5.1.1 ***Show that the assumption that we are always able to determine which candidate is best in line 4 of procedure $HIRE_{ASSISTANT}$, implies that you know a total order on the ranks of the candidates***

Let 2 candidates c_1 and c_2 Since we are always able to determine which candidate is best, means that given any c_1 and c_2 in Candidates set, we can compare them based on a relation R
This Implies that we know a total relation on the ranks of the candidates
Now we need to show that this relation is reflexive, transitive and antisymetric
It's clearly reflexive. We can relate c_1 to itself. the result is just choose it
The relation is transitive. If we choose c_1 over c_2 and we choose c_2 over c_3 we will choose c_1 over c_3
It' is also antisymetric, it was forced that each rank is unique. So c_1 R c_2 and c_2 R c_1 imply c_2 $= c_1$

5.1.2 ***Describe an implementation of the procedure $RANDOM(a, b)$ that makes calls only to $RANDOM(0, 1)$ What is the expected running time of your procedure as a function of a and b?***

$RANDOM(A, B)$ was defined as a procedure such that produces as output some integer in the range $[A, B]$ with probability of $\frac{1}{sizeof([a,b])}$

Lets transform A and B in binary numbers. We then associate A with Zero and B with B $- A$. Now we need at least $\lceil \lg(b-a) \rceil + 1$ bytes to represent it.
Do a for loop from 0 to $\lceil \lg(b-a) \rceil$ and call $RANDOM(0,1)$, multiply and sum on 2^i
i, represents the $i-th$ iteration. at the end of the iteration

```
RANDOM(A, B)
range = b − a
Result = 0
bits = ⌈log(b − a)⌉ + 1
for i = 0 to bits − 1:
    Result += 2^i · RANDOM(0,1)
if Result > range:
    return RANDOM(a, b)
return Result + a
```

This algorithm is not deterministic and there is a chance that never outputs anything

each call of $RANDOM(A, B)$ takes $\lceil \log(b-a) \rceil + 1$ to run. we go from 0 to $b - a$
And our algorithm can achieve $2^{\lceil \log(b-a) \rceil}$ numbers, each number has equal probability
So the probability that we leave a given $RANDOM(A, B)$ without calling another call is equal to $\frac{b-a+1}{2^{\lceil \log(b-a) \rceil + 1}}$ *and each one does $\lceil \log(b-a) \rceil + 1$ as job So*

$$\sum_{i=1}^{\infty} \left(1 - \frac{b-a+1}{2^{\lceil \log(b-a) \rceil + 1}}\right)^{i-1} \left(\frac{b-a+1}{2^{\lceil \log(b-a) \rceil + 1}}\right)^{1} i(\lceil \log(b-a) \rceil + 1). \text{ we choosed a to be } 0 \rightarrow$$

$$\sum_{i=1}^{\infty} \left(1 - \frac{b+1}{2^{\lceil \log(b) \rceil + 1}}\right)^{i-1} \left(\frac{b+1}{2^{\lceil \log(b) \rceil + 1}}\right)^{1} i(\lceil \log(b) \rceil + 1)$$

$$= (\mathbf{1.0}) \left(\frac{b+1}{2^{\lceil \log(b) \rceil + 1}}\right)^{1} (\lceil \log(b) \rceil + 1) \sum_{i=1}^{\infty} \left(1 - \frac{b+1}{2^{\lceil \log(b) \rceil + 1}}\right)^{i-1} i \rightarrow$$

$$S=\sum_{i=1}^{\infty} i\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)^{i-1} \rightarrow$$

$$\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)S+\sum_{i=1}^{\infty}\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)^{i}=$$

$$\sum_{i=1}^{\infty} i\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)^{i}+\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)^{i}=\sum_{i=1}^{\infty}(i+1)\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)^{i}$$

$$\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)S+\sum_{i=1}^{\infty}\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)^{i}-S=\lim_{j\to\infty}\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)^{j}(j+1)-1$$

$$(\mathbf{2.0})\ S\frac{b+1}{2^{\lceil \log(b)\rceil+1}}=1+\sum_{i=1}^{\infty}\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)^{i}$$

$$R=\sum_{i=1}^{\infty}\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)^{i}\rightarrow\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)R=\sum_{i=1}^{\infty}\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)^{i+1}\rightarrow$$

$$\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)R-R=\sum_{i=1}^{\infty}\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)^{i+1}-\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)^{i}\rightarrow$$

$$-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}R=\lim_{j\to\infty}\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)^{j}-\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)=-\left(1-\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)$$

$$\frac{b+1}{2^{\lceil \log(b)\rceil+1}}R+\frac{b+1}{2^{\lceil \log(b)\rceil+1}}=1\rightarrow(R+1)=\frac{1}{\frac{b+1}{2^{\lceil \log(b)\rceil+1}}}=\frac{2^{\lceil \log(b)\rceil+1}}{b+1}\rightarrow\boxed{R=\frac{2^{\lceil \log(b)\rceil+1}}{b+1}-1}$$

Coming back to 2.0 *we have*:

$$S\frac{b+1}{2^{\lceil \log(b)\rceil+1}}=1+\frac{2^{\lceil \log(b)\rceil+1}}{b+1}-1\rightarrow S=\left(\frac{2^{\lceil \log(b)\rceil+1}}{b+1}\right)^{2}$$

coming back to 1.0 *we have*

$$\left(\frac{b+1}{2^{\lceil \log(b)\rceil+1}}\right)^{1}(\lceil \log(b)\rceil+1)\left(\left(\frac{2^{\lceil \log(b)\rceil+1}}{b+1}\right)^{2}\right)=$$

$$(\lceil \log(b)\rceil+1)\left(\left(\frac{2^{\lceil \log(b)\rceil+1}}{b+1}\right)^{1}\right)=\frac{(\lceil \log(b)\rceil+1)\left(2^{\lceil \log(b)\rceil+1}\right)}{b}=O(\lceil \log(b-a)\rceil)$$

5.1.3 ***You wish to implement a program that outputs*** **0** ***with probability*** $\frac{1}{2}$ ***and*** **1** ***with probability*** $\frac{1}{2}$. ***At your disposal is a procedure Biased Random that outputs either*** **0** ***or*** **1** ***with some probability*** p ***and*** **0** ***with probability*** $1-p$. ***You don't know what*** p ***is.*** ***Give an algorithm that used Biased*** $-$ ***Random as subroutine and return an an unbiased answer, returning*** **0** ***with probability*** $\frac{1}{2}$ ***and*** **1** ***with probability*** $\frac{1}{2}$. ***What is the expected running time of your algorithm as a function of*** p ?

We need to map outcomes of Biased Random with same probability
If we call the procedure two times and the return are 1 *and* 0 *or* 0 *and* 1
This happens with same probability, so it turns out we just need to map the output to return 1 *or* 0. *If the output is* 00 *or* 11. *Run the biased again. Our pseudocode would be*:

```
RANDOM( ):
while true do
    a = Biased − Random
    b = Biased − Random
    if a == b:
        continue
    return a
```

Our algorithm will output 1 *or* 0 *with same probability, but again, is not deterministic the case* $a == b$ *happens with probability* $p^2 + (1-p)^2$. *if this happens we have to iterate again so* $\sum_{i=1}^{\infty}\big(2p(1-p)\big)\cdot(p^2+(1-p)^2)^{i-1}i\big(\Theta(1)\big) \rightarrow$

$$\big(2p(1-p)\big)\Theta(1)\sum_{i=1}^{\infty}(p^2+(1-p)^2)^{i-1}i \rightarrow S = \sum_{i=1}^{\infty}(p^2+(1-p)^2)^{i-1}i$$

$$S(p^2+(1-p)^2) + \sum_{i=1}^{\infty}(p^2+(1-p)^2)^i - S$$

$$= \sum_{i=1}^{\infty}(p^2+(1-p)^2)^i i + (p^2+(1-p)^2)^i - (p^2+(1-p)^2)^{i-1}i$$

$$S(p^2+(1-p)^2) + \sum_{i=1}^{\infty}(p^2+(1-p)^2) - S$$

$$= \sum_{i=1}^{\infty}(i+1)(p^2+(1-p)^2)^i - (p^2+(1-p)^2)^{i-1}i$$

$$S(p^2+(1-p)^2) + \sum_{i=1}^{\infty}(p^2+(1-p)^2)^i \ - S = \lim_{n\to\infty}(n+1)(p^2+(1-p)^2)^n - 1 = -1$$

$$S(p^2+(1-p)^2-1) + \sum_{i=1}^{\infty}(p^2+(1-p)^2)^i = -1 \rightarrow$$

$$Z = \sum_{i=1}^{\infty}(p^2+(1-p)^2)^i \rightarrow$$

$$Z(p^2+(1-p)^2)-Z=\sum_{i=1}^{\infty}(p^2+(1-p)^2)^{i+1}-(p^2+(1-p)^2)^i$$
$$=\lim_{n\to\infty}(p^2+(1-p)^2)^k-1\to$$
$$Z(p^2+(1-p)^2)-Z=-1\to Z(p^2+(1-p)^2-1)=-(p^2+(1-p)^2)\to$$
$$S(p^2+(1-p)^2-1)^2+(p^2+(1-p)^2-1)^1\sum_{i=1}^{\infty}(p^2+(1-p)^2)^i$$
$$=-1(p^2+(1-p)^2-1)^2\to$$
$$S=\frac{(p^2+(1-p)^2)-1(p^2+(1-p)^2-1)^1}{(p^2+(1-p)^2-1)^2}$$
$$\big(2p(1-p)\big)\Theta(1)\sum_{i=1}^{\infty}(p^2+(1-p)^2)^{i-1}i=$$
$$\big(2p(1-p)\big)\Theta(1)\frac{-1(p^2+(1-p)^2-1)^1+(p^2+(1-p)^2-1)^2}{(p^2+(1-p)^2-1)^2}=$$
$$\frac{-(2)\Theta(1)}{4p^1(p-1)^1}=\Theta\left(\frac{1}{2p(1-p)}\right)$$

5.2.1 ***In Hire Assistant, assuming that the candidates are presented in a random order what is the probability that you hire exactly one time? What is theprobability that you hire exactly n times?***

You will always hire the best candidate over the n sent to you. So you want that this is the only Candidate you will ever hire. After his hire now one will be hired it all, so if the best candidate is the first one you will only hire the first candidate. The thing is, you will always hire the first Candidate since you initialize Hire Candidate with the worst already set, so the firt candidate and the best will always be hired, so the only case you hire just one is when the first is the best This happens with probability of $\frac{1}{n}$

To hire n times you have to hire everyone. Since there is a total order. we only hire a candidate n_k *if he as better than* n_{k-1} *and we will only hire* n_{k+1} *if he is better than* n_k*. There is only an order that this occurs, so the probabiliy is* $\frac{1}{n!}$*. Why there is only one order?*
Suppose there is another order, so exists a candidate c_1 *who was hired first than* c_2 *in one order and after* c_2 *in the second order. but if that the case* c_2*is better and worse than* c_1 *What is impossible*

5.2.2 ***In Hire Assistant, assuming that the candidates are presented in a random order, what is the probability that you hire exaclty twice?***

You will always hire the first and the best. so they might not be the same. Supose that you hire Someone that is worse than $k(k \geq 2)$ *or more people, then at least one is the best candidate and will be hired, at least other one will also be hired,*

So if you hire a candidate of rank K as your first candidate. Then for you only hire two times you need that the best candidate comes before than the $n-k$ *better candidates than* k*.*

This happens with a probability of $\frac{1}{n-k}\frac{1}{n}$ *So we have*:

$$\frac{1}{n}\sum_{k=1}^{n-1}\frac{1}{n-k}=\frac{1}{n}\sum_{k=1}^{n-1}\frac{1}{k}$$

5.2.3 ***Use indicator random variable to compute the expected value of the sum of n dice***:

Expectation of a single dice is $\frac{1+2+3+4+5+6}{6}=3.5$

Now let J_k *be a random variable that associates the outcome of a* $k-$*rolled dice to its value*

$$let\ X=\sum_k J_k \rightarrow E[X]=E\left[\sum_k J_k\right]=\sum_k [J_k]=3.5n$$

5.2.4 ***This exercise asks you to partly verify that linearity of expectation holds even if the random variables are not independet. Consider two*** $\mathbf{6-}$ ***sided dice that are rolled indepentdly What is the expected value of the sum*****? *Now consider the case where the first die is rolled normally and then the second die is set equal to the value shown on the first die. What is the expected value of the sum*****? *Now consider the case where the first die is roleld and the second is set equal to*** **& *minus the value of the first die. What is the expected value of the sum*****?**

Let X_1 *be a random variable that assocaites the outcome of rolling the first dice to the value the dice gets after rolled and let* X_2 *be the same, but for the second dice*
Now let X be the random variable that associates the outcome of the two dices, to the sum of the values, we then have $X=X_1+X_2 \rightarrow E[X]=E[X_1]+E[X_2]=7$

Let X_1 *be a random variable that assocaites the outcome of rolling the first dice to the value the dice gets after rolled and* X_2 *that associates the outcome of the first event to the value itself*
We then get that $X_2=X_1 \rightarrow X=2X_1=2\cdot 3.5=7$

Let X_1 *be a random variable that assocaites the outcome of rolling the first dice to the value the dice gets after rolled and* X_2 *be associated to* $7-$ *value of outcome of* X_1
We then have $X_2=7-X_1 \rightarrow E[X]=7$

5.2.5 ***Use indicator random variables to solve the following problem, which is known as the hat*** $\mathbf{-}$ ***check*** $\mathbf{-}$ ***problem. Each of n customers gives a hat to a hat*** $\mathbf{-}$***check person at a restaurant. The hat check person gives the hats back to the customers in a random order. What is the expected number of customers who get back their own hat*****?**

We can analyze this problem by enumerating each one of the customers to its own hat
In this case during the exist we will have an exit order and without loosing generality
We can associate the numbers with its position in a list. So if we have the following:
1 3 2 4 7 8 9 5 6 *then we will say that only first and fourth customers got its own hat*

Even with dependence we can say that let X_i be a random variable defined as below:
$\begin{cases} 1 \text{ if person } i \text{ gets its right hat} \\ 0 \text{ if person } i \text{ gets its wrong hat} \end{cases}$

In this case, we can define $X = \sum_{i=1}^{n} X_i$ *and the expected Value* $= 1$

5.2.6 ***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 $(1, 2, \ldots, n)$. Use indicator random variables to compute the expected number of inversion***

Let X_{ij} be a random variable that is defined as:
$\begin{cases} 1 \text{ if happened a inversion between } i \text{ and } j \\ 0 \text{ if there is no inversion between } i \text{ and } j \end{cases}$

We will only analyze the cases where $j > i.$, $i = n$ makes no sense if we count
we are adding a already counted inversion, since it happened with a before index
So we have:
$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_i j$ *. By the property of linearity we have that* $E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]$

Now what is the expected value of X_{ij}*? Its equal to the probability of happening such an inversion*

By symnetry $E[X_{ij}] = \frac{1}{2} \rightarrow E[X] = \frac{1}{2} \sum_{i=1}^{n-1} n - i = \frac{n(n-1)}{4}$

5.3.1 ***Professor Marceau objects to the loop invariant used in the proof of Lemma 5.4. He questions whether it holds prior to the first iteration. He reasons that we could just as easily declare that an empty subarray contains no $0-$permutations. Therefore, the probability that an empty subarray contains a 0 permutation should be 0, thus invalidating the loop invariant prior to the first iteration. Rewrite the procedure $RANDOMLY-PERMUTE$ so that its associated loop invariant applies to a nonempty subarray prior to the first iteration, and modify the proof of Lemma 5.4 for your procedure***

We just change the iteration from $i = 1$ to $i = 2$ and perform the first swap
before entering the loop
We just need to change the initialization for $i = 2$

5.3.2 ***Professor Kelp decides to write a procedure that produces at random any permutation except the identity permutation, in which every element ends up where it started. He proposes the procedure PERMUTE − WITHOUT − IDENTITY. Does this procedure do what Professor Kelp intends?***

```
PERMUTE-WITHOUT-IDENTITY(A, n)
1  for i = 1 to n − 1
2      swap A[i] with A[RANDOM(i + 1, n)]
```

No, there are others permutations that the algorithm will not produce. Suppose we have a array of 3
Let the array be 123. $n = 3$ *so our iteration goes from* 1 *to* 2
In the first we swap 123 *with* 2, *or* 3 *so we can get*:
213 *or* 321 *after that we swap the second number with third and the algorthm ends*:
So we finish with 231 *or* 312 *and there is no* 132 .

5.3.3 ***Consider the PERMUTE − WITH − ALL procedure on the facing page, which instead of swapping element A[i] with a random element from the subarray A[i: n], swaps it with a random element from anywhere in the array. Does PERMUTE − WITH − ALL produce a uniform random permutation? Why or why not?***

```
PERMUTE-WITH-ALL(A, n)
1  for i = 1 to n
2      swap A[i] with A[RANDOM(1, n)]
```

Lets analyze again a case with just 3 *numbers in the array, say it* 1,2,3
123 *generate the outcomes* → 213 *or* 321 → *which generates* 123, 231, 312 *and* 231 →
In the end we have 321, 132, 132, 213, 213, 321, 132, 213.
This is clearly not uniform

5.3.4 ***Professor Knievel suggests the procedure PERMUTE − BY − CYCLE to generate a uniform random permutation. Show that each element A[i] has a 1/n probability of winding up in any particular position in B. Then show that Professor Knievel is mistaken by showing that the resulting permutation is not uniformly random***

```
PERMUTE-BY-CYCLE(A, n)
1  let B[1 : n] be a new array
2  offset = RANDOM(1, n)
3  for i = 1 to n
4      dest = i + offset
5      if dest > n
6          dest = dest − n
7      B[dest] = A[i]
8  return B
```

Ok, lets take again a size 3 *array.* 1,2,3. *Offset can be* 1, 2 *or* 3
lets go trhough the iteration

$Case\ i = 1$	$Case\ i = 2$	$Case\ i = 3$
$dest\ can\ be\ 2,3,4$	$dest\ can\ be\ 3,4,5$	$dest\ can\ be\ 4,5,6$
$2 \rightarrow B[2] = A[1]$	$3 \rightarrow B[3] = A[2]$	$4 \rightarrow B[1] = A[3]$
$3 \rightarrow B[3] = A[1]$	$4 \rightarrow B[1] = A[2]$	$5 \rightarrow B[2] = A[3]$
$4 \rightarrow B[1] = A[1]$	$5 \rightarrow B[2] = A[2]$	$6 \rightarrow B[3] = A[3]$

$So\ we\ end\ up\ with\ the\ following$
$if\ offset = 1 \rightarrow B[2] = A[1], B[3] = A[2]\ e\ B[1] = A[3] \rightarrow 3,1,2$
$if\ offset\ is\ 2 \rightarrow B[3] = A[1], B[1] = A[2]\ e\ B[2] = A[3] \rightarrow 2,3,1$
$if\ offset\ is\ 3 \rightarrow B[1] = 1, B[2] = 2. B[3] = 3 \rightarrow 123$

$We\ have\ only\ 3\ outputs, each\ one\ in\ each\ position, what\ looks\ like\ it\ will\ have\ probability\ of\ \frac{1}{n}$
$But\ this\ is\ obviously\ not\ a\ uniform\ permutation$
$Now.\ lets\ prove\ the\ probability.\ Since\ the\ offset\ is\ chosen, the\ whole\ permutation\ is\ already$
$Decided.\ Since\ we\ have\ n\ possibilities\ of\ offset, each\ number\ can\ be\ in\ a\ position\ with\ probability$
$of\ \frac{1}{n}$

5.3.5 ***Professor Gallup wants to create a random sample of the set*** $\{1, 2, 3, \ldots . n\}$ ***that is, an*** $m -$ ***element subset*** S***, where*** $0 \leq m \leq n$***, such that each*** $m -$ ***subset is equally likely to be created. One way is to set*** $A[i] = i$ ***for*** $i = 1, 2, 3 \ldots n$***, call*** $RANDOMLY - PERMUTE(A)$ ***and then take just the first*** m ***array elements. This method makes*** n ***calls to the RANDOM procedure.***
In Professor Gallup's application, n ***is much larger than*** m***, and so the professor wants to create a random sample with fewer calls to RANDOM***

```
RANDOM-SAMPLE(m, n)
1  S = ∅
2  for k = n − m + 1 to n          // iterates m times
3      i = RANDOM(1, k)
4      if i ∈ S
5          S = S ∪ {k}
6      else S = S ∪ {i}
7  return S
```

Show that the procedure $RANDOM - SAMPLE$ ***on the previous page returns a random*** $m -$ ***subset*** S ***of*** $\{1, 2, 3 \ldots n\}$ ***in which each*** $m -$ ***subset is equally likely, while making only*** m ***calls to Random***

$Lets\ use\ an\ inductive\ proof, we\ need\ to\ show\ that\ each\ value\ that\ appears\ in\ S\ have\ a\ probability\ of$
$\frac{m}{n}. The\ hypothesis\ is\ that\ after\ an\ iteration\ the\ set\ S\ contain\ k - (n - m) elements$
$and\ the\ probability\ of\ each\ one\ is\ \frac{|S|}{k}. For\ the\ base\ case, first\ iterationi\ can\ be\ anything\ with\ equal$
$probability\ and\ clearly\ is\ added\ to\ S. Now\ lets\ S^{*} be\ the\ set\ after\ iteration\ k - 1.$
$We\ then\ suppose\ that\ |S^{*}| = |S| - 1\ with\ each\ value\ having\ probability\ og\ \left|\frac{S^{*}}{k-1}\right|$
$for\ the\ values\ betwwen\ 1 \ldots k - 1. Now\ we\ need\ to\ prove\ it\ for\ k.$
$the\ probability\ that\ an\ X\ be\ in\ S = the\ probability\ of\ t\ being\ in\ S' + probability\ of$
$not\ being\ in\ S^{*} and\ Random(1, k) returns\ t. They\ are\ disjoint\ events\ so:$

$$Probability\ of\ t \in S = \frac{|S^{*}|}{k-1} + \frac{1}{k} \cdot \left(1 - \frac{|S^{*}|}{k-1}\right) = \frac{|S|}{k}$$

5.4.1 ***How many people must there be in a room before the probability that someone has the same birthday as you do is at least*** $\frac{1}{2}$? ***How many people must there be before the probability that at least two people have a birthday on July 4 is greater than*** $\frac{1}{2}$?

Each person is likely to was born in any day of the year, so the probability of a person share the same birthday as me is equal to $\frac{1}{365}$. *Also we know that* $1 -$ *the probability of everyone not having the same burthday as me is equivalent to what we want to find the probability of someone not having the same probability as me is* $\frac{364}{365}$.

$\left(\frac{364}{365}\right)^n$ *is the probability of n people not having the same birthday as me.*

$1 - \left(\frac{364}{365}\right)^n = \frac{1}{2} \rightarrow \left(\frac{364}{365}\right)^n = \frac{1}{2} \rightarrow$ *this leads to* 253 *persons. Since you are in the room* 254.

The probability of at least two people having birthday on july 4 *is equivalent to* $1 -$ *probability of just* 1 *or less*

The probability of no one having 4*th july as birthday is* $\left(\frac{364}{365}\right)^n$

The probability of just one having this birthday is $\frac{n}{365}\left(\frac{364}{365}\right)^{n-1}$ *so we have*

$1 - \left(\frac{364}{365}\right)^n - \frac{n}{365}\left(\frac{364}{365}\right)^{n-1} = \frac{1}{2} \rightarrow n \geq 613$

5.4.2 ***How many people must there be in a room before the probability that two people have the same birthday is at least*** **0.99**? ***For that many people What is the expected number of pair of people who have the same birthday***?

This is the same as asking how many people should have a in a room before the probability that no one shares the same birthday is at maximun 0.01?

In this case we need $\frac{364}{365} \cdot \frac{363}{365} \ldots \frac{366 - n}{365} \rightarrow \frac{365!}{365^n(365 - n)!} = 0.01$

With 57 *people there is atleast* 0.99 *of probability that there is* 2 *people or more sharing birthday*

5.4.3 ***You toss ball into b bins until some bin contain two balls. Each toss is independent and each ball is likely to end up in any bin. What is the expected number of ball tosses***?

The probability of a ball falling in a bin is $\frac{1}{b}$ *and the maximun value of balls is b*

We can calculate the expected number of ball tosses directly, so we have:

$\sum_{i=2}^{B+1} iP(X = i)$. *Now what is* $P(X = i)$?

Lets analyze the case for 2 *balls, the second one must fall in the same as the first*

So we have $\frac{1}{b}$. *In the case of three balls, the second must not fall in the same as the first*

and the third must fall in the same as the first or second so we have: $\frac{b-1}{b}\cdot\frac{2}{b}$

For the 4 iteration we have the same logic, so $\frac{(b-1)}{b}\cdot\frac{b-2}{b}\cdot\frac{3}{b}$ *we can guess that the probability of* $P(X=i)$*must be :*

$\frac{(i-1)(b-1)!}{b^{i-1}(b-i+1)!}\rightarrow$ *we then endup with*

$\sum_{i=2}^{B+1} i\frac{(i-1)(b-1)!}{b^{i-1}(b-i+1)!}$, *in which for* $B=365$ *we have* 24.6 *as the expected number*

5.4.4 ***For the analysis of the birthday paradox, is it important that the birthdays be mutually independent, or is pairwise independence sufficient*?**

pairwise independence is sufficient

5.4.5 ***How many people should be invited to a party in order to make it likely that there are three people with the same birthday*?**

After some Research I discovered that it's not an easy solution with just approximated Answer found, in this way I decided to not try this question since it'snot the scope of this book Below there are some links about it:
https://math.stackexchange.com/questions/25876/probability-of-3-people-in-a-room-of-30-having-the-same-birthday

https://math.stackexchange.com/questions/485462/birthday-problem-for-3-people

---

5.4.6 ***What is the probability that a k − string over a set of size n forms a k −permutation? How does this question relate to the birthday paradox*?**

The first one can be anything, the second must be only $n-1$*, the third must be only* $n-2$*. the k must be only* $n-(k-1)\rightarrow$

$\frac{n-1}{n}\cdot\frac{n-2}{n}\ldots\frac{n-k+1}{n}=\prod_{i=1}^{k-1}1-\frac{i}{n}$*. This is equivalent of a year having n days*

Selecting k people and calculating the probability of no one of them sharing the same birthday

---

5.4.7 ***you toss n balls into n bins, where each toss is independent and the ball is equally likely to end up in any bin. What is the expected number of empty bins? What is the ex pected number of bins with exactly one ball*?**

Lets define the X be a random variable which calculates the number of empty bins
Let define Y_i *be* 1 *if bin i is empty and* 0 *else. then we have*:

$X=\sum_{i=1}^{n}Y_i\rightarrow$ *by the linearity* $E[x]=\sum_{i=1}^{n}E[Y_i]=n\left(\frac{n-1}{n}\right)^n$

Lets use the same logic let Y_i be a random variable with value 1 if bin i contains only one ball

We then have $E[X] = \sum_{i=1}^{n} \Pr obability\ of\ bin\ i\ contain\ only\ one\ ball$.

Suppose that we put one ball at the first play. so the probability in this case of having only one is: $\frac{1}{n}\left(\frac{n-1}{n}\right)^{n-1}$, *in the second play we have the same, for all possibly plays the probability turns out to be* $\left(\frac{(n-1)}{n}\right)^{n-1}$ *then* $E[x] = n\left(\frac{n-1}{n}\right)^{n-1}$

5.4.8 *Sharpen the lower bound on streak length by showing that in n flips of a fair coin, the probability is at least* $1 - \frac{1}{n}$ *that a streak of length* $lgn - 2\lg(lgn)$ *consecutive heads occurs*

Lets assume that n is a 2^{2^k} *for some integer k.*
In this situation we want a contiguous only heads block

The possibilities are of the block beginning in the first flip until the $lgn - 2\lg(lgn)$

Lets divide our n flips into disjoint blocks of size $\lg(n) - 2\lg(\lg(n))$
Thus we have a number of blocks like this $= \frac{n}{\lg(n) - 2\lg(\lg(n))}$.
The probability of one of those blocks are independent and the probabiity that one of them contains all heads is $\left(\frac{1}{2}\right)^{\lg(n)-2\lg(\lg(n))} = \frac{\lg^2 n}{n}$. *What is the probability that no one of these blocks have the all heads property?* $\left(1 - \frac{\lg^2 n}{n}\right)^{\frac{n}{\lg(n)-2\lg(\lg(n))}}$, *we know that if* $n > 4$ *then this is lower than* $\frac{1}{n}$. *So the probability of no one happening is* $< \frac{1}{n}$
$\rightarrow$ *so* $1 - \frac{1}{n} < 1 - probability\ of\ no\ one\ happening = prob\ of\ atleast\ one\ happening$

5.1 With a $b-$bit counter, we can ordinarily only count up to $2^b - 1$.
**With R. Morris's probabilistic counting, we can count up to a much larger value at the expense of some loss of precision. We let a counter value of i represent a count of n_i *for* $i = 0, .. 2^b - 1$, *where the* n_i *form an increasing sequence of nonnegative values* We assume that the initial value of the counter is 0, representing a count of $n_0 = 0$ The INCREMENT operation works on a counter containing the value i in a probabilistic manner. *If* $i = 2^b - 1$, *then the operation reports an overflow error. Otherwise, the INCREMENT operation increases the counter by* 1 *with probability of* $1 - \frac{1}{n_{i+1} - n_i}$. *If we select* $n_i = i$ *for all* $i \geq 0$ *then the counter is an ordinary one. More interesting situation arise if we select , say,* $n_i = 2^{i-1}$ *for* $i > 0$ *or* $n_i = F_i$(*the* $i-th$ *Fibonacci number*).

For this problem assum ethat $n_{2^b-1}$is large enough that the probability of an overflow error is negligible

a) Show that the expected value represented by the counter after n INCREMENT operation have been performed is exactly n

b)The analysis of the variance of the count represented by the counter depends on the sequence of the n_i. Let us consider a simple case: $n_i = 100i$ for all $i \geq 0$. Estimate the variance in the value represented by the register after n INCREMENT operations have been performed

Let X_i represent the increase or not in the value of the counter in the $i-th$ operation

so Lets determine V_n the value of the counter after n operation of INCREMENT

$$V_n = \sum_{i=1}^{n} X_i \rightarrow E[V_n] = \sum_{i=1}^{n} E[X_i]$$

We can see that if $E[X_i] = 1$ always, then we will prove that the expected value is n if the counter does not increase then the expected value is 0, otherwise is $n_{i+1} - n_i$, and the probability is $\frac{1}{n_{i+1} - n_i}$, so the expected value is 1

Now we need to compute the variance. we know that $Var[X_j] = E[X_j^2] - E^2[X_j]$

$Var[X_j] = E[X_j^2] - 1 \rightarrow n_{i+1} - n_i = 100$, with a probability of $\frac{1}{100}$

So its $100 - 1 = 99$ for X_j, there are n of them, so $Var[V_n] = 99n$

5.2 *Searching an unsorted array*.

This problem examines three algorithms for searching for a value x in an unsorted array A consisting of n elements Consider the following randomized strategy: pick a random index i into A. If A[i] = x, then terminate; otherwise, continue the search by picking a new random index into A. Continue picking random indices into A until you find an index j such that A[j] = x or until every element of A has been checked. This strategy may examine a given element more than once, because it picks from the whole set of indices each time.

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

```
RANDOM − SEARCH( ):
initialize an array from 0 to n − 1 with 0 as values
visited = 0
while visited < n:
i = RANDOM
if A[i] == x
    return i
```

$elseif\ !array[i]$:
    $array[i] = 1$
    $visited++$
$return\ NULL$

b) Suppose that there is exactly one index i such that $A[i] = x$.
What is the expected number of indices into A that must be picked before x is found and Random Search terminates?

$$\sum_{i=1}^{\infty} i \cdot \left(\frac{n-1}{n}\right)^{i-1}\left(\frac{1}{n}\right) = \frac{S}{n}\ and\ \ S = \sum_{i=1}^{\infty} i \cdot \left(\frac{n-1}{n}\right)^{i-1} \rightarrow$$

$$\frac{n-1}{n}S - S + \sum_{i=1}^{\infty}\left(\frac{n-1}{n}\right)^{i} = \sum_{i=1}^{\infty}(i+1)\cdot\left(\frac{n-1}{n}\right)^{i} - (i)\cdot\left(\frac{n-1}{n}\right)^{i-1}$$

$$= \lim_{k\to\infty}(k+1)\left(\frac{n-1}{n}\right)^{k} - 1 = -1$$

$$\frac{n-1}{n}S - S + \sum_{i=1}^{\infty}\left(\frac{n-1}{n}\right)^{i} = -1 \rightarrow \frac{S}{n} = 1 + \sum_{i=1}^{\infty}\left(\frac{n-1}{n}\right)^{i}$$

$$R = \sum_{i=1}^{\infty}\left(\frac{n-1}{n}\right)^{i} \rightarrow \frac{n-1}{n}R - R = \sum_{i=1}^{\infty}\left(\frac{n-1}{n}\right)^{i+1} - \left(\frac{n-1}{n}\right)^{i} \rightarrow -\frac{R}{n}$$

$$= \lim_{k\to\infty}\left(\frac{n-1}{n}\right)^{k+1} - \left(\frac{n-1}{n}\right)^{1} \rightarrow R = n-1$$

$\frac{S}{n} = 1 + n - 1 = n.$ *The expected value is 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 must be picked before x is found and $RANDOM - SEARCH$ terminates?
your answer should be a function of n and k

Simillarly we have
$\sum_{i=1}^{\infty} i \cdot \left(\frac{n-k}{n}\right)^{i-1}\left(\frac{k}{n}\right) \rightarrow$ *Applying the same logic above we get* $\rightarrow \frac{n}{k}$

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

Lets X_i denote a random variable whose value is the number of calls for Random − Search until i different position in the array be acessed at least one time. this value is setted to 0 after the acess of a new array position

In this case we have number of calls $= \sum_{i=1}^{n} X_i \rightarrow$

$$E[n^{\circ}\ indices] = \sum_{i=1}^{n} E[X_i] \rightarrow \sum_{i=1}^{n}\sum_{j=1}^{\infty} j\left(\frac{i-1}{n}\right)^{j-1}\frac{n-(i-1)}{n} = \sum_{i=1}^{n}\frac{n}{(1+n-i)} = n\sum_{i=1}^{n}\frac{1}{i} \rightarrow$$

Now consider a deterministic lienar search algorithm. The algorithm, which we call DETERMINISTIC − SEARCH, searches A for x in order until it finds A[i] = x or it reaches the end of the array. Assume tat all possible permutations of the input array are equally likely

e)Suppose that there is exaclty 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?

The probability of x being on the i − th index is $\frac{1}{n}$*, so we have:*
$\sum_{i=1}^{n} \frac{i}{n} \to \frac{(1+n)\cdot n}{n\cdot 2} = \frac{n+1}{2}$*. In the worst case its in the end with n, both* $\Theta(n)$

f)Generalizing your solution to part e, suppose that there are k ≥ 1 indices i such that A[i] = x. What is the average − case running time of Deterministic − Search? What is the worst − case running time of Determinstc − Search? Your answer should be a function of n and k

The worst case is easy, it happens when all x are in the end, so we need to $n - k + 1$
The probability of existing an index being on the $i - th$ *index is equal the probability of no x in the previous index and the probability of existing an x in the* $i - th$

Ok now ke know that if we have k x in the array them the last position we need to check is the position $n - k + 1$*. Furthermore, given the index i, we will find x in i if we don't find x in the previous. what leads us to the following:*
$$\sum_{i=1}^{n-k+1} \frac{ik(n-k)!\,(n-i)!}{n!\,\big(n-(k+i-1)\big)!} = \frac{(n+1)}{k+1}$$

g)Suppose that there are no indices 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?

Clearly it's n, since we will have to pass through all the array

Finally, consider a randomized algorithm SCRAMBLE − SEARCH that first randomly permutes the input array and then runs the deterministic linear search given above on the resulting permuted array.

h) Letting k be the number of the indices i such that A[i] = x, give the worst worst case and expected running time of Scramble − Search for the cases in which k = 0 and k = 1. Generalize your solution to handle the case in which k ≥ 1

This algorithm runs identically to the determinist approach, given that the output of the random part is the same, so its all equal

i) Which of the three searching algorithms would you use? Explain your answer

Since its all equal we choose the determinist one because even though the complexities are the same, we have more work done in the Scramble one.