

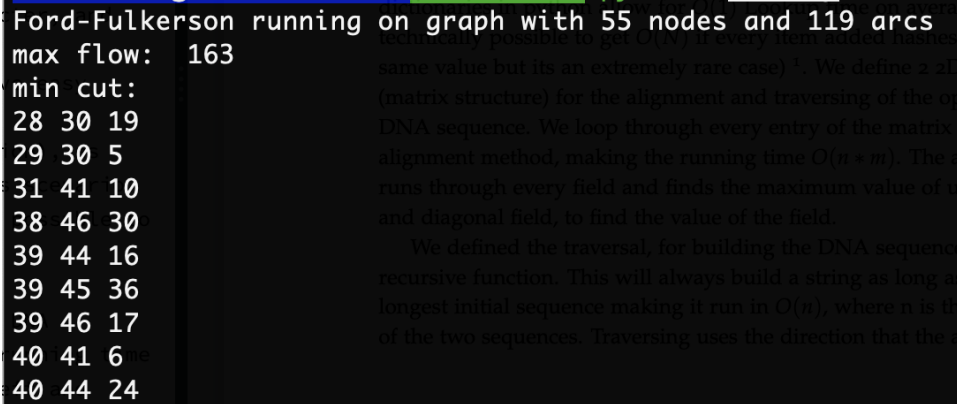
Flow

Silas Paaske, Oliver Jørgensen & Benjamin Kjær

October 26, 2022

Results

When running the implementation the algorithm correctly calculates the max flow (163) and the minimum cut, as seen on the image below.



```
Ford-Fulkerson running on graph with 55 nodes and 119 arcs
max flow: 163
min cut:
28 30 19
29 30 5
31 41 10
38 46 30
39 44 16
39 45 36
39 46 17
40 41 6
40 44 24
```

Implementation details

We have implemented a version of the Ford-Fulkerson algorithm.

To represent an Edge, we created a simple class with 3 fields: u, v & c. To represent the graph (in our Graph class) we use a nested dictionary. We decided on using a dictionary as it allows for constant lookup time. The first key in the dictionary is representing a node index (0 to 54 in the test case), and the subsequent key is another node index, representing a connected edge with its capacity. The value in the last key-value pair is then an instance of the Edge class. For a graph with nodes a, b, c, d and edges: a -> b, a -> c, a -> d; the corresponding dictionary would be:

```
{
  a {
    b: edge(u:a, v:b, c: 42),
    c: edge(u:a, v:c, c: 42),
    d: edge(u:a, v:d, c: 42)
  }
}
```

Space complexity

The primary space consumption of our implementation is the graph representation. The graph instance contains a nested dictionary of size: $O(n * (m * 2))$ - where n is number of nodes, m is number of arcs/edges, multiplied by 2 because we represent the "undirectedness" of the graph by always creating edges in both directions.

In order to "remember" the capacity of every edge (the capacity of an edge is manipulated in the augment method), we create a deep copy of the graph object before computing max flow and min cut. This means that we can print the edges used in the min cut, with their original capacity, but of course it also doubles the amount of space used for Graph objects.

Time complexity

"Let n denote the number of nodes in G , and m denote the number of edges in G . We have assumed that all nodes have at least one incident edge, hence $m \geq n/2$, and so we can use $O(m + n) = O(m)$ to simplify the bounds" - Algorithm Design, TARDOS

Supposing that, as above, that all capacities in the flow network G are integers. Then the Ford-Fulkerson Algorithm can be implemented to run in $O(m * C)$ time. The augment method will never run more times than the total capacity of the network. We use a depth first style of traversing the graph in our augment method. The augment method builds up a recursion stack as it explores edges. It will explore edges which have a capacity greater than the "threshold".