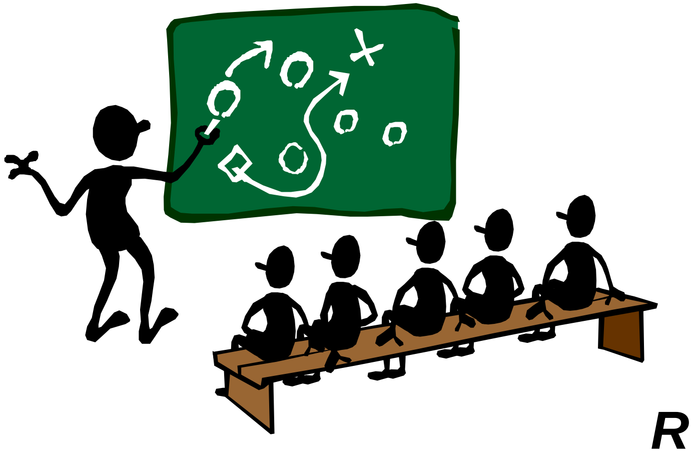
Algorithms – Chapter 5
Randomized Algorithms



Juinn-Dar Huang

Professor

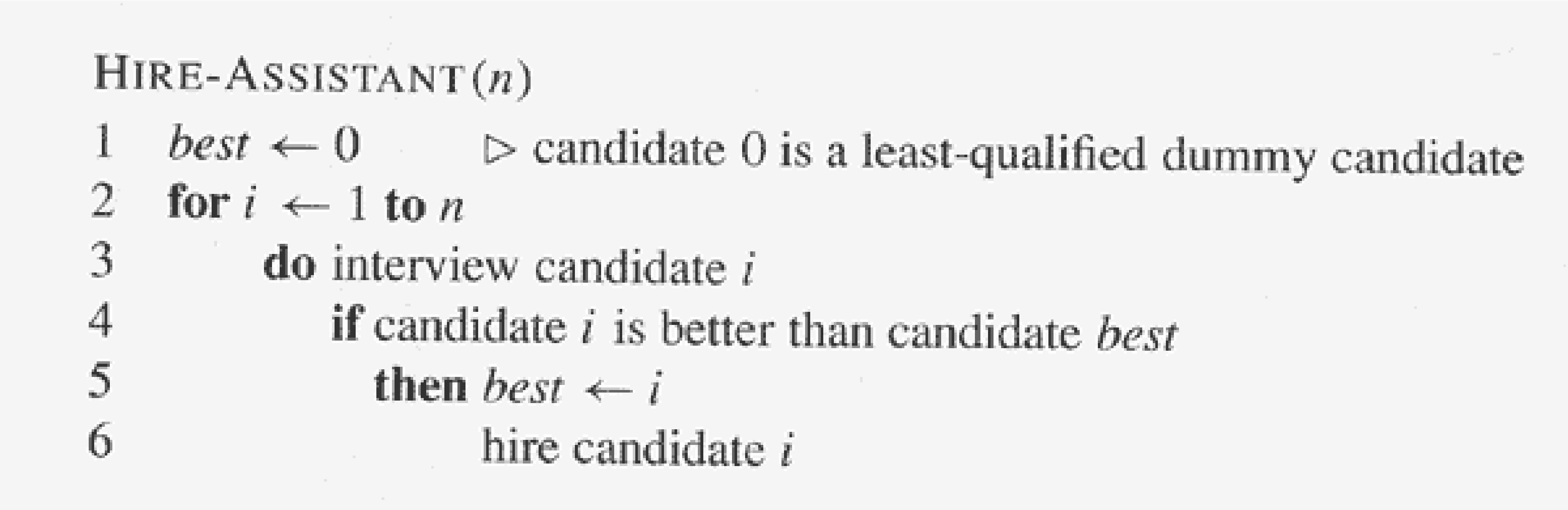
jdhuang@mail.nctu.edu.tw

August 2007

Rev. ’08, ’11, ’12, ’15, ’16, ’18, ’19, ’20, ’21

The Hiring Problem

- Hiring a good assistant is not an easy job…



```
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 cost : O($n*c_i + m*c_h$)
  - n : # of interviewers; m : # of hired people
  - c_i : interview cost; c_h : hiring cost; $c_i << c_h$
  - $n*c_i$ is a constant

Analysis

- Worst-case analysis
  - hire every candidate → $O(n*c_h)$
- Typical (average) case
  - the quality of interviewers should be in random order!
  - → uniform random distribution
  - Answer: $O(\ln n * C_h)$; why?

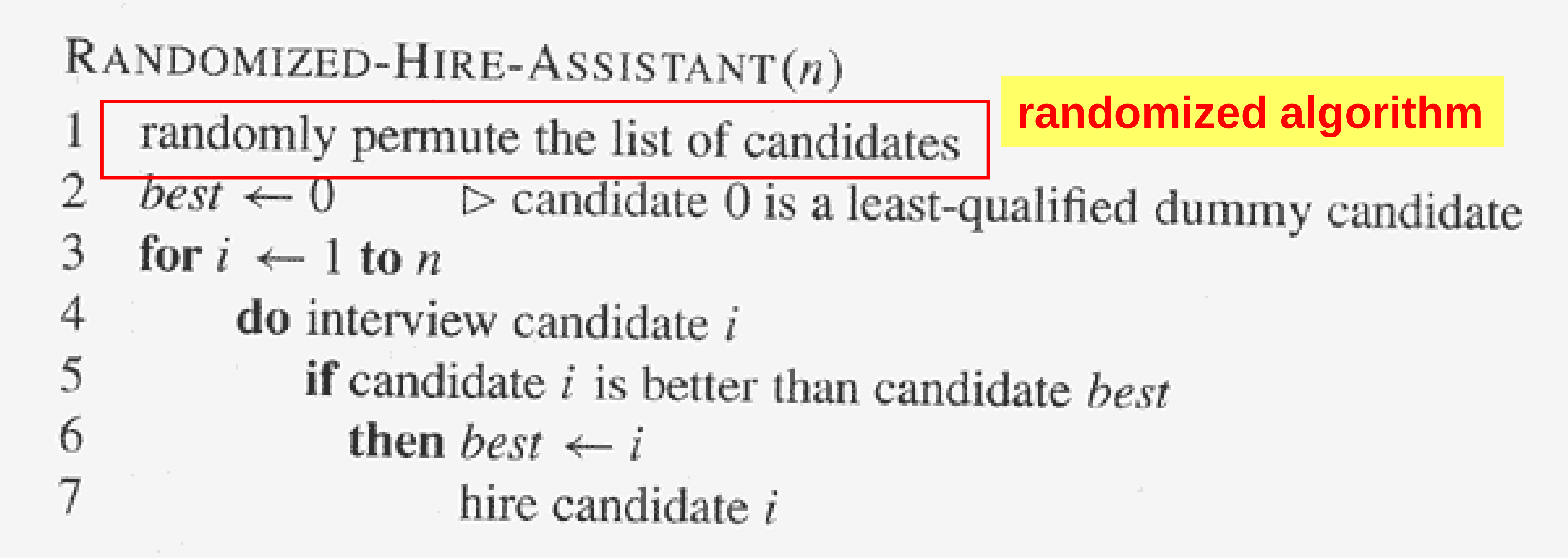
Indicator Random Variables

- Let X_i = I{ candidate i is hired } =
  - 1, if candidate i is hired
  - 0, if candidate i is not hired
- Let X be the random variable representing the number of hires
  - $X = X_1 + X_2 + \ldots + X_n$
- **Key: $E[X_i] = 1/i$**

➡ $$E[X]=E[\sum_{i=1}^{n} X_i]=\sum_{i=1}^{n} E[X_i]=\sum_{i=1}^{n} 1/i=\boxed{\ln n}+O(1)$$

Randomized Algorithms

- An algorithm is randomized if its behavior is determined not only by its input but also by values produced by a random number generator



```
RANDOMIZED-HIRE-ASSISTANT($n$)
randomly permute the list of candidates
$best \leftarrow 0$        ▷ candidate 0 is a least-qualified dummy candidate
for $i \leftarrow 1$ to $n$
    do interview candidate $i$
        if candidate $i$ is better than candidate $best$
            then $best \leftarrow i$
                hire candidate $i$
```

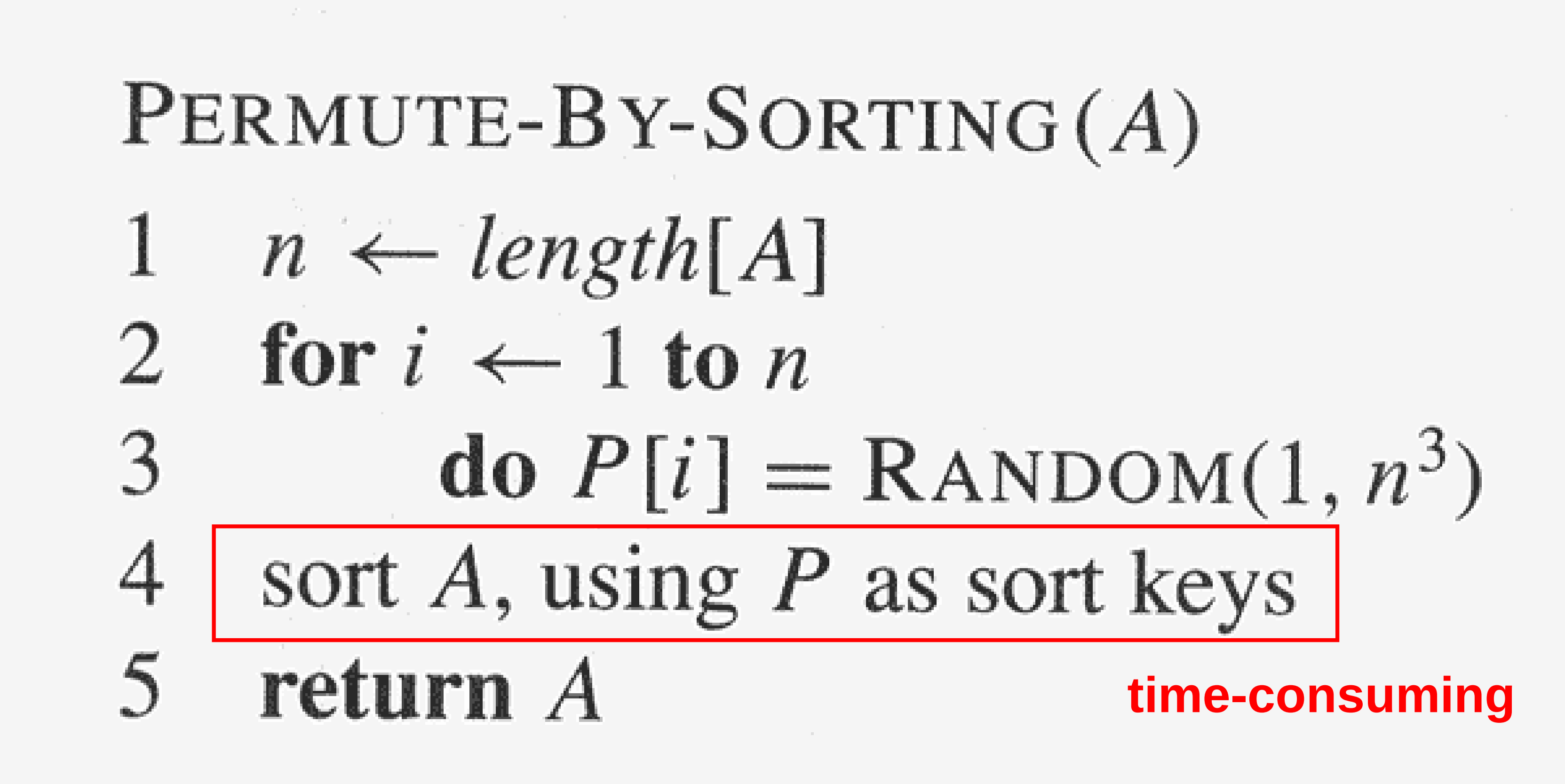
randomized algorithm

Worst-case and average-case time complexity ?

Random Permutations

- Many randomized algorithms randomize the input by permuting the given input array
  - e.g., 撲克牌發牌程式
  - how to make sure it's a uniform random permutation?

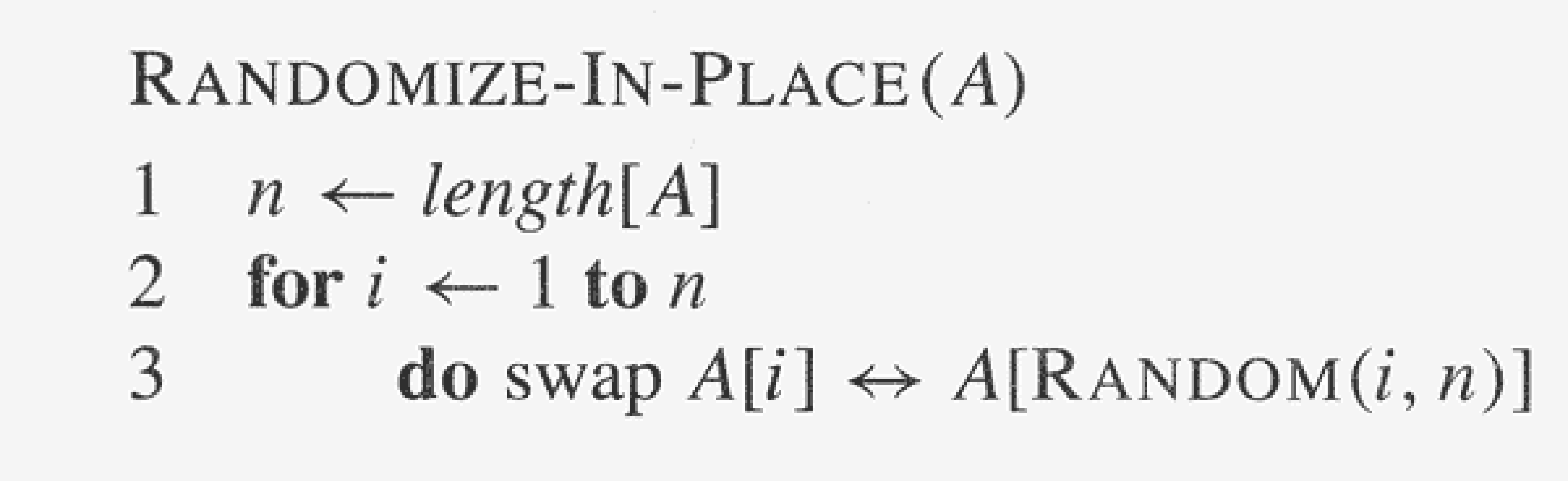
Permute by Sorting



```
PERMUTE-BY-SORTING(A)
n ← $length[A]$
for $i$ ← 1 to $n$
        do $P[i]$ = RANDOM($1, n^3$)
sort $A$, using $P$ as sort keys
return $A$
```

time-consuming

In-Place Randomization



```
RANDOMIZE-IN-PLACE($A$)
n ← length[A]
for i ← 1 to n
        do swap A[i] ↔ A[RANDOM(i, n)]
```

Guarantee to generate a uniform random permutation

Simple, Fast, and Elegant !