Analysis of Algorithms 1
(Fall 2015)
Istanbul Technical University
Computer Eng. Dept.

Chapter 5: Probabilistic Analysis and Randomized Algorithms

Course slides from Jennifer Welch are used in preparation of these slides

Last updated: October 13, 2015

Purpose

- Learn how to conduct **probabilistic analysis** of an algorithm:
  - Make assumptions about probability distributions of inputs
  - Analyze algorithm, computing "expected" running time
- Learn about **randomized algorithms**:
  - Behavior determined not only by inputs but also by random number generator

Contents

- Hiring Problem
- Indicator Random Variables
- Randomized Algorithms

Hiring Problem

- You need to hire a new employee
- The headhunter sends you a different applicant every day for *n* days
- If the applicant is better than the current employee, then fire the current employee and hire the applicant
- Firing and hiring is expensive
- How expensive is the whole process?

Hiring Problem

```
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
```

Hiring Problem (Worst/Best Case)

- Worst case:
  - Headhunter sends you *n* applicants in increasing order of goodness
  - Then you hire (and fire) each one in turn: *n* hires
- Best case:
  - Headhunter sends you best applicant on first day
  - Total cost is just 1 (fire and hire once)

Hiring Problem (Average Case)

Average cost:

- What is meant by average?
- An input to the hiring problem is an ordering of the *n* applicants
- There are *n*! different inputs
- Assume there is some distribution on the inputs
  - For instance, each ordering is equally likely
  - But, other distributions are also possible
- Average cost is expected value

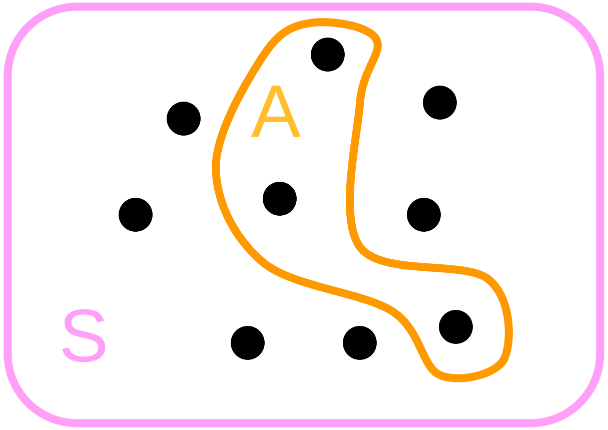
REVIEW

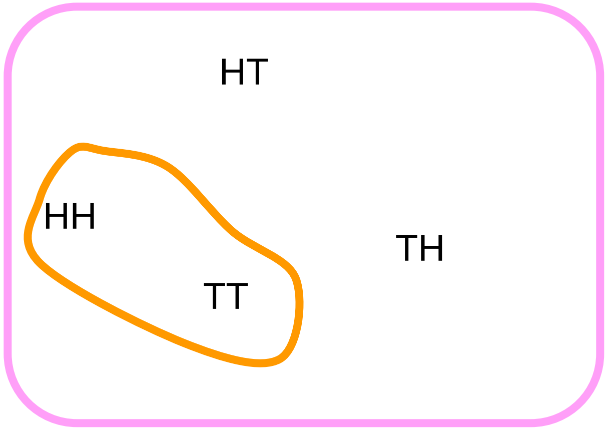
Probability

- Every probabilistic claim ultimately refers to some sample space, which is a set of elementary events
- Think of each elementary event as the outcome of some experiment
  - Ex: flipping two coins gives sample space

    {HH, HT, TH, TT}
- An event is a subset of the sample space
  - Ex: event "both coins flipped the same" is {HH, TT}

Sample Spaces and Events





REVIEW

Probability Distribution

- A probability distribution Pr on a sample space S is a function from events of S to real numbers s.t.
  - $Pr[A] \geq 0$ for every event A
  - $Pr[S] = 1$
  - $Pr[A \cup B] = Pr[A] + Pr[B]$ for every two non-intersecting ("mutually exclusive") events A and B
- Pr[A] is the probability of event A

REVIEW

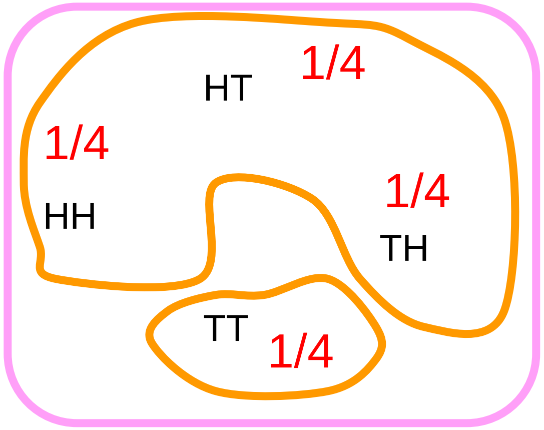
Properties of Probability Distributions

- $\Pr[\emptyset] = 0$
- If $A \subseteq B$, then $\Pr[A] \le \Pr[B]$
- $\Pr[S — A] = 1 — \Pr[A]$ // complement
- $\Pr[A \cup B] = \Pr[A] + \Pr[B] – \Pr[A \cap B]$

  $\le \Pr[A] + \Pr[B]$

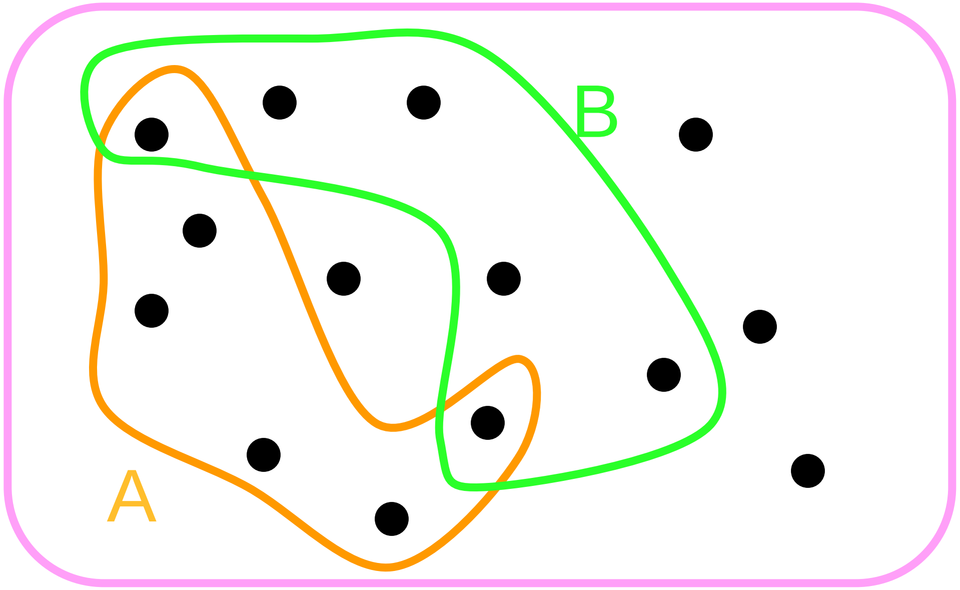
Example

- Suppose Pr[{HH}] = Pr[{HT}] = Pr[{TH}] = Pr[{TT}] = 1/4.
- Pr["at least one head"]
  = Pr[{HH U HT U TH}]
  = Pr[{HH}] + Pr[{HT}] + Pr[{TH}]
  = 3/4.
- Pr["less than one head"]
  = 1 — Pr["at least one head"]
  = 1 — 3/4 = 1/4



REVIEW

Probability Distribution



$$\Pr[A \cup B] = \Pr[A] + \Pr[B] - \Pr[A \cap B]$$

Specific Probability Distribution

REVIEW

- Discrete probability distribution: sample space is finite or countably infinite
  - Ex: flipping two coins once; flipping one coin infinitely often
- Uniform probability distribution: sample space S is finite and every elementary event has the same probability, 1/|S|
  - Ex: flipping two fair coins once

Flipping a Fair Coin

- Suppose we flip a fair coin n times
- Each elementary event in the sample space is one sequence of n heads and tails, describing the outcome of one "experiment"
- Size of sample space is 2^n
- Let A be the event of "k heads and $n\text{–}k$ tails occurring"
- $\Pr[A] = C(n,k)/2^n$
  - There are $C(n,k)$ sequences of length n in which k heads and $n\text{–}k$ tails occur, and each has probability $1/2^n$.

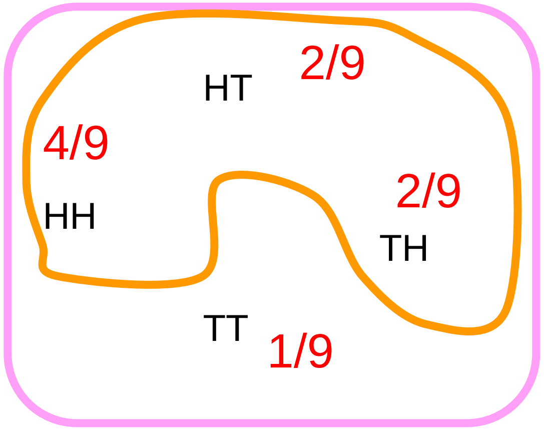
Example

- $n = 5$, $k = 3$
- HHHTT HHTTH HTTHH TTHHH
- HHTHT HTHTH THTHH
- HTHHT THHTH
- THHHT
- Pr(3 heads and 2 tails) = $C(5,3)/2^5$

  = 10/32

Flipping Unfair Coins

- Suppose we flip two coins, each of which gives heads two-thirds of the time
- What is the probability distribution on the sample space?



Pr[at least one head] = 8/9

REVIEW

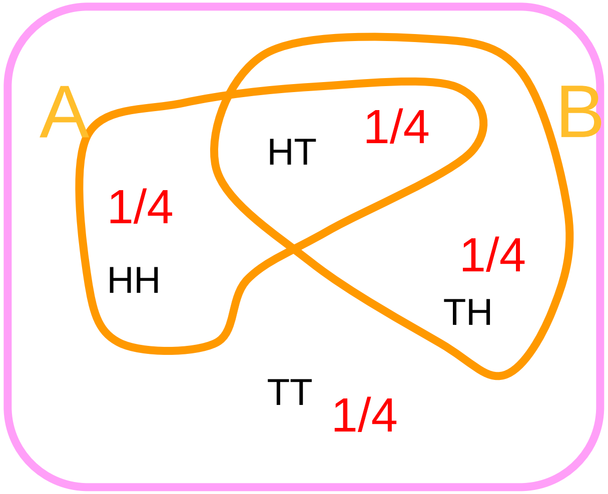
Independent Events

- Two events A and B are independent if $\Pr[A \cap B] = \Pr[A] \cdot \Pr[B]$
  - i.e., probability that both A and B occur is the product of the separate probabilities that A occurs and that B occurs

Independent Events Example

In two-coin-flip example with fair coins:

- A = "first coin is heads"
- B = "coins are different"

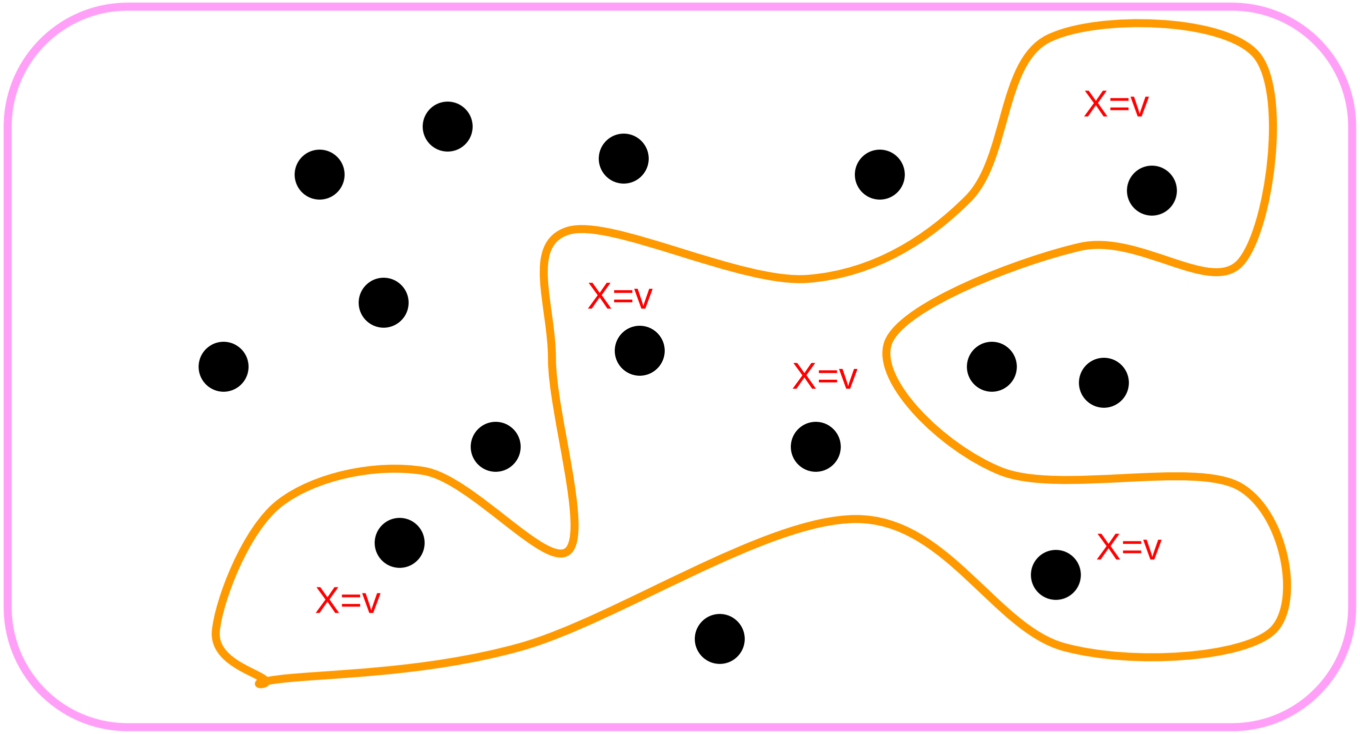


Pr[A] = 1/2
Pr[B] = 1/2
Pr[A ∩ B] = 1/4 = (1/2)(1/2)
so A and B are independent

Discrete Random Variables

- A discrete random variable X is a function from a finite or countably infinite sample space to the real numbers
- Associates a real number with each possible outcome of an experiment
- Define the event "X = v" to be the set of all the elementary events s in the sample space with X(s) = v
- So, Pr["X = v"] is the sum of Pr[{s}] over all s with X(s) = v

Discrete Random Variable



Add up the probabilities of all the elementary events in the orange event to get the probability that X = v

Random Variable Example

- Roll two fair 6-sided dice
- Sample space contains 36 elementary events (1:1, 1:2, 1:3, 1:4, 1:5, 1:6, 2:1,…)
- Probability of each elementary event is 1/36
- Define random variable X to be the maximum of the two values rolled
- What is Pr["X = 3"]?
- It is 5/36, since there are 5 elementary events with max value 3 (1:3, 2:3, 3:3, 3:2, and 3:1)

Independent Random Variables

REVIEW

- It is common for more than one random variable to be defined on the same sample space:
  - X is maximum value rolled
  - Y is sum of the two values rolled
- Two random variables X and Y are independent if for all v and w, the events "X = v" and "Y = w" are independent

Expected Value of a Random Variable

REVIEW

- Most common summary of a random variable is its "average", weighted by the probabilities
  - called expected value, or expectation, or mean

- Definition: $E[X] = \sum_{v} v \Pr[X = v]$

Expected Value Example

- Consider a game in which you flip two fair coins
- You get 3TL for each head but lose 2TL for each tail
- What are your expected earnings?
  - i.e., what is the expected value of the random variable X, where X(HH) = 6, X(HT) = X(TH) = 1, and X(TT) = -4?
- Note that no value other than 6, 1, and -4 can be taken on by X (e.g., Pr[X = 5] = 0)
- $E[X] = 6(1/4) + 1(1/4) + 1(1/4) + (-4)(1/4) = 1$

Properties of Expected Values

- $E[X+Y] = E[X] + E[Y]$, for any two random variables X and Y, even if they are not independent!
- $E[a \cdot X] = a \cdot E[X]$, for any random variable X and any constant a
- $E[X \cdot Y] = E[X] \cdot E[Y]$, for any two *independent* random variables X and Y

Back to Hiring Problem

- We want to know the expected cost of our hiring algorithm, in terms of how many times we hire an applicant
- Elementary event s is a sequence of the n applicants
- Sample space is all n! sequences of applicants
- Assume uniform distribution, so each sequence is equally likely, i.e., has probability 1/n!
- Random variable X(s) is the number of applicants that are hired, given the input sequence s
- What is E[X]?

Solving the Hiring Problem

- Break the problem down using indicator random variables and properties of expectation
- Change viewpoint: instead of one random variable that counts how many applicants are hired, consider n random variables, each one keeping track of whether or not a particular applicant is hired.
- Indicator random variable X_i for applicant *i*: 1 if applicant *i* is hired, 0 otherwise

Indicator Random Variables

The indicator random variable *I*[*A*] associated with event A is defined as

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

- Lemma 5.1
  - Given a sample space S and an event A in the sample space S, let X_A=I{A}
  - Then $E[X_A]=Pr\{A\}$

Indicator Random Variables

- Important fact: $X = X_1 + X_2 + \ldots + X_n$
  - number hired is sum of all the indicator r.v.'s
- Important fact:
  - $E[X_i]$ = Pr["applicant *i* is hired"]
  - Why? Plug in definition of expected value
- Probability of hiring *i* is probability that *i* is better than previous *i* -1 applicants

Probability of Hiring *i*th Applicant

- In general, since all permutations are equally likely, if we only consider the first *i* applicants, the largest of them is equally likely to occur in each of the *i* positions.
- Thus, $\Pr[X_i = 1] = 1/i$

Expected Number of Hires

- Recall that X is random variable equal to the number of hires
- Recall that X = the sum of the X_i's (each X_i is the random variable that tells whether or not the *i*th applicant is hired)
- $E[X] = E[\sum X_i]$

  $= \sum E[X_i]$, by property of E

  $= \sum Pr[X_i = 1]$, by property of X_i

  $= \sum 1/i$, by argument on previous slide

  $\leq \ln n + 1$, by formula for harmonic number

$Hn=1+1/2 + 1/3 + \ldots + 1/n = \ln(n)+O(1)$ see Appendix A.

Discussion of Hiring Problem

- So, average number of hires is ln n, which is much better than worst case number (n)
- But, this relies on the headhunter sending you the applicants in random order
- What if you cannot rely on that?
  - Maybe headhunter always likes to impress you, by sending you better and better applicants
- If you can get access to the list of applicants in advance, you can create your own randomization, by randomly permuting the list and then interviewing the applicants.
- Move from (passive) probabilistic analysis to (active) randomized algorithm by putting the randomization under your control!

Randomized Algorithms

- Instead of relying on a (perhaps incorrect) assumption that inputs exhibit some distribution, make your own input distribution by, say, permuting the input randomly or taking some other random action
- On the same input, a randomized algorithm has multiple possible executions
- No one input elicits worst-case behavior
- Typically we analyze the average case behavior for the worst possible input

Randomized Hiring Algorithm

- Suppose we have access to the entire list of candidates in advance
- Randomly permute the candidate list
- Then interview the candidates in this random sequence
- Expected number of hirings/firings is O(log n) *no matter what the original input is*

Probabilistic Analysis vs. Randomized Algorithm

- Probabilistic analysis of a deterministic algorithm:
  - Assume some probability distribution on the inputs
- Randomized algorithm:
  - Use random choices in the algorithm

How to Randomly Permute an Array

- input: array A[1..n]
- for i := 1 to n do
  - j := value between i and n chosen with uniform probability (each value equally likely)
  - swap A[i] with A[j]

Why Does It Work?

- Show that after *i*th iteration of the for loop:

  A[1..i] equals each permutation of i elements from {1,…,n} with probability (n–i)!/n!

- **Basis:** After first iteration, A[1] contains each permutation of 1 element from {1,…,n} with probability (n–1)!/n! = 1/n
  - True since A[1] is swapped with an element drawn from the entire array uniformly at random

Why Does It Work?

- **Induction:** Assume that after *(i–1)*st iteration of the for loop

  A[1..i–1] equals each permutation of i–1 elements from {1,…,n} with probability

  (n–(i–1))!/n!

- The probability that A[1..i] contains permutation $x_1, x_2, \ldots, x_i$ is the probability that A[1..i–1] contains $x_1, x_2, \ldots, x_{i-1}$ after the *(i–1)*st iteration AND that the *i*th iteration puts x_i in A[i]

Why Does It Work?

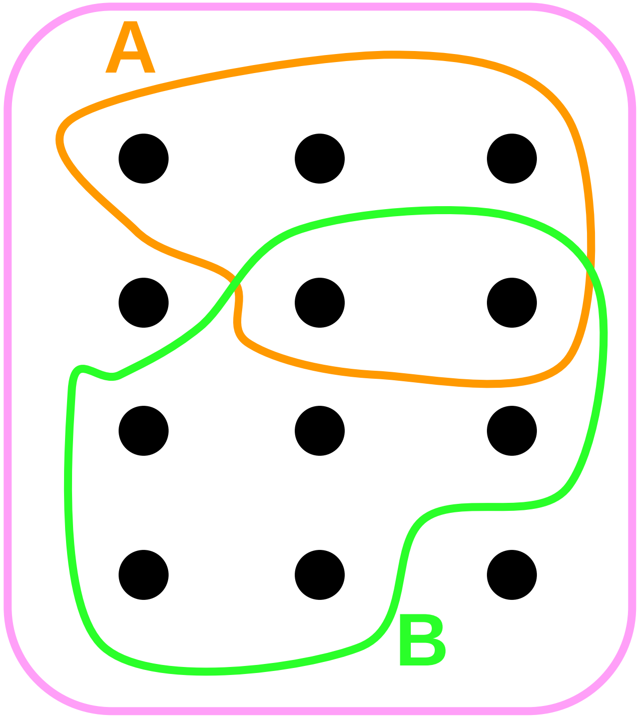
- Let e_1 be the event that A[1..i–1] contains x_1, x_2, …, x_{i-1} after the (i–1)-st iteration
- Let e_2 be the event that the i-th iteration puts x_i in A[i]
- We need to show that $\Pr[e_1 \cap e_2] = (n-i)!/n!$
- Unfortunately, e_1 and e_2 are not independent: if some element appears in A[1..i –1], then it is not available to appear in A[i]
- We need some more probability…

Conditional Probability

- Formalizes having partial knowledge about the outcome of an experiment
- Example: flip two fair coins
  - Probability of two heads is 1/4
  - Probability of two heads when you already know that the first coin is a head is 1/2
- Conditional probability of A given that B occurs is Pr[A|B] is defined to be

  $\Pr[A \cap B]/\Pr[B]$

Conditional Probability



Pr[A] = 5/12
Pr[B] = 7/12
Pr[A∩B] = 2/12
Pr[A|B] = (2/12)/(7/12) = 2/7

Conditional Probability

- Definition is $\Pr[A|B] = \Pr[A \cap B]/\Pr[B]$
- Equivalently, $\Pr[A \cap B] = \Pr[A|B] \cdot \Pr[B]$
- Back to analysis of random array permutation…

Why Does It Work?

- Recall: e_1 is event that $A[1..i-1] = x_1,\ldots,x_{i-1}$
- Recall: e_2 is event that $A[i] = x_i$
- $\Pr[e_1 \cap e_2] = \Pr[e_2|e_1] \cdot \Pr[e_1]$
- $\Pr[e_2|e_1] = 1/(n-i+1)$ because
  - x_i is available in $A[i..n]$ to be chosen since e_1 already occurred and did *not* include x_i
  - every element in $A[i..n]$ is equally likely to be chosen
- $\Pr[e_1] = (n-(i-1))!/n!$ by inductive hypothesis
- So $\Pr[e_1 \cap e_2] = [1/(n-i+1)] \cdot [(n-(i-1))!/n!]$

$$= (n-i)!/n!$$

Why Does It Work?

- After the last iteration (the *n*th), the inductive hypothesis tells us that

  A[1..n] equals each permutation of n elements from {1,…,n} with probability $(n-n)!/n! = 1/n!$
- Thus, the algorithm gives us a uniform random permutation

Randomized Algorithms

```
RANDOMIZED-HIRE-ASSISTANT(n)
randomly permute the list of candidate
best ← 0
for i ← 1 to n
   do interview candidate i
        if candidate i is better than candidate best
          then best ← i
                 hire candidate i
```

PERMUTE-BY-SORTING(A)

$n \leftarrow length[A]$

for $i \leftarrow 1$ **to** n

  do $P[i] \leftarrow$ RANDOM$(1, n^3)$ //Choose a random number in $\{1,\ldots,n^3\}$ //To make sure that all priorities P are unique.

sort A, using P as sort keys

return A

- Lemma 5.4

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

RANDOMIZE-IN-PLACE(A)

1 $n \leftarrow length[A]$

2 **for** $i \leftarrow 1$ **to** n

3 **do** swap $A[i] \longleftrightarrow A[\text{RANDOM}(i,n)]$

- Lemma 5.5

  Procedure RANDOMIZE-IN-PLACE computes a uniform random permutation

Quicksort (More detail in Chapter 7)

- Deterministic quicksort:
  - $\Theta(n^2)$ worst-case running time
  - $\Theta(n \log n)$ average case running time, assuming every input permutation is equally likely
- Randomized quicksort:
  - Do not rely on possibly faulty assumption about input distribution
  - Instead, randomize!

Randomized Quicksort

- Two approaches
- One is to randomly permute the input array and then do deterministic quicksort
- The other is to randomly choose the pivot element at each recursive call
  - called "random sampling"
  - easier to analyze
  - still gives Θ(n log n) expected running time

Randomized Quicksort

- Given array A[1..n], call recursive algorithm RandQuickSort(A,1,n).
- Definition of RandQuickSort(A,p,r):
  - if $p < r$ then
  - q := RandPartition(A,p,r)
  - RandQuickSort(A,p,q–1)
  - RandQuickSort(A,q+1,r)

Randomized Partition

- RandPartition(A,p,r):
  - i := randomly chosen index between p and r
  - swap A[r] and A[i]
  - return Partition(A,p,r)

Partition

- Partition(A,p,r):
  - x := A[r]   // the pivot
  - i := p–1
  - for j := p to r–1 do
  -     if A[j] ≤ x then
  -         i := i+1
  -         swap A[i] and A[j]
  - swap A[i+1] and A[r]
  - return i+1

A[r]: holds pivot
A[p,i]: holds elts ≤ pivot
A[i+1,j]: holds elts > pivot
A[j+1,r-1]: holds elts
not yet processed

Summary

probabilistic analysis of algorithms

randomized algorithms

randomized hiring

randomized quicksort