Chapter 7

Probability.

Definition 7.0.1. *A probability space defined by a tuple* (Ω, P) *such that:*

1. Ω *is a set, called the sample space. Any element* $\omega \in \Omega$ *is named an atomic event. Conceptually, we think of atomic events as possible outcomes of our experiment. Any subset* $A \subset \Omega$ *is an event.*

2. P*, called the probability function, is a function that assigns a number in* $[0, 1]$ *to any event, denoted as* $P : 2^{\Omega} \to [0, 1]$*, and satisfies:*

   (a) *For any event* $A \subset \Omega$*,* $P(A) = \sum_{\omega \in A} P(w)$*.*

   (b) *Normalization, over the atomic events, to* 1*, which means* $\sum_{\omega \in \Omega} P(\omega) = 1$*.*

Claim 7.0.1. *Probability function satisfies the following properties:*

1. $P(\emptyset) = 0$*.*

2. *Monotonic, If* $A \subset B \subset \Omega$ *then* $P(A) \leq P(B)$*.*

3. *Union Bound,* $P(A \cup B) \leq P(A) + P(B)$*.*

4. *Additivity for disjointness events. If* $A \cap B = \emptyset$ *then* $P(A \cup B) = P(A) + P(B)$*.*

5. *Denote by* $\bar{A}$ *the complementary event of* A*, which means* $A \cup \bar{A} = \Omega$*. Then,* $P(\bar{A}) = 1 - P(A)$*.*

Example 7.0.1. *Let's proof the additivity of disjointness property. Let* A, B *disjointness events, so* $A \cap B = \emptyset$ *then*

$$\begin{aligned}
P(A \cup B) &= \sum_{w \in A \cup B} P(w) \\
&= \overbrace{\sum_{w \in A, w \notin B} P(w)}^{P(A)} + \overbrace{\sum_{w \in B, w \notin A} P(w)}^{P(B)} + \overbrace{\sum_{w \in A, w \in B} P(w)}^{0} \\
&= P(A) + P(B)
\end{aligned}$$

Definition 7.0.2. *Let (Ω, P) be a probability space. A random variable X on (Ω, P) is a function $X : \Omega \to \mathbb{R}$. An indicator, is a random variable defined by an event $A \subset \Omega$ as follows*

$$X(\omega) = \begin{cases} 1 & \omega \in A \\ 0 & \omega \notin A \end{cases}$$

Sometimes, we will use the notation $\{X = x\}$ to denote the event A such:

$$A = \{\omega : X(\omega) = x\} := \{X = x\}$$

Result: Sorting $A_1, A_2, ..A_n$

for $i \in [n]$ **do**
    for $j \in [n]$ **do**
        if $A_i < A_j$ **then**
            swap $A_i \leftrightarrow A_j$
        end
    end
end

Algorithm 1: "ICan'tBelieveItCanSort" alg.