Question 1.

The dual is the following linear program:

$$\begin{aligned}
&\max \sum_{S \in \mathcal{S}} y_S \\
&\forall e \in E, \sum_{S \in \mathcal{S}: e \in \delta(S)} y_S \leq w(e) \\
&\forall s \in \mathcal{S}, y_S \geq 0
\end{aligned}$$

Question 2.

Each iteration of the loop considers the variable y_C corresponding to the connected component C of the graph $G' = (V, F)$ containing s. Let's prove that each connected component C can be considered only once. At the end of an iteration, an edge e' is added to F. However, the edge e' corresponds to a dual constraint where y_C appears, that is, $e' \in \delta(C)$. This means that e' is connected to C. Therefore, adding e' to F will also add it to C.

C is therefore strictly monotonic (strictly increasing), so each C can only be considered in one iteration at most, so each y_C can only be increased in one iteration at most.

Question 3.

Since each dual variable can only be increased once, the algorithm cannot run for more iterations than the number of dual variables, therefore the algorithm terminates.

If the algorithm terminates, it means the looping condition is not satisfied anymore, that is, there is a path connecting s to t in F. Returning such a path P, as the algorithm does, gives a solution to the problem.

Question "3.5".

Lemma 1. At any step of the algorithm, the set F is a tree that contains s.

Let's call i the number of edges added to C.

1. Let's consider the case $i = 1$. This is just after the first iteration of the algorithm. That first iteration considered the set $C = \{s\}$ and selected an edge e' such that $e' \in \delta(C)$. That edge e' is therefore connected to s, the only vertex in C. At the end of the iteration, F consists of the single edge e': it is trivially a tree containing s.
2. Now assuming the lemma holds for $i - 1$, let's consider the i^{th} step. That step will add an edge e' to F. We know that, by construction,

this edge is connected to F (see question 2): it only remains to prove that adding this edge keeps F a tree, that is, it doesn't create a cycle. What would it mean to create a cycle? It would mean that e' would join two vertices that were already in C, not only one. But then e' wouldn't belong to $\delta(C)$, therefore increasing y_C couldn't have saturated the dual constraint corresponding to e'. This is in contradiction with the fact that e' was selected by the algorithm.

Therefore, the lemma is proved by induction for the whole duration of the algorithm.

Question 4.

Let's call x^* the optimal fractional solution of the primal LP. Since the primal LP is a minimization problem, by weak duality, $val(y^*) \leq val(x^*)$. But, since the primal LP is a minimization problem, the optimal integral solution also verifies $val(x^*) \leq val(P^*)$. Therefore, $val(y^*) \leq val(P^*)$.

Question 5.

The solution y at the first step of the algorithm, $y \leftarrow 0$, is trivially feasible for the dual. But then, at each step of the algorithm, care is taken not to invalidate any dual constraint. So, at the end of the algorithm, y is still feasible.

Question 6.

Since y is a feasible solution to the dual and the dual is a maximization problem, $val(y) \leq val(y^*) \leq P^*$.

Question 7.

An edge e in P is also a member of F: it was selected at some iteration of the algorithm. This means the dual constraint corresponding to e was saturated during that iteration of the algorithm. In turn this means that

$$\sum_{S \in \mathcal{S}: e \in \delta(S)} y_S = w(e)$$

Question 8.

$$\sum_{e \in P} w(e) = \sum_{e \in P} \sum_{S \in \mathcal{S}: e \in \delta(S)} y_S$$

Question 9.

In the sum given above (question 8), each y_S appears once for each e in P that is also a member of $\delta(S)$. We can therefore turn the sum into:

$$\sum_{e \in P} w(e) = \sum_{S \in \mathcal{S}} y_S \cdot |P \cap \delta(S)|$$

Question 10.

Let's assume there is a $S \in \mathcal{S}$ such that $y_S > 0$ and $|P \cap \delta(S)| \geq 2$. This means there are at least two edges in $\delta(S)$ that are also part of P.

Since $y_S > 0$, we can focus on the step of the algorithm where S was considered (i.e. S was the connected component C in that iteration). Each of the edges therefore were connected, each by a single vertex, to F. Let's name the two vertices v' and v''. There is a single path P' in F leading from s to v' and a single path P'' leading from s to v''. Since P itself is a path leading from s, it can include either P' or P'', but not both at the same time. Therefore it is impossible for P to use both edges.

Therefore, it is impossible to have both $y_S > 0$ and $|P \cap \delta(S)| \geq 2$.

Question 11.

By question 10, we know that $y_S > 0$ implies $|P \cap \delta(S)| \leq 1$. By combining with question 9, we deduce that

$$val(P) = \sum_{e \in P} w(e) \leq \sum_{S \in \mathcal{S}} y_S = val(y)$$

Question 6 lets us finally conclude that $val(P) \leq val(P^*)$, therefore P is optimal.

Question 12.

By ensuring that P is a path and not a larger subset of the tree, we are able to prove the contradiction in question 10. A generic subset of the tree could span any number of edges in $\delta(S)$ such that $y_S > 0$ (this is trivial, for example, in the case of F itself), and we wouldn't be able to reduce the equation in question 9 to the simpler equation in question 11.

Question 13.

Let's consider the first step of the primal-dual algorithm. At that step, $C = \{s\}$ and therefore $\delta(C)$ is the set of edges connected to s. The edge e' which is chosen at that step therefore is of the form (s, i). Furthermore,

since at that point $y = 0$ except for y_C which is being increased, the set of dual constraints can be rewritten as $y_C \leq w(e)$ for each $e \in \delta(C)$. The constraint which is first saturated by increasing y_C is the one with the lowest $w(e)$; in other words, the chosen edge (s, i) is the one with the lowest weight.

This matches exactly how the second vertex i is chosen in Djikstra's algorithm. QED.

Question 14.

The vertex which is added in the next iteration is the vertex j for which an edge (j, i) (for any i connected to j) has its constraint first saturated by increasing y_{C_0}, that is, it is the vertex j with the smallest $l(j)$.

Question 15.

We denote $S' = S \cup \{j\}$ where j is the vertex added to C_0 in the previous iteration. The set of edges appearing in $\delta(S')$ but not in $\delta(S)$ is the set of edges connecting j with another vertex not in S; or in formal notation, $\{(j, k) \in E : k \in V \setminus S\}$.

Question 16.

After the algorithm added j to S, the next step of the algorithm will consider the edges in $\delta(S')$. This means the constraints corresponding to the edges in $\delta(S')$ but not in $\delta(S)$ must now also be considered. Those constraints correspond to the set of edges (j, k) such that $k \notin S$. This means that for each neighbour k of j such as $k \notin S$, $l(k)$ must be updated to also consider $a((j, k))$.

Or in other words, for each neighbour k of j such as $k \notin S$, $l(k)$ must be updated with $min(l(k), w((j, k)) + l(j))$.

Question 17.

We now see that the selection of a vertex j by the primal-dual algorithm according to the values of $l(j)$ and the updating of $l(k)$ at each step of the algorithm matches exactly how Djikstra's algorithm selects a vertex i and updates $d(j)$ for the neighbours of the newly selected vertex. Therefore, both algorithms select exactly the same vertices in the same order.