Lecture Outline

- Proving the Correctness of Recursive Algorithms.
- Induction hypothesis: the recursive calls are correct.
- Example: Merge Sort.

Proof of Correctness for Recursive Algorithms

- In order to prove recursive algorithms, we have to:
  1. Prove the partial correctness (the fact that the program behaves correctly).
     - We assume that all recursive calls with arguments that satisfy the preconditions behave as described by the specification, and use it to show that the algorithm behaves as specified.
  2. Prove that the program terminates.
     - Any chain of recursive calls eventually ends and all loops, if any, terminate after some finite number of iterations.

Example - Merge Sort

```
MERGE-SORT(A, p, r)
if p < r
    q = (p+r)/2
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q+1, r)
    MERGE(A, p, q, r)
```

Precondition:

- Array A has at least 1 element between indexes p and r (p ≤ r).

Postcondition:

- The elements between indexes p and r are sorted.
Example - Merge Sort

- **MERGE-SORT** calls a function **MERGE(A,p,q,r)** to merge the sorted subarrays of A into a single sorted one.
- The proof of **MERGE** (which is an iterative function) can be done separately, using loop invariants.
- We assume here that **MERGE** has been proved to fulfill its postconditions (can do it as a distinct exercise).

**Correctness proof for Merge-Sort**

- **Number of elements to be sorted**: \( n = r-p+1 \)
  - **Base Case**: \( n = 1 \)
    - \( A \) contains a single element (which is trivially "sorted").
  - **Inductive Hypothesis**:
    - Assume that **MergeSort** correctly sorts \( n=1, 2, \ldots, k \) elements.
  - **Inductive Step**:
    - Show that **MergeSort** correctly sorts \( n = k + 1 \) elements.
    - First recursive call \( n_1=q-p+1=(k+1)/2 \) \( \leq k \) \( \Rightarrow \) subarray \( A[p..q] \) is sorted
    - Second recursive call \( n_2=r-q=(k+1)/2 \) \( \leq k \) \( \Rightarrow \) subarray \( A[q+1..r] \) is sorted
    - \( A, p, q, r \) fulfill now the precondition of **Merge**
    - The postcondition of **Merge** guarantees that the array \( A[p..r] \) is sorted \( \Rightarrow \) postcondition of **MergeSort**

**Correctness proofs for recursive algorithms**

- **Base Case**: Prove that **RECURSIVE** works for \( n = \text{small}_\text{value} \)
- **Inductive Hypothesis (strong induction form)**:
  - Assume that **RECURSIVE** works correctly for \( n=\text{small}_\text{value}, \ldots, k \)
- **Inductive Step**:
  - Show that **RECURSIVE** works correctly for \( n = k + 1 \)

**Lecture (Parts 1, 2, and 3) Summary**

- Proving that an algorithm is totally correct means:
  1. Proving that it will **terminate**
  2. Proving that the list of actions applied to the input (satisfying the precondition) imply the output satisfies the postcondition.
- How to prove repetitive algorithms correct:
  - **Iterative** algorithms: use **Loop invariants**, Induction
  - **Recursive** algorithms: use induction using as hypothesis the recursive call

**Bibliography**

- Weiss, Ch. 1 section on induction.
- Goodrich and Tamassia: Induction and loop invariants; see, e.g., http://www.cs.mun.ca/~kol/courses/2711-w09/Induction.pdf
- Erickson, J. Proof by Induction. Available at: http://jeffe.cs.illinois.edu/teaching/algorithms/notes/98-induction.pdf