Best, Worst and Average Cases:

The **best case** running time of an algorithm is the function defined by the minimum number of steps taken on any instance of size.

The **worst case** running time of an algorithm is the function defined by the maximum number of steps taken on any instance of size $n$.

The **average case** running time of an algorithm is the function defined by an average number of steps taken on any instance of size $n$.

In some instances, the above three cases are equivalent (there is one formula only) for the same value of instance characteristic, like in summation array elements example (instance characteristic is $n$). But in other instances, dependency on instance characteristics is not enough to determine the time complexities, like in search problem.

**Sequential search**

write a sequential search function and then find the best, worst, and average case time complexity.

**Solution:**
Function SeqSearch (a: array of element, n:number of element, k: the number that look for it)

Begin

\[ i=n \]
\[ a[0]=k \]

while ( \( a(i) \neq k \) ) do

\[ i=i-1 \]

end while

return i

end
Solution: In this example, the search process begins from the end of a.

The successful Searches:
- The Best case time complexities: When k number found in the position a(n). Therefore, the time complexities in the best case will be: \( T_{BSeqSearch}(n) = 4 = \Theta(1) \)
- The Worst case time complexities: when k found in the first position a(1), So: \( T_{WSeqSearch}(n) = 2 + 2n = \Theta(n) \)
- The Average case time complexities: It is the average of complexities for all cases:
  \[
  T_{ASeqSearch}(n) = \sum_{i=1}^{n} \frac{(n-i+1)}{n} = \frac{\sum_{i=1}^{n} n - \sum_{i=1}^{n} i + \sum_{i=1}^{n} 1}{n} = \frac{n^2 - \frac{(n+1)n}{2} + n}{n} = n - \frac{(n+1)}{2} + 1 = \frac{n+1}{2} = \Theta(n)
  \]

The Failed Searches: When k doesn't exist in the array:
- \( T_{FSeqSearch}(n) = n + 1 = \Theta(n) \)

Maximum element in one dimension array:
Find the max number in one dimension array, then find the best, worst and average cases for time complexities?

Function ArrayMax(a: array of element, n: number of element)

Begin
1. max = a(1)
2. For i = 2 to n
3. If max < a(i) then
4. max = a(i)
5. end if
6. end for
7. return max
end

The instance characteristics of this problem is n
Solution:

The successful cases:

- The Best case time complexities: When \(a(1)\) is the max number. That mean, there is no need to entered in if block (if condition is never be true):
  1. \(\cdots\cdots\cdots 1\)
  2. \(\cdots\cdots\cdots (n-2+1)+1=n\)
  3. \(\cdots\cdots\cdots n-1\)
  4. \(\cdots\cdots\cdots 0\)
  7. \(\cdots\cdots\cdots 1\)

\(T_B_{ArrayMax}(n)=2n+1=\Theta(n)\)

- The Worst case time complexities: when array a is sorted in increasing form, max element is \(a(n)\), So:
  1. \(\cdots\cdots\cdots 1\)
  2. \(\cdots\cdots\cdots (n-2+1)+1=n\)
  3. \(\cdots\cdots\cdots n-1\)
  4. \(\cdots\cdots\cdots n-1\)
  7. \(\cdots\cdots\cdots 1\)

\(T_W_{ArrayMax}(n)=3n=\Theta(n)\)

- The Average case time complexities: It is the average of complexities for all cases from the best to the worst one:

\[
\text{Average case} = \frac{\text{sumasion of cases of } i (1,2,3,\ldots,n)}{\text{number of steps}(n)}
\]

\[
T_A_{ArrayMax}(n)=\frac{\sum_{i=1}^{n}(2n+i)}{n} = \frac{5n+1}{2} = \Theta(n)
\]

Note:

- In average case, the \(i\) counter can be started from 1 or 0 depending on the example.
- if the loop is beginning from last to the first element then the average case will be:
\[ \sum_{i=1}^{n} \frac{(3n-i+1)}{n} = \? \text{ H.W.} \]

The Failed cases: there is no failed case in this example.

**H.W.**

Prove that:

- \[ \sum_{i=1}^{n} \frac{(2n+i)}{n} = \frac{5n+1}{2} \]
- \[ 5n^2 - 6 = \Theta(n^2) \]