Problem 1. F, F, F, F, F, T, T, T, T, T.

Problem 2. Free marks.

Problem 3. First apply k-selection to find the k_1-th smallest integer x_1 in $O(n)$ expected time. Then apply k-selection again to find the k_2-th smallest integer x_2 in $O(n)$ expected time. Then scan S once more to report all the numbers $y \in S$ satisfying $x_1 \leq y \leq x_2$. The last step takes $O(n)$ time.

Problem 4. First, compute the set S of all 5-cycles in G. For this purpose, enumerate all possible sequences $(v_1, v_2, v_3, v_4, v_5)$ where each $v_i \in V$ $(1 \leq i \leq t)$, and check, for every such sequence, whether the edges (v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_5), and (v_5, v_1) exist in the graph G. The problem is now converted to finding the smallest set of edges that hit every 5-cycle in S. This is an instance of the hitting set problem. Now, apply the standard approximation algorithm.

The running time is clearly polynomial in $|V|$. The approximation ratio is at most $1+\ln|S| = O(\log|V|)$.

Problem 5. By the white path theorem, the descendants of c in the DFS-forest are: c, b, d, g, f, and h.

Problem 6. Prof. Goofy's algorithm runs in $O(n^{102})$ time. To find a vertex cover with the smallest size, we can apply his algorithm by setting $k = 1, 2, 3, ...$, respectively, and stop as soon as his algorithm returns a vertex cover for the first time. The overall running time is $O(n^{103})$.

Problem 7. Let $\text{OPT}(\ell)$ be the optimal cost to cut a rod of ℓ meters into segments of 1 meter long. Thus, $\text{OPT}(1) = 0$. For $\ell \geq 2$, we have:

$$\text{OPT}(\ell) = A[\ell] + \min_{i=1}^{\ell-1} \Big(\text{OPT}(i) + \text{OPT}(\ell - i)\Big).$$

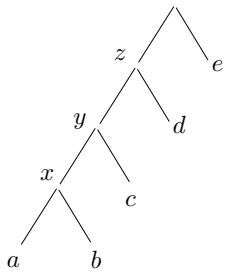
We may compute $\text{OPT}(\ell)$ in ascending order of ℓ. As it takes $O(\ell)$ time to compute $\text{OPT}(\ell)$, the total cost of computing $\text{OPT}(n)$ is $\sum_{\ell=1}^{n} O(\ell) = O(n^2)$.

Problem 8. First, generate a complete graph (i.e., a clique) $G^* = (S, E^*)$ where, for any two distinct vertices $u, v \in S$, the edge $\{u, v\}$ has a weight that equals the shortest path distance between u and v in the original graph G. As every shortest path can be computed in polynomial time, overall the $\Theta(|S|^2)$ shortest paths can be computed in polynomial time.

Then, the value of OPT corresponds to the shortest length of all Hamitonian cycles of G^*. Find a Hamitonian cycle C of length $2 \cdot \text{OPT}$. This can be done in polynomial time as discussed in the lecture.

We can now generate an S-round-trip walk from C as follows. For every two consecutive vertices u, v on C, replace the edge $\{u, v\}$ with a shortest path from u to v. The walk thus obtained has the same length as C, which as mentioned is at most $2 \cdot \text{OPT}$.

Problem 9. Let f_a, f_b, f_c, f_d, and f_e denote the frequencies of a, b, c, d, and e, respectively.



Observe that the algorithm must have (i) first merged a and b into a node x, (ii) then merged x and c into a node y, (iii) proceeded to merge y and d into a node z, and (iv) finally merged z with e. It thus follows that

$$\begin{aligned} f_c &\geq f_a = 1/8 \qquad \text{(due to (i))} && (1)\\ f_d &\geq f_x = f_a + f_b \qquad \text{(due to (ii))} && (2)\\ f_e &\geq f_y = f_a + f_b + f_c \qquad \text{(due to (iii))} && (3) \end{aligned}$$

Therefore

$$\begin{aligned} 1 &= f_a + f_b + f_c + f_d + f_e \\ &\geq 1/8 + 1/8 + f_c + (1/8 + 1/8) + (1/8 + 1/8 + f_c) \end{aligned}$$

which yiels $f_c \leq 1/8$. Combining this with (1) gives $f_c = 1/8$.

We argue that f_d must be precisely $2/8$. First note that (2) indicates $f_d \geq 2/8$. Thus, if $f_d > 2/8$, then $f_e < 1 - (1/8 + 1/8 + 1/8 + 2/8) = 3/8$, which would contradict (3).

It thus follows that $f_e = 3/8$.

Problem 10. Let T^* be an arbitrary MST of G. If e^* is not in T^*, we are done. Next, we consider the opposite scenario where e^* is in T^*.

Remoing e^* from T^* breaks T^* into two connected components (CCs). Let T_1 (resp., T_2) be the first (resp., second) CC. The cycle C must contain another edge $e \neq e^*$ such that e connects a vertex in T_1 with a vertex in T_2. Connecting T_1 and T_2 with e gives another tree T whose total weight cannot exceed that of T^*. Thus, T must be an MST.

Problem 11. Let us first consider the situation where all the edges of G are non-negative. Remove from G all the edges whose weights are positive; let G' be the graph induced by the remaining edges. Thus, G contains a cycle of weight 0 if and only if G' has a cycle. We can detect whether G' is cyclic using DFS in $O(|V| + |E|)$ time.

Now let us consider the general situation where G has negative edges. A crucial observation is that re-weighting does not change the weight of any cycle. Specifically, let $h : V \to \mathbb{R}$ be an arbitrary function. If $w : E \to \mathbb{R}$ be the original weight function of G, construct a new function $w' : E \to \mathbb{R}$ by defining for each edge $(u, v) \in E$:

$$w'(u, v) = w(u, v) + h(u) - h(v).$$

Thus, for any cycle $u_1, u_2, ..., u_t, u_1$, its weight under w' equals

$$w'(u_t, u_1) + \sum_{i=1}^{t-1} w'(u_i, u_{i+1})$$

which can be easily verified to be

$$w(u_t, u_1) + \sum_{i=1}^{t-1} w(u_i, u_{i+1})$$

namely, the cycle's weight under w.

Therefore, the problem boils down to finding a function h that makes $w'(u, v) \geq 0$ for all edges $(u, v) \in E$. As G has no negative cycles, such a function h can be found in $O(|V||E|)$ time, as discussed in the lecture (Johnson's algorithm).