Question 1

(a)

The flow is conserved at node C as the total flow in and total flow out of C is equal:

$$\begin{aligned} I(C) &= f_{(s,C)} + f_{(B,C)} = 2 + 2 \\ O(C) &= f_{(C,t)} = 4 \\ \therefore I(C) &= O(C) \end{aligned}$$

(b)

Consider the cut $U = \{s, A, B, C\}$.

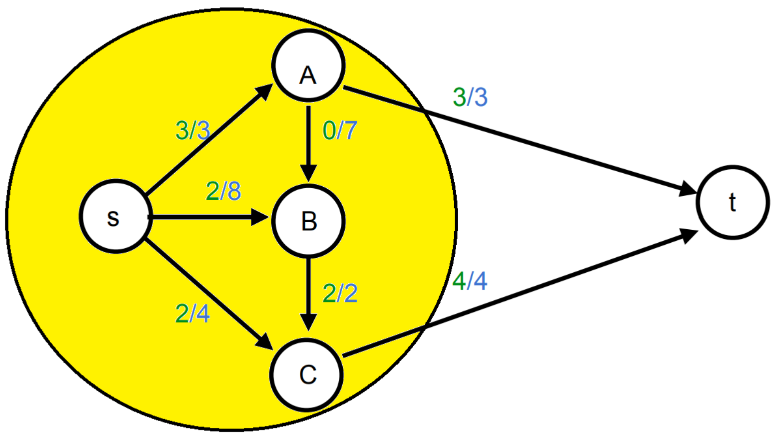

Figure 1: $U = \{s, A, B, C\}$

The capacity of the cut U is the sum of the capacities of the arcs leaving U:

$$\begin{aligned} \text{cap}(U) &= u_{(A,t)} + u_{(C,t)} \\ &= 3 + 4 \\ &= 7 \end{aligned}$$

Meanwhile, the flow out of this cut is equal to:

$$\begin{aligned} \text{val}(f) &= f_{(A,t)} + f_{(C,t)} \\ &= 3 + 4 \\ &= 7 \end{aligned}$$

By the max-flow min-cut theorem, the network flow is optimal when the flow of the network is equal to the capacity of an s-t cut. This is the case, as:

$$\text{cap}(U) = 7 = \text{val}(f)$$

and therefore the current network flow is optimal.

(c)

In this flow network, (1) all arc capacities are integer and (2) the min-cost flow is finite. Thus, by the integrality property of flow network, we can expect an optimal integer solution for this flow network.

(d)

Consider the path $s - B - A - t$.

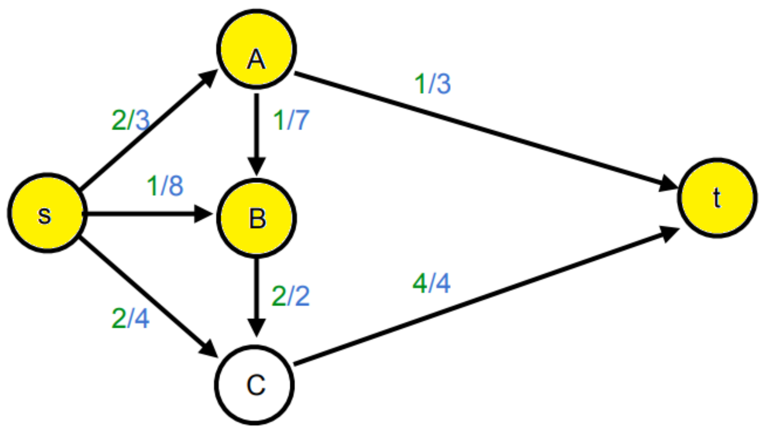

Figure 2: $s - B - A - t$

Since $f_{(s,B)} < u_{(s,B)}$, $f_{(B,A)} > 0$, and $f_{(A,t)} < u_{(A,t)}$, this path is augmenting. Thus, we can push flow along the path as much as $\delta = \max\{u_{(s,B)} - f_{(s,B)}, f_{(B,A)}, u_{(A,t)} - f_{(A,t)}\} = 1$. This gives the following flow network with new flow of $\text{val}(f) = f_{(A,t)} + f_{(C,t)} = 2 + 4 = 6$.

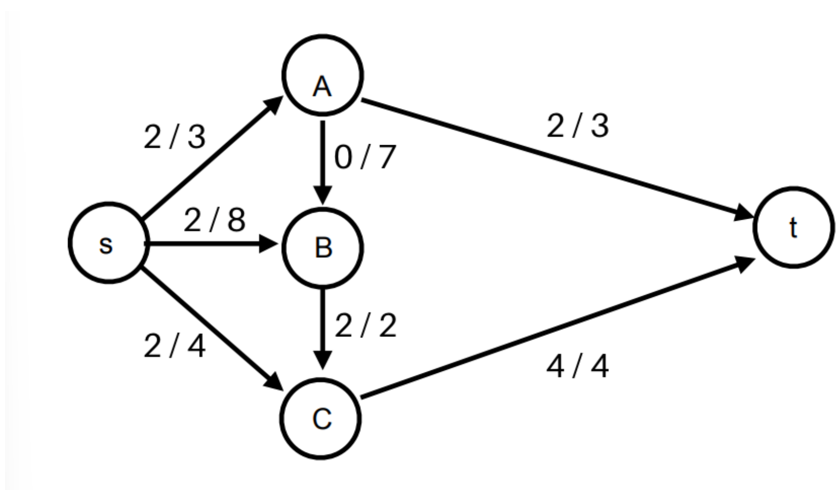

Figure 3: Flow network after the 1st iteration

Next, we select path $s - A - t$.

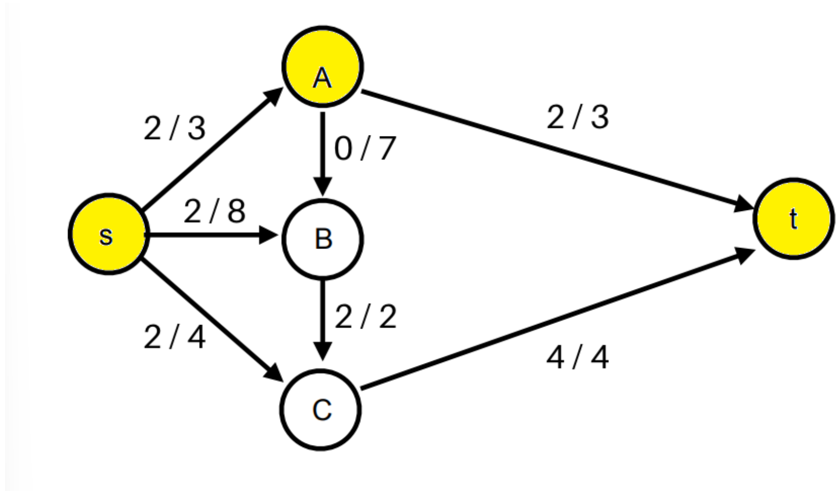

Figure 4: $s - A - t$

Since $f_{(s,A)} < u_{(s,A)}$ and $f_{(A,t)} < u_{(A,t)}$, $s - A - t$ is an augmenting path. Thus, we can push flow along this path as much as $\delta = \max\{u_{(s,A)} - f_{(s,A)}, u_{(A,t)} - f_{(A,t)}\} = 1$.

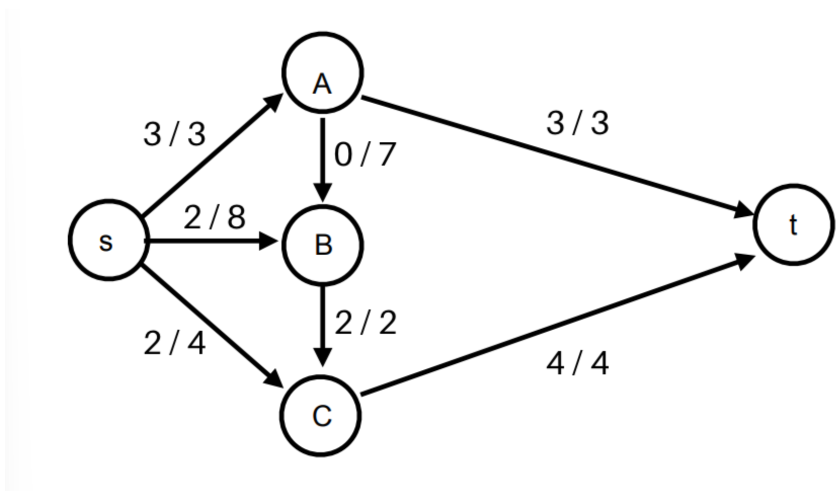

Figure 5: Flow network after the 2nd iteration

This gives the above flow network with new flow of $\text{val}(f) = f_{(A,t)} + f_{(C,t)} = 3 + 4 = 7$. Since now there are no more augmenting paths, the flow network is optimal. Indeed, this flow network is exactly the same as the one discussed in (b).

Question 2

(a)

Find the basic feasible solution as instructed in the problem. We get the following spanning tree, which illustrates our basic feasible solution.

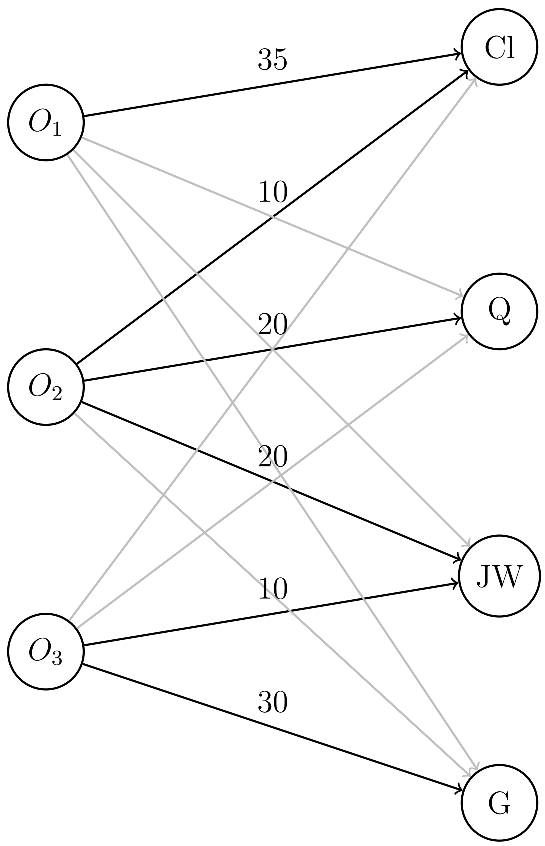

Figure 6: Spanning Tree

The basic arcs are those that appear in the spanning tree in black color, i.e. $(O_1, \mathrm{Cl}), (O_2, \mathrm{Cl})$, $(O_2, \mathrm{Q}), (O_2, \mathrm{JW}), (O_3, \mathrm{JW})$, and (O_3, G).

The rest of the arcs which appear in light grey are the nonbasic arcs, i.e. (O_1, Q), (O_1, JW), (O_1, G), (O_2, G), (O_3, Cl), and (O_3, Q).

The basic arcs have flow as indicated in the spanning tree, whereas the nonbasic arcs have 0 flow.

(b)

Set the simplex multiplier at G to be 0. Using this information and the formula $c(i,j) = y_i - y_j$, calculate the simplex multipliers of the remaining nodes.

$$\begin{aligned} c(O_3, \mathrm{G}) &= y_{O_3} - y_{\mathrm{G}} \\ 5 &= y_{O_3} - 0 \\ &\boxed{\therefore y_{O_3} = 5} \end{aligned}$$

$$\begin{aligned} c(O_3, \mathrm{JW}) &= y_{O_3} - y_{\mathrm{JW}} \\ 16 &= 5 - y_{\mathrm{JW}} \\ &\boxed{\therefore y_{\mathrm{JW}} = -11} \end{aligned}$$

$$\begin{aligned} c(O_2, \mathrm{JW}) &= y_{O_2} - y_{\mathrm{JW}} \\ 13 &= y_{O_2} + 11 \\ &\boxed{\therefore y_{O_2} = 2} \end{aligned}$$

$$\begin{aligned} c(O_2, \mathrm{Q}) &= y_{O_2} - y_{\mathrm{Q}} \\ 12 &= 2 - y_{\mathrm{Q}} \\ &\boxed{\therefore y_Q = -10} \end{aligned}$$

$$\begin{aligned} c(O_2, \mathrm{Cl}) &= y_{O_2} - y_{\mathrm{Cl}} \\ 9 &= 2 - y_{\mathrm{Cl}} \\ &\boxed{\therefore y_{\mathrm{Cl}} = -7} \end{aligned}$$

$$\begin{aligned} c(O_1, \mathrm{Cl}) &= y_{O_1} - y_{\mathrm{Cl}} \\ 8 &= y_{O_1} + 7 \\ &\boxed{\therefore y_{O_1} = 1} \end{aligned}$$

With this, our spanning tree becomes:

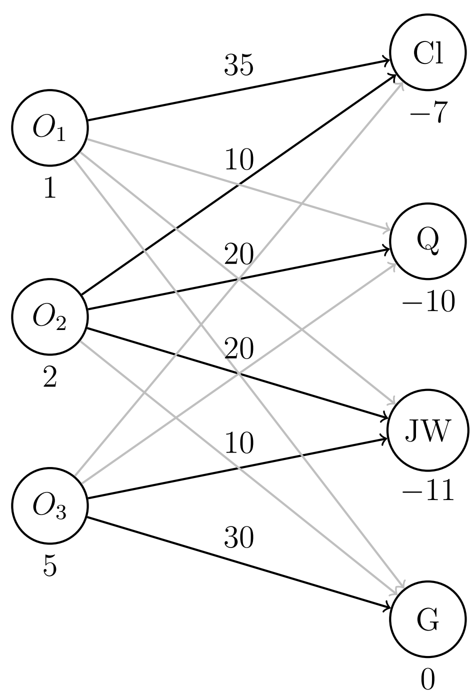

Figure 7: Flow Network, simplex multiplier under each node

(c)

Calculate the reduced costs of the nonbasic arcs using the formula $\bar{c}(i,j) = c(i,j) - y_i + y_j$. This gives us:

$$\begin{aligned}
\bar{c}(O_1, \text{Q}) &= c(O_1, \text{Q}) - y_{O_1} + y_{\text{Q}} \\
&= 6 - 1 - 10 = -5 \\
\bar{c}(O_1, \text{JW}) &= c(O_1, \text{JW}) - y_{O_1} + y_{\text{JW}} \\
&= 10 - 1 - 11 = -2 \\
\bar{c}(O_1, \text{G}) &= c(O_1, \text{G}) - y_{O_1} + y_{\text{G}} \\
&= 9 - 1 + 0 = 8 \\
\bar{c}(O_2, \text{G}) &= c(O_2, \text{G}) - y_{O_2} + y_{\text{G}} \\
&= 7 - 2 + 0 = 5 \\
\bar{c}(O_3, \text{Cl}) &= c(O_3, \text{Cl}) - y_{O_3} + y_{\text{Cl}} \\
&= 14 - 5 - 7 = 2 \\
\bar{c}(O_3, \text{Q}) &= c(O_3, \text{Q}) - y_{O_3} + y_{\text{Q}} \\
&= 9 - 5 - 10 = -6
\end{aligned}$$

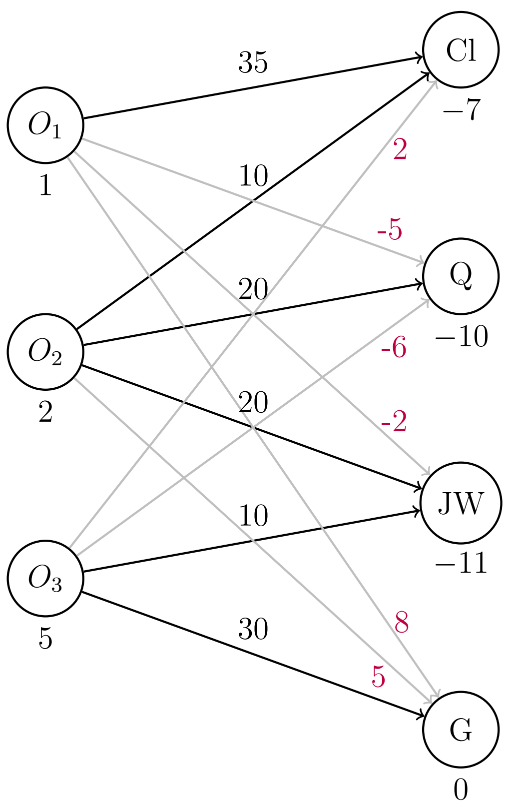

Figure 8: Flow Network, reduced cost of nonbasic arcs in purple

(d)

The current BFS is not optimal due to some of the reduced costs being negative: $\bar{c}(O_1, \text{Q}) = -5$, $\bar{c}(O_1, \text{JW}) = -2$, and $\bar{c}(O_3, \text{Q}) = -6$.

(e)

Arc (O_3, Q) has the most negative reduced cost. As such, we convert this arc into a basic arc and create the following unique cycle:

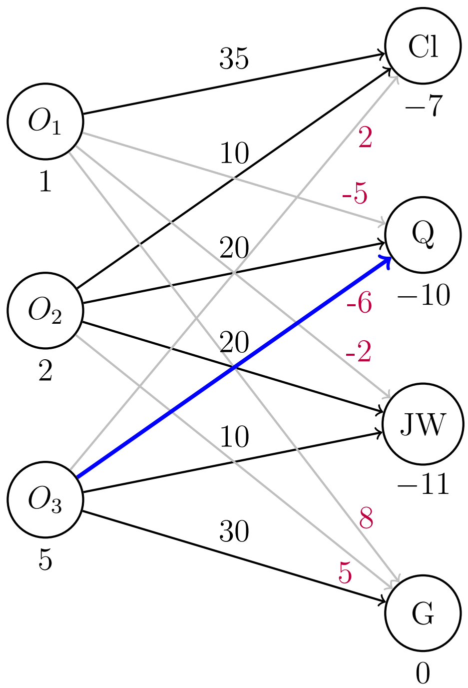

Figure 9: Flow Network, with new arc between O_3 and Q

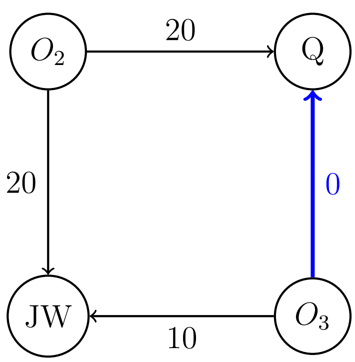

Figure 10: Unique cycle, simplified view

(f)

Orient the cycle such that the nonbasic arc is forward. Increase flow on forward arcs by t and decrease flow on backward arcs by t.

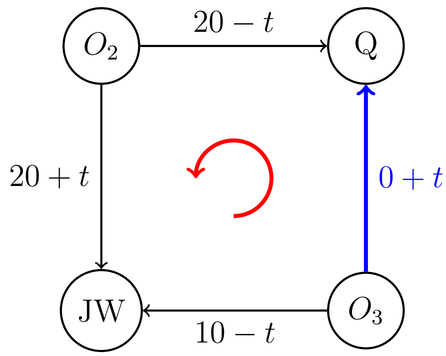

Figure 11: Unique cycle, simplified view

As the flows must be non-negative, we have

$$0 + t \geq 0 \qquad 20 - t \geq 0 \qquad 10 - t \geq 0 \qquad 20 + t \geq 0$$

and thus the biggest value t can take is 10. Using this t value, we push flow into the network and produce the following new basic feasible solution. The arc (O_3, Q) becomes basic whereas the arc (O_3, JW) becomes nonbasic. The remaining arcs remain basic or nonbasic as they were.

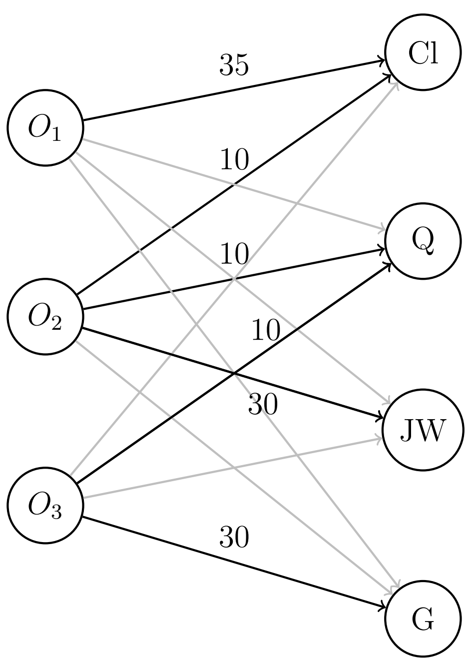

Figure 12: New Flow Network

Question 3

You're tasked with creating an algorithm to insert line breaks in a text, made up of words $w_1, \ldots, w_n$, to maximize aesthetic appeal. The "beauty score" for any arrangement of line breaks is the sum of the scores for each line. The score of an individual line starting with word i and ending with word j is

$$\text{score}(w_i, w_j) = \left\{ \begin{array}{ll} +\infty & \text{if } w_i, w_{i+1}, \ldots, w_j \text{ don't fit in a line} \\ (\text{pagewidth} - \text{total width})^2 & \text{otherwise} \end{array} \right\}$$

where total width means the space occupied by words $w_i, w_{i+1}, \ldots, w_j$ (including empty spaces). Give the pseudocode of a dynamic programming algorithm for solving this problem, including the recursion and the base cases, and explain your rationale.

To solve this problem, define $\text{length}(w_i, w_j)$ as the number of characters needed to write word i to word j. By this information, we can redefine $\text{score}(w_i, w_j)$ into:

$$\text{score}(w_i, w_j) = \left\{ \begin{array}{ll} +\infty & \text{if length}(w_i, w_j) > \text{pagewidth} \\ (\text{pagewidth} - \text{total width})^2 & \text{otherwise} \end{array} \right\}$$

The function $\text{score}(w_i, w_j)$ essentially describes the "cost" of a certain line break arrangement; the lower the score, the more beautiful the arrangement. So we want to minimize this function using Dynamic Programming.

Now to solve the problem, consider the following approach: given a text input, split the text into 2 lines where the cost of the 1st line is minimized. Assuming there are n words in the original text, we let $\text{S}[i] = \min\{\text{score}(w_i, w_j)\}$ be the minimum score to write w_i to w_n. So $\text{S}[1]$ denotes the original problem as it refers to the minimum score required to write w_1 to w_n.

By default, we know that $\text{S}[n+1] = 0$ since there are no more words to be written after n. This forms our base case, which enables us to identify the following recursion pattern for $\text{S}[i]$.

$$\text{S}[i] = \min\{\text{score}(w_i, w_j) + \text{S}[j]\}, \quad \forall j = i+1, \ldots, n+1$$

Hence, we can write the pseudocode to implement the recursion and solve the original problem:

Algorithm 1 Minimize Beauty Score

```
1: S[n+1] = 0
2: for i from n to 1 do
3:     S[i] = min{score(i, j) + S[j] : j ∈ {i+1, ..., n+1}}
4: end for
5: return S[1]
```

For instance, consider the case when $n = 3$ with w_1, w_2, w_3 as the words in a text. We initiate the algorithm from $i = 3$:

$$\begin{aligned} i &= 3 \\ \mathrm{S}[3] &= \min\{\mathrm{score}(3, j) + \mathrm{S}[j]\}, \quad \forall j = 4 \end{aligned}$$

Here, since $j = 4$ and $\mathrm{S}[4] = \mathrm{S}[n + 1] = 0$, we get:

$$\begin{aligned} \mathrm{S}[3] &= \min\{\mathrm{score}(3, 4) + \mathrm{S}[4]\} \\ &= \mathrm{score}(3, 4) \end{aligned}$$

where the value of $\mathrm{score}(3, 4)$ depends on the value of w_3 and can be computed using the definition provided previously ($w_4 = 0$ since the word technically does not exist). Now, we can continue the loop for $i = 2$:

$$\begin{aligned} i &= 2 \\ \mathrm{S}[2] &= \min\{\mathrm{score}(2, j) + \mathrm{S}[j]\}, \quad \forall j = 3, 4 \\ &= \min\{\mathrm{score}(2, 3) + \mathrm{S}[3],\ \mathrm{score}(2, 4) + \mathrm{S}[4]\} \\ &= \min\{\mathrm{score}(2, 3) + \mathrm{score}(3, 4),\ \mathrm{score}(2, 4)\} \end{aligned}$$

where S[2] depends on the values of w_1, w_2, and w_3 and takes whichever is smaller between $\mathrm{score}(2, 3) + \mathrm{score}(3, 4)$ and $\mathrm{score}(2, 4)$. Now, we can continue the loop for $i = 1$:

$$\begin{aligned} i &= 1 \\ \mathrm{S}[1] &= \min\{\mathrm{score}(1, j) + \mathrm{S}[j]\}, \quad \forall j = 2, 3, 4 \\ &= \min\{\mathrm{score}(1, 2) + \mathrm{S}[2],\ \mathrm{score}(1, 3) + \mathrm{S}[3],\ \mathrm{score}(1, 4) + \mathrm{S}[4]\} \\ &= \min\Big\{\mathrm{score}(1, 2) + \min\{\mathrm{score}(2, 3) + \mathrm{score}(3, 4),\ \mathrm{score}(2, 4)\}, \\ &\qquad \mathrm{score}(1, 3) + \mathrm{score}(3, 4),\ \mathrm{score}(1, 4)\Big\} \end{aligned}$$

where $S[1]$ depends on the exact values of w_1, w_2, and w_3 but can be computed if we are given these values. Once we solve for S[1], we would solve an instance of the problem for $n = 3$.