximal flow of the network. In case of rational capacities, the algorithm will also terminate, but the complexity is not bounded. In case of irrational capacities, the algorithm might never terminate, and might not even converge to the maximal flow.

**Applications**

This is an important problem as it arises in many practical situations. Examples include, maximizing the transportation with given traffic limits, maximizing packet flow in computer networks. Some applications where it is desirable to find the maximum flow through a network are:

* Modeling traffic in a road system
* Fluids in pipes
* Currents flowing through an electrical circuit

**Complexity**

The Edmonds-Karp algorithm runs in O(VE2). In each iteration of the algorithm, the shortest path (BFS) between the source and all other vertices must increase monotonically. We need to prove that one iteration of the Edmonds-Karp algorithm is bounded by O(E). We then need to prove that the number of iterations of the algorithm to find the maximum flow of a network is bounded by O(VE) iterations. Proving these two parts implies that the Edmonds-Karp algorithm is bounded by O(VE2). The shortest path increases monotonically in the residual graph. Therefore, the length of one iteration is bounded in the Edmonds-Karp algorithm to O(E).

The intermediate vertices on the shortest path from s to u cannot contain vertices s, u, or t. Therefore, the distance to vertex u is at most |V|−2. Since the distance from the source increases by at least 2 every time an edge becomes critical, the edge (u, v) can become critical at most |V|−2 / 2 times. An edge can become critical at most O(|V|) times. Now, there are a total of O(|E|) edges. Therefore, the number of iterations that the Edmonds-Karp algorithm must go through is O(|V||E|). Whereas V = number of vertexes and E = number of edges.

**Conclusion**

We have presented the Edmonds-Karp algorithm. Starting with a proof of the Ford-Fulkerson theorem, we have verified the generic Ford-Fulkerson method, specialized it to the Edmonds- Karp algorithm, and proved the upper bound O(V E) for the number of outer loop iterations. We then conducted several refinement steps to derive an efficiently executable implementation of the algorithm, including a verified breadth first search algorithm to obtain shortest augmenting paths. Finally, we added a verified algorithm to check whether the input is a valid network, and generated executable code in java. The runtime of our verified implementation compares well to that of an unverified reference implementation in Java.

**References**

Wikipedia February 05, 2022 Edmonds–Karp algorithm Retrieved from URL

<https://en.wikipedia.org/wiki/Edmonds%E2%80%93Karp_algorithm>

Flow Network Theory using Edmonds-Karp Algorithm February 13, 2022 The Edmonds-Karp Algorithm Retrieved from URL

<https://jamieheller.github.io/theory.html#complexity>

Maximum flow - Ford-Fulkerson and Edmonds-Karp - Algorithms for Competitive Programming February 13, 2022 Maximum flow - Ford-Fulkerson and Edmonds-Karp Retrieved from URL <https://cp-algorithms.com/graph/edmonds_karp.html>

Brilliant Math & Science Wiki February 14, 2022 Edmonds-Karp Algorithm Retrieved from URL <https://brilliant.org/wiki/edmonds-karp-algorithm/>

GeeksforGeeks November 10, 2021 Ford-Fulkerson Algorithm for Maximum Flow Problem Retrieved from URL

<https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/>