Outline

Reasoning on Correctness of Iterative Algorithms

Notes
def power(x, y):
    z = 1
    m = 0

    while m < y:
        z = z * x
        m = m + 1

    return z
Integer Power
Algorithm to compute $x^y$

```python
# Pre: x in R, y in N
def power(x, y):
    z = 1
    m = 0
    # loop pre: x in R, y in N, z = 1, m = 0
    while m < y:
        z = z * x
        m = m + 1
    # loop post: z = x^y
    return z
# Post: returns x^y
```
Some notation
Partial Correctness + Loop Invariants
Partial Correctness + Loop Invariants
Loop Invariant and Postcondition
Proof
Proof
Termination

In general
Termination

Of pow
General loop techniques
Correctness by Design
Consider before, during, after
Correctness by Design
Consider before, during, after
Correctness by Design
Consider before, during, after
Correctness by Design
Consider before, during, after
Correctness by Design
Consider before, during, after
Correctness by Design

Iterative Binary Search

# Pre: A is sorted and x comparable with A[0..n-1]
b = 0
e = n - 1
# LI: 0 <= b <= e+1 <= n and A[0..b-1] < x <= A[e+1..n]
while b <= e:
    m = (b + e) // 2
    if A[m] < x:
        b = m + 1
    else:
        e = m - 1
p = b
# Post: 0 <= p <= n and A[0..p-1] < x <= A[p..n-1]
Termination?
Merge Sort

Previously...

MergeSort(A, b, e):
    if b == e:
        return
    m = (b + e) // 2
    MergeSort(A, b, m)
    MergeSort(A, m+1, e)
    for i = b, ..., e:
        B[i] = A[i]
    c = b
d = m + 1
<snip>
Correctness by Design

Merge loop
Correctness by Design

Merge loop
Correctness by Design

Merge loop
# LI: A[b..i-1] contains B[b..c-1] and B[m+1..d-1], sorted, and A[b..i-1] <= B[c..m], B[d..e]

for i = b, ..., e:
  if d > e or (c <= m and B[c] < B[d]):
    A[i] = B[c]
    c = c + 1
  else:  # d <=e and (c > m or B[c] >= B[d])
    A[i] = B[d]
    d = d + 1
Correct?
Merge Sort

# Pre: 0 <= b <= e < len(A), A[b..e] comparable
MergeSort(A, b, e):
   if b == e: return
   m = (b + e) // 2
   MergeSort(A, b, m)
   MergeSort(A, m+1, e)
# LI: left as an exercise!
for i = b, ..., e: B[i] = A[i]
# loop post: B[b..e] = A[b..e]
c = b
d = m + 1
# LI: A[b..i-1] contains B[b..c-1] and B[m+1..d-1],
# sorted, and A[b..i-1] <= B[c..m], B[d..e]
for i = b, ..., e:
   if d > e or (c <= m and B[c] < B[d]):
      A[i] = B[c]
      c = c + 1
   else: # d <= e and (c > m or B[c] >= B[d])
      A[i] = B[d]
      d = d + 1
# loop post: A[b..e] contains B[b..m] and B[m+1..e], sorted
# Post: A[b..e] sorted (ie. contains same elements as before,
#       reordered so that A[b] <= ... <= A[e])