1. Here is a simplified version of the problem which food delivery companies like Swiggy face. Let us call it the k-Swiggy problem. Like in the k-server problem, k delivery persons occupy points in a metric space. Each request consists of a (restaurant,customer) pair. To serve the request, Swiggy sends one delivery person to the restaurant, who picks up the order and delivers it to the customer. Let us assume that Swiggy gets the next request only after it has served the current request, so all the k persons are available to serve any request.

   Mathematically, each request i is a pair of points (s_i, t_i) from the metric space. Such a request is served by moving a server to s_i and then moving the same server from s_i to t_i. The cost incurred is, naturally, the total distance travelled by the k servers (inclusive of the distances from s_i to t_i). Let α_k and β_k denote the deterministic competitive ratios of the k-server and the k-Swiggy problem respectively. (Recall that $k \le \alpha_k \le 2k-1$.)

   (a) [**3 points**] Prove that $\alpha_k \le \beta_k$. (That is, show how you can use any online k-Swiggy algorithm for the k-server problem.)

   Answer: Given an instance I of k-server whose i'th request is r_i, let the instance I' of k-Swiggy be such that its i'th request is (r_i, r_i). Then $\text{OPT}(I) = \text{OPT}(I')$. Given a β_k competitive algorithm B for k-Swiggy, consider the algorithm A for k server which, given r_i, passes request (r_i, r_i) to B and mimics B's server movement. Then we have $A(I) = B(I')$. Thus, A is β_k-competitive, and hence $\alpha_k \le \beta_k$.

   (b) [**7 points**] Prove that $\beta_k \le \alpha_k + 2$. (That is, show how you can use any online k-server algorithm to get a k-Swiggy algorithm while adding at most 2 to the competitive ratio.)

   Answer: Given an instance I' of k-Swiggy whose i'th request is (s_i, t_i), let the instance I of k-server be such that its i'th request is s_i. The optimal server movement for I' is also a feasible server movement for I, and therefore, $\text{OPT}(I) \le \text{OPT}(I')$.

   Given an α_k-competitive algorithm A for k-server, define the algorithm B for k-Swiggy to be the following. On request (s_i, t_i), pass s_i to A. Use the same server to serve (s_i, t_i) as A uses to serve s_i, and bring back that server from t_i to s_i. By induction it is easy to prove that A and B have servers at the same set of locations after serving any sequence of requests, and so the behavior of B on the next request is well defined. The competitiveness guarantee of B can be derived as follows.

   $$B(I') = A(I) + 2\sum_i d(s_i, t_i) \le \alpha_k \cdot \text{OPT}(I) + 2 \cdot \text{OPT}(I') \le (\alpha_k + 2) \cdot \text{OPT}(I'),$$

   where d is the distance function of the metric space. Here we used the fact that for each i, some server moves from s_i to t_i in the optimal solution to I', and hence, $\text{OPT}(I') \ge \sum_i d(s_i, t_i)$. Thus, B is $(\alpha_k + 2)$-competitive, and hence, $\beta_k \le \alpha_k + 2$.

2. Recall the problem of pre-emptive online matching in the edge-arrival setting. Edges arrive one after the other. We can accept an edge only as soon as it comes, and we can discard it in future. We must always maintain a matching M in the graph. To prove competitiveness, we showed that we can maintain values x_v for each vertex v that constitute a feasible dual solution, such that for every new edge arrival, the change in the algorithm's matching's weight is at least the competitive ratio times the change in dual.

   (a) [**7 points**] We call an instance *b-special* if the ratio of the weights of any two edges is an integer power of b, for some parameter $b > 2$. Consider the following algorithm, which we will call algorithm $\mathcal{A}$. Start with $M = \emptyset$. For every new edge e with weight w, if w is greater than the weight of every conflicting edge in M, then remove those conflicting edges from M and add e to M. Else, discard e. Explain how you will maintain the dual variables to prove that on b-special instances, this algorithm has competitive ratio at least

   $$\frac{b-2}{2 \cdot (b-1)}.$$

Answer: Let w_e denote the weight of edge e. We initialize $x_u := 0$ for all vertices u. Whenever we add edge $e = (u, v)$ to M, we set $x_u := \max(x_u, w_e)$ and $x_v := \max(x_v, w_e)$. Thus, the dual variables can only increase over time, and hence, once a dual constraint is satisfied, it remains satisfied in future. Therefore, it is sufficient to prove that $x_u + x_v \geq w_e$ holds just after processing every edge $e = (u, v)$.

The dual variable updates ensure that for any vertex u, x_u is at least the weight of the edge matching u in M, if any. Suppose edge e arrives and is discarded. Then there must be a conflicting edge e' in M of weight $w_{e'} \geq w_e$. Suppose e and e' share vertex u. Then $x_u \geq w_{e'}$, and therefore, $x_u + x_v \geq w_e$. On the other hand, suppose e gets accepted in M. Then our dual variable updation rule ensures $x_u + x_v \geq w_e$. Thus, the dual remains feasible always.

We are left to prove $\Delta\text{primal} \geq (b-2)/(2b-2) \times \Delta\text{dual}$, which implies $((b-2)/(2b-2))$-competitiveness. Suppose edge e arrives and is discarded. Then $\Delta\text{primal} = \Delta\text{dual} = 0$. On the other hand, suppose e gets accepted and edges e_1 and e_2 get discarded as a result. This implies $w_e > w_{e_1}$ and $w_e > w_{e_2}$. Since the instance is b-special, we have $w_e \geq b w_{e_1}$ and $w_e \geq b w_{e_2}$. Now $\Delta\text{primal} = w_e - w_{e_1} - w_{e_2}$ and $\Delta\text{dual} = \Delta x_u + \Delta x_v \leq (w_e - w_{e_1}) + (w_e - w_{e_2})$. Thus,

$$\frac{\Delta\text{primal}}{\Delta\text{dual}} \geq \frac{w_e - w_{e_1} - w_{e_2}}{(w_e - w_{e_1}) + (w_e - w_{e_2})} = \frac{1}{2}\left(1 - \frac{w_{e_1} + w_{e_2}}{(w_e - w_{e_1}) + (w_e - w_{e_2})}\right),$$

and therefore,

$$\frac{\Delta\text{primal}}{\Delta\text{dual}} \geq \frac{1}{2}\left(1 - \frac{w_{e_1} + w_{e_2}}{(bw_{e_1} - w_{e_1}) + (bw_{e_2} - w_{e_2})}\right) = \frac{1}{2}\left(1 - \frac{1}{b-1}\right) = \frac{b-2}{2(b-1)},$$

as required. (If e has less than two conflicting edges then we take either w_{e_1} or w_{e_2} or both to be 0, appropriately.)

(b) Let us now analyze a randomized algorithm for pre-emptive online matching. For a fixed $b > 2$, define $f(w, r)$ to be the largest number less than or equal to w that is of the form b^{z+r} for some integer z (equivalently, $\log_b f(w, r) = \lfloor \log_b w - r \rfloor + r$). The algorithm $\mathcal{A}'$ is as follows. Choose r uniformly at random from $[0, 1]$. For every edge of weight w, reassign its weight to be $f(w, r)$ and pass it to algorithm $\mathcal{A}$. In the end, simply return the output of $\mathcal{A}$. Observe that the instance received by $\mathcal{A}$ is b-special, so the guarantee from the previous question applies.

i. **[6 points]** Prove that

$$\mathbb{E}_{r \sim U[0,1]}[f(w, r)] = w \cdot \frac{b-1}{b \ln b},$$

where $U[0, 1]$ denotes the uniform distribution on $[0, 1]$. (Hint: How is $\log_b f(w, r)$ distributed?)

Answer: We have $\log_b w - \log_b f(w, r) = \log_b w - r + \lfloor \log_b w - r \rfloor$. Here $\log_b w - r$ is distributed uniformly on $(\log_b w - 1, \log_b w]$, and hence its fractional part is distributed uniformly on $[0, 1)$. Thus, $\log_b f(w, r)$ is distributed uniformly on $(\log_b w - 1, \log_b w]$. Therefore,

$$\mathbb{E}_{r \sim U[0,1]}[f(w, r)] = \mathbb{E}_{X \sim U(\log_b w - 1, \log_b w]}[b^X] = \int_{\log_b w - 1}^{\log_b w} b^x dx = \frac{w - w/b}{\ln b} = w \cdot \frac{b-1}{b \ln b}.$$

ii. **[7 points]** Now prove that algorithm $\mathcal{A}'$ has competitive ratio at least

$$\frac{b-2}{2 \cdot (b-1)} \times \frac{b-1}{b \ln b} = \frac{b-2}{2b \ln b}.$$

You may assume that all the earlier claims are true, even if you couldn't prove them. (It is easy to see that the maximum value of the competitive ratio is attained when b satisfies $2 \ln b = b - 2$, and the maximum value is ≈ 0.1867, better than the deterministic competitive ratio.)

Answer: Fix an instance. Let M^* be the maximum weight matching in it. For every $r \in [0,1]$, define $M^*(r)$ and $M(r)$ as follows. $M^*(r)$ is the maximum weight matching in the same underlying graph but with every edge e assigned weight $f(w_e, r)$ instead of its original weight w_e, and $M(r)$ is the matching returned by $\mathcal{A}$ on this reweighted instance. Due to the competitiveness guarantee of $\mathcal{A}$, for every r, we have,

$$\sum_{e \in M(r)} f(w_e, r) \geq \frac{b-2}{2(b-1)} \times \sum_{e \in M^*(r)} f(w_e, r) \geq \frac{b-2}{2(b-1)} \times \sum_{e \in M^*} f(w_e, r),$$

where the second inequality follows from the optimality of $M^*(r)$. Taking expectation over r uniformly distributed on $[0,1]$, we get,

$$\mathbb{E}_{r \sim U[0,1]}\left[\sum_{e \in M(r)} f(w_e, r)\right] \geq \frac{b-2}{2(b-1)} \times \sum_{e \in M^*} \mathbb{E}_{r \sim U[0,1]}[f(w_e, r)] = \frac{b-2}{2(b-1)} \cdot \frac{b-1}{b \ln b} \times \sum_{e \in M^*} w_e,$$

where the equality follows from the previously proven claim applied to every $e \in M^*$. Since $f(w, r) \leq w$ for every w and r, we have,

$$\mathbb{E}_{r \sim U[0,1]}\left[\sum_{e \in M(r)} w_e\right] \geq \mathbb{E}_{r \sim U[0,1]}\left[\sum_{e \in M(r)} f(w_e, r)\right] \geq \frac{b-2}{2b \ln b} \times \sum_{e \in M^*} w_e.$$

Since $\mathcal{A}'$ returns $M(r)$ for $r \sim U[0,1]$, the required competitiveness guarantee of $\mathcal{A}'$ follows.

3. The goal of this question is to analyze another algorithm for the prophet problem. Assume that none of the independent non-negative random variables $X_1, \ldots, X_n$ have point masses, that is, their CDFs are all continuous. Their realizations are seen in the order $X_1, \ldots, X_n$. Let the threshold T be such that $\sum_{i=1}^n \Pr[X_i \geq T] = 1$. (Such a T exists because the CDFs are continuous.) Define the random variables $\overline{X}_1, \ldots, \overline{X}_n$ as $\overline{X}_i = X_i \cdot \mathbb{I}[X_i \geq T]$, (that is, $\overline{X}_i = X_i$ if $X_i \geq T$ and $\overline{X}_i = 0$ otherwise).

   (a) **[4 points]** Prove that $\mathbb{E}[\max_i X_i] \leq \sum_{i=1}^n \mathbb{E}\left[\overline{X}_i\right]$.

   Answer: We prove that for all $x > 0$, $\Pr[\max_i X_i \geq x] \leq \sum_{i=1}^n \Pr[\overline{X}_i \geq x]$. Integrating both sides from $x = 0$ to ∞ proves the claim. For $x \in (0, T)$, we have, $\Pr[\overline{X}_i \geq x] = \Pr[X_i \geq T]$, and therefore,

$$\sum_{i=1}^n \Pr[\overline{X}_i \geq x] = \sum_{i=1}^n \Pr[X_i \geq T] = 1 \geq \Pr[\max_i X_i \geq x],$$

   where the second equality follows by the definition of T. Now for $x \geq T$, we have, $\Pr[\overline{X}_i \geq x] = \Pr[X_i \geq x]$, and therefore,

$$\sum_{i=1}^n \Pr[\overline{X}_i \geq x] = \sum_{i=1}^n \Pr[X_i \geq x] \geq \Pr[\max_i X_i \geq x],$$

   where the inequality follows from the union bound.

   (b) **[6 points]** Define $p_i = \Pr[X_i \geq T]$. Consider the following algorithm. For $i = 1$ to n, on receiving X_i:

   - If $X_i \geq T$, then with probability $1/(2 - \sum_{j<i} p_j)$ accept X_i and stop, and with probability $1 - 1/(2 - \sum_{j<i} p_j)$ reject X_i and continue.
   - Else (i.e. if $X_i < T$), reject X_i and continue.

   (Note that $1/(2 - \sum_{j<i} p_j)$ is a valid probability because $\sum_{j<i} p_j \leq \sum_{j=1}^n p_j = 1$.) Determine the probability that the algorithm doesn't stop before it sees X_i. Hence, prove that the expected reward of this algorithm is $\left(\sum_{i=1}^n \mathbb{E}\left[\overline{X}_i\right]\right)/2$. (Therefore, this algorithm too is $(1/2)$-competitive.)

Answer: We claim that for all i, the probability that the algorithm doesn't stop before it sees X_i is $(2 - \sum_{j<i} p_j)/2$. We prove is by induction on i. The claim is obvious for $i = 1$. Assume that the claim is true for $i - 1$, that is, the probability that the algorithm doesn't stop before it sees X_{i-1} is $(2 - \sum_{j<i} p_j)/2$. Now for the algorithm to not stop before it sees X_i, it shouldn't stop before it sees X_{i-1}, and conditioned on this, it shouldn't accept X_{i-1} either. Therefore, the probability that the algorithm doesn't stop before it sees X_i is

$$\frac{2 - \sum_{j<i-1} p_j}{2} \cdot \left(1 - p_{i-1} \cdot \frac{1}{2 - \sum_{j<i-1} p_j}\right) = \frac{2 - \sum_{j<i-1} p_j}{2} \times \frac{2 - \sum_{j<i} p_j}{2 - \sum_{j<i-1} p_j} = \frac{2 - \sum_{j<i} p_j}{2},$$

as required.

Define the random variables ALG_i as follows. $\mathrm{ALG}_i = X_i$ if the algorithm accepts X_i, and $\mathrm{ALG}_i = 0$ otherwise, so that $\mathrm{ALG} = \sum_{i=1}^n \mathrm{ALG}_i$. Thus, $\mathrm{ALG}_i = \overline{X}_i$ if the algorithm doesn't stop before it sees X_i and the i'th coin toss is favorable; otherwise $\mathrm{ALG}_i = \overline{X}_i = 0$. The random variable $\overline{X}_i$, the event that the algorithm gets to see X_i, and the i'th coin toss are all independent. Therefore,

$$\mathbb{E}[\mathrm{ALG}_i] = \frac{2 - \sum_{j<i} p_j}{2} \times \frac{1}{2 - \sum_{j<i} p_j} \times \mathbb{E}[\overline{X}_i] = \frac{\mathbb{E}[\overline{X}_i]}{2},$$

and hence, we have, $\mathbb{E}[\mathrm{ALG}] = \sum_{i=1}^n \mathbb{E}[\mathrm{ALG}_i] = \left(\sum_{i=1}^n \mathbb{E}\left[\overline{X}_i\right]\right)/2$, as required.