Exercise Sheet #6

14 June 2020 13:23

1. Dijkstra Algorithm
    a. Infinity as it is initialised like that and will never get relaxed because there just is no edge to relax.
    b. It does not work because we will relax the same edges over and over as it always results in a shorter way. This can be resolved by using the Bellman-Ford algorithm which replaces the queue and instead works by relaxing all edges V-1 times, where V is the number of vertices. Afterwards it tests whether it is able to relax them one more time and if yes then there is a negative cycle.
    c.

Iteration number	Selected node with minimum distance	A	B	C	D	E	F	G	H	I
-	-	$\boxed{0}$	∞	∞	∞	∞	∞	∞	∞	∞
1	A		9_A	∞	∞	5_A	∞	∞	$\boxed{3_A}$	∞
2	H	.	9_A	∞	∞	2_H	∞	∞	.	$\boxed{7_H}$
3	I	.	9_A	∞	∞	$\boxed{2_H}$	21_I	∞	.	.
4	E	.	$\boxed{-6_E}$	16_E	∞	.	4_E	∞	.	.
5	B	.	.	$\boxed{-9_B}$	∞	.	4_E	∞	.	.
6	C	.	.	.	-3_C	.	$\boxed{-7_C}$	∞	.	.
7	F	.	.	.	$\boxed{-6_F}$	.	.	-3_F	.	.
8	D	.	.	.	.	.	.	$\boxed{-3_F}$	.	.

    d. A-H-E-B-C-F-G with a length of -3.
2. Shortest path and interactive instance segmentation
    a. Graph and problem definition
    Given is a picture as weighted pixelgridgraph with annotations by the user. To obtain the final object mask by solving a shortest distance problem we connect all positive annotations to a node P with weight 0 and all negative annotations to another node N with weight 0. Then we use a shortest path algorithm that works with negative cycles like Bellman-Ford to calculate the distance from each pixel/node to P and N. The node gets assigned to foreground or background depending on whether it is closer to P or N respectively. We need to account for negative cycles as the weights given are displayed in all real numbers.
    b. Precomputed features
    The weights should be non-negative as it does not make sense that taking edges saves distance, it might just make no difference as it is exactly the same as the pixel marked by the user and then be weighed 0. This would allow us to use the Djikstra algorithm which uses a priority queue and relaxes the outgoing edges of the vertices with the smallest distance instead of relaxing all edges every time which saves computation time. The edge weights could be determined by calculating the absolute colour difference between the pixels.
    c. Deep-learning based model

3. Bonus: Maximum subarray problem
    a. Path: A-5-E-4-I-20-F-2-C-6-D-7-G -> Vertices A,E,I,F,C,D,G included with a length of $5 + 4 + 20 + 2 + 6 + 7 = 44$
    b. You create two new nodes and connect them to all elements/vertices with a weight of zero, one with ingoing and one with outgoing edges. The vertices itself are connected in list order and have their value as weight. Then you run the longest path algorithm on this directed acyclic graph between the two newly created nodes to get the longest path. This way every element can be the start of the array and every element can also be the end.