Consider the algorithms of MERGESORT and QUICKSORT.

QUICKSORT($A$)

```
if |A| > 1 then
        pivot ← A[i]
        partition A into multisets
        L = {elements in A <  pivot}
        E = {elements in A =  pivot}
        G = {elements in A >  pivot}
        QUICKSORT(L)
        QUICKSORT(G)
        A ← L, E, G
fi
```

Let $G(n)$ ="For all arrays $A$ with $n$ elements from a totally ordered domain, if QUICKSORT($A$) is performed, ... unchanged."

We proceed by induction.

Let $n \in \mathbb{N}$ be arbitrary.

Let $A$ be an arbirtrary array with $n$ elements from a totally ordered domain.

**Base Case**

If $n = 0$ or $n = 1$, the test on line 1 fails and thus $A$ is unchanged and vacuously sorted.

**Inductive Step**

Suppose $G(n')$ is true for all $n' \in \mathbb{N}$ with $n' < n$.

The test on line 1 succeeds.

By partitioning, all elements in $L$ are less than all elements in $E$, which are less than all elements in $G$. A multiset of elements in $A$ is the union of the multiset of elements in $L, E, G$.

$A[i] \in E$, so $|L|$ and $|G|$ are less than $|A|$.

By the IH, after QUICKSORT($L$) and QUICKSORT($G$) are performed $L$ and $G$ are sorted in nondecreasing order and the multiset of elements in $L$ and $G$ are unchanged.

After the assignment on line 6, $A$ is sorted in a non-decreasing order. All elements in $L$ are therefore less than all elements in $E$, which in turn are less than all elements in $G$, and thu multiset of elements in $A$ is unchanged. By generalisation, $P(n)$.

By induction, for any $n \in \mathbb{N}.P(n)$.

# 1 Divide and Conquer Algorithms

- divide the problem into smaller parts, often of roughly equal size
- solve each part independently
- combine the solutions for the parts into a solution for the whole problem

## 1.1 Correctness of Iterative Algorithms

$z \leftarrow 0$
$w \leftarrow y$
while $w \neq 0$ do
    $z \leftarrow z + x$
    $w \leftarrow w - 1$
od

What are the values of the variables immediately after iteration $i$ of the loop? ($w = y - i$, $z = ix$)

Let $P(i)$ =“if the loop is executed at least $i$ times, then immediately afer the iteration $i$ we have $w = y - i$ and $z = ix$”.

Note that, by convention, when we talk about the 0th iteration, we are talking about the state immediately before the 1st iteration.

**Lemma 1.1**

Let $x, y \in \mathbb{Z}$. For all $i \in \mathbb{N}$, we have $P(i)$.

**Proof:**

Let $W_i$ and $z_i$ denote the values of $w$ and $z$ immediately after iteration $i$.

**Base Case**

$w_0 = y = y - 0$ by line 2.

$z_0 = 0 = 0 \cdot x$ by line 1,

so $P(0)$ is true.

**Inductive Hypothesis**

Let $i \geq 0$ and assume $P(i)$ is true.

Then $w_i = y - i$ and $z_i = ix$.

From lines 4 and 5, we have that $z_{i+1} = z_i + x$ and $w_{i+1} = w_i - 1$, which by inductive hypothesis means that $z_{i+1} = (i+1)x$ and $w_{i+1} = y - (i+1)$, and hence $P(i+1)$ holds.

By induction, we have that $\forall i \in \mathbb{N}. P(i)$.

**Corollary 1.2**

If the algorithm runs and halts, then at the end $z = xy$.

**Proof:**

Suppose the loop halts immediately after the iteration $i$.

From the termination condition of the loop on line 3 we have that $W_i = 0$. By Lemma 1.1,$w_i = y - i$ and $z_i = ix$, so $i = y$ and $z_i = xy$ .

A **loop invariant** is a predicate that is true each time a particular place in the loop is reached (often the beginning or end).

**Lemma 1.3**

$z = x(y - w)$ is a loop invariant which is true at the beginning and end of every iterations.

**Proof:**

Initially, yrom lines 1 and 2 we can see that $z = 0$ and $w = y$, so $x(y - w) = 0 = z$.

Consider an arbirtrary iteration of the loop.

Let $w'$ and $z'$ denote the values of $w$ and $z$ before the iterations. Let $w''$ and $z''$ denote their variable at the end of the iteration.

Suppose the claim holds at the begining of the iteration so $z' = x(y - w')$.

From lines 4 and 5,

$w'' = w' - 1$ and $z'' = z' + x$, so $x(y - w'') = x(y - (w' - 1)) = x(y - w') + x = z' + x = z''$, so the claim is true at the end of the iterations.

By induction, $z = x(y - 0)$ is true after every iteration.

To prove that an algorithm terminates, we need to show that some nonnegative integral quantity decreases each time through the loop.

**Lemma 1.4**

If $x \in \mathbb{Z}$, $y \in \mathbb{N}$, and the algorithm runs, it eventually halts.

**Proof:**

Before the loop is executed, $w \leftarrow y \in \mathbb{N}$.

In each iteration of the loop, $w$ is decreased by 1, so it is a smaller natural number.

Hence $w$must eventually reach 0, which is the exiting condition of the loop. Thus the loop terminates and the algorithm halts.

**Proof:** [*More Formal*]

Suppose the loop does not terminate.

Let $w_i$ be the value of $w$immediately before the $i$th iteration of the loop.

In each iteration of the loop $w$ is decreased by 1, so $w_{i+1} < w_i$.

Since the loop does not terminate, we have $w_i \neq 0$.

Thus, if $w_i \in \mathbb{N}$, then $w_{i+1} \in \mathbb{N}$.

Also, we know that $w_1 = y \in \mathbb{N}$.

By induction, $w_1, w_2, w_{3,\ldots}$ is a decreasing sequence of natural numbers.

By the Well-Ordering Principle, the set of elements in this sequence has a smallest element.

Suppose that $w_j$ is the smallest element.

But $w_{j+1} < w_j$, which contradicts the assumption that $w_j$ is the smallest element. Thus, the loop eventually terminates.

**Proof:**

Prove by induction that $w_i = y - i$.

Then $w_y = 0$, and so the algorithm terminates.

Now we consider the algorithm of multiplication by shifting and addition.

MULT(m, n)

$x \leftarrow m$
$y \leftarrow n$
$z \leftarrow 0$
while $x \neq 0$ do
    if $x \mod 2 = 1$
        then $z \leftarrow z + y$
            $x \leftarrow x$ div 2
            $y \leftarrow y \cdot 2$
    fi
od
return $z$

Preconditions: $m \in \mathbb{N}$, $n \in \mathbb{R}$

Postconditions: $z = mn$

To derive loop invariants, it is worthwhile to look at examples.

Suppose $m = 11 = 1011_2$, $n = 20 = 10100_2$.

Let $x, y \neq z$ denote the values of $x, y \neq z$ immediately after the iteration $i$.

Therefore,

| $i$ | $x_i$ | $(x_i)_2$ | $x_i \mod 2$ | $y_i$ | $z_i$ |
|---|---|---|---|---|---|
| 0 | 11 | 1011 | 1 | 20 | 0 |
| 1 | 5 | 101 | 1 | 40 | 20 |
| 2 | 2 | 10 | 0 | 80 | 60 |
| 3 | 1 | 1 | 1 | 160 | 60 |
| 4 | 0 | 0 | 0 | 320 | 220 |

Let $m[k-1], \dots, m[0]$ denote the binary representation of $m$:

$$m = \sum_{j=0}^{k-1} m[j]2^j.$$

Thus, $m[j] \in \{0, 1\}$ for $j = 0, \dots, k-1$.

Hence, $x_i = \sum_{i-i}^{k-1} m[j]2^{k-i}$ and $\lfloor m/2 \rfloor = \left\lfloor \sum_{j=0}^{k-1} m[j]2^{j-1} \right\rfloor = \sum_{i=1}^{k-1} m[j]2^{j-1}$.

Note that $z_0 = 0$.

Therefore, $z_{i+1} = z_i + m[i] \cdot n2^i$, and thus

$$\begin{aligned}
z_{i+1} &= z_i + m[i] \cdot n2^i && (1)\\
&= z_{i-1} + m[i-1] \cdot n2^{i-1} + m[i]n2^i && (2)\\
&= n \sum_{i=0}^{i} m[i] \cdot 2^i && (3)\\
&= n(\sum_{j=0}^{k-1} m[j]2^j - \sum_{j=i+1}^{k-1} m[j]2^j) && (4)\\
&= n(m - x_{i+1}2^{i+1}) && (5)\\
&= nm - x_{i+1}y_{i+1} && (6)
\end{aligned}$$

**Corollary 1.5** (Partial Correctness)

Let $m \in \mathbb{N}$ or $n \in \mathbb{R}$.

If the algorithm MULT$(m, n)$ is run and it halts, then, when it halts, $Z = nm$.

**Proof:**

From the termination condition of the loop $x = 0$, so $z = nm - xy$, and by the result already obtained, $z = nm - xy = nm$.