3.2.2 Space Complexity:

Space complexity is nothing but the amount of memory space that an algorithm or a problem takes during execution of that particular problem.

Space needed by an algorithm is given by:

$S(P) = C$ (fixed part) + S_p (variable part)

Fixed part: is independent of instance characteristic. i.e. space for simple variables, constants, etc (int a; int b)

Variable part: is space for variables whose size is dependent on particular problem instance.
Example: array

Example:

i.
```
Algorithm max(A, n)
{
    Result = A[1];
    For i = 2 to n do
    If A[i]>result then
            Result = A[i];
    Return = A[i];
}
```
Answer:
Variables i, n, result = 1 unit each // 4 bytes each, which is fixed, can't be changed
Variable A = n units // n * 4 bytes, completely depends on 'n'
Total = n + 4
Removing constant value:
Space complexity: O(n)

ii.
```
Algorithm abc(d, e, f)
{
    Return d + e * f + (d + e + f)/(d + e) + 4.0
}
```
Answer:
d: 4 bytes, fixed
e: 4 bytes, fixed
f: 4 bytes, fixed
total: 12 bytes at least
removing constant values:
space complexity: O(1), constant space

iii.
```
Algorithm sum(a, n)
{
    s=0.0;
for i=1 to n do
    s=s+a[i];
return s;
}
```

Answer:

Variables i, n, s = 4 bytes each i.e. (4 + 4 + 4 = 12)// fixed part

Variable a: n * 4 bytes // variable part

Total =(4*n+12) bytes

Removing constant values:

Total = 4*n

Space complexity: O(n)