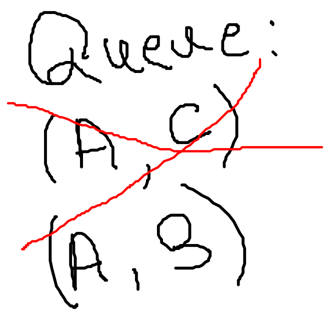
Supervision work
Supervision 16
26 April 2016

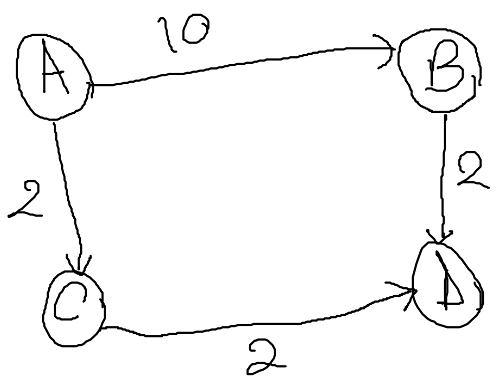
Tudor Avram
Homerton College, tma33@cam.ac.uk

April 25, 2016

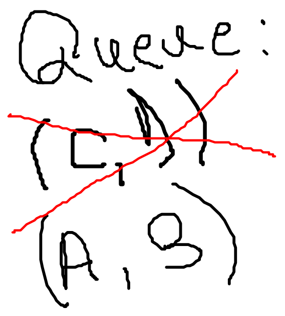
1 Shortest Paths

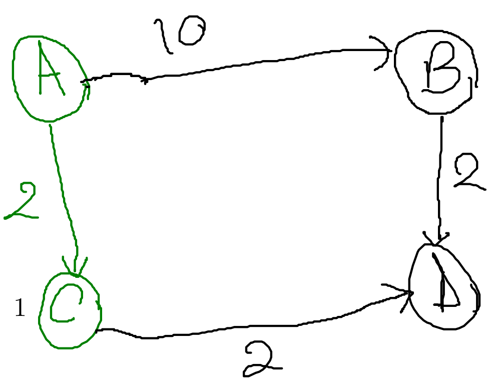
1.1 Exercise 1





STEP 1 : We go from A to C, as it is the shortest path at this point.

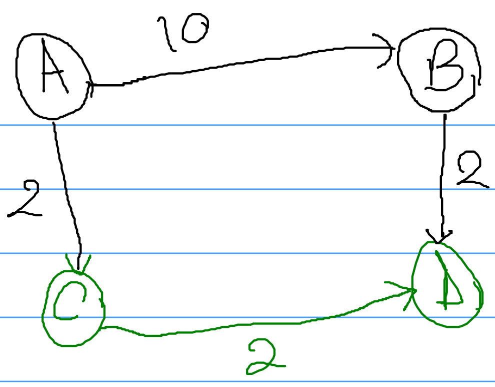




STEP 2 : We go from C to D, as it is the shortest edge as this point

Queue:

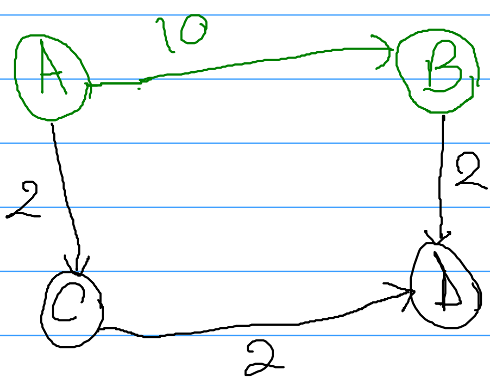
~~(A, B)~~



STEP 3 : At this point, we can't reach B from D, so we try the edge (A,B)

Queue:

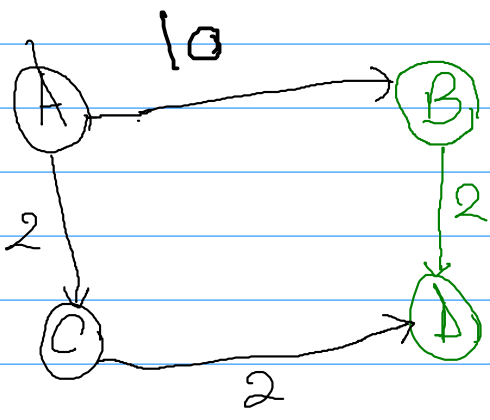
~~(B, D)~~



STEP 4 : Now we found a new minimum path from A to B (i.e. has a total weight of 10. We now try the only edge available from B - (B,D)

Queue:

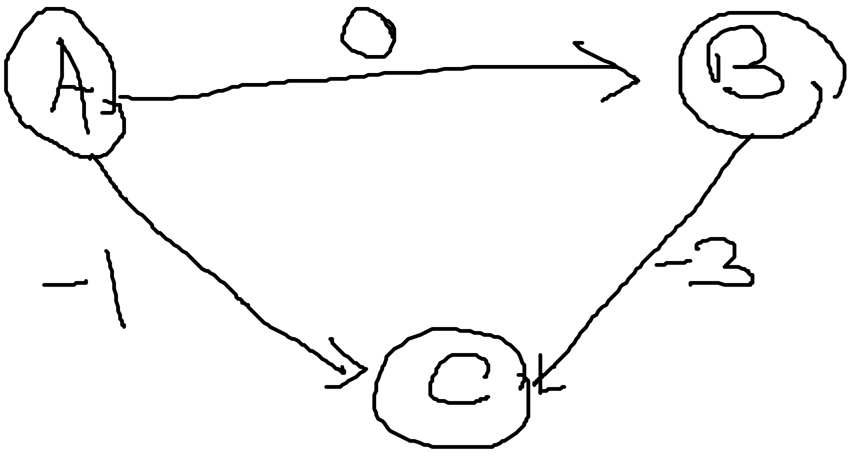
empty



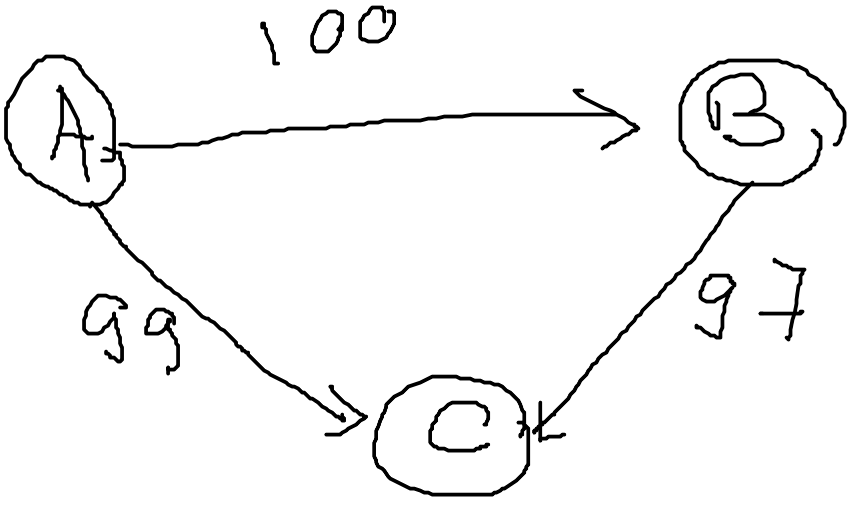
STEP 5 : At this point our queue is empty, so our algorithm terminates. The minimum path from A to B is (A,B), with weight 10.

1.2 Exercise 2

At first sight, the idea to add the same constant to the weight of every edge seems to be a valid approach to make Dijkstra's algorithm work on negative-edges graphs. In practice, though, it is not true. This happens because when adding the same value, we affect the total cost of the paths and this may give us answers that are not correct. We can take the following example :

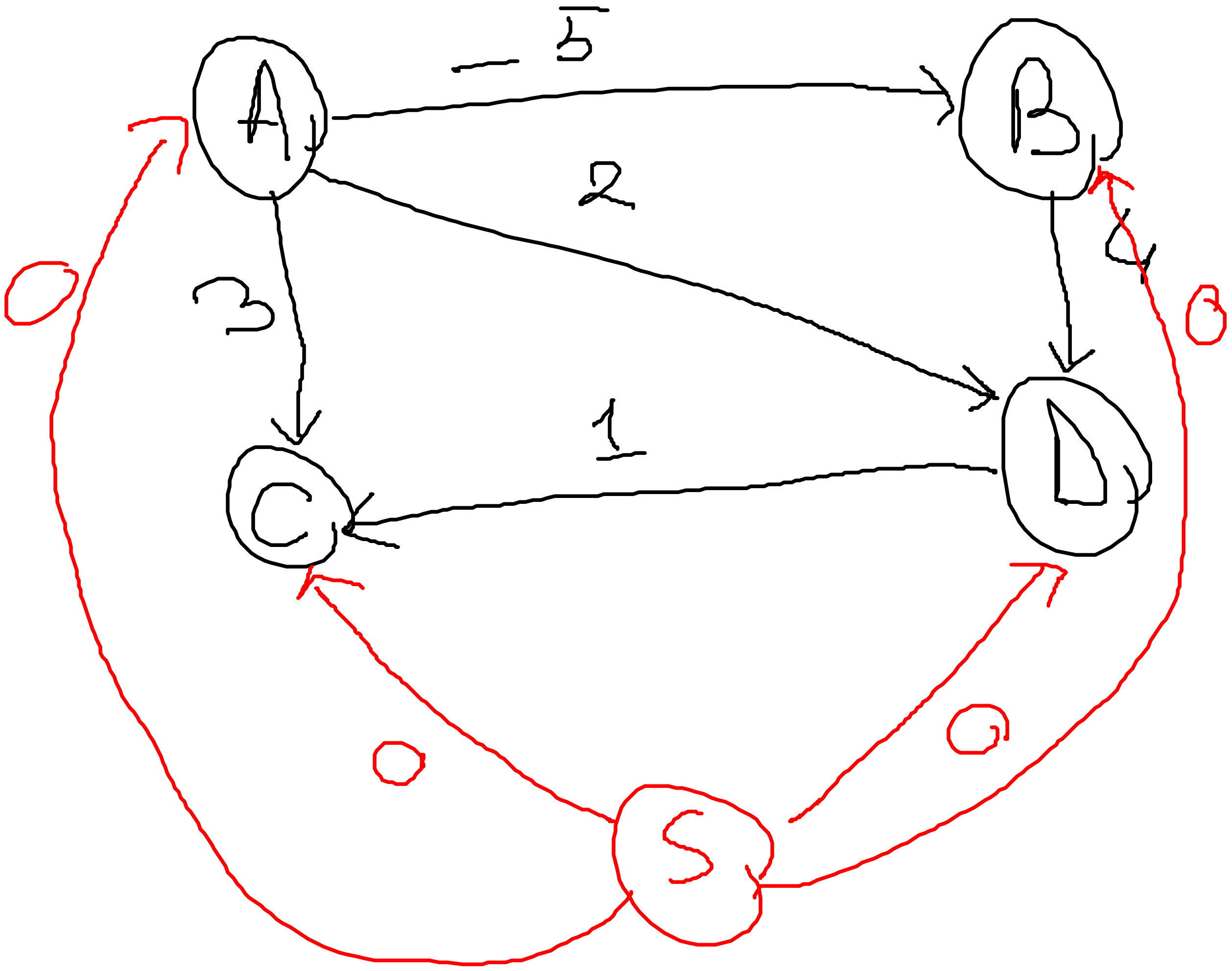


It is obvious that the shortest path from A to C is A-B-C, with a total weight of -5. We can't run the Dijkstra's algorithm on this graph, though, so we try to add a constant (let's say 100) to make all the weights positive. The new graph would look like this :



Now we can see that the shortest path after we added 100 to every edge's weight, the shortest path from A to C is not longer A-B-C. It is the direct edge from A to C. Therefore, we can conclude that adding the same constant to every edge of the graph is not a reliable solution to Dijkstra algorithm's negative edges problem.

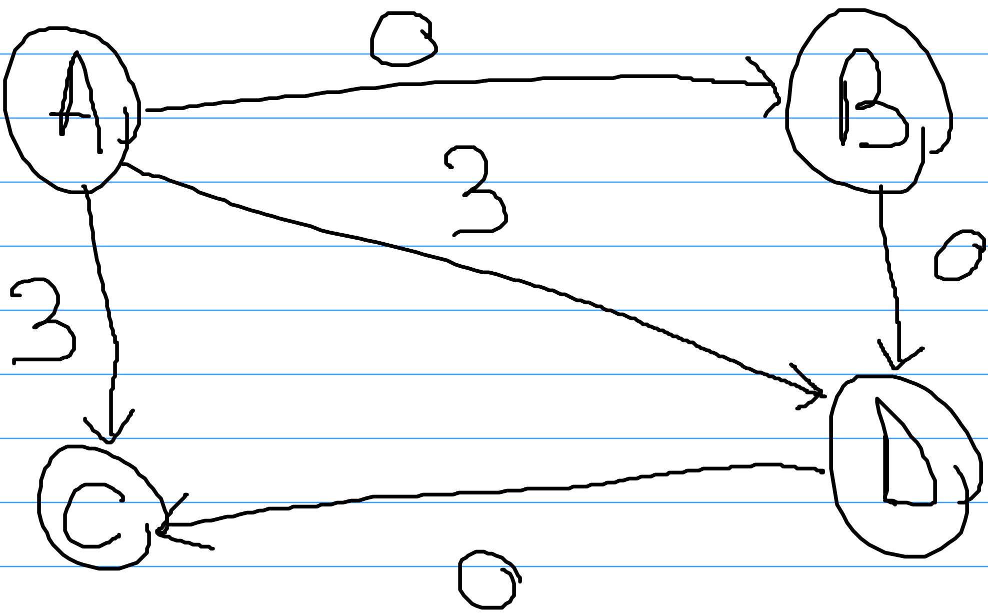
1.3 Exercise 3



After using the Bellman-Ford algorithm on the graph above, I obtained the following costs. (I used node S as the source).

	A	B	C	D
d	0	-5	0	-1

So, the re-weighted graph would look like this :



$$\begin{tikzcd}
A \arrow[r, "0"] \arrow[d, "3"'] \arrow[dr, "3"] & B \arrow[d, "0"] \\
C & D \arrow[l, "0"]
\end{tikzcd}$$

Then, I applied Dijkstra's algorithm using every node as source, and the resulting costs were :

	A	B	C	D
A	▨	0	3	0
B	▨	▨	0	0
C	▨	▨	▨	▨
D	▨	▨	0	▨

table

P

*Note : * ▨ means that there is no possible path there.

Now we only have to re-calculate the weights of the paths. The actual table would look like this :

	A	B	C	D
A	[illegible]	−5	3	−1
B	[illegible]	[illegible]	5	4
C	[illegible]	[illegible]	[illegible]	[illegible]
D	[illegible]	[illegible]	1	[illegible]

table P'

Note the initial re-weighting of an edge was calculated by the following formula :

$$W'_{i,j} = W_{i,j} + d_i - d_j$$

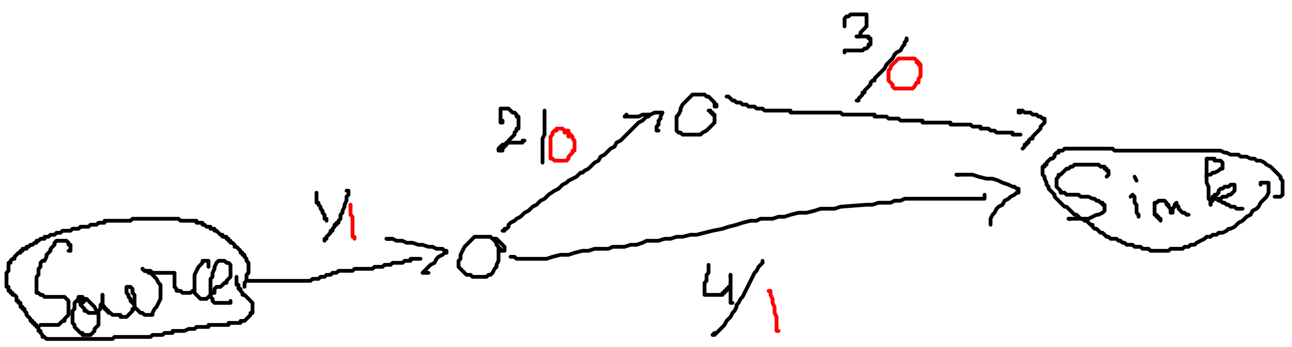
Thus, the formula for the final re-weighting is given by :

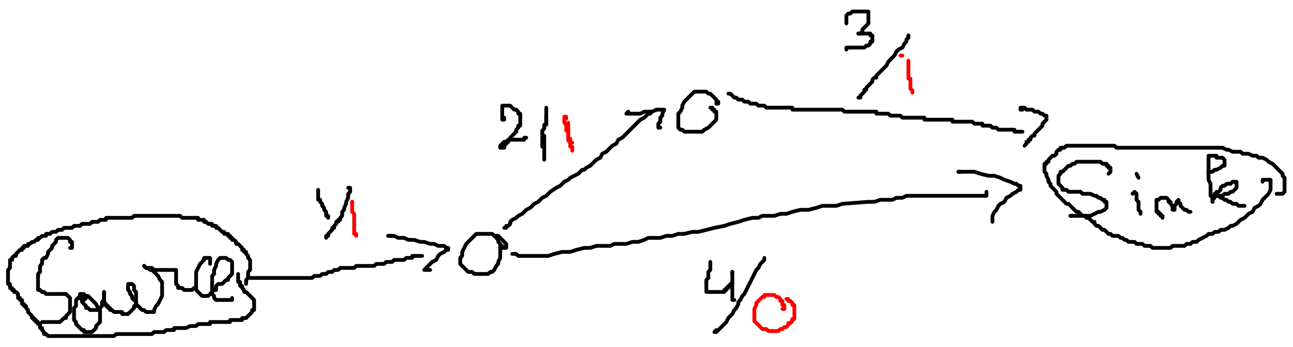
$$P'_{i,j} = P_{i,j} - d_i + d_j$$

2 Flows

2.1 Exercise 5

"If all edges in a flow network have different capacities, then there is a unique maximum flow." This sentence is not true, as is shown by the following example, which has two possible maximum flows.





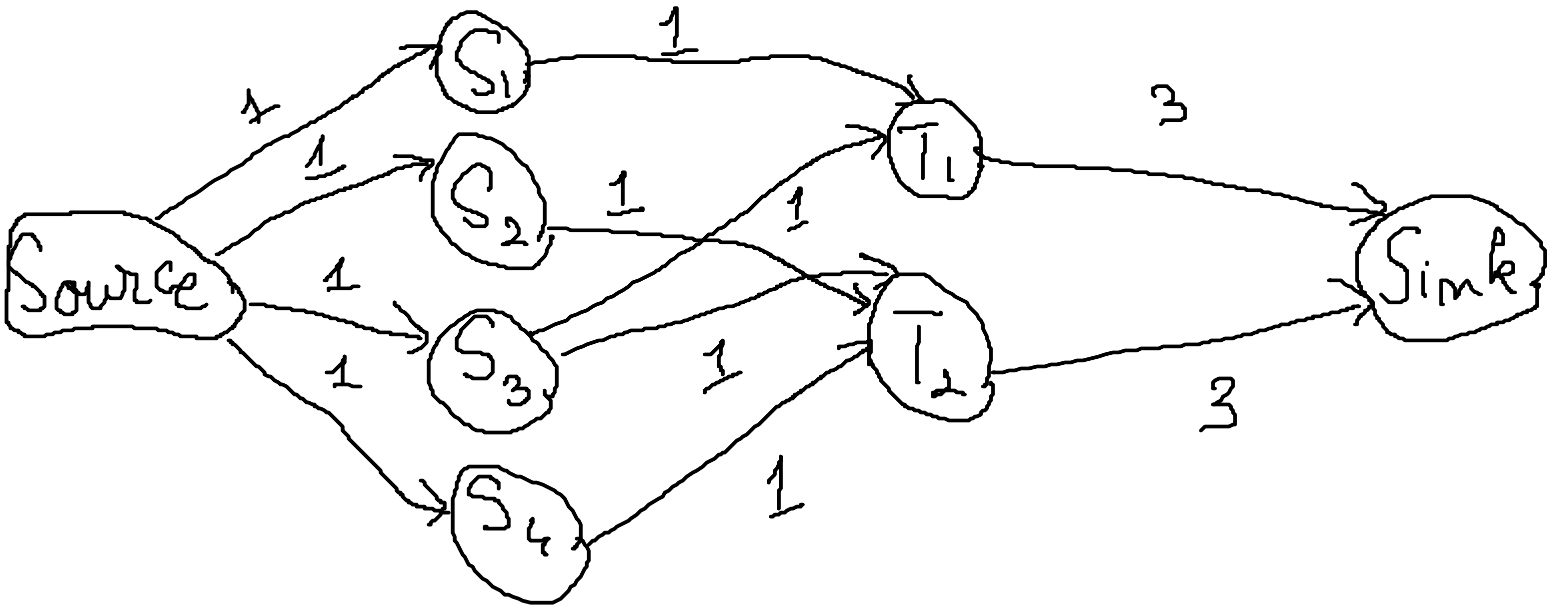
2.2 Exercise 6

In order to check if f is a maximum (s,t)-flow in G we need to follow 2 steps :

1. First we need to apply the Ford-Fulkeson algorithm in order to find the value of the maximum flow in our network.
2. Then, after we compare the value we found in step 1 to the value returned f. If the two numbers are equal, we can state that f is one of the possible maximum flows.

2.3 Exercise 7

In order to solve the given problem, we can use the flow-determining algorithm on a directed graph that has the students, the supervisions times, a source and a sink as vertices. The source is connected to the student-nodes by edges of weight 1 and the supervision-time-nodes are all connected to the sink by edges of weight 3. The nodes that represent the students and supervision times are connected by edges of weight 1, considering their preferences. For example, if student 1 can make supervision 2 and 4, the node representing him/her will be connected to the correspondent nodes for the supervision slots.
In order to respect the minimum number of 2 students/ supervision slot, we need to prioritise the paths that go through supervision slots with the maximum number of students.
Here is an example of how the constructed graph would look like :



3 Geometric Algorithms

3.1 Exercise 8

Note : From the question (i.e. "any three") I understood that it asks to check if *all* the points are collinear. I suppose from the given complexity that what it wants us to do is to find all the triplets of collinear points.

In order to solve this, I consider every single point in the set as being an origin (Po) and compute the gradient it makes with every single other point P in the set. All these gradients are stored in an array, which is then sorted. Now we only have to go through the sorted array and for every pair of equal elements we have 3 collinear points.

Thus the overall number of iterations is $n * (n + log(n) + n) = 2 \times n^2 + n \times log(n)$. So the complexity is $\boldsymbol{O(n^2 \times log(n))}$, as requested.

3.2 Exercise 9

Let's assume that when we enter the n^{th} point in our coordinate system, we have C(n-1) as our convex hull up to this point. Now, we have to check if our new point is contained by C(n-1) or not. We can do this in $O(n)$ complexity.
If it is so, we don't have to do anything, as C(n-1) still is a valid hull.
If not, we just have to "adjust" C(n-1), which can be easily done in $O(1)$.

Therefore, we can compute a one-point-at-a-time convex hull with $\sum_1^{n-1} k = \frac{n \times (n-1)}{2}$ operations. So, we can safely assume that the overall complexity is $\boldsymbol{O(n^2)}$.