# Chapter 5: Probabilistic Analysis

Average-Case Analysis
Finding the Maximum Example
Randomizing an Array Example

## Average-Case Analysis

- In practice, many algorithms perform better than their worst-case.
- The **average case** is analyzed by:
  1. construct a probabilistic model of the input
  2. determine the probabilities and running times (or costs) of alternate executions
  3. calculate expected running time (or cost)
- Through **randomization**, one can often ensure that the probabilistic model is true.

### Example 1: Rolling Dice

**Scenario:** Pay $n$ to roll a die $n$ times.

**Payoff:** Maximum value of the rolls.

**Question:** What's the best value of $n$?

- Expected value sums probabilities times values.
- For $n = 1$, expected value is:
  
  \[
  \frac{1}{6} \times (1 - 1) + \frac{1}{6} \times (2 - 1) + \frac{1}{6} \times (3 - 1) + \frac{1}{6} \times (4 - 1) + \frac{1}{6} \times (5 - 1) + \frac{1}{6} \times (6 - 1) = 2.5
  \]

- Expected value for $n = 2$? $n = 3$?
- Ignoring cost, what is the prob. that the maximum value is $v$ after $n$ rolls?
- For prob. that $v$ is 6, easiest to determine prob. of not rolling a six = $(5/6)^n$.
  - So prob. that $v$ is 6 = $1 - (5/6)^n$. 

## Example 1: Rolling Dice

- **Scenario:** Pay $n$ to roll a die $n$ times.
  - **Payoff:** Maximum value of the rolls.
  - **Question:** What’s the best value of $n$?
- **Expected value sums probabilities times values.**
- For $n = 1$, expected value is:
  
  \[
  \frac{1}{6} \times (1 - 1) + \frac{1}{6} \times (2 - 1) + \frac{1}{6} \times (3 - 1) + \frac{1}{6} \times (4 - 1) + \frac{1}{6} \times (5 - 1) + \frac{1}{6} \times (6 - 1) = 2.5
  \]

- **Expected value for $n = 2$? $n = 3$?**
- **Ignoring cost, what is the prob. that the maximum value is $v$ after $n$ rolls?**
- For prob. that $v$ is 6, easiest to determine prob. of not rolling a six = $(5/6)^n$.
  - So prob. that $v$ is 6 = $1 - (5/6)^n$. 

Example 2: Finding the Maximum

MAXIMUM(A)
  max ← A[1]
  for i ← 2 to A.length do
    if max < A[i] then
      max ← A[i]
  return max

Problem: How many assignments to max?
- Best-case: 1 (When does this happen?)
- Worst-case: n (When does this happen?)
- Average-case: \( (n + 1)/2 \) is incorrect

Example 2: Probablistic Model
- Assume A has n distinct numbers. (What is the effect of duplicates?)
- Assume each permutation of the numbers is equally likely. (How can randomization guarantee this?)
- How many permutations are there?
  - What is the probability of the best case?
  - What is the probability of the worst case?

Example 2: Analysis
- On iteration i, max is assigned a value iff A[i] is the maximum of the first i numbers.
- Probability that A[i] is the maximum of the first i numbers = \( 1/i \)
- Probability of assignment = \( 1/i \), cost = 1
  - Prob. of no assignment = \( (i-1)/i \), cost = 0
- On iteration i, the expected cost is: \( (1/i)(1) + ((i-1)/i)(0) = 1/i \)
- Over the initial assignment and n−1 iterations, the expected cost is:
  \[ \sum_{i=1}^{n-1} \frac{1}{i} \]
  which is between \( \ln n \) and \( 1 + \ln n \)

Example 3: Random Permutation of an Array

RANDOMIZE-IN-PLACE(A)
  n ← A.length
  for i ← 2 to n do
    swap A[i] ↔ A[Random(1, i)]

Random(a, b) returns an integer r, a ≤ r ≤ b.
- r is equally likely to be any integer between a and b, inclusive.
Example 3: Analysis

- For any permutation of the first $i$ values, there is exactly one way to permute the first $i-1$ values, and then swap $A[i]$ into the correct position.
- So after iteration $i$, the first $i$ values have been randomly permuted, making the loop invariant true the next iteration.