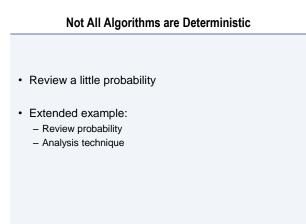
			Math Background: Review	/ & Beyond
		1.	Asymptotic notation	
Design & Analysis of Algorithms COMP 482 / ELEC 420	<b>1</b>	2.	Math used in asymptotics	
		3.	Recurrences	
John Greiner		4.	Probabilistic analysis	<u>To do:</u> [CLRS] 5 #2





$$Pr\{Event_1 | Event_2\} = \frac{Pr\{Event_1 \cap Event_2\}}{Pr\{Event_2\}}$$

# Random VariablesRandom VariablesWhat is the probability of getting exactly 1 "tail" when<br/>flipping two normal coins? $Pr{X=x} = \sum_{\{s \in S \mid X(s)=x\}} Pr{s}$ Pr{X=x} = \sum\_{\{s \in S \mid X(s)=x\}} Pr{s}X = #Tails, x=1<br/>S = TwoNormalCoinsFlipped = {HH, HT, TH, TT}Expected v<br/> $- E[X] = \Sigma$ $Pr{HTails=1}$ $= \sum_{(coins \in TwoNormalCoinsFlipped | #Tails(coins)=1} Pr{coins}$ $= Pr{HT} + Pr{TH}$ $= \sum_{(coins \in TwoNormalCoinsFlipped | #Tails(coins)=1} Pr{coins}$ = E[X+Y] $= \sum_{(coins \in TwoNormalCoinsFlipped | #Tails(coins)=1} Pr{coins}$ $= Pr{HT} + Pr{TH}$ $= \sum_{(coins \in TwoNormalCoinsFlipped | #Tails(coins)=1} Pr{coins}$ $= Pr{HT} + Pr{TH}$ = E[X+Y]= E[X+Y]= E[X+Y]

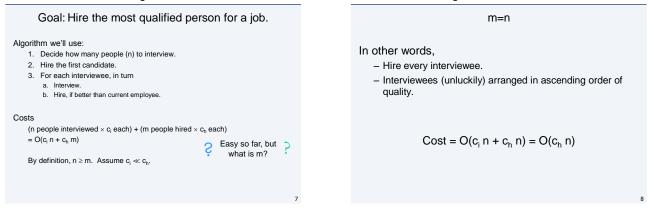
Random variables X,Y "independent"  $\leftrightarrow$ Pr{X=x  $\land$  Y=y} = Pr{X=x}  $\times$  Pr{Y=y}

Expected values of random variables:

 $- \mathsf{E}[\mathsf{X}] = \Sigma_{\mathsf{x}} (\mathsf{x} \times \mathsf{Pr}\{\mathsf{X}=\mathsf{x}\})$ 

- $\label{eq:expectation} \begin{array}{l} \ E[X+Y] = E[X] + E[Y] & \mbox{linearity of expectation} \\ \ E[a \times X] = a \times E[X] \end{array}$
- $E[X \times Y] = E[X] \times E[Y]$  if X,Y independent

### **Hiring Problem**



# Hiring Problem: Average-Case

What is m likely to be?

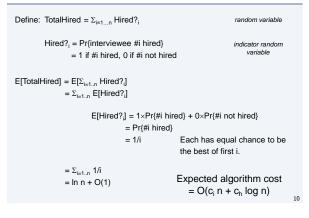
Wrong answer: m can be 1...n, so use the mean m=(n+1)/2.

Instead, average cost over all possible interviewee orders. Number each interviewee 1...n. Consider each permutation of the set {1,...,n}.

To compute this average, use probabilistic techniques to avoid listing every possible permutation.

### Averaging via Expected Values

Hiring Problem: Worst-Case



### **On-Line vs. Off-Line Hiring Problem**

Let's say the candidates come from an employment agency.

Algorithm's cost is greatly affected by the agency.

- Agency could send best applicants first.
  - Typical of well-run agencies.
  - Waste of money to interview later applicants.
- Agency could send worst applicants first.
  - Maximizes their fees.

We can randomize order, to minimize chance of worst case.

- Protects against poor input distributions. (Accidental or malicious.)

11

- Can still produce worst case, but only with low probability.



Let n = # elements in A For i = 1 to n, swap A[i] with A[random(i...n)] Simple algorithm, with an interesting proof that it computes a uniform random permutation.

The key – use a loop invariant:

Before i<sup>th</sup> iteration, for any (i-1)-permutation of elements from 1...n, A[1...i-1] contains the permutation with probability ((n-i+1)!)/(n!).

2

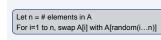
# Randomizing Order of an Array

Proving loop invariant: Before ith iteration, for any (i-1)-

permutation of elements from 1...n,

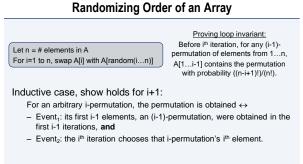
A[1...i-1] contains the permutation

with probability ((n-i+1)!)/(n!).



Base case, i=1:

A[1..0] contains the 0-permutation (no data) with probability (n!)/(n!)=1. Holds trivially.



 $Pr{Event_1 \cap Event_2} = Pr{Event_1} \times Pr{Event_2 | Event_1}$  $= (n-i+1)!/n! \times 1/(n-i+1)$ 

Inductive hypothesis Each remaining element equally likely

### Randomizing Order of an Array

 Let n = # elements in A

 For i=1 to n, swap A[i] with A[random(i...n)]

 Before i<sup>th</sup> iteration, for any (i-1)-permutation of elements from 1...n, A[1...i-1] contains the permutation with probability ((n-i+1)!)/(n!).

 Inductive case, show holds for i+1:

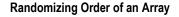
For an arbitrary i-permutation, the permutation is obtained  $\leftrightarrow$ 

- Event<sub>1</sub>: its first i-1 elements, an (i-1)-permutation, were obtained in the first i-1 iterations, and
- Event2: the ith iteration chooses that i-permutation's ith element.

 $Pr{Event_1 \cap Event_2} = Pr{Event_1} \times Pr{Event_2 | Event_1}$ 

 $= (n-i+1)!/n! \times 1/(n-i+1)$ 

= (n-i)!/n!	Algebra, cancelling term	
= (n-(i+1)+1)!/n!	Algebra	15



Let n = # elements in A For i=1 to n, swap A[i] with A[random(i...n)]

Proving loop invariant: Before i<sup>th</sup> iteration, for any (i-1)permutation of elements from 1...n, A[1...i-1] contains the permutation with probability ((n-i+1)!)/(n!).

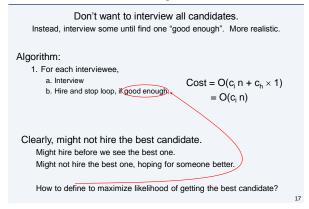
At termination (i=n+1)

Invariant  $\rightarrow$  any n-permutation of elements 1...n occurs in A[1..n] with probability 1/(n!).

I.e., uniform random distribution.

16

# **On-Line Hiring Problem**



# On-Line Hiring Problem

One possible algorithm:

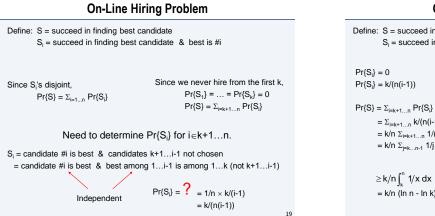
Interview k candidates, and hire the first one after that with a better score than previously found.

BestScore := -∞ For i=1 to k, BestScore = max(BestScore, Score(i)) For i=k+1 to n, If Score(i) > BestScore, Then Hire #i & Quit Hire #n

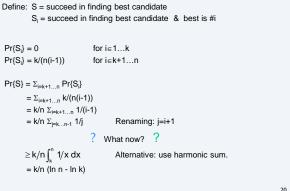
Strategy to find best k:

- Fix k, & compute the probability of getting best candidate.
- 2. Find k to maximize probability.

18



# **On-Line Hiring Problem**



On-Line Hiring Problem				
$Pr\{S\} \ge k/n \ (ln \ n \ - \ ln \ k)$	Find k to maximize Pr{S}.			
In k = In n/e k = n/e	ero: lug in: $Pr{S} \ge (n/e)/n (ln n - ln (n/e))$ = 1/e (ln n - ln n + ln e) $= 1/e \times 1$ = 1/e $\approx 0.368$ Algorithm has better than 1/3 ance of getting best candidate.			