Assignment-2
Computing Space Complexity

Space Complexity by Experimental Analysis

Algorithm/Pseudo Code

The following algorithm/pseudo code is given:

```
T ← 1
For i = 1 to n
   T ← T * A[i]
Return T
```

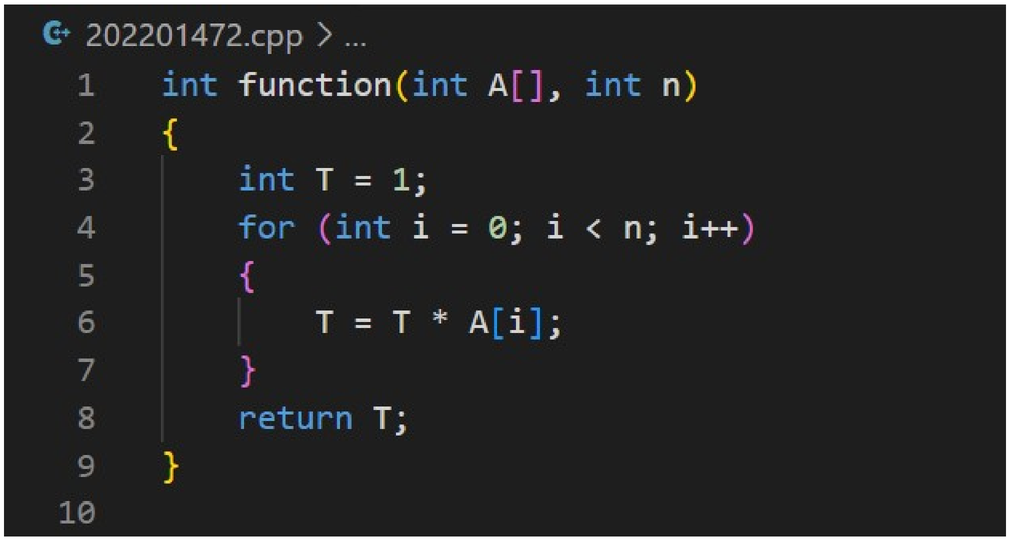
Code

Here is the complete code for the given program in C++:

```
int fun(int A[] , int n)
{
     int T = 1;
     for( int i = 0 ; i < n ; i++)
     {
              T = T * A[i];
     }
     return T;
}
```

Space Complexity

Here in this programme the size of array is n and the array is type of integer. Since the integers take 4 bytes of memory in 64 bit architecture, the total memory taken by the array is $4n$,where n is the number of elements in an array.



```
202201472.cpp > ...
int function(int A[], int n)
{
    int T = 1;
    for (int i = 0; i < n; i++)
    {
        T = T * A[i];
    }
    return T;
}
```

Figure 1: C++ Programme

The integer variables n, T and i will take 4 bytes each.Let the for loop in the code requires $C1$ bytes of memory to store the condition statement and iteration statement and the return statement in function takes $C2$ bytes of memory. So the total amount of memory used by the above algorithm can be represented as

$$S(n) = 4n + 12 + C1 + C2$$

where n is the number of elements in an input array and $S(n)$ has unit bytes. The space taken by the array A depends on the input size and the space taken by integer variables(n, i, T) is constant.

For the purpose of experiment doing approximation, let's say $C1$ is 8 bytes as the condition statement checks condition between two integers and $C2$ be 4 bytes as the value returned by the function is integer. Now the the space taken by the programme can be written as

$$S(n) = 4n + 12 + 8 + 4$$
$$S(n) = 4n + 24$$

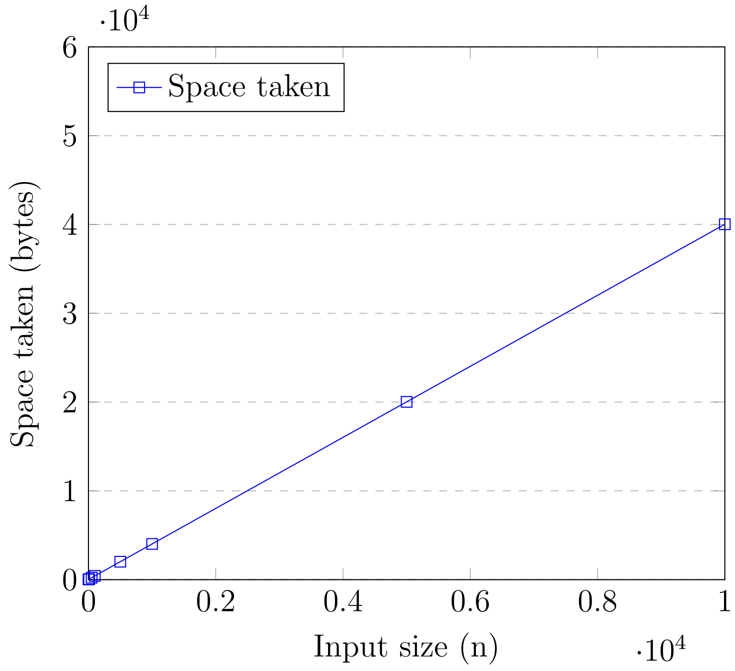
Observation Table

Let's observe the space(in bytes) taken by the program for different input sizes(n).

Input Size (n)	Space Taken (bytes)
1	28
10	64
50	224
100	424
500	2024
1000	4024
5000	20024
10000	40024

Graph

Let's plot the graph of the space taken by the program for different input sizes. We will assume that the size of the array is the same as the input size.



$$Space(n) = \mathcal{S}(n) = 4n + 24$$

The graph shows that the space taken by the program increases linearly with the input size, which is consistent with the O(n) space complexity of the program.

Expression

The expression for the space taken by the program in terms of the input size n can be written as:

$$Space(n) = \mathcal{S}(n) = 4n + 24$$

Space Complexity by Tilda Approximation

1. The space complexity expression for the above programme is,

$$\mathcal{S}(n) = 4n + 24$$

2. According to tilda approximation we can ignore the lower order terms. Here the lower order term is constant 24.

$$\mathcal{S}(n) = 4n$$

3. According to tilda approximation we can also ignore the leading constants of the higher oder term. Here the higher order term is $4n$ which is linear term having 4 as leading constant.

$$\mathcal{S}(n) = n$$

4. Hence the space complexity expression in tilda approximation will look like,

$$\mathcal{S}(n) = n$$

Space Complexity by Asymptotic Analysis

1. In asymptotic analysis we always talk about n tends to infinity. so as $n \to \infty$, we can ignore the lower order terms as well as leading coefficients.

$$\mathcal{S}(n) = n$$

2. The Big-O notation provides the upper bound for an algorithm. In Big-O notation we can write the space complexity of above algorithm as follows,

$$Space(n) = \mathcal{O}(\mathcal{S}(n)) = \mathcal{O}(n)$$

3. Space complexity $\mathcal{O}(n)$ implies that the given algorithm will always take linear time to execute the programme and it will never take space more than linear.