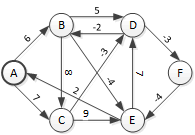
Nalongsone Danddank Student ID : 14958950 StarID: jf3893pd

Email: [nalongsone.danddank@my.metrostate.edu](mailto:nalongsone.danddank@my.metrostate.edu)

Homework 7 (*15/15* pts)[[1]](#footnote-1)

1. [6/6 pts]: Perform the first two iterations of the Bellman-Ford algorithm on the following graph. Show the results in the following tables. Appendix 1 shows the filled out tables for the Bellman Ford algorithm on a similar graph. the table is also shown in the lecture slides for the graph used there.



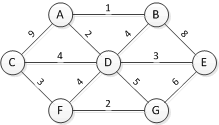
Assume that edges are processed in the following order. Fill out this table for iteration 1. The start values are given. Do not update a distance and predecessor unless it results in a shorter path (so don’t update if you find a new path equal in distance to a previous one).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Node | Start | | Rel. AB | | Rel. AC | | Rel. BC | | Rel. BD | | Rel. BE | | Rel. CD | | Rel. CE | | Rel. DB | | Rel. DF | | Rel. EA | | Rel. ED | | Rel. FE | |
| s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π |
| A | 0 | ∅ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B | ∞ | ∅ | 6 | A |  |  |  |  |  |  |  |  |  |  |  |  | 2 | D |  |  |  |  |  |  |  |  |
| C | ∞ | ∅ |  |  | 7 | A |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D | ∞ | ∅ |  |  |  |  |  |  | 11 | B |  |  | 4 | C |  |  |  |  |  |  |  |  |  |  |  |  |
| E | ∞ | ∅ |  |  |  |  |  |  |  |  | 2 | B |  |  |  |  |  |  |  |  |  |  |  |  | -3 | F |
| F | ∞ | ∅ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 | D |  |  |  |  |  |  |

Fill out this table for iteration 2. Take the start values from the end values of iteration 1. Do not update a distance and predecessor unless it results in a shorter path (so don’t update if you find a new path equal in distance to a previous one).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Node | Start | | Rel. AB | | Rel. AC | | Rel. BC | | Rel. BD | | Rel. BE | | Rel. CD | | Rel. CE | | Rel. DB | | Rel. DF | | Rel. EA | | Rel. ED | | Rel. FE | |
| s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π |
| A | 0 | ∅ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | -1 | E |  |  |  |  |
| B | 2 | D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| C | 7 | A |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D | 4 | C |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| E | -3 | F |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| F | 1 | D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. [3/3 pts]: For the following graph, show the order in which the edges are added to the MST when using Kruskal’s algorithm. If you have a choice of two edges to add, add the edge whose vertex pair comes first alphabetically. (So if you have the choice of adding edge BE or edge CD, add BE.) Number the edges in the MST from 1 to 6, leaving the non-MST edges unnumbered.



Edges:

AB: \_\_1\_

AC: \_\_\_

AD: \_\_2\_

BD: \_\_

BE: \_\_\_

CD: \_\_6\_

CF: \_4\_\_

DE: \_\_5\_

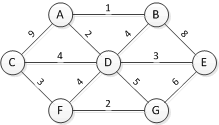
DF: \_\_\_

DG: \_\_\_

EG: \_\_\_

FG: \_3\_\_

1. [3/3 pts]: For the following graph, show the order in which the edges are added to the MST when using Prim’s algorithm, starting from node ***D***. If you have a choice of two edges to add, add the edge whose vertex pair comes first alphabetically. (So if you have the choice of adding edge BE or edge CD, add BE.) Number the edges in the MST from 1 to 6, leaving the non-MST edges unnumbered.



Edges:

AB: \_2\_\_

AC: \_\_\_

AD: \_1\_\_

BD: \_\_\_

BE: \_\_\_

CD: \_\_4\_

CF: \_\_5\_

DE: \_\_3\_

DF: \_\_\_

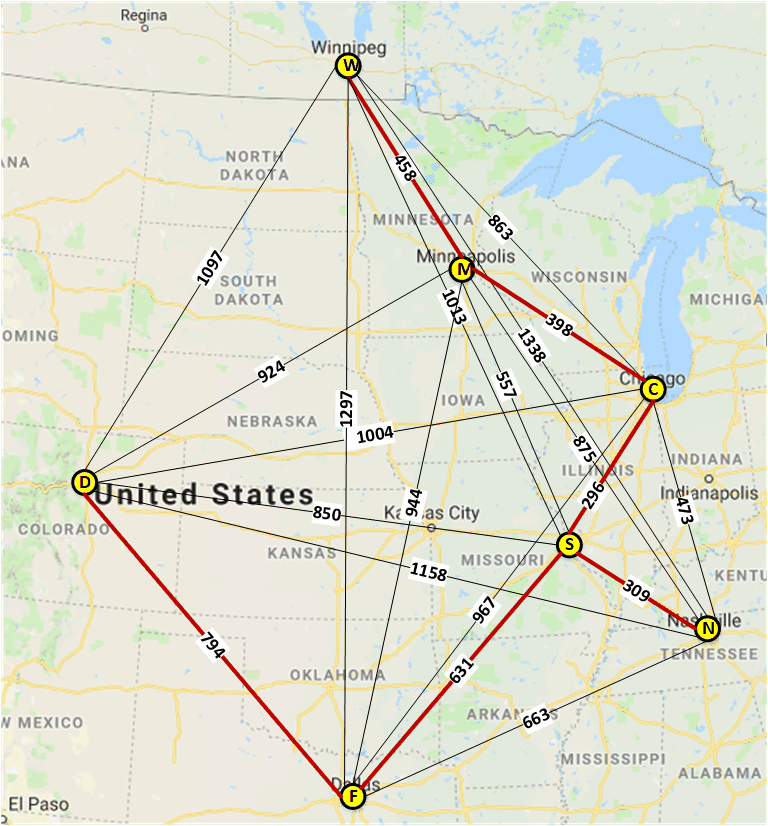
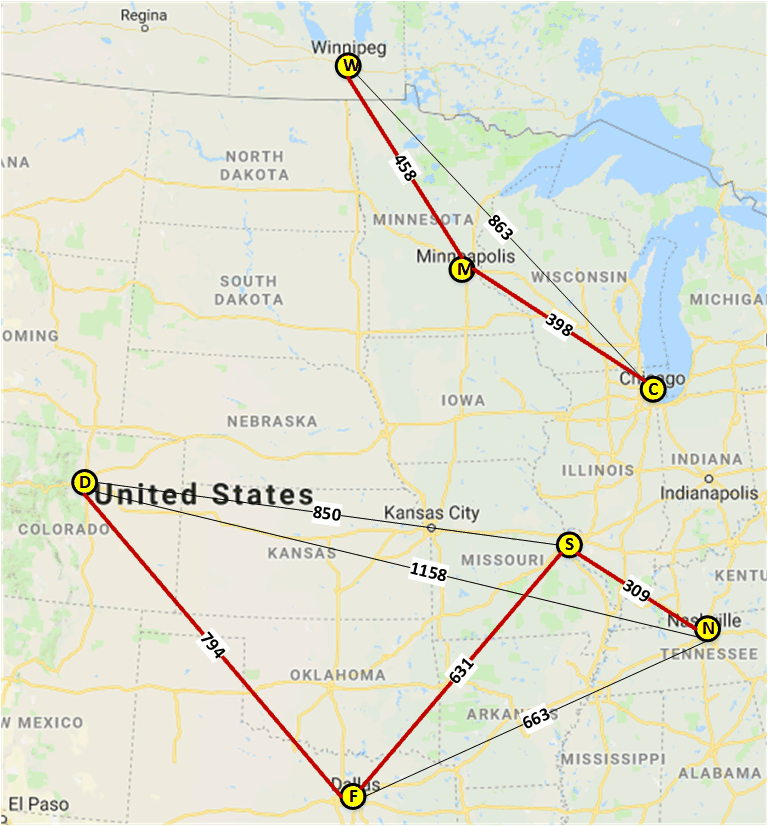
DG: \_\_\_

EG: \_\_\_

FG: \_\_6\_

1. [3/4 pts]: Suppose that we have a graph G = (V,E), and a minimum spanning tree T = E’, where E’ ⊆ E. Suppose that we remove one “interior” edge of the MST, that is an edge each of whose endpoints is attached to another edge of the MST). This divides the MST into two trees. Prove that each of these two trees is an MST on its respective set of nodes.

*Example: (Note that I’m asking for a general proof, not a proof of this example) In the left graph, the thick maroon lines are the MST. In the right graph, when I remove Chicago 🡨 🡪 St Louis, the maroon lines are the MSTs of the subgraphs. The thick lines are the MSTs of the subgraphs.*



Solution:

As we know that total number of MST’s edge E is number of MST’s Vectors minus one.

-> E = V -1 .

For example: following above left MST Graph we have number of V = 7

and number of edge E = V -1 = 7 - 1 = 6

And we can see the right MST graph two small MST graph also have the same rule:

E1 = V1 -1 = 4-1 = 3 and E2 = V2 - 1 = 3 - 1 = 2.

-> Following this question said that we take one edge off from original graph to divide to two graph, so we have V = V1 + V2 and E = E1 + E2 +1 , E1 = V1 -1 and E2 = V2 -1.

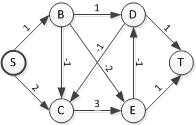
-> Then, we have that E = (V1 -1) + (V2 -1) + 1 = (V1 + V2) -1 -1 +1 = (V1 + V2) - 1 = V - 1.

So, we can get E = V -1 from two small graph that divide from original one.

Also following above two graph: E = 4 -1 + 3 - 1 +1 = 6

So we proved that divides the MST into two MST’s trees by remove one interior edge it off, like question above. *But how do we know that the two smaller “MSTs” are actually MSTs of the smaller trees?*

**Appendix A: Bellman Ford example**

****

Assume that edges are processed in the following order. Fill out this table for iteration 1. The start values are given.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Node | Start | | Rel. SB | | Rel. SC | | Rel. BC | | Rel. BD | | Rel. BE | | Rel. CE | | Rel. DC | | Rel. DT | | Rel. ED | | Rel. ET | |
| s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π |
| S | 0 | ∅ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B | ∞ | ∅ | 1 | S |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| C | ∞ | ∅ |  |  | 2 | S | 0 | B |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D | ∞ | ∅ |  |  |  |  |  |  | 2 | B |  |  |  |  |  |  |  |  | -2 | E |  |  |
| E | ∞ | ∅ |  |  |  |  |  |  |  |  | -1 | B |  |  |  |  |  |  |  |  |  |  |
| T | ∞ | ∅ |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 3 | D |  |  | 0 | E |

Fill out this table for iteration 2. Take the start values from the end values of iteration 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Node | Start | | Rel. SB | | Rel. SC | | Rel. BC | | Rel. BD | | Rel. BE | | Rel. CE | | Rel. DC | | Rel. DT | | Rel. ED | | Rel. ET | |
| s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π | s.d | π |
| S | 0 | ∅ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B | 1 | S |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| C | 0 | B |  |  |  |  |  |  |  |  |  |  |  |  | -3 | D |  |  |  |  |  |  |
| D | -2 | E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| E | -1 | B |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| T | 0 | E |  |  |  |  |  |  |  |  |  |  |  |  |  |  | -1 | D |  |  |  |  |

1. Note that > 15 points are possible, so basically this homework includes a little extra credit. [↑](#footnote-ref-1)