Edmonds–Karp algorithm

In computer science and graph theory, 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. It is asymptotically slower than the relabel-to-front algorithm, which runs in $O(V^3)$ time, but it is often faster in practice for sparse graphs. The algorithm was first published by Yefim (Chaim) Dinic in 1970 and independently published by Jack Edmonds and Richard Karp in 1972. Dinic's algorithm includes additional techniques that reduce the running time to $O(V^2E)$.

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 a shortest path that has available capacity. This can be found by a breadth-first search, as we let edges have unit length. 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, 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*.

Pseudocode

For a more high level description, see Ford–Fulkerson algorithm.

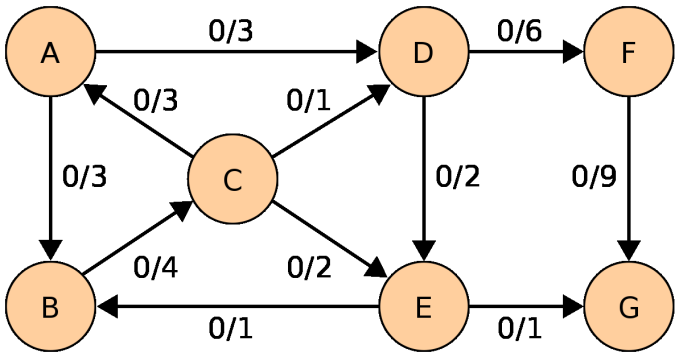
```
algorithm EdmondsKarp
    input:
        C[1..n, 1..n] (Capacity matrix)
        E[1..n, 1..?] (Neighbour lists)
        s             (Source)
        t             (Sink)
    output:
        f             (Value of maximum flow)
        F             (A matrix giving a legal flow with the maximum value)
    f := 0 (Initial flow is zero)
    F := array(1..n, 1..n) (Residual capacity from u to v is C[u,v] - F[u,v])
    forever
        m, P := BreadthFirstSearch(C, E, s, t, F)
        if m = 0
            break
        f := f + m
        (Backtrack search, and write flow)
        v := t
        while v ≠ s
            u := P[v]
            F[u,v] := F[u,v] + m
            F[v,u] := F[v,u] - m
            v := u
    return (f, F)

algorithm BreadthFirstSearch
```

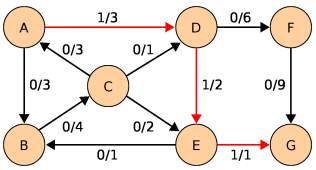
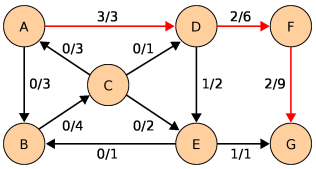
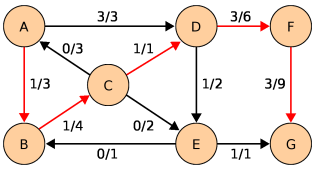
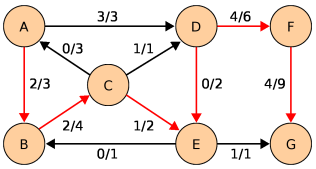
```
    input:
        C, E, s, t, F
    output:
        M[t]            (Capacity of path found)
        P               (Parent table)
    P := array(1..n)
    for u in 1..n
        P[u] := -1
    P[s] := -2 (make sure source is not rediscovered)
    M := array(1..n) (Capacity of found path to node)
    M[s] := ∞
    Q := queue()
    Q.push(s)
    while Q.size() > 0
        u := Q.pop()
        for v in E[u]
            (If there is available capacity, and v is not seen before in search)
            if C[u,v] - F[u,v] > 0 and P[v] = -1
                P[v] := u
                M[v] := min(M[u], C[u,v] - F[u,v])
                if v ≠ t
                    Q.push(v)
                else
                    return M[t], P
    return 0, P
```

Example

Given a network of seven nodes, source A, sink G, and capacities as shown below:



In the pairs f/c written on the edges, f is the current flow, and c is the capacity. The residual capacity from u to v is $c_f(u,v) = c(u,v) - f(u,v)$, the total capacity, minus the flow that is already used. If the net flow from u to v is negative, it *contributes* to the residual capacity.

Capacity	Path
	Resulting network
$\min(c_f(A,D), c_f(D,E), c_f(E,G)) =$ $\min(3-0, 2-0, 1-0) =$ $\min(3, 2, 1) = 1$	A, D, E, G
	
$\min(c_f(A,D), c_f(D,F), c_f(F,G)) =$ $\min(3-1, 6-0, 9-0) =$ $\min(2, 6, 9) = 2$	A, D, F, G
	
$\min(c_f(A,B), c_f(B,C), c_f(C,D), c_f(D,F), c_f(F,G)) =$ $\min(3-0, 4-0, 1-0, 6-2, 9-2) =$ $\min(3, 4, 1, 4, 7) = 1$	A, B, C, D, F, G
	
$\min(c_f(A,B), c_f(B,C), c_f(C,E), c_f(E,D), c_f(D,F), c_f(F,G)) =$ $\min(3-1, 4-1, 2-0, 0-(-1), 6-3, 9-3) =$ $\min(2, 3, 2, 1, 3, 6) = 1$	A, B, C, E, D, F, G
	

Notice how the length of the augmenting path found by the algorithm (in red) never decreases. The paths found are the shortest possible. The flow found is equal to the capacity across the minimum cut in the graph separating the source and the sink. There is only one minimal cut in this graph, partitioning the nodes into the sets $\{A, B, C, E\}$ and $\{D, F, G\}$, with the capacity

$$c(A, D) + c(C, D) + c(E, G) = 3 + 1 + 1 = 5.$$

Notes

References

1. Algorithms and Complexity (see pages 63–69). http://www.cis.upenn.edu/~wilf/AlgComp3.html

Article Sources and Contributors

Edmonds–Karp algorithm *Source*: http://en.wikipedia.org/w/index.php?oldid=568278410 *Contributors*: Aaron Hurst, Amelio Vázquez, Balloonguy, Cburnett, Chopchopwhitey, Cosmi, Cquan, Darth Panda, Giftlite, Glrx, Hashproduct, Headbomb, Htmnssn, Jamelan, John of Reading, Katieh5584, Kristjan Wager, Kubek15, LiuZhaoliang, Magioladitis, Martani, Michael Hardy, Mihai Capotă, Niloofar piroozi, Nils Grimsmo, Nixdorf, Nkojuharov, Ohad trabelsi, Parudox, Pkirlin, Poor Yorick, Pt, Pugget, RJFJR, Sesse, Simonfl, TPReal, WangPublic, Zero0000, قلی زادگان, 66 anonymous edits

Image Sources, Licenses and Contributors

Image:Edmonds-Karp flow example 0.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Edmonds-Karp_flow_example_0.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: en:User:Cburnett

Image:Edmonds-Karp flow example 1.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Edmonds-Karp_flow_example_1.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: en:User:Cburnett

Image:Edmonds-Karp flow example 2.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Edmonds-Karp_flow_example_2.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: en:User:Cburnett

Image:Edmonds-Karp flow example 3.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Edmonds-Karp_flow_example_3.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: en:User:Cburnett

Image:Edmonds-Karp flow example 4.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Edmonds-Karp_flow_example_4.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: en:User:Cburnett

License