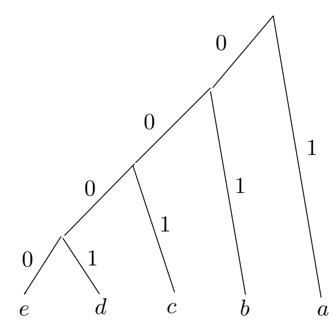
Problem 1.

1. T
2. F
3. T
4. T
5. F
6. F
7. T

Problem 2. $\text{opt}(1) = 3$, $\text{opt}(2) = 6$, $\text{opt}(3) = 9$, $\text{opt}(4) = 12$, $\text{opt}(5) = 15$, $\text{opt}(6) = 18$, and $\text{opt}(7) = 21$.

Problem 3. ad, cd, de, ab, ef.

Problem 4.



Problem 5. Let A be the input array for the inversion counting problem. Construct a set P of n points as follows: for each $i \in [1, n]$, add to P the point $p_i = (i, -A[i])$. Observe that (i, j) is an inversion if and only if point p_j dominates p_i. We run a dominance counting algorithm to find, for each point p_j ($j \in [1, n]$), the number c_j of points dominated by p_j. Then, the number of inversions in A can be obtained as $\sum_{j=1}^{n} c_j$. As P can be constructed in $O(n)$ time, the whole algorithm uses $f(n) + O(n)$ time to solve the counting inversion problem.

Problem 6. We can trivially encode each letter in $\log_2 n$ bits: assign the $\log_2 n$-bith binary representation of i to the i-th letter for each $i \in [1, n]$. This gives a prefix code whose average length is $\log_2 n$. As Huffman's algorithm constructs an optimal prefix code, the code's average length must be at most $\log_2 n$.

Problem 7.

$$\text{opt}(n) = \begin{cases} 0 & \text{if } n = 0 \\ \max\{P[n], (-c) + \max_{i=1}^{n-1}(P[i] + \text{opt}(n-i))\} & \text{otherwise} \end{cases} \tag{1}$$

Problem 8. This problem can be converted to k-selection. First, find the median m of S in $O(n)$ expected time. Then, construct a set $T = \{|x - m| \mid x \in S\}$. Finally, use k-selection to find the k-th smallest number of T in $O(n)$ expected time. If $|x - m|$ is the number returned, then output every $y \in S$ satisfying $|y - m| \leq |x - m|$.

Problem 9. Let $I_1, I_2, ..., I_t$ be the sequence of intervals picked by the algorithm. We will prove the claim: for each $i \in [1, t]$, there is an optimal solution containing $\{I_1, I_2, ..., I_i\}$.

To prove the base case ($i = 1$), notice that I_1 must be the longest interval in $\mathcal{I}$ starting from 0. Take an arbitrary optimal solution T. Clearly, T must contain an interval I' covering 0. Replacing I' with I_1 gives another optimal solution.

Assuming that the claim holds for $i = k < t$, next we will prove its correctness for $i = k + 1$. Let T be an arbitrary optimal solution containing $I_1, I_2, ..., I_k$. Consider the value a at Line 2 right before our algorithm picks I_{k+1}. Clearly, T must contain an interval I' covering $a + 1$. Replacing I' with I_{k+1} gives another optimal solution.

Problem 10. First, find the largest element (i.e., the $2^{\log_2 n}$ smallest) of S in $O(n)$ time. Then, use k-selection to find the $(n/2)$-th smallest element e_1 of S in $O(n)$ expected time. Remove from S all the elements that are greater than e_1. Now, $|S| = n/2$. Use k-selection again to find the $(n/4)$-th smallest element e_2 of S in $O(n/2)$ expected time. Remove from S all the elements that are greater than e_2. Use k-selection again to find the $(n/8)$-th smallest element e_3 of S in $O(n/8)$ expected time. Remove from S all the elements that are greater than e_3. Repeat in the same fashion until S has only one element. The total expected time is $O(n) + O(n/2) + O(n/4) + ... + O(1) = O(n)$.