**Graph algorithms - Flows**

**Transport graph and maximum flow**

In a transport graph, we have a source vertex, where a producer of some commodity is located, a destination vertex, and all the other vertices act as intermediates. Each edge represents the possibility to transport a certain amount of that commodity; the amount is the *capacity* of that edge.

The goal is to plan how to transport a maximum amount of that commodity from the source to the destination.

More formally, we have:

* a directed graph G=(X,E);
* a source vertex *s* and a destination vertex *t*;
* each edge *(x,y)* has a positive capacity *cap(x,y)*;

A *flow* can be established through the graph. A flow is an assignment of a flow value to each edge, such that:

* the flow through each edge is between zero and the capacity of that edge: 0 ≤ flow(x,y) ≤ cap(x,y)
* for each vertex, except for the source or the destination, the inbound flow is equal to the outbound flow: for any x∈X, ∑y∈Nin(x)flow(y,x) = ∑y∈Nout(x)flow(x,y)

For the source vertex, there is a positive net outbound flow. That value is called the *total value of the flow*. It is vflow = ∑y∈Nout(s)flow(s,y) - ∑y∈Nin(s)flow(y,s).

It can be easily shown that the total flow value is equal to the net inbound flow into the destination vertex: vflow = ∑y∈Nin(t)flow(y,t) - ∑y∈Nout(t)flow(t,y).

The classical problem to be solved is to set a maximum flow in the transport graph, that is, to maximize the total flow value among all possible flows.

**Cuts, capacities, and the flow across a cut**

To analize the flow, we need the concept of a *cut*. A cut is, essentially, a partitioning of the vertices into two sets: one containing the source and the other containing the destination. Then, we analyse the capacities and the flow along the edges between vertices in one subset and the other subset.

The *net flow* across the cut is the total "left to right" flow (the total flow along the edges leading from the set containing the source to the set containing the destination), minus the "right to left" flow. Formally, assuming that the cut is (A, X\A), with s∈A and t∈ X\A:  
flow(A,X\A) = ∑(x,y)∈E, x∈ A, y ∈ X\Aflow(x,y) - ∑(x,y)∈E, x∈ X\A, y ∈ Aflow(x,y)

The capacity of the cut is, however, only the "left to right" capacity: cap(A,X\A) = ∑(x,y)∈E, x∈ A, y ∈ X\Acap(x,y)

It is clear that, for any cut, the flow across the cut is less than or equal to the capacity of the cut. The maximum flow is obtained when all "right to left" edges are saturated and all "left to right" edges have zero flow.

On the other hand, the flow across any cut is the same, and is equal to the total value of the flow. (Actually, the total value of the flow is the flow across a cut that has only the source vertex on the "left", and all other vertices on the "right".)

**The naïve algorithm**

A flow of zero everywhere is clearly a valid flow. Starting from it, we can increase the flow while keeping it valid by the following approach:

* find a path from the source to the destination, consisting only of non-saturated edges;
* compute the *capacity of the path* as being the smalles of the residual capacities of its edges (the residual capacity of an edge is the difference between the capacity and the current flow through that edge);
* increase the flow, on all the edges of the path, with a value equal to that capacity.

It is clear that, by following the steps above, the flow remains valid. However, we may end up with a flow that cannot be increased by this approach, yet a flow of larger total value still exists.

**Ford-Fulkerson algorithm and Ford-Fulkerson theorem**

A correct algorithm can be devised starting from the (incorrect) naïve algorithm. This algorithm (Ford-Fulkerson) is the following:

1. again, start with a zero flow
2. for the current flow, construct a *residual graph* containing the same vertices as the original graph, but:
   * for each non-saturated edge of the original graph, put an edge with the same direction and with the remaining capacity (the capacity is the difference between the original capacity and the current flow through that edge)
   * for each edge with non-zero flow, put an edge in the reverse direction and with a capacity equal to the value of that flow (this edge signifies that the flow can be increased in the reverse direction by decreasing the forward flow). Note that this is the only difference compared to the naïve algorithm.
3. find a path from source to destination in the residual graph;
4. compute the capacity of the above path;
5. update the flow: for forward edges, increase the flow by a value equal to the capacity of the path; for backward edges, decrease the flow on the corresponding forward edges by the same value.
6. repeat the steps 2-5 until no path can be found in the residual graph from source to destination

To show that the algorithm produces the optimal flow, consider the cut having on the left-hand side all the vertices that are accessible from the source in the residual graph (since there is no path to destination, the destination is on the right-hand side of the cut). No edge can exist in the residual graph across the cut from left to right. Because of the way the residual graph was constructed, it follows that the left-to-right edges in the original graph are saturated and the right-to-left edges have zero flow. So, that cut is saturated. So, no larger flow can exist.

It follows that the value of the maximum flow is equal to the capacity of the minimum cut. This statement is called the Ford-Fulkerson theorem.

**Example**

Input graph:

A diagram of a network

Description automatically generated

After augmenting path 1,2,3,6 (capacity = 1):

A diagram of a number

Description automatically generated with residual graph A diagram of a network

Description automatically generated

After augmenting path 1,4,5,6 (capacity = 3):

A diagram of a number

Description automatically generated with residual graph A diagram of a diagram

Description automatically generated

After augmenting path 1,2,5,6 (capacity = 1):

A diagram of a number with Great Pyramid of Giza in the background

Description automatically generated with residual graph A diagram of a diagram

Description automatically generated

The naïve algorithm stops here. Ford-Fulkerson, however, finds the augmenting path 1,2,5,4,3,6 (capacity = 1):

A diagram of a number with Great Pyramid of Giza in the background

Description automatically generated with residual graph A diagram of a network

Description automatically generated

No augmenting path can be found any more. The cut {1} to {2,3,4,5,6} is saturated (capacity=6, flow=6).