```
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
```

Worst-case: *n* hires

Average-case analysis using indicator variables: ln *n* + O(1) hires

- assumes each of the *n!* permutations of rankings is equally likely

Expected analysis: average run-time, <u>regardless of the input</u>

achieved by randomizing the input

A randomized algorithm for the hiring problem

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
```

Randomly permuting arrays

PERMUTE-BY-SORTING(A)

$n = A.length$
let $P[1\,..\,n]$ be a new array
for $i = 1$ **to** n
    $P[i] = \text{RANDOM}(1, n^3)$
sort A, using P as sort keys

Makes it very likely that the *n* values will all be unique

Randomly permuting arrays

RANDOMIZE-IN-PLACE(A)

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