Karlsruhe Institute of Technology

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

Exercise Sheet 11
Game Theory & Yao's Principle

Probability and Computing

Exercise 1 – Lower bounds for randomized sorting

Let $n \in \mathbb{N}$ and **Inputs** be the set of all permutations of $\{1, \dots, n\}$. Let **Algos** be the set of all comparison-based *deterministic* sorting algorithms.

(i) Give an $A \in$ **Algos** (without proof) with $\max_{I \in \textbf{Inputs}} C(A, I) = O(n \log n)$.

(ii) Warm-up: For $n = 3$ there exists a randomized algorithm $\mathcal{A}$ that beats every deterministic algorithm in the following sense:

$$\min_{A \in \textbf{Algos}} \max_{I \in \textbf{Inputs}} C(A, I) = 3 > 2 + \tfrac{2}{3} = \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)].$$

Our plan in the following is to show that this advantage disappears in O-notation.

(iii) Show that there is no "hard input", that is, that the following holds:

$$\max_{I \in \textbf{Inputs}} \min_{A \in \textbf{Algos}} C(A, I) = n - 1.$$

In order to immediately focus on the best possible cost for an input *distribution*, we need some preparation:

(iv) Show that in a binary tree with k leaves, the average depth of a leaf is at least $\lfloor \log_2(k) \rfloor$.
Hint: To do so, show that the average leaf depth is minimal for balanced trees; more precisely, that any tree in which the leaf depths differ by at least 2 can be rearranged such that the average leaf depth decreases while the number of leaves remains the same.

(v) Now conclude using Yao's principle that:

$$\min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] = \Omega(n \log n).$$

Solution 1

(i) Mergesort.

(ii) We first show the left equality. Let $A \in \mathbf{Algos}$ be arbitrary. On input $I = (x, y, z)$ we may assume, by symmetry (between the elements), that A first compares x and y and finds $x < y$. By symmetry (between $<$ and $>$) we may assume that it next compares x and z. Now it may be the case that $x < y$ and $x < z$ hold; then the order of y and z is still undetermined and another comparison is necessary. Hence there exists an input I such that $C(A, I) = 3$. Therefore $\max_{I \in \mathbf{Inputs}} C(A, I) = 3$. Since A was arbitrary, this implies:

$$\min_{A \in \mathbf{Algos}} \max_{I \in \mathbf{Inputs}} C(A, I) = 3.$$

A randomized algorithm can "guess" which element is the middle element and compare the other two elements with it. If the algorithm guessed correctly, the complete order is determined after two comparisons. Otherwise, a third comparison is necessary. This $\mathcal{A}$ satisfies $\mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] = \frac{1}{3} \cdot 2 + \frac{2}{3} \cdot 3 = 2 + \frac{2}{3}$ for every input I, as desired.

(iii) The order of "min" and "max" allows us to tailor the algorithm to the input I. Let π be the permutation[1] that sorts I. We define A depending on I as follows:

```
Algorithm A(I′):
    swap elements of I′ according to π
    fail ← FALSE
    for i = 1 to n − 1 do // check whether sorted
        if I′[i] > I′[i + 1] then
            fail ← TRUE
            break
    if fail then
        MergeSort(I′)
```

We have to consider three things:

The algorithm is correct, i.e. $A \in \mathbf{Algos}$. This is clear: The algorithm first reorders its input I' according to π and checks whether I' is sorted as a result. If so, it does nothing further; otherwise, it uses MergeSort.

The algorithm is fast for I. If the input I' coincides with I (the input to which A is tailored), then I' is sorted after the swaps according to π. Then only the $n - 1$ comparisons of the loop are needed to verify this.

It cannot be done any faster. Suppose fewer than $n - 1$ comparisons were made. Consider the graph $G = ([n], E)$ that contains an edge between i and j if the ith element of the input was compared with the jth element of the input. Since

[1] By "permutation" one sometimes means an ordering of a set (as at the beginning of the exercise), sometimes an operator that reorders a given sequence (this is what is meant here).

$|E| < n-1$, G is disconnected. Hence the relative order of these connected components is not determined. (Formally: If one input is consistent with the information of the decision tree, then so is another input. Thus the algorithm does not yet "know" what the input was, which is necessary in order to construct the output.)

(iv) Let T be a binary tree with k leaves and minimal average leaf depth. Then T is a full binary tree, i.e. every vertex has 0 or 2 children (reason: internal vertices with only one child can be removed and the average leaf depth can be reduced). Let L and L' be the maximum and minimum depth of a leaf in T, respectively. We now show that $L' \geq L-1$. Suppose this is not the case; then $L' \leq L-2$. Let ℓ_1 be a leaf at depth L and let ℓ_2 be its sibling (which exists because T is full), which must also be a leaf (by choice of L as the maximum depth). Let ℓ' be a leaf at depth L'. We now reattach ℓ_1 and ℓ_2 and make them children of ℓ'. This again yields a binary tree. The sum of leaf depths changes for the following reasons:

- The leaves ℓ_1 and ℓ_2 now have depth $L'+1$ instead of L.
- The vertex p, which was the parent of ℓ_1 and ℓ_2, is now a leaf of depth $L-1$.
- The vertex ℓ' at depth L' is no longer a leaf.

Thus the sum of leaf depths changes by

$$2((L'+1)-L) + (L-1) - L' = L' - L + 1 \leq L - 2 - L + 1 = -1,$$

i.e. it has decreased. Hence the average leaf depth has decreased, which contradicts the choice of T.

Therefore $L' \geq L-1$, i.e. the leaf depths differ by at most 1. If the average leaf depth of T were less than $\lfloor \log_2(k) \rfloor$, then at least one leaf would have depth less than $\lfloor \log_2(k) \rfloor$ and all leaves would have depth at most $\lfloor \log_2(k) \rfloor$. Hence there would be fewer than $2^{\lfloor \log_2(k) \rfloor} \leq k$ leaves—a further contradiction.

Thus T has average leaf depth at least $\lfloor \log_2(k) \rfloor$. Since T was chosen to have minimal average leaf depth, this also holds for all other binary trees.

(v) Consider the decision tree T_A describing an arbitrary $A \in \mathbf{Algos}$. Without loss of generality, we consider only algorithms that perform a comparison "$x < y$" only if both "true" and "false" are still possible outcomes, i.e. every computation path is followed by at least one input. This means that T_A is a full binary tree. For each of the $n!$ possible inputs, a different permutation of the elements is required; hence each input must lead to a different leaf in A. Thus A has exactly $n!$ leaves. By (iii), the average leaf depth of A is at least $\lfloor \log_2(n!) \rfloor$. If we now consider the uniform distribution $\mathcal{I}$ on the inputs, then the expected number of comparisons that A performs for $I \sim \mathcal{I}$ is exactly the average leaf depth of T_A, and hence also at least $\lfloor \log_2(n!) \rfloor$. From $n! \geq (n/2)^{n/2}$ it follows that $\lfloor \log_2(n!) \rfloor \geq \lfloor (n/2)\log_2(n/2) \rfloor = \Omega(n \log n)$. Thus, by Yao's theorem,

$$C \overset{\text{Yao}}{\geq} \min_{A \in \mathbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}}[C(A,I)] \geq \lfloor \log_2(n!) \rfloor = \Omega(n \log n).$$

Exercise 2 – Yao's principle without game theory

Prove Yao's principle without resorting to game-theoretic theorems (no Nash theorem, Loomis theorem, etc.). That is, prove that in the setting of the lecture, for an arbitrary distribution $\mathcal{A}_0$ on **Algos** and an arbitrary distribution $\mathcal{I}_0$ on **Inputs**, the following holds:

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}_0}[C(A,I)] \geq \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_0}[C(A,I)].$$

Hint: The game-theoretic framework of the lecture is not required here, because we do not wish to show that "=" is achievable.

Solution 2

We have:

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}_0}[C(A,I)] \geq \mathbb{E}_{A \sim \mathcal{A}_0, I \sim \mathcal{I}_0}[C(A,I)] \geq \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_0}[C(A,I)].$$

In words: In the middle, both A and I are chosen *at random*. On the left, I is not chosen at random but with the goal of maximization, which makes the result larger. On the right, A is not chosen at random but with the goal of minimization, which makes the result smaller. This already solves the exercise.

Remark. Formally, one could write the first step (and analogously the second) in more fine-grained detail by using the following two abstract observations:

1. $\max_{x \in X} f(x) \geq \mathbb{E}_{x \sim \mathcal{X}}[f(x)]$.
2. $\mathbb{E}_{x \sim \mathcal{X}}[\mathbb{E}_{y \sim \mathcal{Y}}[g(x,y)]] = \mathbb{E}_{x \sim \mathcal{X}, y \sim \mathcal{Y}}[g(x,y)]$.

where f and g are functions, $\mathcal{X}$ is a distribution on a set X, and $\mathcal{Y}$ is a distribution on a set Y.

The first equation allows one to upper bound a maximum by an expectation, and the second equation allows one to convert a two-stage random experiment and an "expected expectation" into a single-stage random experiment. Applying these equations with $X = \textbf{Inputs}$, $Y = \textbf{Algos}$, $\mathcal{X} = \mathcal{I}_0$, $\mathcal{Y} = \mathcal{A}_0$, $f(I) = \mathbb{E}_{A \sim \mathcal{A}_0}[C(A,I)]$, and $g(A,I) = C(A,I)$ yields the first inequality above.

Exercise 3 – Recommendation: Simulating the Evolution of Teamwork

The YouTube channel *Primer* deals with evolutionary game theory. In the following video, among other things, all possible 2-player games with two pure strategies are classified according to how many and which types of Nash equilibria they possess. The video is entertaining and invites active thinking, but is only marginally relevant to the lecture.

https://www.youtube.com/watch?v=TZfh8hpJIxo