WIA2005: Algorithm Design and Analysis

Semester 2, Session 2016/17

Lecture 4: Probabilistic and Randomized Algorithms

Learning Objectives

- Know what is Probabilistic Analysis
- Know Randomized Algorithm
- Problem: Hiring an assistant.
  - Pseudocode
  - Probabilistic analysis
  - Randomize algorithm

Probabilistic Analysis

- Assumes that the inputs to the problem are chosen from a known probability distribution.
- The algorithm itself is deterministic.

Problem: Hiring an assistant.

- Number of candidates = n.
- Interview one candidate per day. (Cost of interviewing a candidate = α.)
- If the candidate is better than the current assistant, must fire the assistant and hire the candidate. (Cost of hiring any candidate = ch)
- Goal: Estimate the cost of this strategy.

Pseudocode:

```
HIRE-ASSISTANT(n)
best = 0                    // candidate 0 is a least-qualified dummy candidate
for i = 1 to n
    interview candidate i
    if candidate i is better than candidate best
        best = i
        hire candidate i
```

Note

- Since all candidates must be interviewed, the interview cost $n\alpha$ is unavoidable. (This cost is incurred for every input.)
- If m candidates are hired, the hiring cost is mc_h. This cost varies with the input. So, we focus on this cost.

- Worst-case analysis:
- Candidate list is in increasing order of quality; that is, candidate i is better than candidate $i - 1$, $1 \le i \le n$.
- Every candidate is hired. Therefore, hiring cost = nc_h.

Probabilistic analysis:

- Assumptions: Each candidate i has a rank, denoted by r(i).
- Larger the rank, the better is the candidate.
- Candidates are totally ordered by the ranks; that is, no two candidates have the same rank.
- So, the sequence (r(1), r(2), . . . , r(n)) can be considered as a permutation of (1, 2, . . . , n).
- Candidates come in a random order. More precisely, the ranks form a uniform random permutation of (1, 2, . . . , n); that is, each of the n! permutations is equally likely (occurs with probability = 1/n!).

Randomized Algorithm:

- No assumption about probability distribution of inputs.
- The algorithm uses random numbers (or coin tosses).
- May produce different outputs for the same input at different times.

Indicator Random Variable

- Method for converting between possibilities and expectations.
- Suppose we are given a sample space S and an event A. Then the ***indicator random variable*** I{A} associated with event A is defined as

$$\mathrm{I}\{A\} = \begin{cases} 1 & \text{if } A \text{ occurs ,} \\ 0 & \text{if } A \text{ does not occur .} \end{cases}$$

Example (Coin Flip)

- Sample space is S={H,T} , with Pr{H} = Pr{T} = 1/2.
- Indicator random variable X_H , associated with the coin coming up heads, which is the event H.
- This variable counts the number of heads obtained in this flip, and it is 1 if the coin comes up heads and 0 otherwise. We write

$$X_H = \mathrm{I}\{H\} = \begin{cases} 1 & \text{if } H \text{ occurs ,} \\ 0 & \text{if } T \text{ occurs .} \end{cases}$$

Cont.

- The expected number of heads obtained in one flip of the coin is simply the expected value of our indicator variable X_H :

$$\begin{aligned}
\mathrm{E}[X_H] &= \mathrm{E}[\mathrm{I}\{H\}] \\
&= 1 \cdot \Pr\{H\} + 0 \cdot \Pr\{T\} \\
&= 1 \cdot (1/2) + 0 \cdot (1/2) \\
&= 1/2\,.
\end{aligned}$$

Lemma 0

- Given a sample space S and an event A in the sample space S,
  - let $X_A = I\{A\}$
- Then $E[X_A] = Pr\{A\}$

Proof

- By the definition of an indicator random variable and the definition of expected value, we have

$$
\begin{aligned}
\mathrm{E}[X_A] &= \mathrm{E}[\mathrm{I}\{A\}] \\
&= 1 \cdot \Pr\{A\} + 0 \cdot \Pr\{\overline{A}\} \\
&= \Pr\{A\} ,
\end{aligned}
$$

where $\overline{A}$ denotes $S - A$, the complement of A.

Coin Flip n times?

- X_i= I [the i^{th} flip H}
- X = random variable denoting the total number of heads in the n coin flips, so that

$$X = \sum_{i=1}^{n} X_i \, .$$

- Expected number of H

$$\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}\left[\sum_{i=1}^{n} X_i\right] \\
&= \sum_{i=1}^{n} \mathrm{E}[X_i] \\
&= \sum_{i=1}^{n} 1/2 \\
&= n/2 \, .
\end{aligned}$$

Analysis of hiring problem using IRV

- Now we will apply IRV on hiring problem.

Lemma 1

- Under the above assumptions, the expected hiring cost is $O(c_h \log n)$.

Proof 1

- Let Xi be the indicator random variable associated with the event in which the i[th] candidate is hired. Thus,

$$X_i = \mathrm{I}\{\text{candidate } i \text{ is hired}\}$$
$$= \begin{cases} 1 & \text{if candidate } i \text{ is hired} , \\ 0 & \text{if candidate } i \text{ is not hired} , \end{cases}$$

$$X = X_1 + X_2 + \cdots + X_n$$

- We know that,

$$\mathrm{E}[X_i] = \Pr\{\text{candidate } i \text{ is hired}\}$$

and we must therefore compute the probability that lines 5–6 of HIRE-ASSISTANT are executed.

Cont.

- Candidate i is hired, in line 6, exactly when candidate i is better than each of candidates 1 through i 1. Because we have assumed that the candidates arrive in a random order, the first i candidates have appeared in a random order.
- Any one of these first i candidates is equally likely to be the best-qualified so far.
- Candidate i has a probability of 1/i of being better qualified than candidates 1 through i - 1 and thus a probability of 1/i of being hired.

$$\mathrm{E}[X_i] = 1/i$$

Cont.

- Now we can compute E[X] :

$$
\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}\left[\sum_{i=1}^{n} X_i\right] && \text{(by equation (5.2))} \\
&= \sum_{i=1}^{n} \mathrm{E}[X_i] && \text{(by linearity of expectation)} \\
&= \sum_{i=1}^{n} 1/i && \text{(by equation (5.3))} \\
&= \ln n + O(1) && \text{(by equation (A.7))} \ .
\end{aligned}
$$

A randomized algorithm

- The list of candidates is an input to the algorithm.
- Algorithm randomly permutes the list to generate the order in which candidates are interviewed.
- If this is done in such a way that each of the n! permutations is equally likely, then the previous analysis can be used.

- Key point: No particular input elicits worst-case behavior. (The algorithm performs badly only when the random permutation generated turns out to be "unlucky".)

RANDOMIZED-HIRE-ASSISTANT(n)

```
randomly permute the list of candidates
best = 0                // candidate 0 is a least-qualified dummy candidate
for i = 1 to n
    interview candidate i
    if candidate i is better than candidate best
        best = i
        hire candidate i
```

Lemma 2: Under the above assumptions, the expected hiring cost for the randomized algorithm is $O(c_h \log n)$.

Proof: After permuting the input array, we have achieved a situation identical to that of the probabilistic analysis of HIRE-ASSISTANT.

Randomly permuting an array

- Goal: Each of the n! permutations must be equally likely.

- Array A[1 .. n] contains some permutation of
- (1, 2, . . . , n). (The initial order makes no difference.)
- A function (Random) generating uniformly distributed random numbers is available. In particular, Random(a, b) generates a ran- dom integer in the range [a .. b], where each integer in the range is generated with probability 1/(b − a + 1).

Method 1 – Permute-by-Sorting

- For each element A[i], generate a random priority value P [i] in the range [1 .. n3]. (We assume that all the priority values are distinct.)
- Sort array A into increasing order, using the priority values as keys.
- Output the sorted array A.

Pseudocode:

```
PERMUTE-BY-SORTING(A)
n = A.length
let P[1..n] be a new array
for i = 1 to n
        P[i] = RANDOM(1, n^3)
sort A, using P as sort keys
```

Example:

Suppose

A = (1, 2, 3, 4)
P = (30, 9, 53, 7).

Resulting in Permutation:
(4, 2, 1, 3)

Lemma 3

- Procedure PERMUTE-BY-SORTING produces a uniform random permutation of the input, assuming that all priorities are distinct.

Proof

- We start by considering the particular permutation in which each element A[i] receives the ith smallest priority.
- We shall show that this permutation occurs with probability exactly 1/n!.
- For i = 1, 2,..,n, let E_i be the event that element A[i] receives the i^{th} smallest priority. Then we wish to compute the probability that for all i, event E_i occurs, which is

$$\Pr\{E_1 \cap E_2 \cap E_3 \cap \cdots \cap E_{n-1} \cap E_n\}$$

Cont.

- The probability equal to:

$$\Pr\{E_1\} \cdot \Pr\{E_2 \mid E_1\} \cdot \Pr\{E_3 \mid E_2 \cap E_1\} \cdot \Pr\{E_4 \mid E_3 \cap E_2 \cap E_1\} \\ \cdots \Pr\{E_i \mid E_{i-1} \cap E_{i-2} \cap \cdots \cap E_1\} \cdots \Pr\{E_n \mid E_{n-1} \cap \cdots \cap E_1\}$$

$$\begin{aligned} \Pr\{E_1 \cap E_2 \cap E_3 \cap \cdots \cap E_{n-1} \cap E_n\} &= \left(\frac{1}{n}\right)\left(\frac{1}{n-1}\right)\cdots\left(\frac{1}{2}\right)\left(\frac{1}{1}\right) \\ &= \frac{1}{n!}, \end{aligned}$$

Method 2: Randomize In-Place

- Permutes the given array in place.
- In iteration i, the element A[i] is chosen randomly from among the elements A[i] through A[n].
- Subsequent to iteration i, A[i] is never altered.

Pseudocode

```
RANDOMIZE-IN-PLACE(A)
n = A.length
for i = 1 to n
        swap A[i] with A[RANDOM(i, n)]
```

Lemma 4

- Procedure RANDOMIZE-IN-PLACE computes a uniform random permutation.

Proof

- Loop invariant: Just prior to iteration i of the for loop, for each possible (i−1)-permutation, the subarray A[1 .. i − 1] contains this (i − 1)-permutation with probability (n − i + 1)!/n!.

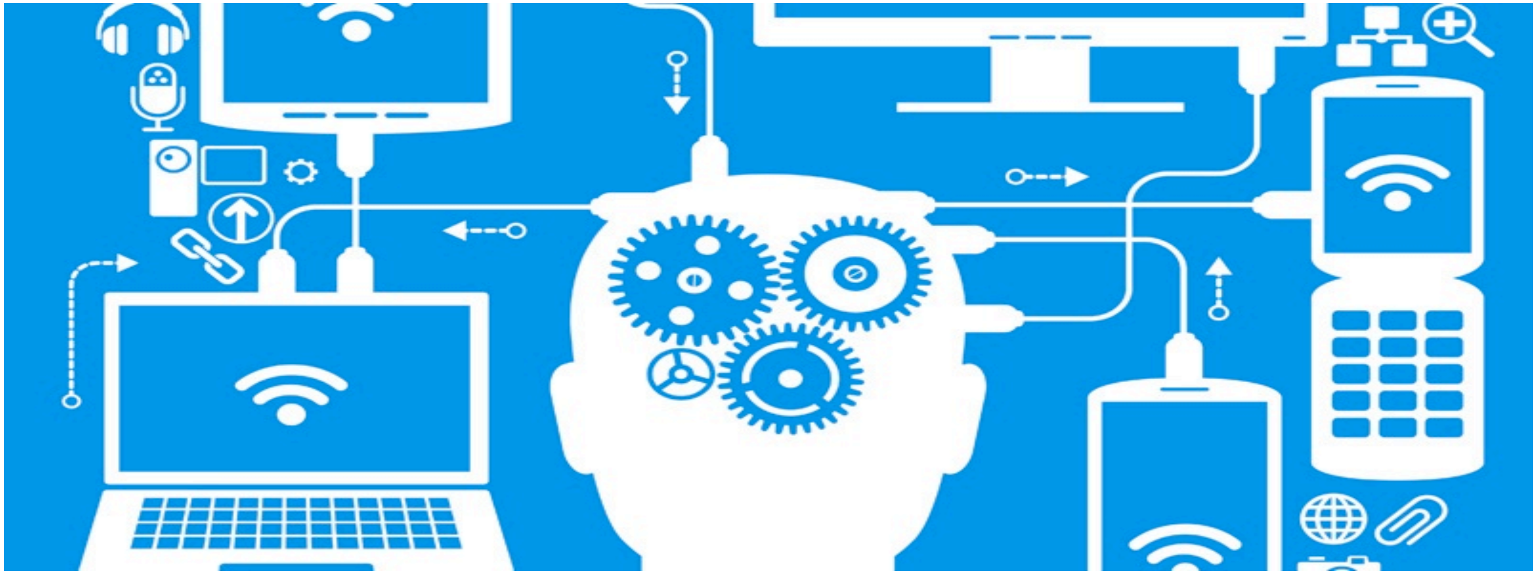
- We need to show that this invariant
  - Is true prior to the first loop iteration
  - Each iteration of the loop maintains the invariant, and that the invariant provides a useful property to show correctness when the loop terminates.

Let’s reflect

Reference

- Cormen, Lieserson and Rivest, Introduction to Algorithms, Third Edition, MIT Press, 2009.

In the next lecture..



Lecture 5: Order Statistics