Algorithm 1.1 Sequential Search
Problem: Is the key *x* in the array *S* of *n* keys?
Inputs (parameters): positive integer *n*, array of keys *S* indexed from 1 to *n*, and a key *x*.
Outputs: *location*, the location of *x* in *S* (0 if *x* is not in *S*).

```
void seqsearch ( int n,
                 const keytype S[ ],
                 keytype x,
                 index & location )
{
    location = 1;
    while (location <= n && S[location] != x)
        location++;
    if (location > n)
        location = 0;
}
```

Algorithm 1.2 Add Array Members
Problem: Add all the numbers in the array *S* of *n* numbers.
Inputs: positive integer *n*, array of numbers *S* indexed from 1 to *n*.
Outputs: *sum*, the sum of the numbers in *S*.

```
number sum ( int n, const keytype S[ ] )
{
    index i;
    number result;

    result = 0;
    for (i = 1;  i <= n;  i++)
        result = result + S [ i ];
    return result;
}
```

basic operation - fundamental operation in the algorithm (i.e., operation done the most)
problem size - input size. For algorithms involving arrays, the problem size is the number of elements. Typically denoted as "n".

1. a) For both of the above algorithms, what are their basic operations?

b) For both of the above algorithms, what are their problem sizes?

c) For both of the above algorithms, we want to derive a function for the number of times that the basic operation is performed relative to the *problem size.*

Algorithm 1.2 Add Array Members

```
number sum ( int n, const keytype S[ ] )
{
    index i;
    number result;
    result = 0;
    for (i = 1;  i <= n;  i++)
        result = result + S [ i ];
    return result;
}
```

Done once. Assume time to do once is $c_1 = 100$

Done n times. Assume time to do loop once is $c_2 = 10$

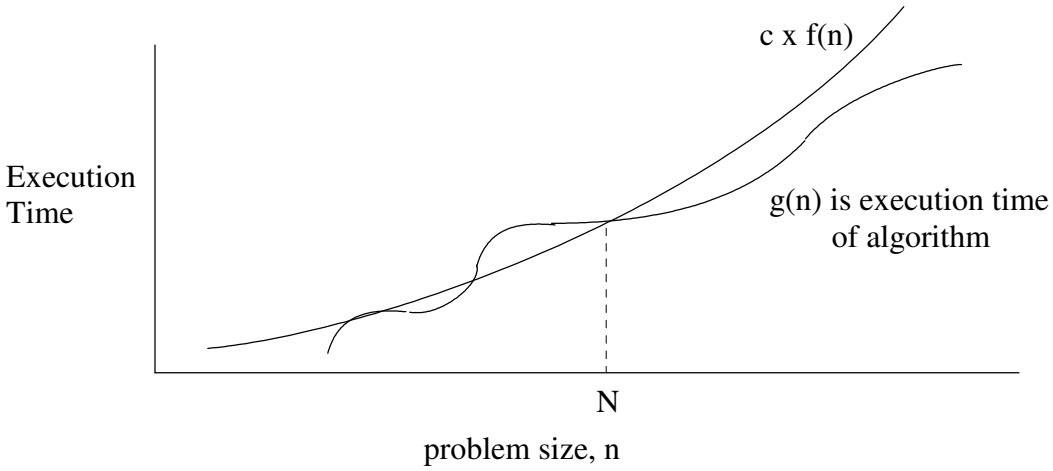
Every-case time complexity analysis, $T(n) = c_1 + c_2 n = 100 + 10 n$

What determines the length of c_1 and c_2?

When n is "sufficiently large," the 10 n term dominates. For what value of n, does 10 n = 100?

Big-oh Definition - asymptotic upper bound
For a given complexity function $f(n)$, $O(f(n))$ is the set of complexity functions $g(n)$ for which there exists some positive real constant c and some nonnegative integer N such that for all $n \geq N$,

$$g(n) \leq c \times f(n).$$



$T(n) = c_1 + c_2 n = 100 + 10 n$ is $O(n)$.

"Proof": Pick c = 110 and N = 1, then $100 + 10 n \leq 110 n$ for all $n \geq 1$.

$100 + 10 n \leq 110 n$

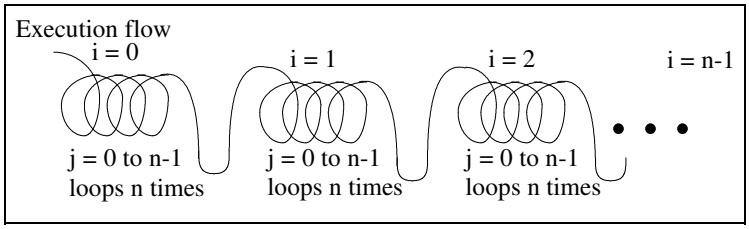
$100 \leq 100 n$

$1 \leq n$

2. Consider the following `someLoops` function.

```
int someLoops(int n) {
    """Returns the sum of values"""
    int total = 0;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
           total = total + i + j;
        } // end for j
    } // end for i
    return total
```



a) What is the basic operation of `someLoops` (i.e., operation done the most) ?

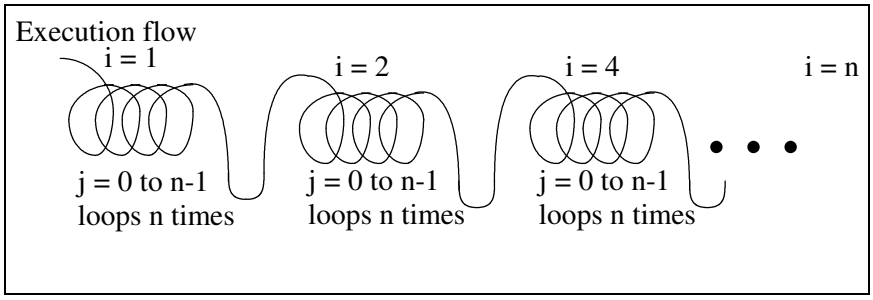
b) How many times will the basic operation execute as a function of `n`?

c) What is the big-oh notation for `someLoops`?

d) If we input n of 10000 and `someLoops` takes 10 seconds, how long would you expect `someLoops` to take for n of 20000?

4. Analyze the below algorithm to determine its big-oh notation, *O*( ).

```
i = 1
while (i <= n) {
   for (j=0; j < n; j++) {
      // something of O(1)
   } // end for
   i = i * 2
} // end while
```

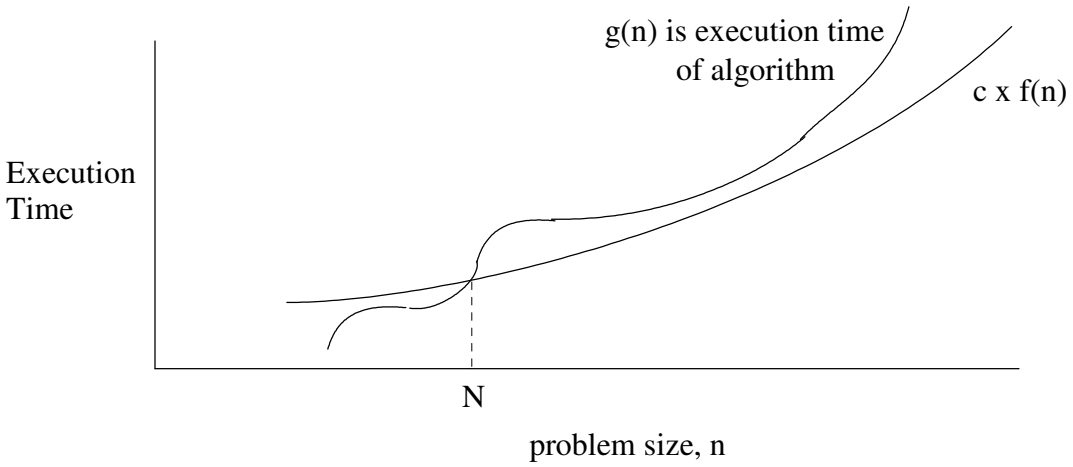


Problem with big-oh:

If T(n) is $O(n)$, then it is also $O(n^2)$, $O(n^3)$, $O(n^3)$, $O(2^n)$, ... since these are also upper bounds.

Omega Definition - asymptotic lower bound
For a given complexity function $f(n)$, $\Omega(f(n))$ is the set of complexity functions $g(n)$ for which there exists some positive real constant c and some nonnegative integer N such that for all $n \geq N$,

$$g(n) \geq c \times f(n).$$



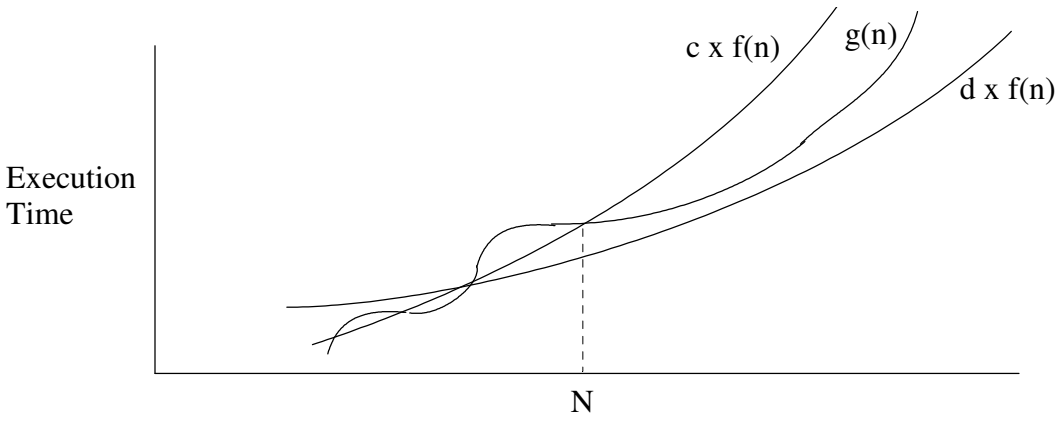
$T(n) = c_1 + c_2 n = 100 + 10 n$ is $\Omega(n)$.

"Proof": We need to find a c and N so that the definition is satisfied, i.e.,
$100 + 10 n \geq c n$ for all $n \geq N$.

What c and N will work?

Theta Definition - asymptotic upper and lower bound, i.e., a "tight" bound or "best" big-oh
For a given complexity function $f(n)$, $\theta(f(n))$ is the set of complexity functions $g(n)$ for which there exists some positive real constants c and d and some nonnegative integer N such that for all $n \geq N$,

$$c \times f(n) \leq g(n) \leq d \times f(n).$$



$T(n) = c_1 + c_2 n = 100 + 10 n$ is $\theta(n)$.