STEP COUNT METHOD IN ALGORITHM

The operation-count method of estimating time complexity omits accounting for the time spent on all but the chosen operations. In the step-count method, we attempt to account for the time spent in all parts of the algorithm. As was the case for operation counts, the step count is a function of the problem size.

A step is any computation unit that is independent of the problem size. Thus 10 additions can be one step; 100 multiplications can also be one step; but n additions, where n is the problem size, cannot be one step. The amount of computing represented by one step may be different from that represented by another. For example, the entire statement

return a+b+b*c+(a+b-c)/(a+b)+4; can be regarded as a single step if its execution time is independent of the problem size.

We may also count a statement such as

x = y; as a single step.

To determine the step count of an algorithm, we first determine the number of steps per execution (s/e) of each statement and the total number of times (i.e., frequency) each statement is executed. Combining these two quantities gives us the total contribution of each statement to the total step count. We then add the contributions of all statements to obtain the step count for the entire algorithm.

The step count method is one of the method to analyze the algorithm. In this method, we count number of times one instruction is executing. From that we will try to find the complexity of the algorithm.

1. Suppose we have one algorithm to perform sequential search. Suppose each instruction will take c1, c2, …. amount of time to execute, then we will try to find out the time complexity of this algorithm

Algorithm	Number of times	Cost
seqSearch(arr, n, key)	1	c1
i := 0	n+1	c2
while i < n, do	n	c3
if arr[i] = key, then	0/1	c4
break		
end if		
done		
return i	1	c5

Now if we add the cost by multiplying the number of times it is executed, (considering the worst case situation), we will get

Cost=c1+(n+1)c 2+nc3+c 4+c 5

Cost=c1+nc 2+c2+nc 3+c 4+c5

Cost=n(c 2+c3)+c 1+c 4+c5

Cost=n(c 2+c3)+C

Considering the c1 + c4 + c5 is C, so the final equation is like straight line y = mx + b. So we can say that the function is linear. The complexity will be O(n).

2. You define what a "step" means for the algorithm (usually statements), then the total of those steps using variables such as N can be calculated.

- First define steps:

```
1  int mean(int a[], size_t n)
2  {
3      int sum = 0;                 // 1 step
4      for (int i = 0; i < n; i++)  // 1 step
5          sum += a[i];             // 1 step
6      return sum;                  // 1 step
7  }
```

- Next determine the frequency of the steps based on N:

```
1  int mean(int a[], size_t n)
2  {
3      int sum = 0;                 // 1 step * 1
4      for (int i = 0; i < n; i++)  // 1 step * (N+1)
5          sum += a[i];             // 1 step * N
6      return sum;                  // 1 step * 1
7  }
```

- Add up the steps: 1 + (N+1) + N + 1
- Reduce: 2N + 3
- Throw away factors that don't grow with N and you're done: O(N)