1 Markov chain

i. Boston has days which are either sunny or rainy and can be modeled as a two-state Markov chain. Every sunny day is followed by another sunny day with probability 0.8. Every rainy day is followed by another rainy day with probability 0.6.

1. Draw the state diagram and write down the transition matrix for Boston weather?

2. If yesterday was sunny what is the probability of tomorrow being sunny?

3. What is the probability that it will be a sunny day a year from today?
Solution:

1. Let sunny be state 1 and rainy be state 2. The state diagram is:

   ![Diagram](image)

Figure 1: State diagram with transitions for Boston weather

The transition matrix \( P \) is:

\[
\begin{pmatrix}
0.8 & 0.2 \\
0.4 & 0.6
\end{pmatrix}
\]

2. The two-state transition matrix \( P^2 \) is:

\[
\begin{pmatrix}
0.72 & 0.28 \\
0.56 & 0.44
\end{pmatrix}
\]

from which we see that \( P(\text{tomorrow sunny} \mid \text{yesterday sunny}) \) is 0.72.

3. In one year’s time the chain would have (almost) achieved the steady state distribution. Let \((\pi_1, \pi_2)\) denote the steady state distribution. Then we solve:

\[
(\pi_1 \pi_2)
\begin{pmatrix}
0.8 & 0.2 \\
0.4 & 0.6
\end{pmatrix}
= (\pi_1 \pi_2)
\]

along with the requirement (since it’s a distribution) that \( \pi_1 + \pi_2 = 1 \). Solving we get \((\pi_1, \pi_2) = (\frac{2}{3}, \frac{1}{3})\). Thus \( P(\text{year from today sunny}) \) is \( \frac{2}{3} \).

ii. A machine can be in one of 3 states - state 1: running, state 2: under repair and state 3: idle - given by the transition matrix \( P \):

\[
\begin{pmatrix}
0 & 0.5 & 0.5 \\
1 & 0 & 0 \\
1 & 0 & 0
\end{pmatrix}
\]

1. Draw the state diagram and describe the transitions the machine makes.
2. Compute $P^n$ for all $n \geq 1$.

3. Does the machine have a steady state distribution?

**Solution:**

1. The machine switches from running (state 1) to under repair (state 2) or idle (state 3) with equal probabilities. And then it always returns to running.

2. For all odd $n$ $P^n = P$ and for all even $n$ $P^n$ is:

\[
\begin{pmatrix}
1 & 0 & 0 \\
0 & 0.5 & 0.5 \\
0 & 0.5 & 0.5
\end{pmatrix}
\]

3. Starting from running (state 1) it will always be in under repair (state 2) or idle (state 3) after an odd number of steps and back to running (state 1) after an even number of steps. So, clearly the machine does not have a steady state distribution.
2 Searching A

Moe, Larry, and Curly have just purchased three new computers. Moe’s computer is 20 times faster than Larry’s and 50 times faster than Curly’s. However, Moe runs Ordered-Linear-Search on his computer, while Larry and Curly run Chunk-Search and Binary-Search, respectively. For the questions below, feel free to approximate using guess and test or use Wolfram Alpha for a precise answer.

1. How large must \( N \) be so that Curly’s computer starts to outperform Moe’s?

\[
\text{Solution: } \text{Solve for } 50 \times \log_2 n < n. \ n = 439 \text{ is the crossover point.}
\]

2. How large must \( N \) be so that Larry’s computer starts to outperform Moe’s?
3. How large must N be so that Curly’s computer starts to outperform Larry’s?

Solution: Larry’s is 2.5 times faster than Curly’s. Solve for $2.5 \cdot \log_2 n < 2 \cdot \sqrt{n}$. This is 50.

4. Moe and Curly both run a search on the same data set. Despite the fact that Curly’s machine is 50 times slower than Moe’s, Curly’s machine performs the search 100 times faster than Moe’s machine. How large is the data set?

Solution: $100 \cdot 50 \cdot \log_2 n = n$ thus $n = 81,580$

5. Suppose that Moe switches to Chunk-Search. On this same data set, will Moe’s machine now outperform Curly’s? Explain.

Solution: For $81,579$ $50 \cdot \log_2 81,579 = 815$ while $\sqrt{81,579} = 285$. Moe’s computer is about 3 times faster.
3 Searching B

i. We will consider the problem of search in ordered and unordered arrays.

   1. How many steps does it take to search for a given element in an unordered array of length 1024?

   2. Describe a fast algorithm to search for an element in an ordered array.

   3. How many steps does your fast algorithm (from the previous part) take on an ordered array of length 256?
Solution:

1. Since the array is unordered, in the worst case, every element will need to looked at so the number of steps is 1024.

2. Binary search is a fast (indeed, the fastest) algorithm for search in an ordered array. Given an array $A$ of $n$ elements with values or records $A_0, A_1, \ldots, A_{n-1}$ sorted such that $A_0 \leq A_1 \leq \ldots \leq A_{n-1}$ and target value $T$, the following subroutine uses binary search to find the index of $T$ in $A$.

   1. Set $L$ to 0 and $R$ to $n - 1$.
   2. If $L > R$, the search terminates as unsuccessful.
   3. Set $m$ (the position of the middle element) to the floor (the largest previous integer) of $(L + R)/2$.
   4. If $A_m < T$, set $L$ to $m + 1$ and go to step 2.
   5. If $A_m > T$, set $R$ to $m - 1$ and go to step 2.
   6. Now $A_m = T$, the search is done; return $m$.

3. On an ordered array of length 256 the number of comparison steps is at most 9.

ii. A company database has 10,000 customers sorted by last name, 30% of whom are known to be good customers. Under typical usage of this database, 60% of lookups are for the good customers. Two design options are considered to store the data in the database:

   • Put all the names in a single array and use binary search.
   • Put the good customers in one array and the rest of them in a second array. Only if we do not find the query name on a binary search of the first array do we do a binary search of the second array.

Given these options, answer the following.

1. Calculate the expected worst-case performance for each of the two structures above, given typical usage. Which of the two structures is the best option?

2. Suppose that over time the usage of the database changes, and so a greater and greater fraction of lookups are for good customers. At what point does the answer to the previous part change?
3. Under typical usage again, suppose that instead of binary search we had used linear search. What is the expected worst-case performance of each of the two structures and which is the better option? Where is the cross-over this time?

**Solution:**

**Binary Search**

Using one array:
\[\lceil \log(10000 + 1) \rceil = 14.\]

Using two arrays:
before you lookup the array of bad customer, you look up the array of good customers first.

Two arrays, binary search: 
\[0.6 \times \lceil \log(3000 + 1) \rceil + 0.4 \times (\lceil \log(3000 + 1) \rceil + \lceil \log(7000 + 1) \rceil) = 17.2\]

In this scenario, option 1 is the best option

**Crossover point:**
When about 84.6 percent of lookups are performed for good customers, option 2 has a better worst case performance.
\[x \times \lceil \log(3000 + 1) \rceil + (1 - x) \times (\lceil \log(3000 + 1) \rceil + \lceil \log(7000 + 1) \rceil) = 14\]. Then: \(x = 0.846\).

**Ordered Linear Search**

Using one array:
10,000

Using two arrays:
\[0.6 \times 3000 + 0.4 \times (3000 + 7000) = 5800\]
In this scenario, option 2 is better.

**Crossover point:**
There is no crossover in this scenario because option 2 cannot perform worse than option 1 under any circumstances.
4 Sorting

i. Briefly describe how INSERTION-SORT works. How many element examinations are needed for an array of length $n$ in the worst case?

\textit{Solution:} Insertion sort works iteratively. In each iteration the next element from the unsorted array/list (portion) is removed and inserted into the right place in the sorted array/list (portion) until all the elements are sorted.

The worst case occurs when the array is sorted in reverse order. So the worst case time complexity of insertion sort is quadratic: $1 + 2 + \ldots + n = n(n + 1)/2$.

ii. Suppose that a number $S$ and a sorted array $a_1 < a_2 < \cdots < a_{n-1} < a_n$ of distinct numbers are given.

1. What is the running time of the “brute force” algorithm to determine whether $a_i + a_j = S$ for some $1 \leq i < j \leq n$?

2. Give an algorithm for the above problem which runs in linear time.
Solution:

1. The “brute force” algorithm examines all possible \( \binom{n}{2} \) pairs of numbers and checks whether any of them sums up to \( S \). The running time is proportional to \( \binom{n}{2} \) and hence quadratic. Note that the “brute force” algorithm does not use the fact that the array is sorted.

2. The trick is to create a new array \( B = (S - a_1, S - a_2, \ldots, S - a_n) \) and then check whether \( B \) and the original array \( A = (a_1, a_2, \ldots, a_n) \) contain any common elements. A common element in the list \( B \) is of the form \( S - a_i \), while in the list \( A = a_j \). Those elements being the same means

\[
S - a_i = a_j \implies a_i + a_j = S
\]

How do we find common elements in arrays \( A \) and \( B \)? We simply merge the two arrays taking note whenever we encounter the same numbers (we can use the merge because both lists are sorted). Since creating the array \( B \) takes linear time, and so does the merging process, the running time of this algorithm is linear.

Alternative solution: start with \( i = 1, j = n \) and continuously evaluate \( a_i + a_j \) against \( S \). If sum of the two current elements is smaller, increase \( i \); if it is bigger decrease \( j \). Keep doing so until the sum is \( S \) (DONE) or until \( j = i + 1 \) when we can declare “NOT FOUND”.

5 Series and sequences

1. Recognize the following sequence and write it in concise form. Compute a close-form summation of the first \( n \) terms.
5, 7, 9, 11, 13, 15, ...

Solution: Arithmetic progression \( 2x + 3 \) for \( x = 1, 2, 3, 4, 5 \... \)

\[
\sum_{k=1}^{n} (2k + 3) = 2 \sum_{k=1}^{n} k + 3n = 2 \cdot \frac{n(n + 1)}{2} + 3n = n(n + 1) + 3n = n(n + 4)
\]

2. Recognize the following sequence and write it in concise form. Compute a close-form summation of the first \( n \) terms.
3, 9, 19, 33, 51, 73, 99, ...

Solution: Quadratic progression \( 2x^2 + 1 \) for \( x = 1, 2, 3, 4, 5 \... \)

\[
\sum_{k=1}^{n} (2k^2 + 1) = 2 \sum_{k=1}^{n} k^2 + n = 2 \cdot \frac{n(n + 1)(2n + 1)}{6} + n = \\
= \frac{n(n + 1)(2n + 1) + 3n}{3} = \frac{n(2n^2 + 3n + 1) + 3n}{3} = \frac{n(2n^2 + 3n + 4)}{3}
\]
3. Recognize the following sequence and write it in concise form. Compute a close-form summation of the first $n$ terms.
48, 96, 192, 384, 768, 1536, ...

**Solution:** Geometric progression $24 \times 2^x$ for $x = 1, 2, 3, 4, 5, ...

$$\sum_{k=1}^{n}(24 \times 2^k) = 24 \sum_{k=1}^{n}2^k = 24(\sum_{k=0}^{n}2^k - 1) = 24(\frac{2^{n+1} - 1}{2 - 1} - 1) = 24(2^{n+1} - 2) = 48(2^n - 1)$$

4. Prove that $\frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots + \frac{1}{n^2} < 1$ for any natural number $n$.

**Solution:**

$$\frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots + \frac{1}{n^2} < \frac{1}{1 \times 2} + \frac{1}{2 \times 3} + \frac{1}{3 \times 4} + \cdots + \frac{1}{(n-1) \times n} < \sum_{k=1}^{n}(\frac{1}{k} - \frac{1}{k+1}) = 1 - \frac{1}{n+1}$$
6 Series Formula by Induction

1. \( S_n = 1 \cdot 2 + 2 \cdot 3 + ... + (n-1) \cdot n = \frac{n(n-1)(n+1)}{3} \)

   **Solution:** Base case \( n = 1 \) : \( S_1 = 0 = \frac{1(1-1)(1+1)}{3} \)
   Base case \( n = 2 \) : \( S_2 = 1 \cdot 2 = \frac{2(2-1)(2+1)}{3} \)
   Induction Step: Given \( S_n \) formula, we want to prove \( S_{n+1} = \frac{(n+1)(n+1-1)(n+1+1)}{3} \).

   \[
   S_{n+1} = S_n + n(n+1) = \frac{n(n-1)(n+1)}{3} + n(n+1) = \frac{n(n-1)(n+1) + 3n(n+1)}{3} = \frac{n(n+1)(n-1+3)}{3} = \frac{n(n+1)(n+2)}{3} = \frac{(n+1)(n+1-1)(n+1+1)}{3}
   \]

2. \( S_n = \sum_{k=1}^{n} k^3 = \left( \sum_{k=1}^{n} k \right)^2 \)

   **Solution:** Base case \( n = 1 \) : \( S_1 = 1^3 = 1^2 = (1)^2 \)
   Base case \( n = 2 \) : \( S_2 = 1^3 + 2^3 = (1 + 2)^2 \)
   Induction Step: Given \( S_n \) formula, we want to prove \( S_{n+1} = \sum_{k=1}^{n+1} k^3 = \left( \sum_{k=1}^{n+1} k \right)^2 \)

   \[
   S_{n+1} = S_n + (n+1)^3 = \left( \sum_{k=1}^{n} k \right)^2 + (n+1)^3 = \left( \frac{n(n+1)}{2} \right)^2 + (n+1)^3 = \frac{n^2(n+1)^2}{4} + \frac{4(n+1)^3}{4} = \frac{(n+1)^2}{4}(n^2 + 4n + 4) = \frac{(n+1)^2}{4}(n+2)^2 = \left( \frac{(n+1)(n+2)}{2} \right)^2 = \left( \sum_{k=1}^{n+1} k \right)^2
   \]
3. \(S_n = (1 - \frac{1}{4})(1 - \frac{1}{9})(1 - \frac{1}{16}) \ldots (1 - \frac{1}{n^2}) = \frac{n+1}{2n}\)

\textit{Solution:} Base case \(n = 2 : S_2 = (1 - \frac{1}{4}) = \frac{2+1}{2}\)

Inductive Step: Given \(S_n\) formula, we want to prove \(S_{n+1} = \frac{n+1+1}{2(n+1)}\)

\[S_{n+1} = S_n + \frac{1}{(n+1)^2} \Rightarrow S_{n+1} = S_n \cdot \frac{n+1}{2n} \cdot \frac{(n+1)^2 - 1}{(n+1)^2} = \frac{(n+1-1)(n+1+1)}{2n(n+1)} = \frac{(n+1)}{2(n+1)}\]

4. \(S_n = \sum_{k=1}^{n} (3n - 2)^2 = n(6n^2 - 3n - 1)/2\)

\textit{Solution:} Base case \(n = 1 : (3 - 2)^2 = 1(6 - 3 - 1)/2\)

Inductive Step: Given \(S_n\) formula, we want to prove \(S_{n+1} = (n+1)(6(n+1)^2 - 3(n+1) - 1)/2\)

\[S_{n+1} = S_n + (3(n+1)-2)^2 = n(6n^2-3n-1)/2+(9n^2+1+6n) = \frac{1}{2}(6n^3-3n^2-n+18n^2+2+12n) = \frac{1}{2}(n+1)(6n^2+9n+2) = \frac{1}{2}(n+1)(6n^2+6n+6+3n-3-1) = \frac{1}{2}(n+1)(6(n+1)^2 - 3(n+1) - 1)\]

5. \(S_n = 1 \cdot 2 + 2 \cdot 2^2 + 3 \cdot 2^3 + 4 \cdot 2^4 + \ldots + n \cdot 2^n = 2 + (n-1)2^{n+1}\).

\textit{Solution:} Base case \(n = 1 : S_1 = 1 \cdot 2 = 2 + (1-1)2^{1+1}\)

Inductive Step: Given \(S_n\) formula, we want to prove \(S_{n+1} = 2 + n \cdot 2^{n+2}\)

\[S_{n+1} = S_n + (n+1)2^{n+1} = 2+(n-1)2^{n+1} + (n+1)2^{n+1} = 2+2^{n+1}(n-1+n+1) = 2+2n \cdot 2^{n+1} = 2+n \cdot 2^{n+2}\]
6. \( S_n = \sum_{k=1}^{n} k \cdot k! = (n + 1)! - 1 \)

**Solution:** Base case \( n = 1 : S_1 = 1 \cdot 1! = 2! - 1 \)
Inductive Step Given \( S_n \) formula, we want to prove \( S_{n+1} = (n + 2)! - 1 \)

\[
S_{n+1} = S_n + (n+1)(n+1)! = (n+1)! - 1 + (n+1)(n+1)! = (n+1)!(n+1+1) - 1 = (n+2)! - 1
\]

7. \( S_n = \sum_{k=1}^{n} (2k - 1)^2 = n(4n^2 - 1)/3 \)

**Solution:** Base case \( n = 1 : S_1 = (2 - 1)^2 = 1 = 1(4 \cdot 1^2 - 1)/3 \)

Induction step: give \( S_n \) formula, we want to prove that \( S_{n+1} = (n + 1)(4(n + 1)^2 - 1)/3 \)

\[
S_{n+1} = S_n + (2(n + 1) - 1)^2 = n(4n^2 - 1)/3 + (2n + 1)^2 = n(2n - 1)(2n + 1)/3 + 3(2n + 1)^2/3 = \\
= \frac{1}{3}(2n+1)(n(2n-1)+3(2n+1)) = \frac{1}{3}(2n+1)(2n^2-n+6n+3) = \frac{1}{3}(2n+1)(2n^2+2n+3n+3) = \\
= \frac{1}{3}(2n + 1)(2n(n + 1) + 3(n + 1)) = \frac{1}{3}(n + 1)(2n + 1)(2n + 3) = \frac{1}{3}(n + 1)(4n^2 + 8n + 3) = \\
= \frac{1}{3}(n + 1)(4n^2 + 4n + 4 - 1) = \frac{1}{3}(n + 1)(4(n + 1)^2 - 1)
\]
8. \[ S_n = \sum_{i=1}^{n} (-1)^i * i^2 = (-1)^n \frac{1}{2} n(n+1) \]

**Solution:** Base case \( n = 1 \): (-1)1^2 = (-1)\frac{1 \times (1+1)}{2}

Induction step: assuming \( S_n \) true, we want to prove \( S_{n+1} = (-1)^{n+1} \frac{1}{2} (n+1)(n+2) \)

\[ S_{n+1} = S_n + (-1)^{n+1} (n+1)^2 = (-1)^n \frac{1}{2} n(n+1) + (-1)^{n+1} (n+1)^2 = \]
\[ = (-1)^n \frac{1}{2} (n+1)(n-2(n+1)) = (-1)^n \frac{1}{2} (n+1)(-n-2) = (-1)^{n+1} \frac{1}{2} (n+1)(n+2) \]

9. Prove that \( n! > 3^n > 2^n > n^2 > n \log_2(n) > n > \log_2(n) \) for \( n \geq 7 \)

**Solution:** Base case \( n = 7 \):
\[ 7! = 5040 > 3^7 = 2187 > 2^7 = 128 > 7^2 = 49 > 7 \times \log_2(7) = 19.65 > 7 > \log_2(7) = 2.81 \]

Induction step: using inequalities for \( n \), we want to prove that
\( (n+1)! > 3^{n+1} > 2^{n+1} > (n+1)^2 > (n+1) \log_2(n+1) > n+1 > \log_2(n+1) \)

We start from the left side:
\( (n+1)! = n!(n+1) > 3^n(n+1) > 3^n * 3 = 3^{n+1} \]
\[ 2^{n+1} = 2^n * 2 > n^2 * 2 = n^2 + n^2 \geq n^2 + 7n = n^2 + 2n + 5n > n^2 + 2n + 1 = (n+1)^2 \]

We have proved so far the first three inequalities. Here is the proof for the last one:
\[ 2^{n+1} > n^2 \Rightarrow n + 1 > \log_2(n^2) \geq \log_2(7n) = \log_2(n + 6n) > \log_2(n + 1) \]

Finally the inequalities fourth and fifth:
\( n + 1 > \log_2(n + 1) \Rightarrow (n + 1)^2 > (n + 1) \log_2(n + 1) > (n + 1) \log_2(7) > n + 1 \)
7 Induction proofs

PB 1 Show that 5 divides $8^n - 3^n$ for any natural number $n$.

Solution: Base case $n = 0: 8^0 - 3^0 = 1 - 1 = 0$ is a multiple of 5
Induction Step: Assuming $8^n - 3^n = 5k$ we want to prove that $5(8^{n+1} - 3^{n+1})$

$8^{n+1} - 3^{n+1} = 8 \cdot 8^n - 3 \cdot 3^n = 5 \cdot 8^n + 3(8^n - 3^n) = 5 \cdot 8^n + 5k = 5(8^n + k)$ thus multiple of 5

Solution without induction:
$8^n - 3^n = (8 - 3) \sum_{k=0}^{n} 8^k \cdot 3^{n-k}$

which is a multiple of 5 due to the first factor.

PB 2 Prove that $S_n = 17n^3 + 103n$ is divisible by 6 for all integers $n$.

Solution: Since $17 = -1 \mod 6$ and $103 = 1 \mod 6$, we have $S_n = -n^3 + n \mod 6$
Direct proof (w/out induction):
$S_n \mod 6 = -n^3 + n = -n(n - 1)(n + 1)$ which is a product of three consecutive integers, so among them must be a multiple of 2 and a multiple of 3. Thus the product is a multiple of 6.

Solution with induction for positive range. Base case $n = 0 ⇒ S_0 = 0; n = 1 ⇒ S_1 = 120$
Induction step : Assuming $S_n = 6k$, show that $S_{n+1} \mod 6 = 0$

$S_{n+1} \mod 6 = -(n + 1)^3 + n + 1 = -n^3 - 3n^2 - 3n - 1 + n + 1$
$= S_n - 3n^2 - 3n = 6k - 3n(n + 1) = 6(k - \frac{n(n + 1)}{2})$

which is a multiple of 6 because $\frac{n(n + 1)}{2}$ is integer.

For negative $n$, not that $S_n = -S_{-n}$ which must be multiple of 6 proven above for positive $n$
**PB 3 Binary trees height**  Prove that depth (height) of a binary tree with \(n\) nodes is at least \(\lfloor \log_2(n) \rfloor\) (depth is the max number of edges on a path from root to a leaf).

*Solution:* Base case \(n = 1\), depth = 0 \(\geq \log(1) = 0\)
Base case \(n = 2\), depth = 1 \(\geq \log(2) = 1\)

Strong Induction Step: Will assume the property is true for any \(k < n\), and will prove it for \(n\). In particular the \(k\)-s for which we are going to need it are the number of nodes in the Left and Right subtrees.

Let's say the root of the binary tree has a left subtree with \(p\) nodes and a right subtree with \(q\) nodes. Then \(n = 1 + p + q\). Lets assume (without loss of generality) that \(p \geq q\).

\(p < n\) so by induction hypothesis we know \(\text{depth}_L \geq \lfloor \log_2(p) \rfloor\)

\[
 \text{depth} = 1 + \max(\text{depth}_L, \text{depth}_R) \geq 1 + \text{depth}_L \geq 1 + \lfloor \log_2(p) \rfloor = 1 + \lfloor \log_2(\frac{2p}{2}) \rfloor = 1 + \lfloor \log_2(2p) - 1 \rfloor = \lfloor \log_2(2p) \rfloor
\]

If \(n\) is even then \(p \geq q = n - p - 1 \Rightarrow 2p > n \Rightarrow \lfloor \log_2(2p) \rfloor \geq \lfloor \log_2(n) \rfloor\)

If \(n\) is odd then \(\lfloor \log_2(n - 1) \rfloor = \lfloor \log_2(n) \rfloor\)
and \(p \geq q = n - p - 1 \Rightarrow 2p \geq n - 1 \Rightarrow \lfloor \log_2(2p) \rfloor \geq \lfloor \log_2(n - 1) \rfloor = \lfloor \log_2(n) \rfloor\)
**PB 4 Polygon sum of angles**  Prove that the sum of the interior angles of a convex polygon with \( n \) sides is \((n - 2)\pi\). You can assume known that the sum of angles of any triangle is \( \pi \).

**Solution:** Base case is the triangle \( n = 3 \) for which the property is given as known: sum of triangle angles is \((3 - 2)\pi = \pi\).

Induction step: given the property holds for a convex polygon with \( n \) sides, we want to prove it for a convex polygon with \( n + 1 \) sides. If \( P = \text{ABDC...} \) is the polygon with \( n + 1 \) sides, cutting a triangle \( \text{ABC} \) out of it results in a convex polygon with \( n \) sides.

So we can sum the angles of \( P \) as the sum of angles in triangle \( \text{ABC} \) plus the sum of angles of \( n \)-sides-polygon \( \text{ACDE...} \). Applying induction hypothesis we get
\[
\pi + (n - 1 - 2)\pi = (n - 2)\pi
\]

**PB 5 Exponential approximation around 0**  Show that if real number \( x > -1 \) and \( n \) is any natural number, then \((1 + x)^n \geq 1 + nx\)

**Solution:**
Base case \( n = 0 \) : \((1 + x)^0 = 1 = 1 + 0x\)
Base case \( n = 1 \) : \((1 + x)^1 = 1 + x = 1 + 1x\)
Induction step: assuming \((1 + x)^n \geq 1 + nx\), we want to prove that \((1 + x)^{n+1} \geq 1 + (n + 1)x\).

Note that \(x > -1\) means \(x + 1 > 0\) thus multiplying an inequality with \((1 + x)\) preserves the sign.

\[
(1 + x)^{n+1} = (1 + x)^n(1 + x) \geq (1 + nx)(1 + x) = 1 + nx^2 + x + nx = 1 + (n + 1)x + nx^2 \geq 1 + (n + 1)x
\]

\[PB 6 \text{ Fermat's Theorem again } \star.\] Prove Fermat’s little theorem, by induction over \(a\) (use binomial theorem)

For any \(p\) prime and reminder \(0 < a < p\), we have \(a^{p-1} = 1 \mod p\)

**Solution:** Base case \(a = 1\) : \(1^{p-1} = 1 \mod p\)

Induction step: given \(a^{p-1} = 1 \mod p\) and \(0 < a < p - 1\), we want to prove \((a+1)^{p-1} = 1 \mod p\)

First we note two things:

- \(a^{p-1} = 1 \mod p \Rightarrow a^p = a \mod p\).
- \(\frac{p!}{k!(p-k)!}\) for \(1 \leq k \leq p - 1\) is a multiple of \(p\) because the numerator contains the factor \(p\) but the denominator does not (and cannot made it from other factors since \(p\) is prime). Then

\[
(a + 1)^p = \sum_{k=0}^{p} \binom{p}{k} a^k * 1^{p-k} = a^0 + a^p + \sum_{k=1}^{p-1} \binom{p}{k} a^k =
1 + a + \sum_{k=1}^{p-1} \frac{p!}{k!(p-k)!} a^k =
a + 1 \mod p
\]

Since \(a + 1\) is a proper reminder \(mod p\) prime, it has an inverse \((a + 1)^{-1}\) so

\[
(a + 1)^{p-1} = (a + 1)^p(a + 1)^{-1} = (a + 1)(a + 1)^{-1} = 1 \mod p
\]
PB 7 Will everyone get the same grade? Jimmy found a proof that every student in CS1800 will get the same exact grade, by induction over the number $n$ of students in the class:

- Define the statement $P_n$= “in a class of size $n$, all students get the same grade”
- Base step $n = 1$, $P_n$ is true, since there is only one student
- Inductive step: If a the class has size $n$, consider all $n$ subsets of size $n - 1$. Since $P_{n-1}$ is true, then all these subsets of students will get the same grade (per subset). But the subsets intersect, so it means all students will get the same grade, or $P_n$ is true.

Where is Jimmy wrong?

Solution: Jimmy is wrong in Inductive Step, specifically $P_1 \Rightarrow P_2$ (the other implications, and the best case do hold true). For $n = 2$ there are two subsets of $n - 1 = 1$ each; they DO NOT OVERLAP (INTERSECT) so the argument is flawed.