```
int search(int x, int[] A, int n)
//@requires n == \length(A);
/*@ensures (\result == -1  && !is_in(x, A, 0, n))
          || (0 <= \result && \result < n && A[\result] == x);
@*/
{
    for(int i=0; i<n; i++)
    //@loop_invariant 0 <= i && i <= n;
    //@loop_invariant !is_in(x,A,0,i);
    {
        if ( A[i] == x ) {
            return i;
        }
    }
    return -1;
}
```

```
int search ( x, array A, n ){
    int i;
    for ( i=0; i<n; i++ ){
        if ( A[i] == x )
            return i;
    }
    return -1;
}
```

```
$ cc -O search.c0 → ./a.out
$ time ./a.out
```

We want a measure of runtime / memory used / resources used.

① general ← (a) given the code we should be able to compute.

(b) independent of hardware used.

② · usefulness :- helps in distinguishing "good" programs from "bad".

③ mathematically rigorous.

n+3 units of memory

```
int search(x, array A, n){      <- 1, n, 1
    int i;                      <- 1 operation
    for(i=0; i<n; i++){
        if(A[i]==x)
            return i;
    }
    return -1;
}
```

2, 1, 4n, +1

4n+3 ← $x \notin A[0, n)$

4n+3 steps in the worst case

worst case happens when element x is missing from A.

① Our runtime for search in the worst case is 4n+3.

② run time doesn't seem to depend on elements inside A & the value of x

$T(n) = 4n+3$

runtimes with logs "usually" look like

$$RT(n) = \underline{a}\, n + b$$

Comparing running times of two different codes.

$$S(n) = 4n + 3$$

$$B(n) = 100 \log n$$

4n+3 ↑ good.

$B(n)$ is better than $S(n)$

if for $n \geq n_0$ we have $B(n) < S(n)$

ignore runtimes on small inputs.

$T(n) = 4n+3$

$G(n) = 5n+4.$

$T(n)$ & $G(n)$ are equally good.

Deciding when a runtime is better than another.

"T is better than G" if there exists a natural number n_0 & a real no. $c > 0$ s.t. for all $n \geq n_0$; $T(n) \leq c\, G(n)$

Big-O

- "T is better than G"
  ≡ $T \in O(G)$.
  ≡ "T is in big-O of G"

$f \in O(g)$ iff

$\exists\, n_0$ a natural number, a real $c > 0$
s.t. $\forall\, n \geq n_0$: $f(n) \leq c\, g(n)$

$S(n) = 4n+3$.

$S(n) \in O(n)$, $n_0 = 4$, $c = 5$
$(4n+3 \leq 5n)$, $\forall n \geq n_0 = 4$

$4n+3 \notin O(n)$.

exercise:

- for any $f(n) = an+b$
  $f(n) \in O(n)$

ex 2 $f(n) = an^2 + bn + c$. ✓

$f(n) \in O(n^2)$.

ex 3 if $f(n)$ is any polynomial of degree k then

$f(n) \in O(n^k)$.

$O(\log n)$, $O(n)$, $O(n^2)$... $O(n^k)$...

$O(1)$ ← "algorithm" which runs in constant time

f and g are given as inputs.

$f \in O(g)$ if

[$\exists$ n_0 a natural number, a real $c > 0$
s.t. $\forall\ n \geq n_0 : f(n) \leq c\, g(n)$]

find
n_0 , c

$f(n) \leq c\, g(n) \quad \forall n \geq n_0$

$4n+3 \qquad 5n+4$

n_0, c

$4n+3$ $5n+4$

n_0, c .

$4n+3 \leq c\,(5n+4)$

$4n+3 \leq 5cn+4c$

$(4n-5cn) \leq 4c-3$

$(4-5c)n \leq 4c-3$

$c = 2$

$(4-10)n \leq 8-3$

$-6n \leq 5$ ✓

$(4n+3) \leq 2(5n+4)$

$\forall \quad n \geq 0$

$4n+3 \leq 10n+8$

$0 \leq 5+6n$

$c = 2 \quad ; \quad n_0 = 0.$

$f(n) \in c\, g(n).$

$4n+3 \in O(5n+4).$

$=$

$5n+4 \in O(4n+3)$

$f(n) = n$

$g(n) = n^2$.

$n \in O(n^2)$.

bnA.

$n^2 \notin O(n)$.

ex $an + b \in O(n)$.