Design and Analysis of Algorithms

Lecture 3—Randomized Algorithms

Lei Wang

Dalian University of Technology

October 7, 2008

Probabilistic Analysis and Randomized Algorithms

Goals

- Explain the difference between probabilistic analysis and randomized algorithms,
- present the technique of indicator random variables, and
- give another example of the analysis of a randomized algorithm (permuting an array in place).

The hiring problem

Scenario:

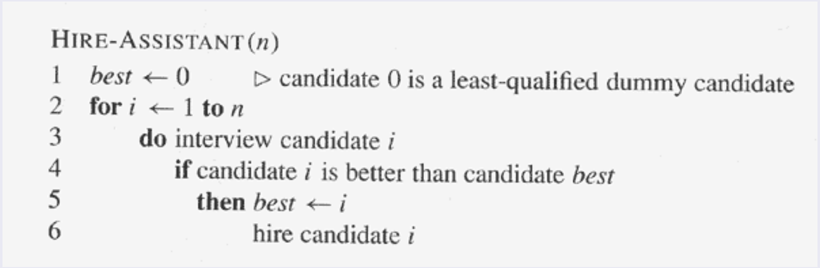
- You are using an employment agency to hire a new office assistant.
- The agency sends you one candidate each day.
- You interview the candidate and must immediately decide whether or not to hire that person. But if you hire, you must also fire your current office assistant even if it's someone you have recently hired.
- Cost to interview is c_i per candidate (interview fee paid to agency).
- Cost to hire is c_h per candidate (includes cost to fire current office assistant + hiring fee paid to agency).
- You are committed to having hired, at all times, the best candidate seen so far.

The hiring problem (cont.)

Goal

Determine what the price of this strategy will be.
Pseudocode to model this scenario: Assumes that the candidates are numbered 1 to n and that after interviewing each candidate, we can determine if it's better than the current office assistant.

HIRE-ASSISTANT(n)



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

The cost

If n candidates, and we hire m of them, the cost is $O(nc_i + mc_h)$.

- Have to pay nc_i to interview, no matter how many we hire.
- So we focus on analyzing the hiring cost mc_h.
- mc_h varies with each run—it depends on the order in which we interview the candidates.
- This is a model of a common paradigm: we need to find the maximum or minimum in a sequence by examining each element and maintaining a current “winner.” The variable m denotes how many times we change our notion of which element is currently winning.

Worst-case analysis

- In the worst case, we hire all n candidates.
- This happens if each one is better than all who came before. In other words,

Worst-case analysis

- In the worst case, we hire all n candidates.
- This happens if each one is better than all who came before. In other words, if the candidates appear in increasing order of quality.

Worst-case analysis

- In the worst case, we hire all n candidates.
- This happens if each one is better than all who came before. In other words, if the candidates appear in increasing order of quality.
- If we hire all n, then the cost is $O(nc_i + nc_h) = O(nc_h)$ (since $c_h > c_i$).

Probabilistic analysis

In general, we have no control over the order in which candidates appear. We could assume that they come in a random order:

- Assign a rank to each candidate: $rank(i)$ is a unique integer in the range 1 to n.
- The ordered list $\langle rank(1), rank(2), \ldots, rank(n) \rangle$ is a permutation of the candidate numbers $\langle 1, 2, \ldots, n \rangle$.
- The list of ranks is equally likely to be any one of the $n!$ permutations.
- Equivalently, the ranks form a ***uniform random permutation***: each of the possible $n!$ permutations appears with equal probability.

Randomized algorithms

We might not know the distribution of inputs, or we might not be able to model it computationally. Instead, we use randomization within the algorithm in order to impose a distribution on the inputs.

For the hiring problem: Change the scenario:

- The employment agency sends us a list of all n candidates in advance.
- On each day, we randomly choose a candidate from the list to interview (but considering only those we have not yet interviewed).
- Instead of relying on the candidates being presented to us in a random order, we take control of the process and enforce a random order.

Randomized algorithms (cont.)

What makes an algorithm randomized:

An algorithm is randomized if its behavior is determined in part by values produced by a random-number generator.

- RANDOM(a, b) returns an integer r, where $a \leq r \leq b$ and each of the $b - a + 1$ possible values of r is equally likely.
- In practice, RANDOM is implemented by a ***pseudorandom-number generator***, which is a deterministic method returning numbers that "look" random and pass statistical tests.

Indicator random variables

Given a sample space and an event A, we define the ***indicator random variable***

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

Lemma

For an event A, let $X_A = I\{A\}$. Then $E[X_A] = \Pr\{A\}$.
Proof Letting $\overline{A}$ be the complement of A, we have

$$\begin{aligned} E[X_A] &= E[I\{A\}] \\ &= 1 \cdot \Pr\{A\} + 0 \cdot \Pr\{\overline{A}\} \qquad \text{(definition of expected value)} \\ &= \Pr\{A\}. \end{aligned}$$

Some examples...

Analysis of the hiring problem

Assume that the candidates arrive in a random order. Let X be a random variable that equals the number of times we hire a new office assistant. Define indicator random variables $X_1, X_2, \ldots, X_n$, where $X_i = I\{\text{candidate } i \text{ is hired}\}$.

Useful properties:

- $X = X_1 + X_2 + \cdots + X_n$.
- **Lemma** $\Rightarrow E[X_i] = \Pr\{\text{candidate } i \text{ is hired}\}$.
- $\Pr\{\text{candidate } i \text{ is the best so far}\} = 1/i$, which implies $E[X_i] = 1/i$.
- we can get $E[X] = E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} 1/i = \ln n + O(1)$ $\Rightarrow$ the expected hiring cost $O(c_h \cdot \ln n)$, much better than $O(c_h n)$.