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 probabilisitic model is true.

Example 1: Finding the Maximum

```
MAXIMUM(A)
    max ← A[1]
    for i ← 2 to length(A) 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 1: 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 1: 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:

  $\sum_{i=1}^{n} \frac{1}{i}$ which is between $\ln n$ and $1 + \ln n$

Example 2: Random Permutation of an Array

RANDOMIZE-IN-PLACE(A)
  $n \leftarrow length(A)$
  for $i \leftarrow 2$ **to** n **do**
    swap $A[i] \leftrightarrow A[\text{RANDOM}(1, i)]$

- RANDOM(a, b) returns an integer r, $a \le r \le b$.
- r is equally likely to be any integer between a and b

Example 2: Analysis

- Loop invariant: Before iteration i, $A[1]$ through $A[i-1]$ is a random permutation of the first $i-1$ values.
- 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.