1 Data Structures and Notation

1.1 List

The concept of 'List' here is used to refer to a one-dimensional vector. The following operations are referred to:

$V_1 + +V_2$: refers to the concatenation of two vectors V_1 of dimensions $1xm$ and V_2 of dimensions $1xn$, such that the final vector has the dimensions $1x(m + n)$

$V[i : j]$: refers to a vector composed of elements at indices $i, i+1....j$ from the original vector

$length(V)$: refers to a function that returns the length of the given vector

1.2 Stack

2 State Space and Initial State

2.1 Initial State

Initial state is a tuple whose first element is a list of integers ls, and second element is a *stack* with the tuple $(1, length(ls)$ as its only element.
i.e: $X_0 = (ls, [(1, length(ls)])$

2.2 Final State

The final state of the transition system is when the stack is empty.

3 Output Space

Output space is the same as the input space, necessitating no mapping function between them.

4 Transition Function

Define a sub-function **Partition** as the following, where ++ denotes the list-append operation:

$Partition(L, start, end, pivot)$:

$ls := L[start : end]$

$smaller = \{x|x \in ls, x < pivot\}$

$greater = \{x|x \in ls, x \geq pivot\}$

$\forall i, start \leq i \leq end$:

$L[i] := ls[i]$

return L

The transition function is therefore as follows:

$transition - function(ls, stack)$:

$start, end := stack[0]$

$pivot_index := n, suchthat 1 \leq n \leq length(ls)$

$ls' := Partition(ls, start, end, ls[pivot_index])$

$if(pivot - 1) > start$:

$stack' = [(start, pivot - 1)] + +stack$

$if(pivot + 1) < end$:

$stack' = [(pivot + 1, end)] + +stack$

return $(ls', stack')$

5 Proof of Termination

Since the transition system terminates when the list *stack* is empty, proof of termination equates to a proof that stack will eventually become empty.

At each iteration

As element 2 is a decreasing sequence

As element 1 is an increasing sequence

6 Proof of Correctness

Correctness rests on the invariant that at any state of the system, a pivot element can be chosen from list ls such that:

- Elements before the pivot's index are lesser than pivot
- Elements after the pivot's index are greater than pivot.

6.1 Base Case

When $start = end$, i.e: the list is a singlet list, then invariant holds.

6.2 Inductive Step

Assume $quicksort(ls)$ is correct for a list of length N.

Take $quicksort(ls')$, which is a list of length $N + 1$.

Choosing a pivot from ls' and then calling partition on ls' gives us two sub-lists, one $greater$ than pivot, the other $lesser$ than pivot.

$0 \leq length(greater), length(lesser) \leq N$. Therefore, by assumption, $quicksort(greater)$ and $quicksort(lesser)$ are correct.

$ls' = lesser + +[pivot] + +greater$, where all elements in $greater$ are greater than pivot, and all elements in $lesser$ are lesser than pivot. Therefore, $quicksort(ls')$ is correct if $quicksort(ls)$ is correct.

Thus proven inductively, the invariant holds for all lists of length greater than or equal to 1.

Therefore $quicksort$ system is correct.