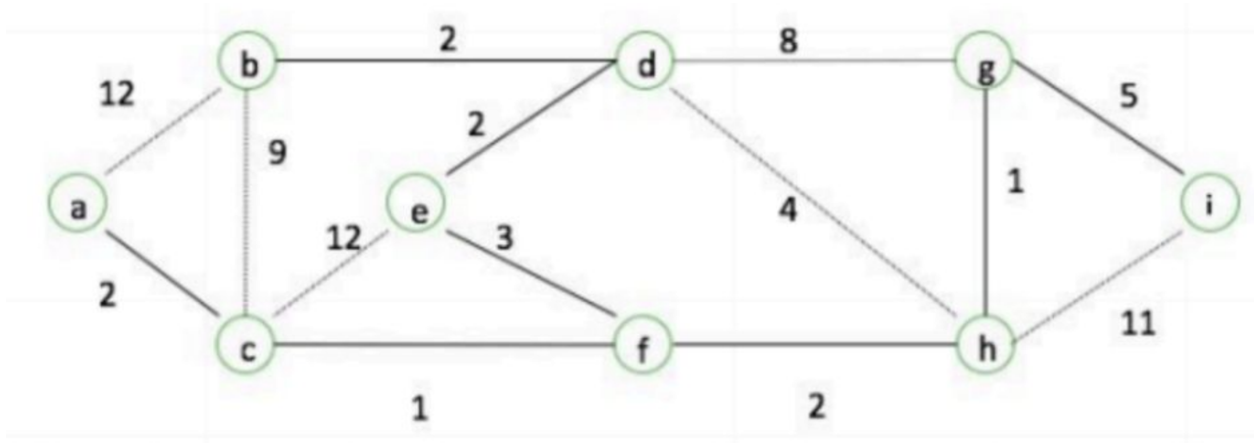
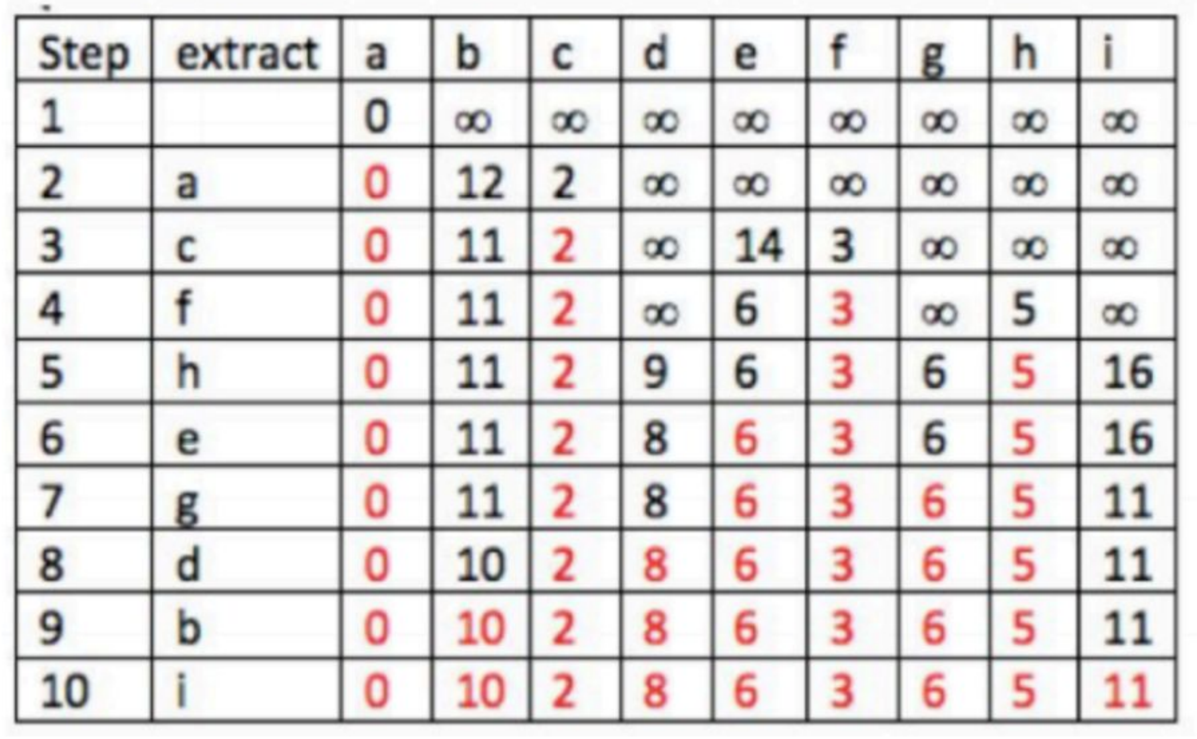
Solution 12

1.





Step	extract	a	b	c	d	e	f	g	h	i
1		0	∞	∞	∞	∞	∞	∞	∞	∞
2	a	0	12	2	∞	∞	∞	∞	∞	∞
3	c	0	11	2	∞	14	3	∞	∞	∞
4	f	0	11	2	∞	6	3	∞	5	∞
5	h	0	11	2	9	6	3	6	5	16
6	e	0	11	2	8	6	3	6	5	16
7	g	0	11	2	8	6	3	6	5	11
8	d	0	10	2	8	6	3	6	5	11
9	b	0	10	2	8	6	3	6	5	11
10	i	0	10	2	8	6	3	6	5	11

2. https://walkccc.github.io/CLRS/Chap25/25.1/

3. https://walkccc.github.io/CLRS/Chap25/25.2/

4.

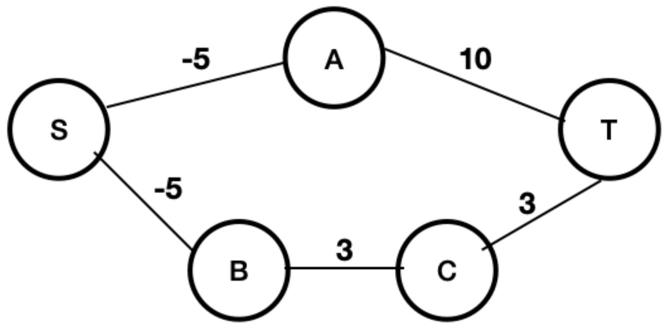
There are two ways of doing this:

1. By creating virtual nodes and bfs - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.20.6245&rep=rep1&type=pdf
2. By optimizing the min-heap using buckets - https://courses.csail.mit.edu/6.006/fall10/handouts/recitation11-5.pdf

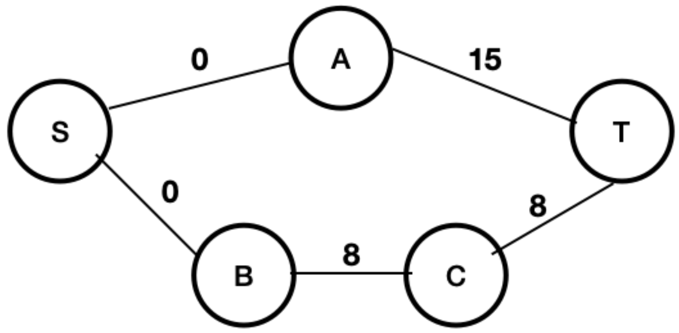
5.

No, this algorithm does not give correct results. Below is a counter-example.

In the origin graph, the stortest path from S to T is S->B->C->T, which has a cost of -5 + 3 + 3 = 1.



If we adjust weights by adding to each edge the absolute value of the lowest edge weight of graph, the new graph becomes following.



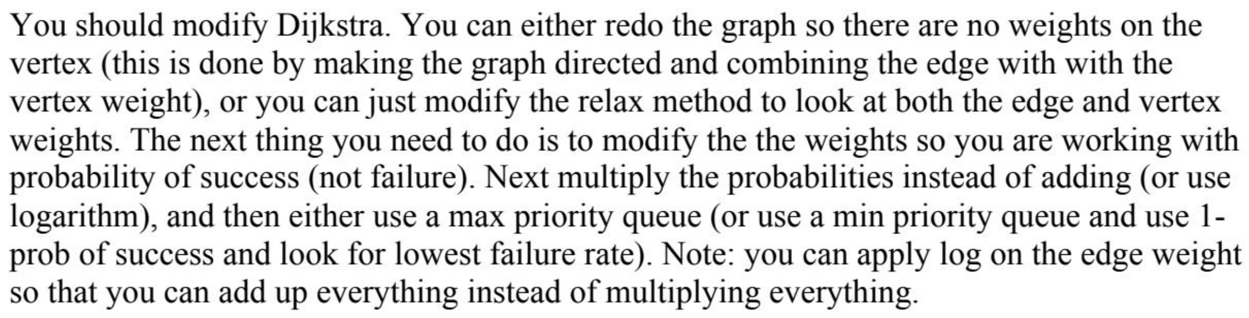
As we can see, the shortest path from S to T now becomes S->A->T with a cost of 0 + 15 = 15.

Therefore, this aproch does not give correct results.

6.

https://www.geeksforgeeks.org/detecting-negative-cycle-using-floyd-warshall/

7.



You should modify Dijkstra. You can either redo the graph so there are no weights on the vertex (this is done by making the graph directed and combining the edge with with the vertex weight), or you can just modify the relax method to look at both the edge and vertex weights. The next thing you need to do is to modify the the weights so you are working with probability of success (not failure). Next multiply the probabilities instead of adding (or use logarithm), and then either use a max priority queue (or use a min priority queue and use 1-prob of success and look for lowest failure rate). Note: you can apply log on the edge weight so that you can add up everything instead of multiplying everything.