ADw3

Name : **Peter Garamvoelgyi** Student ID: **2018280070**

CLRS 5.2-3. Use indicator random variables to compute the expected value of the sum of n dice. (pg. 122)

Let X be a random variable representing the sum of n dice.
Let X_i be a random variable representing the value of the i'th dice.
As all n dice are identical, we can replace these n random variables with just one X_{single}

$$X = \sum_{i=1}^{n} X_n = n \cdot X_{single}$$

Let $X_{single,k}$ be an indicator random variable

$$X_{single,k} = I\{\text{the throw result is } k\} = \begin{cases} 1 & \text{if the result of the throw is } k \\ 0 & \text{otherwise} \end{cases}$$

The connection between these random variables is given by

$$X_{single} = \sum_{k=1}^{6} k \cdot X_{single,k}$$

Of course, exactly one of $X_{single,1}, \dots, X_{single,6}$ will take on the value 1.

We rely on the linearity of the expected value in our calculations:

$$E[X_{single}] = E\left[\sum_{k=1}^{6} k \cdot X_{single,k}\right] = \sum_{k=1}^{6} k \cdot E[X_{single,k}] = \frac{21}{6} = 3.5$$

And now we can simply calculate the final result:

$$E[X] = E[n \cdot X_{single}] = n \cdot E[X_{single}] = 3.5n$$

The expected value of the sum of n dice is: $\mathbf{3.5}\boldsymbol{n}$.

CLRS 5.2-5. Let $A[1..n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an inversion of A. Suppose that the elements of A form a uniform random permutation of $\langle 1, 2, \dots, n \rangle$. Use indicator random variables to compute the expected number of inversions. (pg. 122)

Let X be a random variable corresponding to the number of inversions.

Let us introduce the indicator random variable

$$X_{i,j} = \begin{cases} 1 & \text{if } A[i] > A[j] \\ 0 & \text{otherwise} \end{cases} \qquad i < j$$

Given this definition, we can formulate X as follows:

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{i,j}$$

The expected value of a single indicator random variable is

$$E[X_{i,j}] = P(X_{i,j} = 1) = \frac{(n-2)! \cdot \sum_{k=1}^{n-1} k}{n!} = \frac{1}{n(n-1)} \cdot \frac{n(n-1)}{2} = \frac{1}{2}$$

Explanation: Let us partition the possibilities for the two numbers at positions i and j based on $A[j]$. If $A[j] = 1$ then we have $n - 1$ options for $A[i]$. If $A[j] = 2$ then we have $n - 2$ options for $A[i]$. The sum of these is given by $\sum_{k=1}^{n-1} k$. The rest of the numbers can be in any order: $(n-2)!$.

Using the linearity of the expected value, we can get the final solution:

$$E[X] = E\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{i,j}\right] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{i,j}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} = \sum_{k=1}^{n-1} \frac{1}{2} = \frac{n(n-1)}{4}$$

The expected value of the number of inversions is $\frac{n(n-1)}{4}$.

CLRS 5-2. Searching an unsorted array (pg. 143)

a. Write pseudocode for a procedure RANDOM-SEARCH to implement the strategy above. Be sure that your algorithm terminates when all indices into A have been picked.

Input: A sequence of n numbers $A = \langle a_1, a_2, \dots, a_n \rangle$ and a value v
Output: An index i s.t. $v = A[i]$ or the special value NIL if v does not appear in A.

```
RANDOM-SEARCH(A, v):
visited := ⟨false, false, …, false⟩_n
num_visited := 0
loop forever
    i := RANDOM(1, n)
    if A[i] = v then
        return i
    if !visited[i] then
        visited[i] := true
        num_visited := num_visited + 1
    if num_visited = n then
        return NIL
end loop
```

b. Suppose that there is exactly one index i such that $A[i] = x$. What is the expected number of indices into A that we must pick before we find x and RANDOM-SEARCH terminates?

Let us introduce the indicator random variable

$$X_i = \begin{cases} 1 & \text{if we needed to pick } i \text{ indices} \\ 0 & \text{otherwise} \end{cases} \qquad p(X_i = 1) = \frac{1}{n} \cdot \left(\frac{n-1}{n}\right)^{i-1}$$

… characterized by a geometric distribution.

Let X be a random variable corresponding to the number of indices we must pick.

$$X = \sum_{i=1}^{\infty} i \cdot X_i$$

$$E[X] = E\left[\sum_{i=1}^{\infty} i \cdot X_i\right] = \sum_{i=1}^{\infty} i \cdot E[X_i] = \sum_{i=1}^{\infty} i \left(\frac{1}{n}\right)\left(\frac{n-1}{n}\right)^{i-1} = \frac{1}{1/n} = n$$

c. Generalizing your solution to part (b), suppose that there are $k \geq 1$ indices i such that $A[i] = x$. What is the expected number of indices into A that we must pick before we find x and RANDOM-SEARCH terminates? Your answer should be a function of n and k.

Let us introduce the indicator random variable

$$X_i = \begin{cases} 1 & \text{if we needed to pick } i \text{ indices} \\ 0 & \text{otherwise} \end{cases} \qquad p(X_i = 1) = \frac{k}{n} \cdot \left(\frac{n-k}{n}\right)^{i-1}$$

$$E[X] = E\left[\sum_{i=1}^{\infty} i \cdot X_i\right] = \sum_{i=1}^{\infty} i \cdot E[X_i] = \sum_{i=1}^{\infty} i\left(\frac{k}{n}\right)\left(\frac{n-k}{n}\right)^{k-1} = \frac{n}{k}$$

d. Suppose that there are no indices i such that $A[i] = x$. What is the expected number of indices into A that we must pick before we have checked all elements of A and RANDOM-SEARCH terminates?

The question is: How many indices do we have to draw until we checked all of them?

The probability of checking an unchecked entry once we have checked $i - 1$ is

$$p(n_i) = \frac{n-i+1}{n} \qquad E[n_i] = \frac{n}{n-i+1}$$

where n_i is a random variable with a geometric distribution.

The number of draws required until we have checked all entries is

$$X = \sum_{i=1}^{n} n_i$$

$$E[X] = E\left[\sum_{i=1}^{n} n_i\right] = \sum_{i=1}^{n} E[n_i] = \sum_{i=1}^{n} \frac{n}{n-i+1} = n\sum_{i=1}^{n} \frac{1}{i} = n\big(\ln b + O(1)\big)$$

(Following the derivation in CLRS 5.4.2)

e. Suppose that there is exactly one index i such that $A[i] = x$.What is the average-case running time of DETERMINISTIC-SEARCH? What is the worst-case running time of DETERMINISTIC-SEARCH?

```
DETERMINISTIC-SEARCH(A, v):
for i := 1 to A.length
    if A[i] = v then
        return i
return NIL
```

Let us introduce the indicator random variable

$$X_i = \begin{cases} 1 & \text{if } A[i] = v \\ 0 & \text{otherwise} \end{cases}$$

Let X be a random variable corresponding to the number of indices we must check.

$$X = \sum_{i=1}^{n} i \cdot X_i$$

We can estimate the average-case running time using the expected value:

$$E[X] = E\left[\sum_{i=1}^{n} i \cdot X_i\right] = \sum_{i=1}^{n} i \cdot E[X_i] = \sum_{i=1}^{n} i \cdot \frac{(n-1)!}{n!} = \frac{n(n+1)}{2} \cdot \frac{1}{n} = \frac{n+1}{2}$$

The worst-case running time is when the target element is the last in the array. In this case we need n comparisons.

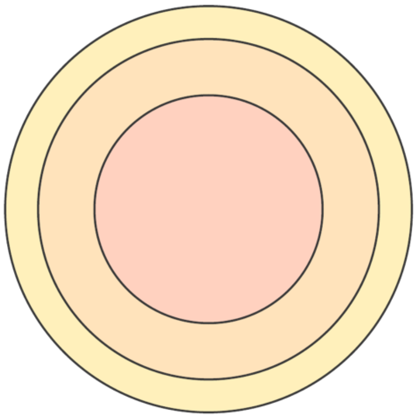
CLRS 8.4-4. We are given n points in the unit circle, $p_i = (x_i, y_i)$, such that $0 < x_i^2 + y_i^2 \leq 1$ for $i = 1, 2, \ldots, n$. Suppose that the points are uniformly distributed; that is, the probability of finding a point in any region of the circle is proportional to the area of that region. Design an algorithm with an average-case running time of $\Theta(n)$ to sort the n points by their distances $d_i = \sqrt{x_i^2 + y_i^2}$ from the origin. (*Hint*: Design the bucket sizes in BUCKET-SORT to reflect the uniform distribution of the points in the unit circle.) (pg. 204)

Let us partition the circle into k concentric rings of equal area. The area of the circle is $1^2\pi = \pi$. The area of the i'th ring is given by the formula

$$r_i^2\pi - r_{i-1}^2\pi = \frac{\pi}{k} \quad \rightarrow \quad r_i^2 = r_{i-1}^2 + \frac{1}{k}$$

… where r_i is the outer radius and r_{i-1} is the inner one.

$$\begin{aligned} & & r_0 &= 0 \\ r_1^2 &= 0 + \frac{1}{k} \quad \rightarrow & r_1 &= \sqrt{\frac{1}{k}} \\ r_2^2 &= \frac{1}{k} + \frac{1}{k} \quad \rightarrow & r_2 &= \sqrt{\frac{2}{k}} \\ & & &\vdots \\ & & r_k &= 1 \end{aligned}$$



Such a partitioning has multiple desirable properties:

1. As the points are uniformly distributed within the circle, the probability that a given point is in ring i is simply $1/k$.
2. The distance of points within ring i from the origin is larger than the distance of points within ring $i - 1$.

Let us use these rings as our buckets for BUCKET-SORT. Our algorithm becomes:

1. Iterate through all points and put each into the correct bucket
$$\text{ring}_i = \lfloor d_i^2 \cdot k \rfloor$$
2. Sort the points in each bucket based on their distances from the origin.
3. Reconstruct the sorted result by iterating through rings $1, 2, \ldots, k$.

If we choose $k \approx n$ then the average complexity of this algorithm is $\Theta(n)$.