Q_1

a.

PF:

1. Assume each of the clusters $\{a, b\}, \{e\}, \{d, c, i, f, g, h\}$ is a tree.

2. Assume $T \subseteq T_{\text{min}}$ for some MST T_{min}.

As $\{a, b\} \neq \{d, c, i, f, j\}$, we add (a, h) to T.

WTP:

1. Each of the clusters $\{a, b, d, c, i, f, g, h\}, \{e\}$ is a tree.

2. Assume $T = \{(g, h), (c, i), (f, g), (c, f), (a, b), (c, d), (a, h)\} \subseteq T_{\text{min}}$ for some MST T_{min}.

Consider $\{a, b\}$ and $\{d, c, i, f, g, h\}$ in our assumption

we have $\{a, b\}$ and $\{d, c, i, f, g, h\}$ two trees

$\Rightarrow 1^\circ$ no circle in $\{a, b\}$ and $\{d, c, i, f, g, h\}$ $(*)$

2° $\{a, b\}$ are connected with each other and $\{d, c, i, f, g, h\}$ are connected with each other

$\because \{a, b\} \neq \{d, c, i, f, g, h\}$

$\therefore (a, h)$ is added to T

$\Rightarrow \{a, b, d, c, i, f, g, h\}$ are connected with each other (1)

suppose there is a circle in $\{a, b, d, c, i, f, g, h\}$

by $(*)$

there are at least 2 paths that connect $\{a, b\}$ and $\{d, c, i, f, g, h\}$

$\because (a, h)$ is a path (edge) that connects $\{a, b\}$ and $\{d, c, i, f, g, h\}$

so before adding (a, h). there is still a path that connects $\{a, b\}$ and $\{d, c, i, f, g, h\}$

then after adding the path. $\{a, b\}$ and $\{d, c, i, f, g, h\}$ should already be merged together before adding (a, b)

$\Rightarrow$ this is a contradiction

$\Rightarrow$ The supposition is false and there is no circle in $\{a, b, d, c, i, f, g, h\}$ (2)

By (1) and (2).

$\{a, b, d, c, i, f, g, h\}$ is a tree and so is $\{e\}$

Call the set T before processing $(a, h, 8)$ T_1 and the set T after processing $(a, h, 8)$ T_2

we have $T_1 \subseteq T_{\text{min}}$ for some MST T_{min}, let's call the T_{min} "T_{mink}"

if $(a, h) \in T_{\text{mink}}$, then $T_2 = T_1 \cup \{(a, h)\} \subseteq T_{\text{mink}}$

$\Rightarrow T_2 \subseteq T_{\text{min}}$ for some MST T_{min}

if $(a, h) \notin T_{\text{mink}}$

$\Rightarrow$ divide vertices into two sets S and $V - S$

$S = \{a, b\}$

$V - S = \{d, c, i, f, g, h, e\}$

there is no T_1 edge between S and $V - S$

in T_{mink} , there exists a unique path that connects a and u, and there is an edge on this path that connects S and $V - S$

In the graph, we have such edges (a, h) and (b, c)

$\because (a, h) \notin T_{\text{mink}}$

$\therefore (b, c) \in T_{\text{mink}}$

$\because$ weight $(a, h) =$ weight $(b, c) = 8$

$\therefore T_{\text{mink}}{}' = T_{\text{mink}} - \{(b, c)\} + \{(a, h)\}$ has the same weight as T_{mink}

T_{mink} is disconnected after removing (b, c), but reconnected after adding (a, h)

In T_{mink} , there is no circle.

So there is no circle in $T_{\text{mink}}{}'$ as we use one unique edge to link these two parts: $\{a, b\}$ and $\{h, g, f, c, i, d, e\}$.

$\Rightarrow T_{\text{mink}}{}'$ is a MST

$\because T_2 \subseteq T_{\text{mink}}{}'$

$\therefore T = \{(g, h), (c, i), (f, g), (c, f), (a, b), (c, d), (a, h)\} \subseteq T_{\min}$ for some MST $T_{\min}$

QED.

b.

PF:

Assume T contains vertices a, b and PQ contains vertices c, d, e, f, g, h, i.

Assume for each v in PQ, priority (v) = minimum weight of any edge between v and T.

Assume $T \subseteq T_{\min}$ for some MST $T_{\min}$

WTP.

After de-queuing h,

T contains a, b, h and PQ contains c, d, e, f, g, i

for each v in PQ, priority (v) = minimum weight of any edge between V and T

$T \subseteq T_{\min}$ for some MST $T_{\min}$

After de-queuing h, (a, h) is added to T

so T has h and PQ only lose h

$\Rightarrow T$ contains a, b, h and PQ contains c, d, e, f, g, i

As T is only extended a vertex h, the vertices in PQ whose priorities are charged can only be the adjacency of h, as they have a new edge to the new T.

By the algorithm and the graph, only i and g need to update their priority and they will still be the minimum weight of any edge between $i||g$ and new T (“the decrease-priority ()” lets g and i always choose the minimum weight).

From our assumption, c, d, e, f have the priorities that are the minimum weight of any edge between $c||d||e||f$ and old T.

The dequeue operation doesn't affect $c||d||e||f$:

1. The new T's new edge (a, h) still doesn't generate a new edge to $c||d||e||f$
2. Priorities of $c||d||e||f$ aren't changed.

$\Rightarrow$ $c||d||e||f$ have priorities that are equal to minimum weight of any edge between $c||d||e||f$ and new T. (4)

By (3) (4). for each v in new PQ, priority (v) = minimum weight of any edge between v and new T

Call T before de-queuing h be T_3, T after de-queuing h be T_4.

We have $T_3 \subseteq T_{\min}$ for some MST.

Call the $T_{\min}$ be T_{minkb}.

If $(a, h) \in T_{\text{minkb}}$, then $T_4 = T_3 \cup \{(a, h)\} \subseteq T_{\text{minkb}}$.

$\Rightarrow$ After de-queuing $h, T \subseteq T_{\min}$ for some MST $T_{\min}$

If $(a, h) \notin T_{\text{minkb}}$,

then divide V into $\{a, b\}$, and $\{h, i, c, d, e, f, g\}$ two parts.

There is a unique path in T_{minkb} that connects a and h, and there is a unique edge that connects these two parts.

In the given graph, they are (a, h) and (b, c)

$\because (a, h) \notin T_{\text{minkb}}$

$\therefore$ The unique edge is (b, c)

$\because$ weight (b, c) = weight $(a, h) = 8$

$\therefore T_{\text{minkb}}{}' = T_{\text{minkb}} - \{(b, c)\} + \{(a, h)\}$ has the same weight as T_{minkb}

T_{minkb} is disconnected after removing (b, c) and reconnected after adding (a, h) with no circles produced.

$T_{\text{minkb}}{}'$ is a MST

$T_4 = T_3 \cup \{(a, h)\} \subseteq T_{\text{minkb}}{}'$

$\therefore$ After de-queuing $h, T \subseteq T_{\min}$ for some MST $T_{\min}$

QED.

Q_2

a.

PF:

Consider two cases:

1° The graph has no negative weights

Then by line 1: minimum_weight > 0

$\therefore c = 0$

$\therefore \forall e \in E$, weight $'(e) =$ weight (e)

$\Rightarrow$ line 5 is actually using the Prim algorithm on the given graph directly.

$\Rightarrow$ the output of this new algorithm is the MST of both the given graph with modified weight and the given graph with original weight.

2° The graph has negative edges

$\Rightarrow$ minimum_weight < 0

$\Rightarrow c \geq -$minimum_weight > 0

$\Rightarrow$ for an arbitrary edge (u, v) in E

weight $'(u, v) =$ weight $(u, v) + c \geqslant$ minimum_weight $+(-$minimum_weight $) = 0$

$\Rightarrow \forall e \in E$, weight $'(e) \geqslant 0$

After line 5,

according to the lecture, a MST of the given graph with modified weights is produced.

Let's suppose the MST be $T'_{\min} = \{e'_1, e'_2 \ldots e'_t\} \subseteq E \quad (t = |V| - 1)$ ①

WTP: T'_{min} is also a MST in the given graph with original weights

As T'_{min} is a MST in the modified graph,

it connects all the vertices of the modified graph and no circle

$\because$ The given graph with modified weights has the same V and E as the given graph with original weights

$\Rightarrow T'_{min}$ connects all the vertices in the original given graph and no circle $(*)$

Now we need to show that $T'min$ is also the spanning tree with minimum weight of the original graph.

Suppose T'_{min} in the original-weight graph is not the one with minimum weight.

$\Rightarrow$ there exists a MST T_{min} in the original graph whose total weight is smaller than T'_{min}'s.

let $T_{\min} = \{e_1, e_2 \ldots e_m\}$ $(m = |V| - 1)$

by ①

$m = t$

$\therefore T_{\min} = \{e_1, e_2 \ldots e_t\}$

$W\left(T_{\min}\right) = \sum\limits_{i=1}^{t} \text{ weight } \left(e_i\right) < \sum\limits_{i=1}^{t} \text{ weight } \left(e'_i\right) = W\left(T'_{\min}\right)$

$$
\begin{aligned}
\Rightarrow W'\left(T'_{\min}\right) = \sum_{i=1}^{t} \text{weight}'\left(e'_i\right) &= \sum_{i=1}^{t}\left(weight\left(e'_i\right)+c\right) \\
&= \sum_{i=1}^{t} \text{weight}\ \left(e'_i\right)+tc \\
&= W\left(T'_{\min}\right)+tc \\
&> W\left(T_{\min}\right)+tc \\
&= \sum_{i=1}^{t}\ \text{weight}\ \left(e_i\right)+tc \\
&= \sum_{i=1}^{t}\left(weight\left(e_i\right)+c\right) \\
&= \sum_{i=1}^{t}\ \text{weight}\ '\left(e_i\right) \\
&= W'\left(T_{\min}\right)
\end{aligned}
$$

$\because T'_{\min}$ is a MST in the modified-weight graph

$\therefore W'\left(T'_{\min}\right) \leqslant W'\left(T_{\min}\right)$

$\Rightarrow$ There is a contradiction.

$\Rightarrow$ Our supposition is false

$\therefore T'_{min}$ has the minimum weight in the original given graph (#)

By (*) and (#),

T'_{min} is also a MST in the original given graph

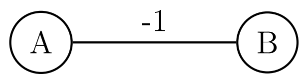
$\Rightarrow$ The new algorithm outputs an MST.

QED.

b.

We can't get the shortest path to some vertices, as there may be no shortest path to some vertices in a graph with negative weights (no lower bound for the weights of some paths)

Consider this graph:



As we do not need the path to be simple, then the path from A to B can be:

$\{A, B\}$ weight $= -1$

$\{A, B, A, B\}$ weight $= -3$

$\{A, B, A, B, A, B\}$ weight $= -5$

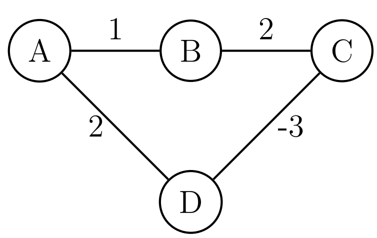
$\underbrace{\{A, B, A, B, ..., A, B\}}_{n\text{“}A,B\text{”}}$ weight $= -2n + 1$

we can get infinite paths by walking back and forth between A and B n times ($n \in N$) and stop at B, the weights of those paths will decrease as n increase and will never come to an end as n has no upper bound. (i.e. the weight can be $-\infty$)

$\Rightarrow$ then we can't get the shortest path from A to B

c. Disprove,

Consider this graph:



if we set A be the starting point, then by Dijkstra's algorithm:

①

A	B	C	D	Distance tree: {}
0	∞	∞	∞	

②

B D C Distance tree: $\{\}$

1 2 ∞

A A

③

D C Distance tree: $\{(A, B, 1)\}$

2 3

A B

④

C Distance tree: $\{(A, B, 1), (A, D, 2)\}$

-1

D

⑤ Distance tree: $\{(A, B, 1), (A, D, 2), (D, C, -1)\}$

$\Rightarrow \delta(D) = 2$

but in our graph $\delta(D)$ should be $1 + 2 - 3 = 0$

$0 < 2$

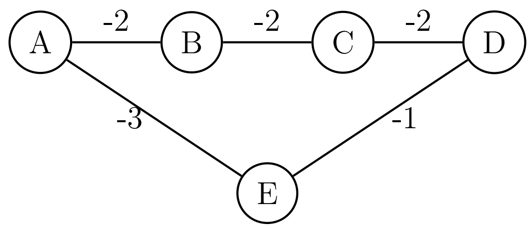
$\therefore$ We fail to find the single-source shortest simple path from A to D with Dijkstra's algorithm.

This is because if we want to have $\delta_{\text{fin}}(D) = 0$, we need to dequeue A, B, C first. But in the algorithm $p(C) = \delta_{\text{fin}}(C) = 3 > p(D) = \delta_{\text{fin}}(D) = 2$. This means we have to dequeue D first in the algorithm and that makes our algorithm fail to find the single-source shortest simple path from A to D.

QED.

d. Disprove

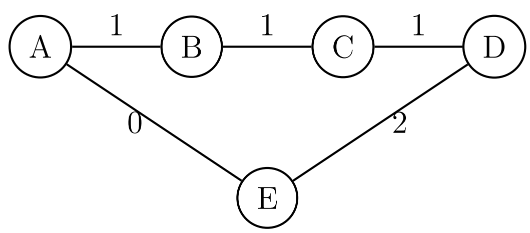
Consider the graph



the single-source shortest simple path from A to D is $\{A, B, C, D\}$ weight $= -6$

but by the modified Dijkstra's algorithm:

The graph will become



After line 5 in modified Dijkstra's algorithm:

In the single-source shortest simple paths that output by line 5, the shortest path from A to D will be $\{A, E, D\}$ with weight $= 2$

Now put it back to the original graph.

$\{A, E, D\}$ is weighted -4 , which is larger than the actual shortest simple path from A to D $\{A, B, C, D\}$ with weight $= -6$.

$\Rightarrow$ The new algorithm fails to output the right shortest single path for A to D.

$\Rightarrow$ the new algorithm doesn't output all the single-source shortest simple paths

$\Rightarrow$ so disprove the statement.

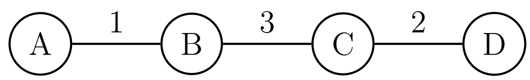
QED.

Q$_3$

a.

Disprove:

Consider the graph:



According to the new algorithm,

$L = \{(A, B, 1), (C, D, 2), (B, C, 3)\}, V = \{A, B, C, D\}$

After processing $(A, B, 1)$ and $(C, D, 2)$, A, B, C, D are all deleted from V and now we are going to process $(B, C, 3)$.

However, based on the algorithm, B and C are not in V, so nothing will be processed and (B, C) won't be added to T.

So after this algorithm finished, $T = \{(A, B), (C, D)\}$ will be returned.

But B and C now are disconnected.

$\therefore T$ is not a MST

$\Rightarrow$ This algorithm can't always output a minimum spanning tree

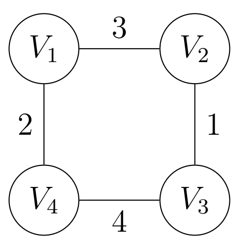
It's a false algorithm.

QED.

b.

Disprove:

Consider the graph



According to the algorithm,

$V\_S = \{v_1, v_2)$ $\quad E\_S = \{(v_1, v_2)\}$

$V\_T = \{v_3, v_4\}$ $\quad E\_T = \{(v_3, v_4)\}$

$\therefore MST\_S = \{(v_1, v_2)\}$, $MST\_T = \{(v_3, v_4)\}$

$\because$ weight $(v_2, v_3) = 1 <$ weight $(v_1, v_4) = 2$

$e = \{(v_2, v_3)\}$

$\therefore MST = \{(v_1, v_2), (v_3, v_4)\,(v_2, v_3)\}$ will be returned

The weight will be $1 + 3 + 4 = 8$.

But the MST in this graph is $\{(v_4, v_1), (v_1, v_2), (v_2, v_3)\}$

So the returned one is not a MST and this algorithm can't guarantee to output a MST.

$\Rightarrow$ The new algorithm is false.

QED.