1st call: MergeSort$(A, 1, |A|)$ 19 Oct 2022

MergeSort(A, p, r)

```
*  if p<r
     q = ⌊(p+r)/2⌋
     MergeSort(A, p, q)
     MergeSort(A, q+1, r)
     Merge(A, p, q, r)
   endif
```

The decrementing fcn:

Given (A, p, r), let $\mathcal{S}$ denote all states in the execution of MergeSort with inputs (A, p, r).

$$d: \mathcal{S} \longrightarrow \mathbb{N}$$

$$s \longmapsto r-p$$

↳ things accessible to us for the computation of $d(s)$:

- variables: $\underbrace{A, p, r}_{\text{input}}, q$
- implicitly: $i \leftarrow$ recursion depth
- I can define $n = |A|$

Need to show:

→ d is well-defined $(d(s) \in \mathbb{N})$

→ d is decrementing: Recursion: top of recursive calls < top of cur.
Loop: next time at the top of loop, we are strictly less

MERGE(A, p, q, r)

```
n ← q - p + 1
m ← r - q
L ← new array of length n+1 with L[n+1] = ∞
R ← new array of length m+1 with R[m+1] = ∞
for i = 1 to n
|  L[i] ← A[p+i-1]
end for
for j = 1 to m
|  R[j] ← A[q+j]
end for
i ← 1
j ← 1
for k = p to r
    if L[i] ≤ R[j]
    |  A[k] ← L[i]
    |  i++
    else
    |  A[k] ← R[j]
    |  j++
    end if else
```

Decrementing Functions

$d_1 : S \to \mathbb{N}$

$s \mapsto n - i$

$d_2 : S \to \mathbb{N}$

$s \mapsto m - j$

$d_3 : S \to \mathbb{N}$

$s \mapsto r - k$

OR

$d_4 : S \to \mathbb{N}$

$s \mapsto n + m - i - j$

$= (n-i) + (m-j)$

example:

MERGESORT $([6, 8, \pi], 1, 3)$

(*) $d(S) = 2$

1: ✓

2: $q = \lfloor 4/2 \rfloor = 2$

3: MERGESORT $(A, 1, 2)$

4: MERGESORT $(A, 3, 3)$

⋮

compare!

recursive call

MERGESORT $([6, 8, \pi], 1, 2)$

(*) $d(S) = 1$

⋮

another recursion

MERGESORT $([6, 8, \pi], 3, 3)$

(*) $d(S) = 3 - 3 = 0 \in \mathbb{N}$

A = the loop terminates
B = the loop is correct

A, B statements

A TRUE
and

$A \Rightarrow B$

then we know:

B is true!

1. the loop terminates
and
2. if the loop terminates, then the loop is correct

$\therefore$ the loop is correct

need to prove both of these!

Loop (& Recursion) Invariants

Statements: (a sentence that evaluates T/F)

L = the loop invariant

{ L_i = the loop inv. at the start of iteration i }

Q = ~~what the loop is t…~~ post-condition(s)
what should be true at the end of the loop. (ie., what the loop is supposed to accomplish)

P = pre conditions

G = the loop guard (what keeps you in the loop)

The three parts to proving partial correctness of a loop: (If loop ends, then it was correct)

1. Initialization (like the base case in induction)

   Prove $P \Rightarrow L$

2. Maintenance (like I.A + Inductive step in induction)

   $L_i \wedge G \Rightarrow L_{i+1}$

   "if ~~when entering a loop~~ the LI holds, then the next time I am at the top of the loop, the L.I. still holds!

3. End

   $L \wedge \neg G \Rightarrow Q$