

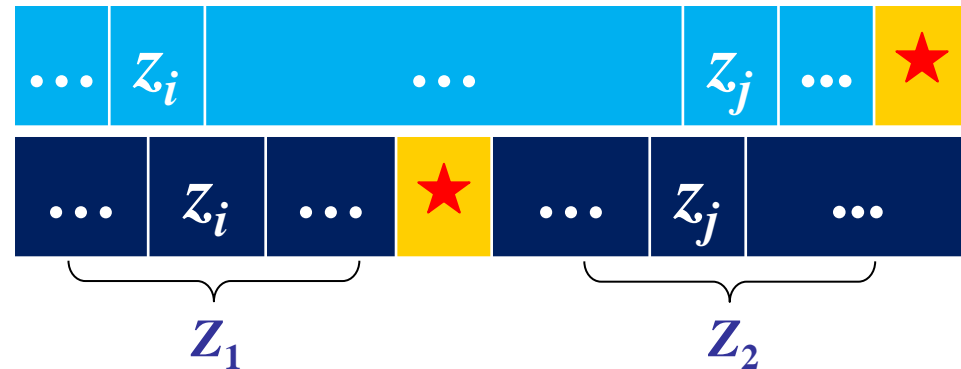
Chapter 5

Probabilistic Analysis and Randomized Algorithms

Why is QUICKSORT quick?

```
QUICKSORT( $A, p, r$ )
```

```
1  if  $p < r$   
2     $q = \text{PARTITION}(A, p, r)$   
3    QUICKSORT( $A, p, q - 1$ )  
4    QUICKSORT( $A, q + 1, r$ )
```



$$T(n) = T(x) + T(n-x-1) + \Theta(n) \quad ?$$

(1) $T(n) = T(n/2) + T(n/2 - 1) + \Theta(n)$ (分区位置是中间时)

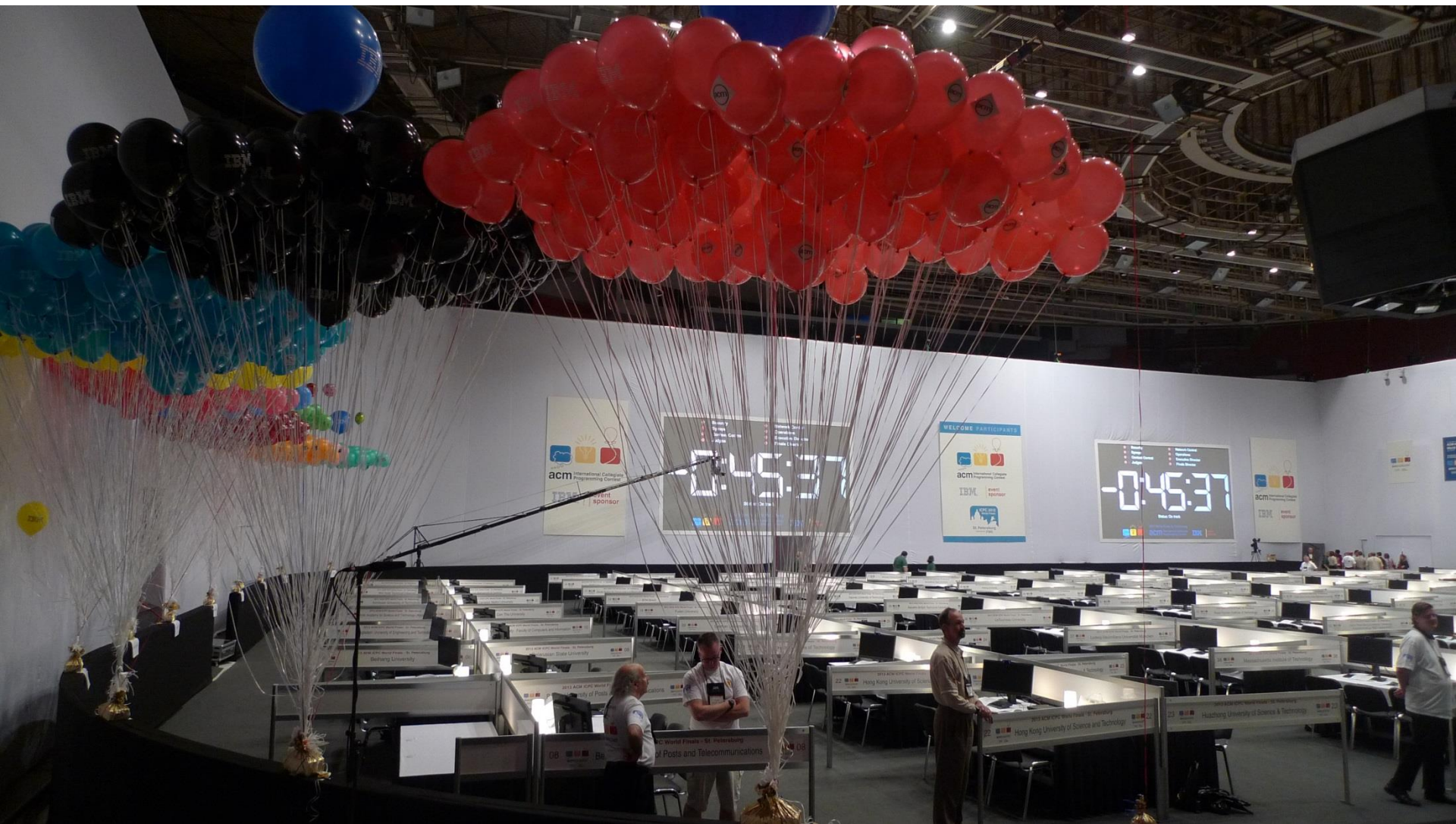
(2) $T(n) = T(a) + T(n-a-1) + \Theta(n)$ (a 是常数)

(3)

.....

5 Probabilistic Analysis and Randomized Algorithms

- Explain the difference between probabilistic analysis & randomized algorithms.
(概率分析与随机算法)
- Present the technique of indicator random variables.
(用指示(器)随机变量来对算法进行概率分析)
- Give another example of a randomized algorithm.
(一个随机算法实例)





2013 ACM-ICPC World Final, St. Petersburg, Russia



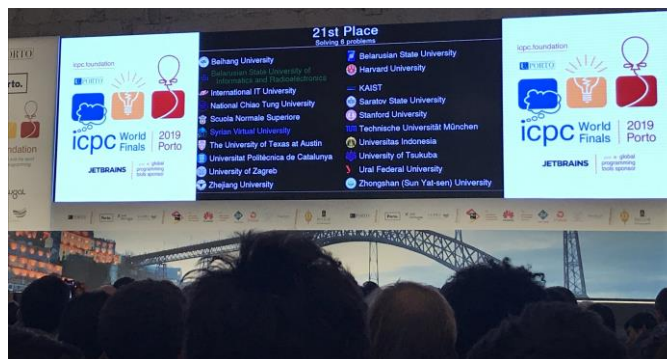
2015 ACM-ICPC World Final, Marrakech, Morocco



2016 ACM-ICPC World Final, Phuket, Thailand



2019 ICPC World Final, Porto, Portugal

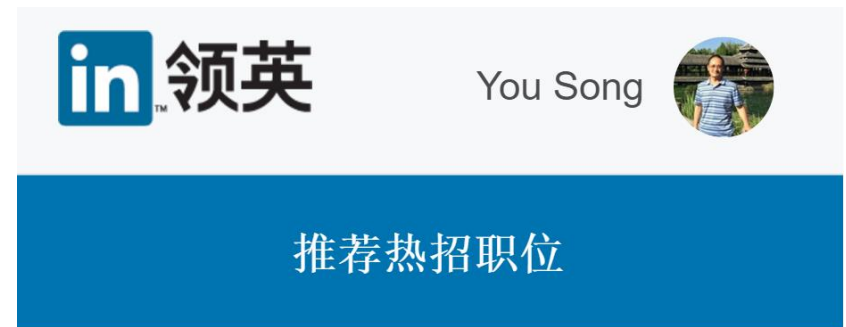


5.1 The hiring problem (雇佣问题, or 助手问题)



Scenario (情景): hire the best office assistant in a month

- You are using an employment agency to hire a new office assistant. (猎头 (代理) 公司帮你物色办公助理候选人)



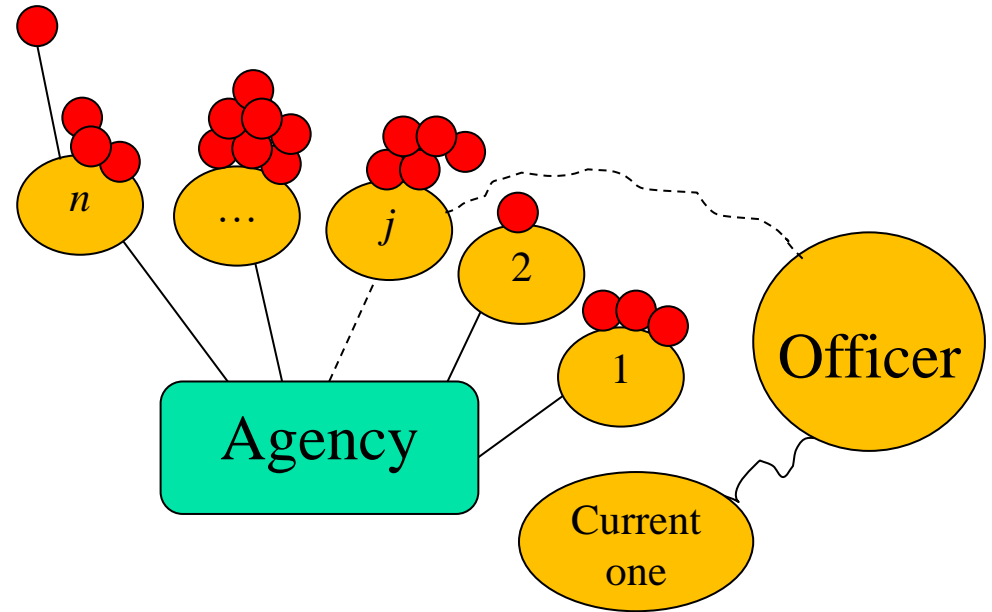
Computer Vision and Machine Learning Engineer

苹果 · Beijing, CN



8 位前同事在这里工作

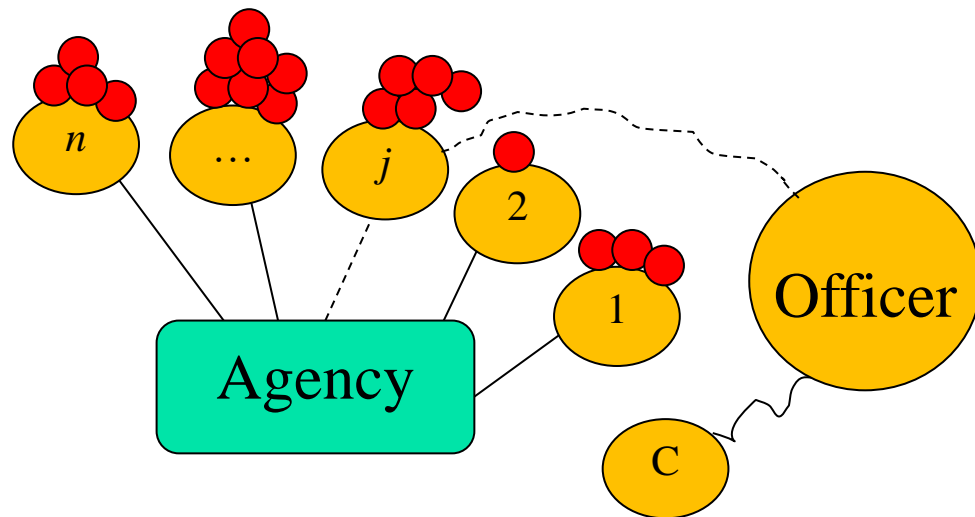
5.1 The hiring problem



Scenario (情景): hire the best office assistant in a month

- You are using an employment agency to hire a new office assistant.
(猎头(代理)公司帮你物色办公助理候选人)
- The agency sends you one candidate each day.
(1, 2, ..., n 表示候选人的编号, 红色圆圈候选人的工作能力)
- You interview the candidate and must immediately decide whether or not to hire him. But if you hire, you must also fire your current one.
-

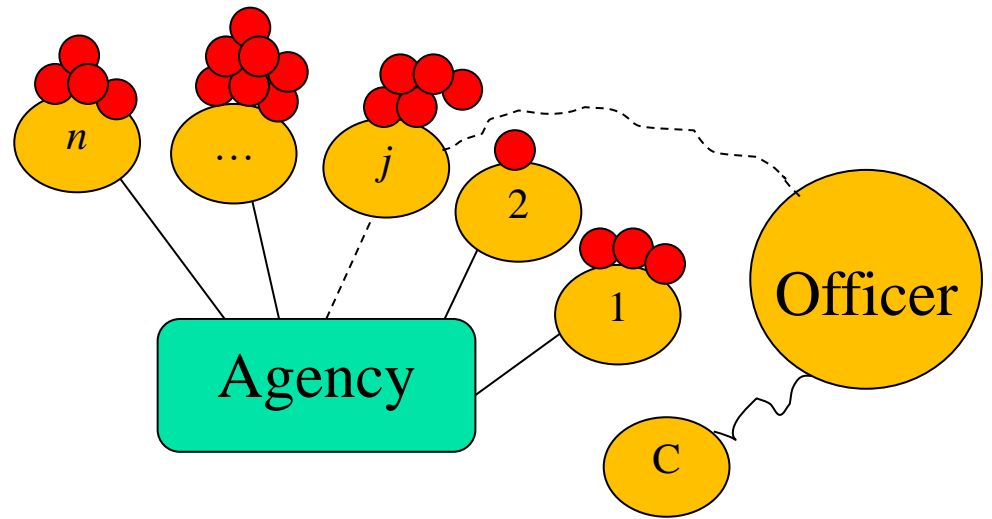
5.1 The hiring problem



- 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 him. But if you hire, you must also fire your current one.
- Cost to interview is 1K per candidate (interview fee paid to agency). (面试一个候选人支付代理公司 1K)
- Cost to hire is 10K per candidate, includes:
cost to fire current office assistant + hiring fee paid to agency

Goal: Determine what the price of this strategy will be?

5.1 The hiring problem



Pseudocode to model this scenario: Assumes that the candidates are numbered 1 to n . Uses a dummy candidate 0 that is worse than all others, so that the first candidate is always hired.

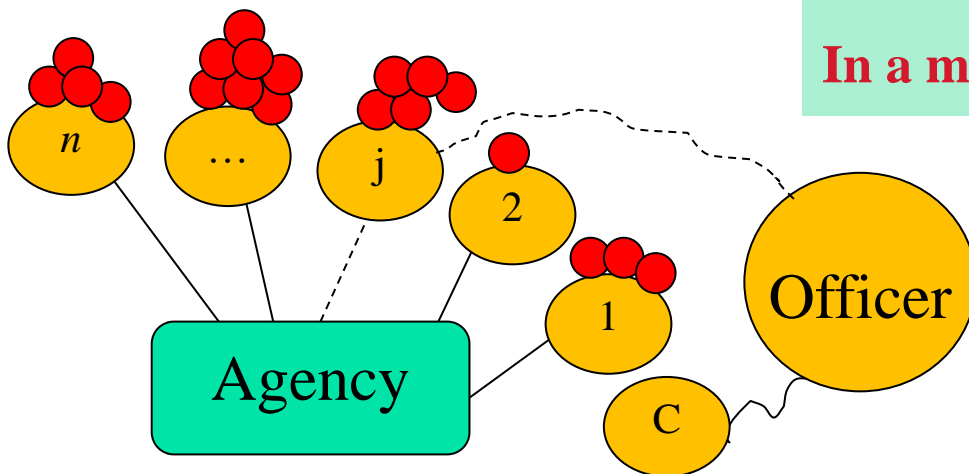
HIRE-ASSISTANT(n)	cost	times
$best \leftarrow 0$ // candidate 0 is a least-qualified dummy candidate		
for $i \leftarrow 1$ to n		
interview candidate i	c_i	n
if candidate i is better than candidate $best$		
$best \leftarrow i$		
hire candidate i	c_h	m

Cost: n candidates, we hire m of them, cost is $T(n) = nc_i + mc_h$

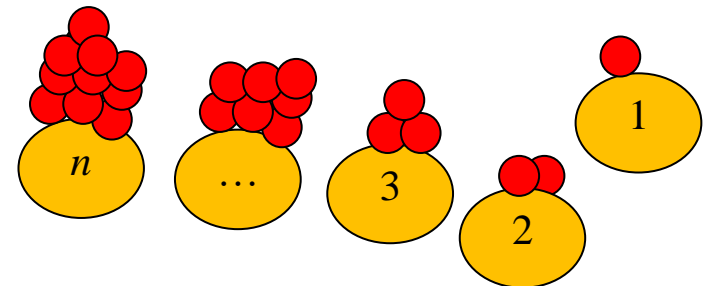
5.1.1 Worst-case analysis

Cost: n candidates, we hire m of them, cost is $T(n) = nc_i + mc_h$

- We focus on analyzing the hiring cost mc_h ? ($c_h \gg c_i$)
- mc_h varies with each run of the algorithm: it depends on the order in which we interview the candidates.
- **Worst-case analysis**
 - ◆ We hire all n candidates. $T(n) = nc_i + nc_h$? When?
 - ◆ **The candidates appear in increasing order of quality.**



In a month, $T(30) = 30 + 30 \cdot 10 = 330$ K



5.1.1 Worst-case analysis

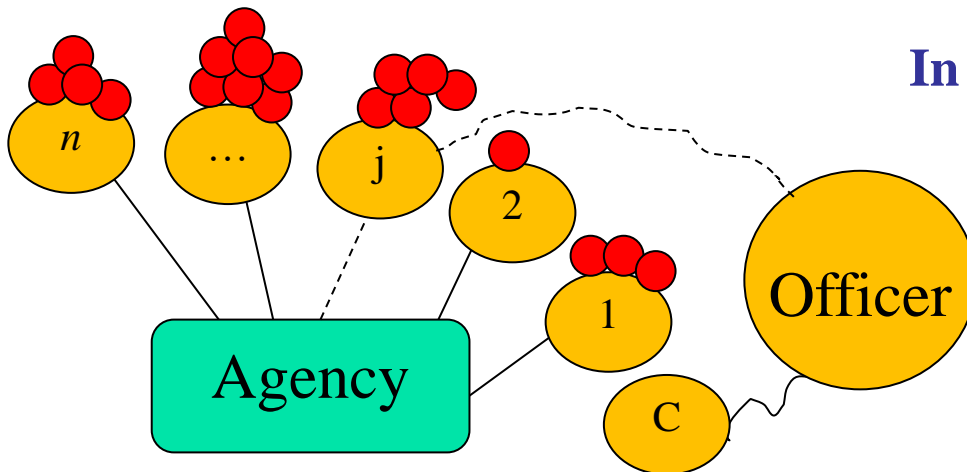
Cost: n candidates, we hire m of them, cost is $T(n) = nc_i + mc_h$

- Worst-case analysis

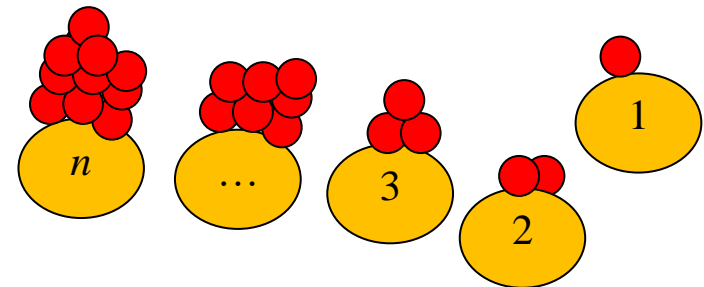
- ◆ We hire all n candidates. $T(n) = nc_i + nc_h$.

- Average-case ?

- Best-case ?



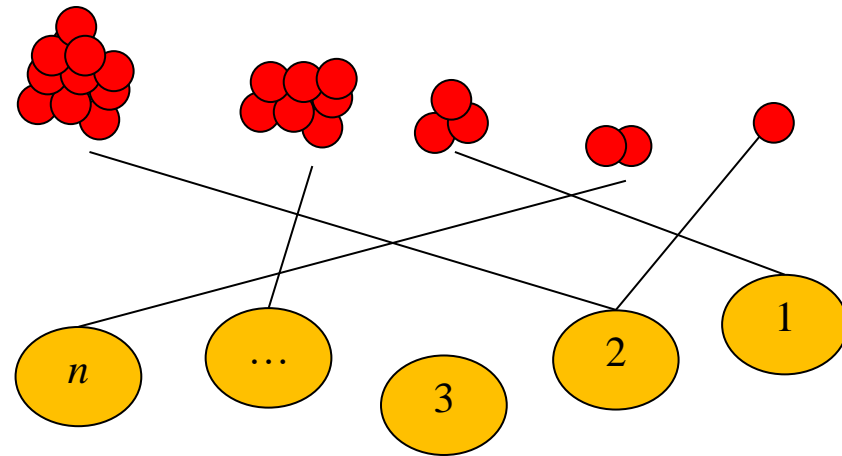
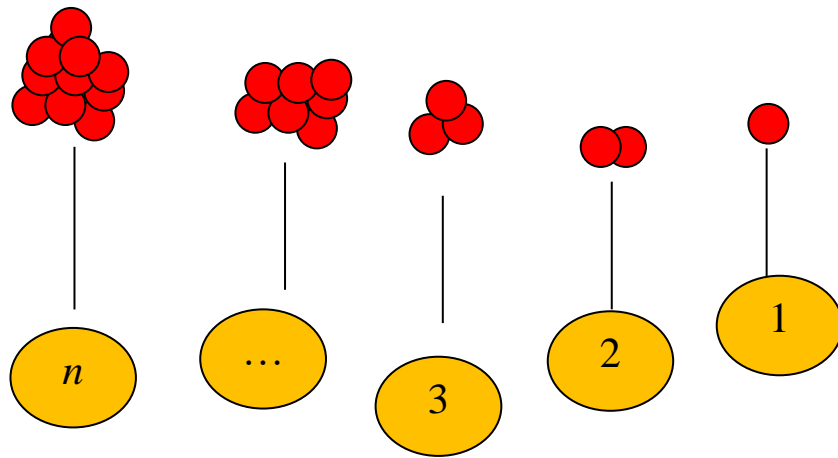
In a month, $T(30) = 30 + 30 \cdot 10 = 330$ K



5.1.2 Probabilistic analysis

Assume that candidates come in a random order:

- Assign a rank to each candidate: $\text{rank}(i)$ is a unique integer in the range 1 to n . (对每一个候选人分配一个“名次”)
- The list $\langle \text{rank}(1), \dots, \text{rank}(n) \rangle$ is a permutation of the candidate numbers $\langle 1, \dots, n \rangle$, such as $\langle 5, 2, 1, 28, 9, \dots, 11 \rangle$
- The ranks form a uniform random permutation: each of the possible $n!$ permutations appears with equal probability



5.1.2 Probabilistic analysis

HIRE-ASSISTANT(n)	cost	times
$best \leftarrow 0$ // candidate 0 is a least-qualified dummy candidate		
for $i \leftarrow 1$ to n		
interview candidate i	c_i	n
if candidate i is better than candidate $best$		
$best \leftarrow i$		
hire candidate i	c_h	m

Essential idea of probabilistic analysis: We must use knowledge of, or make assumptions about, the distribution of inputs.

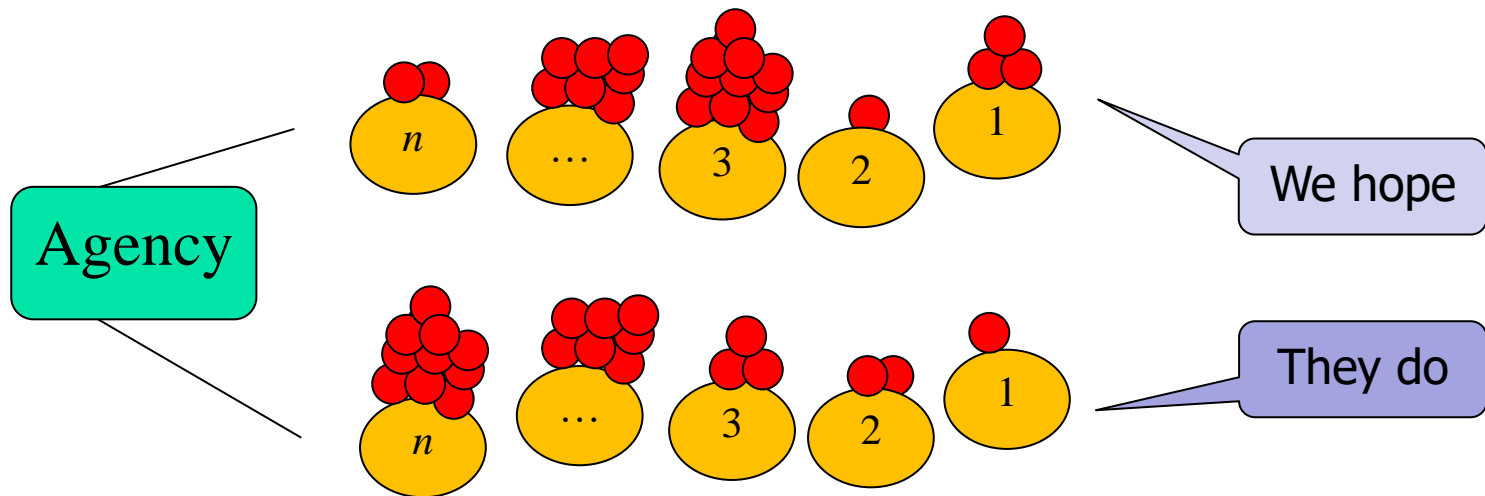
（概率分析的本质：需已知或假定输入的概率分布）

- The expectation is taken over this distribution.
（依据概率分布来求期望，期望值即是平均 hire 的人数）
- Section 5.2 contains a probabilistic analysis of the hiring problem.

5.1.3 Randomized algorithms

Idea

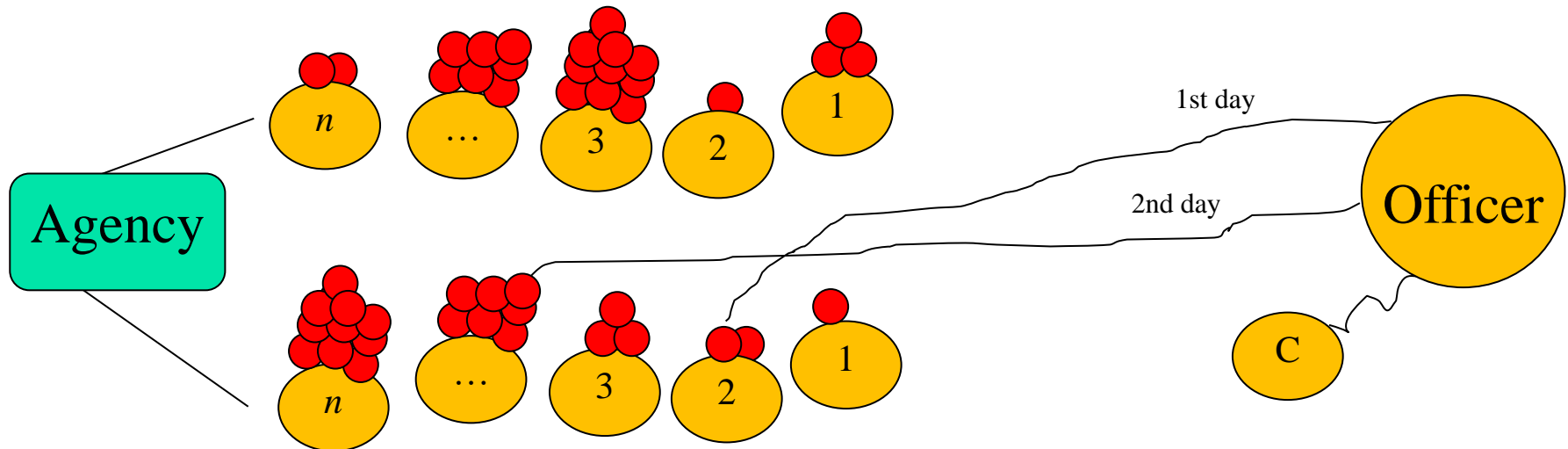
- ◆ We might not know the distribution of inputs, or we might not be able to model it computationally.
(我们不知道输入的分布，也不可能为输入的分布进行可计算的建模)
- ◆ Instead, we use randomization within the algorithm in order to impose a distribution on the inputs.
(在算法中通过对输入进行随机化处理，从而为输入强加一种分布)



5.1.3 Randomized algorithms

For the hiring problem, change the scenario:

- ◆ The employment agency sends us a list of all n candidates in advance.
(猎头公司预先提供 n 个候选人列表)
- ◆ On each day, we randomly choose a candidate from the list to interview.
(每天, 我们随机选取一人进行面试)
- ◆ 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. (Chap 5.3)
(无须担心候选人是否被随机提供, 我们通过随机运算预处理可以控制候选人的随机顺序)



5.1.3 Randomized algorithms

What makes an algorithms randomized: An algorithm is randomized if its behavior is determined in part by values produced by a *random-number generator*.

(算法随机化：由随机数产生器生成数值.....)

- ◆ **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. (RANDOM实际上由一个确定的算法〔伪随机产生器〕产生，其结果表面上看上去像是随机数)

5.1.3 Randomized algorithms

Random number generator (随机数产生器)

- ◆ Most random number generators generate a sequence of integers by the following recurrence
- ◆ X_0 = a given integer (seed), $X_{i+1} = aX_i \bmod M$
- ◆ For example, for $X_0=1$, $a=5$, $M=13$, we have $X_1=5\%13=5$, $X_2=25\%13=12$, $X_3=60\%13=8$. Each integer in the sequence lies in the range $[0, M - 1]$.

A probabilistic analysis of the hiring problem

5.2 Indicator random variables (指示随机变量)



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

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

➤ **Lemma**

For an event, let $X_A = I\{A\}$. Then $E[X_A] = \Pr\{A\}$.

Proof Letting $\sim A$ be the complement of A , we have

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

5.2 Indicator random variables (指示随机变量)

➤ *Lemma*

For an event A , let $X_A = I\{A\}$.

Then $E[X_A] = \Pr\{A\}$.



- **Simple example: Determine the expected number of heads when we flip a fair coin one time.**

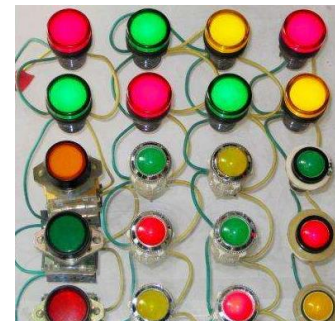
(投一次硬币，正面向上的期望〔平均数〕)

- ◆ Sample space is $\{H, T\}$
- ◆ $\Pr\{H\} = \Pr\{T\} = 1/2$
- ◆ Define indicator random variable $X_H = I\{H\}$.
 X_H counts the number of heads in one flip.
- ◆ Since $\Pr\{H\} = 1/2$, lemma says that $E[X_H] = 1/2$.

5.2 Indicator random variables (指示随机变量)

➤ Lemma

For an event A , let $X_A = I\{A\}$.
Then $E[X_A] = \Pr\{A\}$.



- **Slightly more complicated example:** Determine the expected number of heads when in n coin flips.

(投 n 次硬币, 正面向上的期望〔平均数〕)

- ◆ Let X be a random variable for the number of heads in n flips. (令随机变量 X 表示投 n 次硬币正面向上的数)

- ◆ Then,
$$E[X] = \sum_{k=0}^n k \cdot \Pr\{X = k\}$$

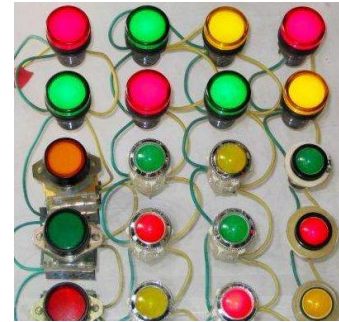
a slightly more complicated? Yes!

- ◆ Instead, we'll use indicator random variables.....

5.2 Indicator random variables (指示随机变量)

□ Lemma

For an event A , let $X_A = I\{A\}$.
Then $E[X_A] = \Pr\{A\}$.



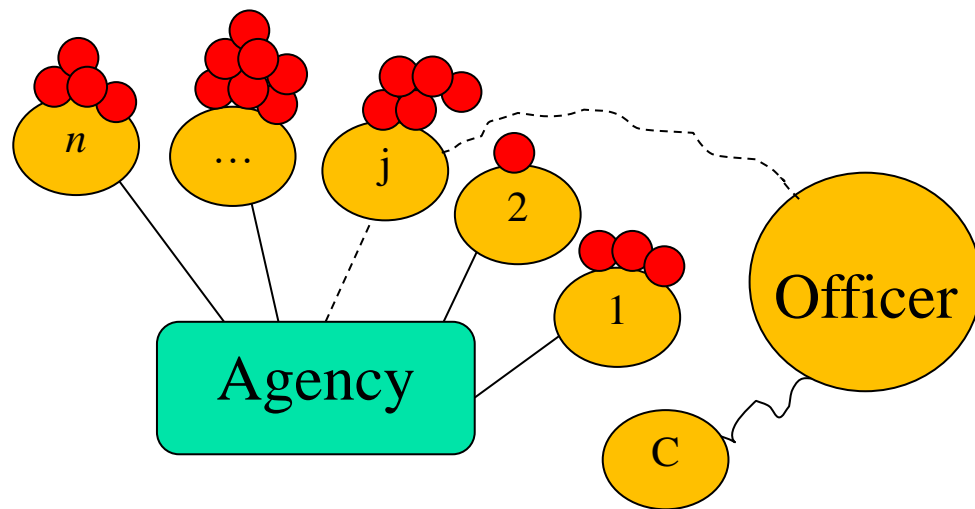
- **Slightly more complicated example:** Determine the expected number of heads when in n coin flips.

(投 n 次硬币, 正面向上的期望〔平均数〕)

- ◆ For $i=1, \dots, n$, define $X_i = I\{\text{the } i\text{th flip results in event } H\}$
- ◆ Then, $X = \sum_{i=1}^n X_i$
- ◆ Lemma says that $E[X_i] = \Pr\{H\} = 1/2$ for $i=1, 2, \dots, n$
- ◆ Expected number of heads is

$$E[X] = E\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n E[X_i] = \sum_{i=1}^n 1/2 = n/2$$

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 new office assistant.

(令随机变量 X 表示我们雇用新助手的人数)

- Use a probabilistic analysis

Then

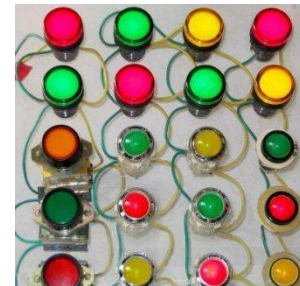
$$E[X] = \sum_{i=1}^n i \Pr\{X = i\}$$

The calculation would be cumbersome (计算烦琐)

Analysis of the hiring problem

Assume that the candidates arrive in a random order.

random variable X = the number of times we hire new office assistant (随机变量 X 表示我们雇用新助手的人数)



- **Use indicator random variables**

- ◆ Define indicator random variables X_1, X_2, \dots, X_n , where $X_i = I\{\text{candidate } i \text{ is hired}\}$

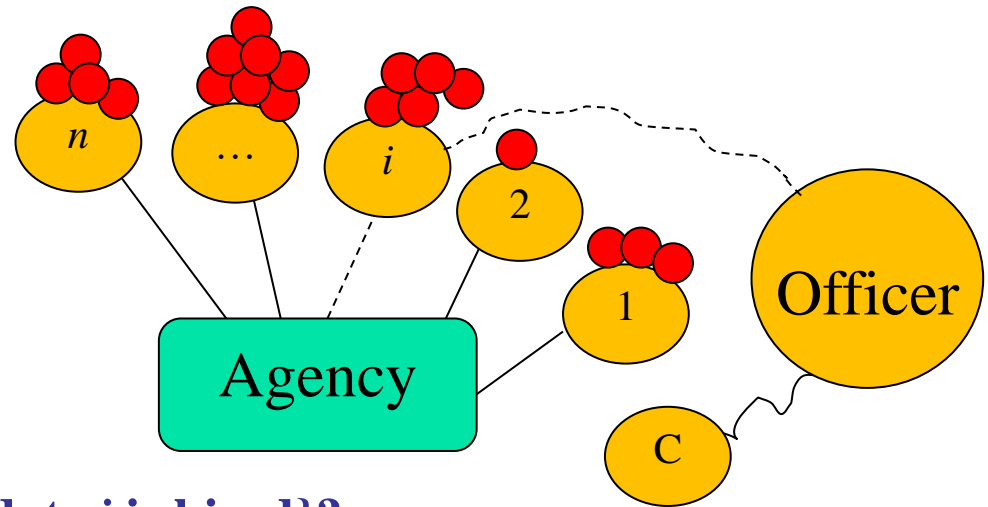
- ◆ Useful properties:

$$X = X_1 + X_2 + \dots + X_n$$

Lemma $\Rightarrow E[X_i] = \Pr\{\text{candidate } i \text{ is hired}\}.$

We need to compute $\Pr\{\text{candidate } i \text{ is hired}\}?$

Analysis of the hiring problem

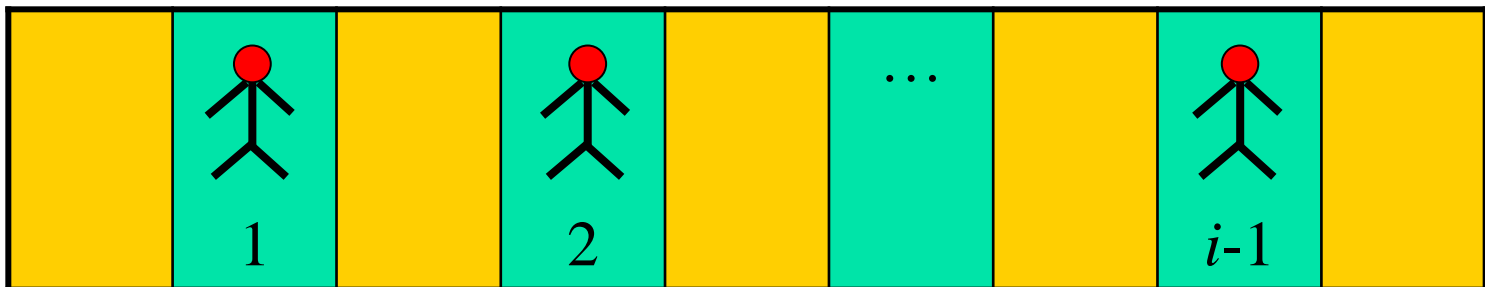


We need to compute $\Pr\{\text{candidate } i \text{ is hired}\}$?

- ◆ i is hired $\Leftrightarrow i$ is better than each of candidates $1, 2, \dots, i-1$.
- ◆ Assumption that the candidates arrive in random order \Rightarrow any one of these candidates is equally likely to be the best one.

(若候选人随机到来, 则每一个候选人为最佳人选的概率相等)

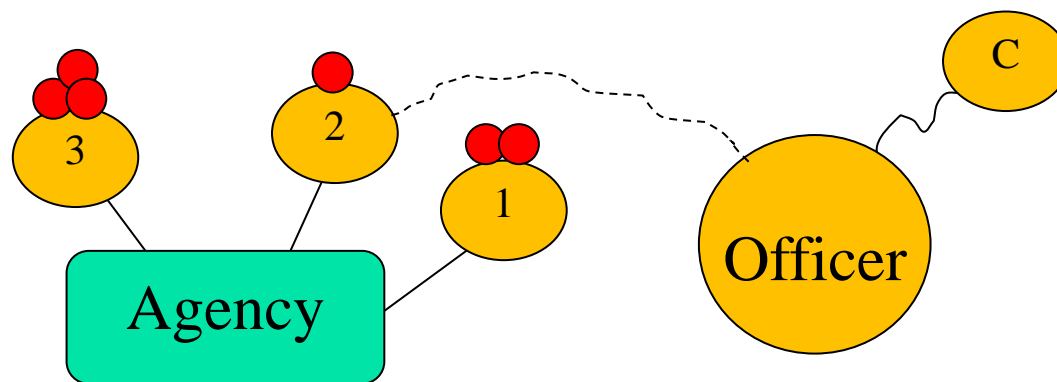
- ◆ Thus, $E\{X_i\} = \Pr\{\text{candidate } i \text{ is the best so far}\} = 1/i$?



Analysis of the hiring problem

We need to compute $\Pr\{\text{candidate } i \text{ is hired}\}$?

- ◆ i is hired $\Leftrightarrow i$ is better than each of candidates $1, 2, \dots, i-1$.
- ◆ Thus, $E\{X_i\} = \Pr\{\text{candidate } i \text{ is the best so far}\} = 1/i$?



viewed : viewing , Pr of hired

已面试：待面试，待面试人被雇佣的概率

0 : {1,2,3}, 1

1:{2, 3}, $1/3 * 1$

2:{1, 3}, $1/3 * 1/2$

3:{1, 2}, $1/3 * 0$

{1, 2}:3, $1/3 * 1$

{1, 3}:2, $1/3 * 0$

{2, 3}:1, $1/3 * 0$

第一天面试的人的资历可能是1or2or3，每种情况的Pr是1/3；若是第一种情况，第二天面试的人被雇佣的Pr是1，则条件概率 $\Pr = 1/3 * 1$ ；其他情况相似。

Analysis of the hiring problem

HIRE-ASSISTANT(n)

$best \leftarrow 0$ // candidate 0 is a least-qualified dummy candidate

for $i \leftarrow 1$ to n

 interview candidate i

 if candidate i is better than candidate $best$

$best \leftarrow i$

 hire candidate i

cost times

c_i n

c_h m

$$T(n) = nc_i + mc_h$$

We need to compute $\Pr\{\text{candidate } i \text{ is hired}\}$?

- ♦ i is hired $\Leftrightarrow i$ is better than each of candidates $1, 2, \dots, i-1$.
- ♦ Assumption that the candidates arrive in random order \Rightarrow any one of these candidates is equally likely to be the best one. (若候选人随机到来, 则每一个候选人为最佳人选的概率相等)
- ♦ Thus, $E\{X_i\} = \Pr\{\text{candidate } i \text{ is the best so far}\} = 1/i$?

Then

$$E[X] = E\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n E[X_i] = \sum_{i=1}^n 1/i = \ln n$$

Thus, the expected hiring cost is $nc_i + (\ln n)c_h$, which is much better than the worst-case cost of $nc_i + nc_h$.

$$(nc_i + mc_h): 30 + 3.4 * 10 = 64 \quad \text{vs} \quad 30 + 30 * 10 = 330$$

$$(\ln 30 = 3.4\dots)$$

$$\ln n \quad \text{vs} \quad n$$

■ Analysis of the hiring problem

Harmonic series

For positive integers n , the n th *harmonic number* is

$$\begin{aligned} H_n &= 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n} \\ &= \sum_{k=1}^n \frac{1}{k} \\ &= \ln n + O(1). \end{aligned} \tag{A.7}$$

(We shall prove a related bound in Section A.2.)

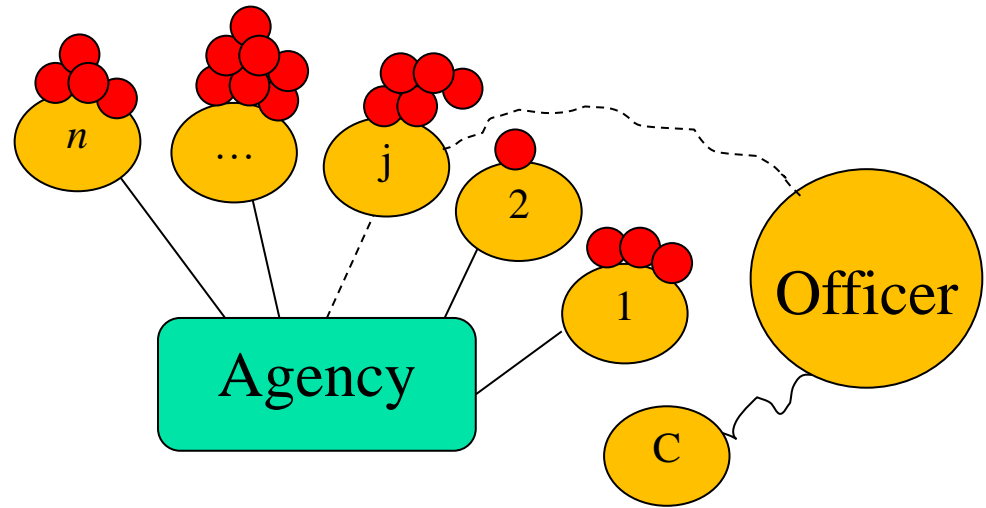
$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathbb{E}[X_i] = \sum_{i=1}^n 1/i = \ln n$$

Thus, the expected hiring cost is $nc_i + (\ln n)c_h$, which is much better than the worst-case cost of $nc_i + nc_h$.

$(nc_i + mc_h)$: $30 + 3.4 \cdot 10 = 64$ vs $30 + 30 \cdot 10 = 330$. ($\ln 30 = 3.4 \dots$)

$\ln n$ vs n

5.3 Randomized algorithms

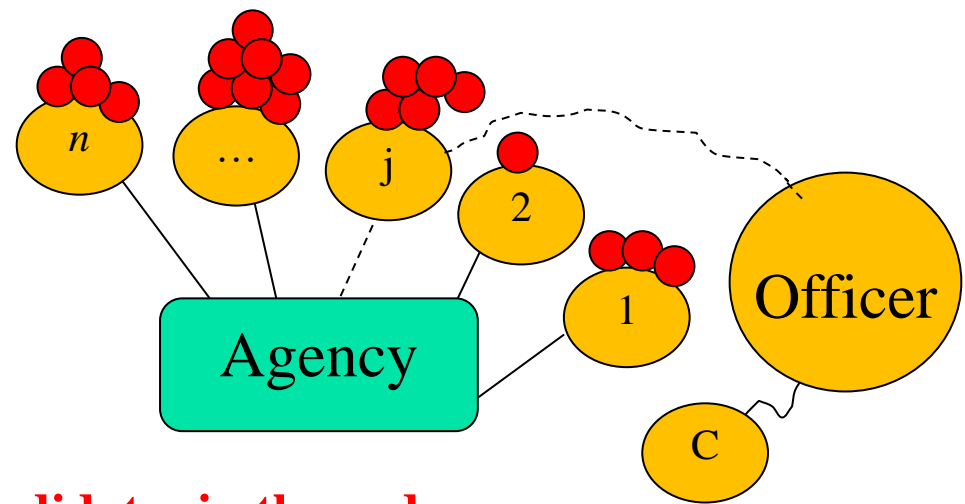


5.3.1 The hiring problem

For the hiring problem, the algorithm is deterministic.

- For any given input, the number of times we hire a new office assistant will always be the same. (给定输入, 则雇用的人数确定)
- The number of times we hire a new office assistant depends only on the input. (雇用的人数〔资源消费〕依赖于输入)
 - ◆ Some rank orderings will always produce a high hiring cost. Example: $\langle 1, 2, 3, 4, 5, 6 \rangle$, where each is hired.
 - ◆ Some will always produce a low hiring cost. Example: $\langle 6, *, *, *, *, * \rangle$, where only the first is hired.
 - ◆ Some may be in between.

5.3.1 The hiring problem



Instead of always interviewing the candidates in the order presented, what if we first randomly permuted this order.

- **The randomization is now in the algorithm, not in the input distribution.**
(随机化过程在算法中，而不是在输入中体现)
- **Given a particular input, we can no longer say what its hiring cost will be. Each time we run the algorithm, we can get a different hiring cost.**
(算法的运行时间与输入无关)
- **No particular input always elicits worst-case behavior.**
(算法的最坏运算时间不取决于特定的输入)
- **Bad behavior occurs only if we get “unlucky” numbers from the random-number generator.**
(只有当随机数产生器产生很不幸运的数时，算法的运算时间最坏)

5.3.1 The hiring problem

Algorithm for randomized hiring problem

RANDOMIZED-HIRE-ASSISTANT(n)

Randomly permute the list of candidates

HIRE-ASSISTANT(n)

□ *Lemma*

The expected hiring cost of **RANDOMIZED-HIRE-ASSISTANT** is $nc_i + (\ln n)c_h$.

Proof

After permuting the input array, we have a situation identical to the probabilistic analysis of deterministic **HIRE-ASSISTANT**.

(对输入矩阵进行随机置换后, 情况同**HIRE-ASSISTANT**相同)

5.3.2 Randomly permuting an array

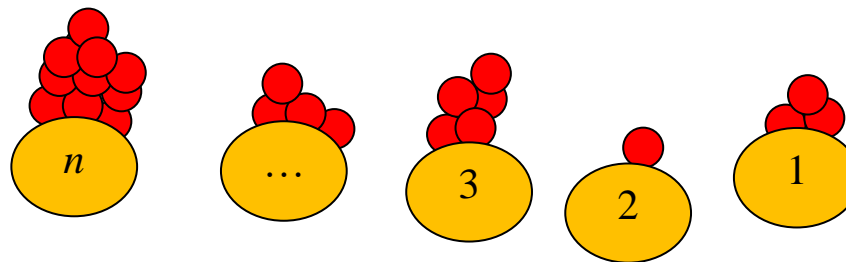
Goal:

Produce a uniform random permutation (each of the $n!$ permutations is equally likely to be produced),

that is

$$A = \langle 1, 2, 3, \dots, n \rangle$$

The numbers of permutation of A is $P_n^n = n!$, each of that is equally likely to be produced.



5.3.2 Randomly permuting an array

(1) Permute-by-sorting

The method is not very good

- Assume the given array A contains the element 1 through n .
(设数组 $A = \langle 1, 2, \dots, n \rangle$)
- Assign each element $A[i]$ a **random priority** $P[i]$, then sort the elements of A according to these priorities. Example
(为每个元素 $A[i]$ 分配一个随机数 $P[i]$ 作为其优先权, 然后依据这些优先权对数组 A 进行排序。)
 - ◆ If initial array is $A = \langle 1, 2, 3, 4 \rangle$, choose random priorities $P = \langle 36, 3, 97, 19 \rangle$, then produce an array $B = \langle 2, 4, 1, 3 \rangle$

```
PERMUTE-BY-SORTING(A)
n = length[A]
for(i=1; i<=n; i++)
    P[i] = RANDOM(1, n3)
sort A, using P as sort keys
return A
```

We use a range of 1 to n^3 in RANDOM to make it likely that all the priorities in P are unique.(Exercise 5.3-5)

5.3.2 Randomly permuting an array

(2) RANDOMIZE-IN-PLACE

The method is better

```
RANDOMIZE-IN-PLACE( $A, n$ )  
for( $i=1; i \leq n; i++$ )  
    swap( $A[i], A[\text{RANDOM}(i, n)]$ )
```

1	2	3	...	n
$A(1)$	$A(2)$	$A(3)$...	$A(n)$

1	2	3	...	i_1	...	n
$A(i_1)$	$A(2)$	$A(3)$...	$A(1)$...	$A(n)$

1	2	3	...	i_2	...	n
$A(i_1)$	$A(i_2)$	$A(3)$...	$A(2)$...	$A(n)$

Idea:

- In iteration i , choose $A[i]$ randomly from $A[i .. n]$.
- Will never alter $A[i]$ after iteration i . (第 i 次迭代后不再改变 $A[i]$)

Merit:

- It runs in linear time without requiring sorting ($O(n)$).
- It needs fewer random bits (n random numbers in the range 1 to n rather than the range 1 to n^3) (仅需更小范围的随机数产生器)
- No auxiliary array is required. (不需要辅助空间)

5.3.2 Randomly permuting an array

(2) RANDOMIZE-IN-PLACE

```
RANDOMIZE-IN-PLACE( $A, n$ )  
for( $i=1; i \leq n; i++$ )  
    swap( $A[i], A[\text{RANDOM}(i, n)]$ )
```

Correctness:

- Given a set of n elements, a k -permutation is a sequence containing k of the n elements. There are $n!/(n-k)!$ possible k -permutations ? (给定 n 个元素, 从其中任取 k 个元素进行排列, 则有 $n!/(n-k)!$ 种不同的 k -排列, 或 k -置换 ?)

□ *Lemma*

RANDOMIZE-IN-PLACE computes a uniform random permutation.

Proof Use a loop invariant:

1	2	3	...	n
$A(1)$	$A(2)$	$A(3)$...	$A(n)$

1	2	3	...	i_1	...	n
$A(i_1)$	$A(2)$	$A(3)$...	$A(1)$...	$A(n)$

1	2	3	...	i_2	...	n
$A(i_1)$	$A(i_2)$	$A(3)$...	$A(2)$...	$A(n)$

$$P_n^k = C_n^k \cdot P_k^k = \frac{n!}{k!(n-k)!} \cdot k! = \frac{n!}{(n-k)!}$$

5.3.2 Randomly permuting an array

(2) RANDOMIZE-IN-PLACE

```
RANDOMIZE-IN-PLACE(A, n)
for(i=1; i<=n; i++)
    swap(A[i], A[RANDOM(i, n)])
```

□ Lemma

RANDOMIZE-IN-PLACE computes a uniform random permutation.

Proof Use a loop invariant:

Loop invariant: Just prior to the i th iteration of the for loop, for each possible $(i-1)$ -permutation, subarray $A[1 .. i-1]$ contains this $(i-1)$ -permutation with probability $(n-i+1)!/n!$?

(第 i 次迭代之前, 对 $(i-1)$ -置换, 任意一个 $(i-1)$ -置换 $A[1 .. i-1]$ 的概率为 $(n-i+1)!/n!$?)

1	2	3	...	n
$A(1)$	$A(2)$	$A(3)$...	$A(n)$

1	2	3	...	i_1	...	n
$A(i_1)$	$A(2)$	$A(3)$...	$A(1)$...	$A(n)$

1	2	3	...	i_2	...	n
$A(i_1)$	$A(i_2)$	$A(3)$...	$A(2)$...	$A(n)$

$$1/P_n^k = 1 / \frac{n!}{(n-k)!} = \frac{(n-k)!}{n!}$$

5.3.2 Randomly permuting an array

(2) RANDOMIZE-IN-PLACE

```
RANDOMIZE-IN-PLACE(A, n)
for(i=1; i<=n; i++)
    swap(A[i], A[RANDOM(i, n)])
```

1	2	3	...	n
A(1)	A(2)	A(3)	...	A(n)

1	2	3	...	i_1	...	n
A(i_1)	A(2)	A(3)	...	A(1)	...	A(n)

1	2	3	...	i_2	...	n
A(i_1)	A(i_2)	A(3)	...	A(2)	...	A(n)

□ **Lemma** RANDOMIZE-IN-PLACE computes a uniform random permutation.

Proof Use a loop invariant:

Loop invariant: A[1 .. i-1] contains each (i-1)-permutation with probability $(n-i+1)!/n!$.

- **Initialization:** Just before first iteration, $i=1$. Loop invariant says for each possible 0-permutation, subarray A[1 .. 0] contains this 0-permutation with probability $n!/n!=1$. A[1 .. 0] is an empty subarray, and a 0-permutation has no elements. So, A[1 .. 0] contains any 0-permutation with probability 1.

(空集包含空置换的概率为1)

5.3.2 Randomly permuting an array

(2) RANDOMIZE-IN-PLACE

```
RANDOMIZE-IN-PLACE( $A, n$ )  
for( $i=1; i \leq n; i++$ )  
    swap( $A[i], A[\text{RANDOM}(i, n)]$ )
```

□ **Lemma** RANDOMIZE-IN-PLACE computes a uniform random permutation.

Proof **Loop invariant:** $\Pr\{A[1 \dots i-1] \text{ contains each } (i-1)\text{-permutation}\} = (n-i+1)!/n! .$

- **Maintenance:** Assume that prior to the i th iteration, $\Pr\{A[1 \dots i-1] \text{ contains each } (i-1)\text{-permutation}\} = (n-i+1)!/n! ,$ we will show that after the i th iteration, $\Pr(A[1 \dots i] \text{ contains each } i\text{-permutation}) = (n-i)!/n! .$

(第 i 次迭代前, 设 $(i-1)$ -置换 $A[1 \dots i-1]$ 中, 任一置换发生的概率为 $(n-i+1)!/n! ,$ 则需证明在第 i 次迭代后, 任一 i -置换 的概率为 $(n-i)!/n! .$)

Consider a particular i -permutation $R = \langle x_1, x_2, \dots, x_i \rangle$. It consists of an $(i-1)$ -permutation $R' = \langle x_1, x_2, \dots, x_{i-1} \rangle$, followed by x_i . (考虑一个特别的 i -置换 R , 其前 $i-1$ 个元素组成 $(i-1)$ -置换 R' , 最后一个元素为 x_i)

5.3.2 Randomly permuting an array

(2) RANDOMIZE-IN-PLACE

```
RANDOMIZE-IN-PLACE(A, n)
for(i=1; i<=n; i++)
    swap(A[i], A[RANDOM(i, n)])
```

□ **Lemma** RANDOMIZE-IN-PLACE computes a uniform random permutation.

Proof **Loop invariant:** $\Pr\{A[1 \dots i-1] \text{ contains each } (i-1)\text{-permutation}\} = (n-i+1)!/n! .$

● **Maintenance:**

i -permutation $R = \langle x_1, x_2, \dots, x_i \rangle = \langle x_1, x_2, \dots, x_{i-1} \rangle \cup x_i = R' \cup x_i .$

Let E_1 be the event that the algorithm actually puts R' into $A[1 \dots i-1] .$

By the loop invariant, $\Pr\{E_1\} = (n-i+1)!/n! .$

Let E_2 be the event that the i th iteration puts x_i into $A[i]$.

We get the i -Permutation R in $A[1 \dots i]$ if and only if both E_1 and E_2 occur \Rightarrow the probability that the algorithm produces R in $A[1 \dots i]$ is

$\Pr\{E_2 \cap E_1\} = ? . \dots$ (令事件 E_1 表示算法实际输出 $(i-1)$ -置换 R' 为 $A[1 \dots i-1]$ ，根据循环不变量， $\Pr\{E_1\} = (n-i+1)!/n!$ ，令事件 E_2 表示第 i 次迭代后输出 $A[i]$ 为 x_i ，则当且仅当 E_1 和 E_2 同时发生时我们得到 i -置换 R 为 $A[1 \dots i]$ ，其概率为 $\Pr\{E_2 \cap E_1\} = ?$) ...

5.3.2 Randomly permuting an array

(2) RANDOMIZE-IN-PLACE

```
RANDOMIZE-IN-PLACE(A, n)
for(i=1; i<=n; i++)
    swap(A[i], A[RANDOM(i, n)])
```

□ *Lemma* RANDOMIZE-IN-PLACE computes a uniform random permutation.

Proof **Loop invariant:** $\Pr\{A[1 \dots i-1] \text{ contains each } (i-1)\text{-permutation}\} = (n-i+1)!/n!$.

- **Maintenance:**

... ..

$$\Pr\{E_2 \cap E_1\} = \Pr\{E_2 | E_1\} \Pr\{E_1\}.$$

i	$i+1$...	n
$A(j_i)$	$A(j_{i+1})$...	$A(j_n)$

The algorithm choose x_i randomly from the $n-i+1$ possibilities in $A[i \dots n] \Rightarrow \Pr\{E_2 | E_1\} = 1/(n-i+1)$? Thus,

$$\begin{aligned} \Pr\{E_2 \cap E_1\} &= \Pr\{E_2 | E_1\} \Pr\{E_1\} \\ &= \frac{1}{n-i+1} \cdot \frac{(n-i+1)!}{n!} = \frac{(n-i)!}{n!} \end{aligned}$$

5.3.2 Randomly permuting an array

(2) RANDOMIZE-IN-PLACE

```
RANDOMIZE-IN-PLACE(A, n)  
for(i=1; i≤n; i++)  
    swap(A[i], A[RANDOM(i, n)])
```

□ *Lemma* RANDOMIZE-IN-PLACE computes a uniform random permutation.

Proof **Loop invariant:** $\Pr\{A[1 \dots i-1] \text{ contains each } (i-1)\text{-permutation}\} = (n-i+1)!/n! .$

- **Termination:**

At termination, $i=n+1$, it is true prior to the i th iteration, so we conclude that $A[i \dots n]$ is a given n -permutation with probability $(n-n)!/n! = 1/n! .$ ■ (lemma)

*5.4 Further applications

- 自学

Exercises and problems

- **Exercises**

所有课后习题。

编程验证习题。

- **Class-exercises**

雇用问题中，仅雇用1个人的概率，即 $\Pr(X=1)$?

雇用问题中，n个人都会被雇佣的概率，即 $\Pr(X=n)$?

雇用问题中，仅雇用2个人的概率，即 $\Pr(X=2)$?

（数学推导和编程验证相结合）