Part VII
Shortest path problems

1 Problem setting (p. 74)

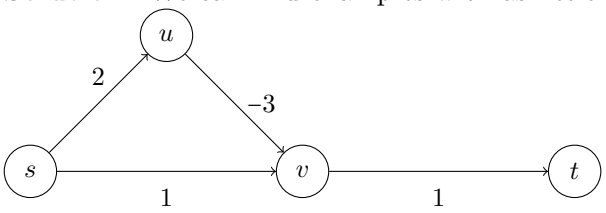
2 Relaxation (p. 76)

3 Dijkstra's algorithm (p. 79)

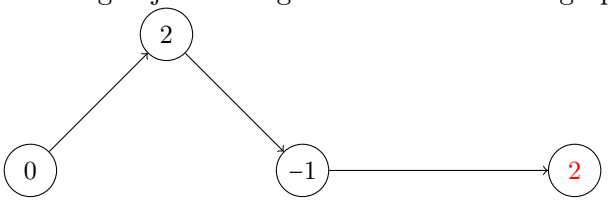
3.1 On Dijkstra's algorithm (p. 81)

1. Give a simple example of a directed graph with negative-weight edges for which Dijkstra's algorithm produces incorrect answers.

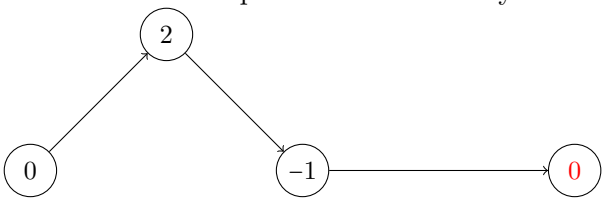
Solution: We can find examples with as little as 4 nodes:



Running Dijkstra's algorithm on the above graph produces:



while the shortest path tree is obviously



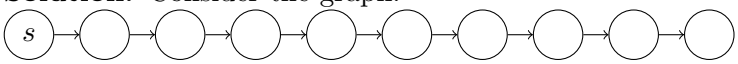
In the above example the algorithm produces an incorrect answer because it dequeues v before the shortest path to v has been found.

4 Bellman-Ford algorithm (p. 82)

4.1 On Bellman-Ford algorithm (p. 84)

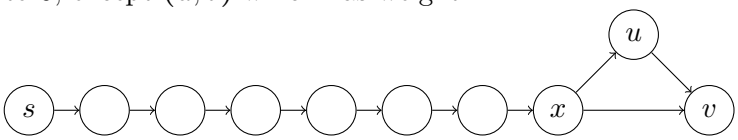
1. Give an example of a graph $G = (V, E)$ with $|V| = 10$, where $d(v) > \delta(s, v)$ after $|V| - 2$ iterations when running the Bellman-Ford algorithm.

Solution: Consider the graph:



where every edge has weight 0. Suppose the order of appearance of edges in E is from the right to the left. Then the last vertex cannot be reached by BF in $|V| - 2$ iterations.

Another example: Consider the following graph, where all edges have weight equal to 0, except (u, v) which has weight -1:



Suppose (u, v) appears before (x, u) in E when running BF algorithm. Then after $|V| - 2$ iterations of BF algorithm we have $d(v) = 0$ but $\delta(s, v) = -1$.

5 Shortest paths in a DAG (p. 85)

Misplaced exercise from page 16:

6. Given a weighted acyclic graph $G = (V, E)$. Give an $O(|V| + |E|)$ algorithm to find a path with maximum weight in G. ☆

Solution:

Step 1: adding a universal source s and a universal sink t
Set $V' = V \cup \{s, t\}$
Set $E' = E \cup \{(s, v) | v \in V\} \cup \{(v, t) | v \in V\}$
Step 2: finding a path with maximum weight
for $v \in V$ **do**
    Set $\alpha'((s, v)) = 0$ and $\alpha'((v, t)) = 0$
end for
for $e \in E$ **do**
    Set $\alpha'(e) = -\alpha(e)$
end for
DAG-SHORTEST-PATH$(G' = (V', E', \alpha'), s)$

Note, that in the RELAX operation in DAG-SHORTEST-PATH we use α' and not α. Both step 1 and 2 take $O(|V| + |E|)$ time. A maximum weight path in G can be found in $O(|V| + |E|)$ time as $t, \pi(t), \pi(\pi(t)), \pi(\pi(\pi(t))), \ldots, s$ is a minimum weight path from s to t (in reverse order) in G'.

6 Shortest simple paths (p. 85)

7 Systems of difference constraints (p. 86)

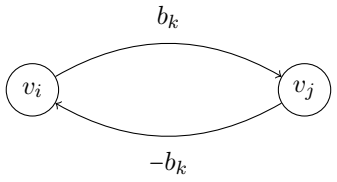
7.1 On systems of difference constraints (p. 88)

1. Argue that the solution determined by the Bellman-Ford algorithm for a system of difference constraints is such that $x_i \leq 0$ for all i.

Solution: In the constraint graph based on a system of difference equations we have an edge (v_0, v_i) with weight 0, for every i. Hence there exists a path of weight zero from v_0 to v_i for every i. The solution of the difference equations problem determined by the Bellman-Ford algorithm is of the form $x_i = \delta(v_0, v_i) \leq \alpha(v_0, v_i) = 0$, as $\delta(v_0, v_i)$ denotes the shortest path length.

2. Can we also use the Bellman-Ford algorithm if some constraints are of the form $x_j = x_i + b_k$?

Solution: Yes, as we can replace $x_j = x_i + b_k$ by two inequalities: $x_j - x_i \leq b_k$ and $x_i - x_j \leq -b_k$. (Note, that



can't create a negative weight cycle on its own.)
Another solution: If you have $x_j = x_i + b_k$, replace x_j by $x_i + b_k$ in other (in)equalities. Keep doing this until you are left only with inequalities. Use BF on the remaining inequalities, once/if a solution is found, plug it in the equalities.

3. Show that the solution determined by the Bellman-Ford algorithm maximizes $\sum_{i=1}^{n} x_i$ if we add the constraints $x_i \leq 0$. [Hint: show that for any solution $(y_1, \ldots, y_n)$ with $y_i \leq 0$ for all i, we have $y_i \leq x_i = \delta(v_0, v_i)$.]

Solution: Denote b_k as $b_{i,j}$ for $x_j - x_i \leq b_k$. Suppose that $v_0 = v_{i_0}, v_{i_1}, \ldots, v_{i_t} = v_i$ is a shortest path from v_0 to v_i in the constraint graph. We then have

$$x_i = \delta(v_0, v_i) = \sum_{k=0}^{t-1} \alpha(v_{i_k}, v_{i_{k+1}}) = \sum_{k=1}^{t-1} \alpha(v_{i_k}, v_{i_{k+1}}) = \sum_{k=1}^{t-1} b_{i_k, i_{k+1}}$$
$$\geq \sum_{k=1}^{t-1} (y_{i_{k+1}} - y_{i_k}) = y_{i_t} - y_{i_1} = y_i - y_{i_1} \geq y_i,$$

where we've used $\alpha(v_0, v_{i_1}) = 0$ in the third equality, the fact that $(y_1, \ldots, y_n)$ is a solution in the first inequality and $y_{i_1} \leq 0$ in the second inequality. It follows that for every solution $(y_1, \ldots, y_n)$ we have

$$\sum_{i=1}^{n} y_i \leq \sum_{i=1}^{n} x_i,$$

hence $\sum_{i=1}^{n} x_i$ is maximized by the Bellman-Ford algorithm.

4. How can we solve a system of difference constraints if we demand that x_i must be an integer for all i?

Solution: Set $b'_k = \lfloor b_k \rfloor$ for every k and solve using Bellman-Ford algorithm for inequalities of the form $x_j - x_i \leq b'_k$ instead of $x_j - x_i \leq b_k$. If there are no negative weight cycles then Bellman-Ford algorithm will return a solution $(x_1, \ldots, x_n)$, with $x_i = \delta(v_0, v_i)$ for every i. As every edge in the associated constraint graph has an integer weight, $\delta(v_0, v_i)$ is a sum of integers and thus an integer. As $\lfloor b_k \rfloor \leq b_k$, a solution to the new problem is also a solution to the original problem.
Note, it is possible for the associated graph to the original problem not to have any negative weight cycles while for the associated graph to the new problem to have such cycles. For example:

$$x_1 - x_2 = 1/2$$

becomes (by using exercise 2):

$$\begin{cases} x_1 - x_2 \leq 1/2 \\ x_2 - x_1 \leq -1/2 \end{cases}$$

Another example:

$$\begin{cases} x_2 - x_1 \leq 1/2 \\ x_3 - x_2 \leq 1/2 \\ x_4 - x_3 \leq 1/2 \\ x_1 - x_4 \leq -1 \end{cases}$$

In both examples taking the floor of b_k's produces cycles in the constraint graph.

8 All pairs shortest paths (p. 89)

8.1 Floyd-Warshall (p. 89)

8.2 Johnson's algorithm (p. 90)