

Computer Programming

Dr. Deepak B Phatak

Dr. Supratik Chakraborty

Department of Computer Science and Engineering
IIT Bombay

Session: Reasoning about loops

Quick Recap of Relevant Topics



- Sequential and conditional execution of statements
- Iteration/looping constructs
- Solving simple problems with iteration constructs in C++

Overview of This Lecture



- Reasoning about loops as a design and post-design activity
 - Pre-conditions
 - Post-conditions
 - Loop invariants
 - Loop variants

Reasoning About Loops



- Loop: Part of program with an iterative construct
 - Won't distinguish between “**for ...**”, “**while ...**”, “**do ...while ...**” loops for this discussion
- What does a loop compute ?
 - Enter the loop with some relation among variables : **pre-condition**
 - Exit the loop with some relation among variables: **post-condition**
 - Loop incrementally changes variables such that we move pre-condition to post-condition

Recall: Min/Max Example



Given positive integers m and n, find $3^{\min(m, n)}$ and $2^{\max(m, n)}$

```
int minMN = 0, maxMN = 0, threeRaisedMin = 1, twoRaisedMax = 1;  
int m, n, i, j;  
for (i = m, j = n; ((i >= 1) || (j >= 1)); i--, j--) // Iterate max(m, n) times  
{ if ((i >= 1) && (j >= 1)) {  
    minMN++; threeRaisedMin *= 3; // Conditionally iterate min(m,n) times  
}  
    maxMN++; twoRaisedMax *= 2; // Executed max(m, n) times  
}
```

Recall: Min/Max Example

Given positive integers m and n , find $3^{\min(m, n)}$ and $2^{\max(m, n)}$

```

// PRECONDITION: integers m >=1, n >=1, minMN =0, maxMN =0
//                                twoRaisedMax =1, threeRaisedMin =1

for (i = m, j = n; ((i >= 1) || (j >= 1)); i--, j--) // Iterate max(m, n) times
{ if ((i >= 1) && (j >= 1)) {
    minMN++; threeRaisedMin *= 3; // Conditionally iterate min(m,n) times
}
  maxMN++; twoRaisedMax *= 2; // Executed max(m, n) times
}

// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)
//                                twoRaisedMax =2^max(m,n), threeRaisedMin =3^min(m,n)
  
```

Reasoning About Loops

Given positive integers m and n , find $3^{\min(m, n)}$ and $2^{\max(m, n)}$

```
// PRECONDITION: integers m >=1, n >=1, minMN =0, maxMN =0  
//                                twoRaisedMax =1, threeRaisedMin =1
```

Loop incrementally changes variables such that starting from pre-condition, eventually post-condition holds

```
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)  
//                                twoRaisedMax = $2^{\max(m,n)}$ , threeRaisedMin = $3^{\min(m,n)}$ 
```

Designing Loops: A Design Activity



Given positive integers m and n , find $3^{\min(m, n)}$ and $2^{\max(m, n)}$

```
// PRECONDITION: integers m >=1, n >=1, minMN =0, maxMN =0  
//  
for (i = m, j = n; (i >= 1) && (j >= 1); )  
{   if ((i >= 1) && (j >= 1))  
    {      minMN = i * j;  
      maxMN = i + j;  
      twoRaisedMax = 2 * 2;  
      threeRaisedMin = 3 * 3;  
      i++;  
      j++;  
    }  
}  
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)  
//  
//           twoRaisedMax = $2^{\max(m,n)}$ , threeRaisedMin = $3^{\min(m,n)}$ 
```

How do we incrementally change variables to effect transformation from pre-condition to post-condition?

Verifying Loops: A Post-Design Activity



Given positive integers m and n , find $3^{\min(m, n)}$ and $2^{\max(m, n)}$

```
// PRECONDITION: integers m >=1, n >=1, minMN =0, maxMN =0
//                                     twoRaisedMax =1, threeRaisedMin =1
for (i = m, j = n; ((i >= 1) && (j >= 1)) && (twoRaisedMax <= 2max(m,n)) && (threeRaisedMin <= 3min(m,n)); i++, j++)
{ if ((i >= 1) && (j >= 1))
    minMN++;
    maxMN++;
    twoRaisedMax = twoRaisedMax * 2;
    threeRaisedMin = threeRaisedMin * 3;
}
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)
//                                     twoRaisedMax =2max(m,n), threeRaisedMin =3min(m,n)
```

Does this loop really effect transformation from pre-condition to post-condition?

Does this loop really effect transformation from pre-condition to post-condition?

Reasoning About Loops



- Designing and verifying loops requires similar kind of reasoning
- A loop iteratively transforms relations between variables such that
 - **Pre-condition** holds when we start iterating for first time
 - **Post-condition** holds when we exit loop
 - Pre-condition, post-condition special cases of relation that holds invariantly every time we are about to iterate: **loop-invariant**
 - **Desirable:** Integer valued “metric” (e.g. value of counter) monotonically changes towards fixed value : **loop-variant** (ensures loop termination)

Reasoning About Loops

// PRECONDITION: integers $m \geq 1$, $n \geq 1$, $\minMN = 0$ $\maxMN = 0$
 // $\text{twoRaisedMax} = 1$, $\text{threeRaisedMin} = 1$

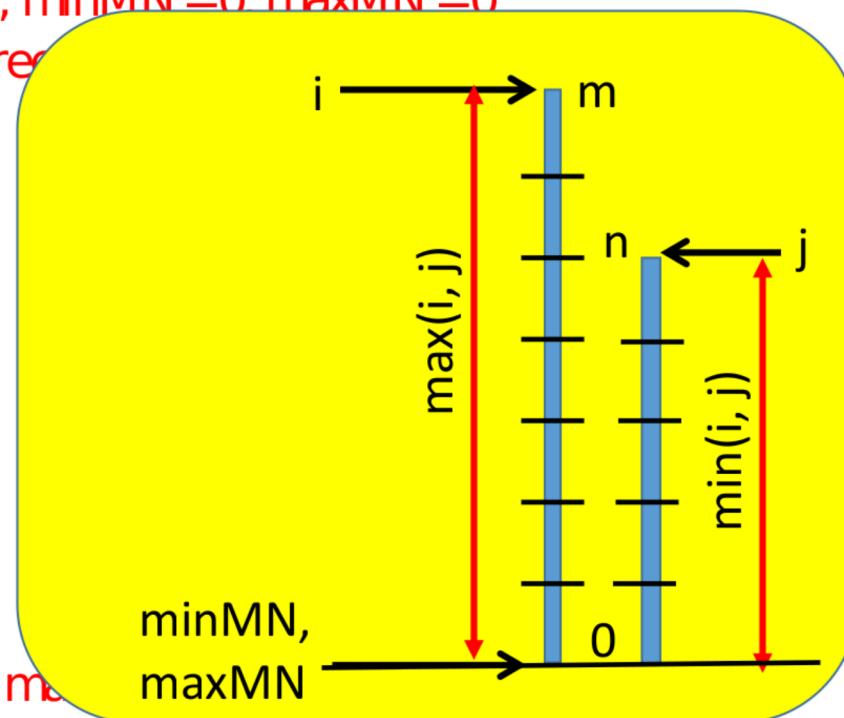
// LOOP INVARIANT:

//

// LOOP VARIANT:

```
for (i = m, j = n; ((i >= 1) || (j >= 1)); i--, j--) {
    if ((i >= 1) && (j >= 1)) {
        minMN++;
        threeRaisedMin *= 3;
    }
    maxMN++;
    twoRaisedMax *= 2;
}
```

// POSTCONDITION: $\minMN = \min(m, n)$, $\maxMN = \max(m, n)$
 // $\text{twoRaisedMax} = 2^{\max(m,n)}$, $\text{threeRaisedMin} = 3^{\min(m,n)}$



Reasoning About Loops

```
// PRECONDITION: integers m >=1, n >=1, minMN =0 maxMN =0
//                                twoRaisedMax =1, threeRaisedMin =1
```

```
// LOOP INVARIANT:
```

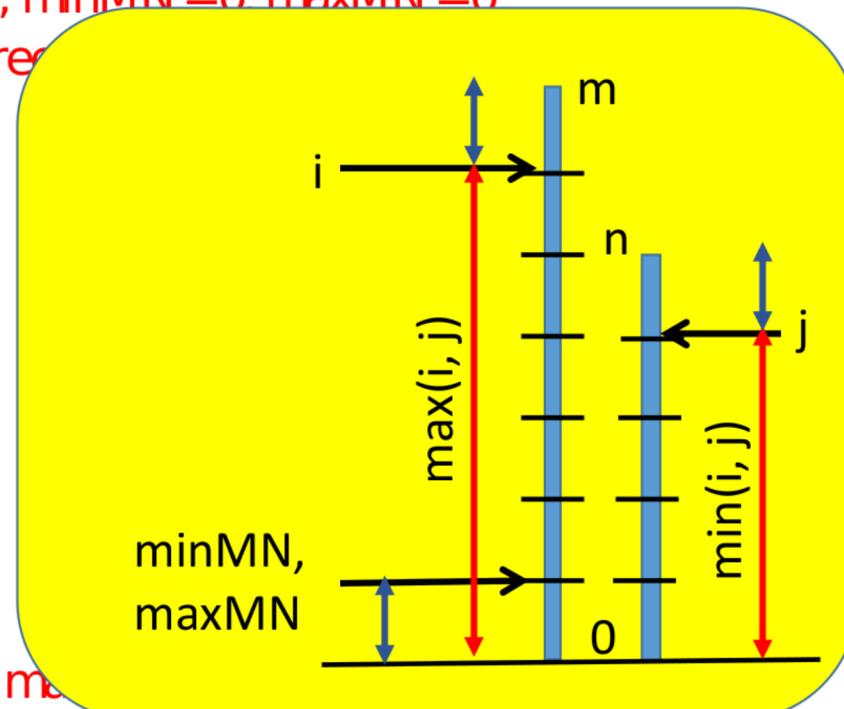
```
//
```

```
// LOOP VARIANT:
```

```
for (i = m, j = n; ((i >= 1) || (j >= 1)); i--, j--)
{ if ((i >= 1) && (j >= 1)) {
    minMN++; threeRaisedMin *= 3;
}
    maxMN++; twoRaisedMax *= 2;
}
```

```
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)
```

```
//                                twoRaisedMax =2max(m,n), threeRaisedMin =3min(m,n)
```



Reasoning About Loops

```
// PRECONDITION: integers m >=1, n >=1, minMN =0 maxMN =0
//                                twoRaisedMax =1, threeRaisedMin =1
```

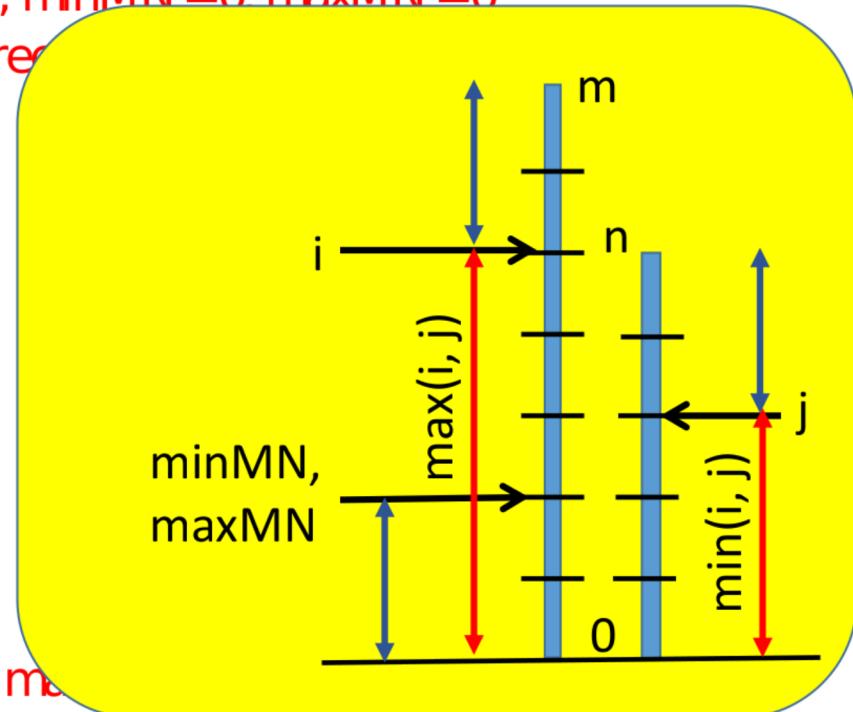
```
// LOOP INVARIANT:
```

```
//
```

```
// LOOP VARIANT:
```

```
for (i = m, j = n; ((i >= 1) || (j >= 1)); i--, j--)
{ if ((i >= 1) && (j >= 1)) {
    minMN++; threeRaisedMin *= 3;
}
    maxMN++; twoRaisedMax *= 2;
}
```

```
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)
//                                twoRaisedMax =2max(m,n), threeRaisedMin =3min(m,n)
```



Reasoning About Loops

```
// PRECONDITION: integers m >=1, n >=1, minMN =0 maxMN =0
//                                twoRaisedMax =1, threeRaisedMin =1
```

```
// LOOP INVARIANT:
```

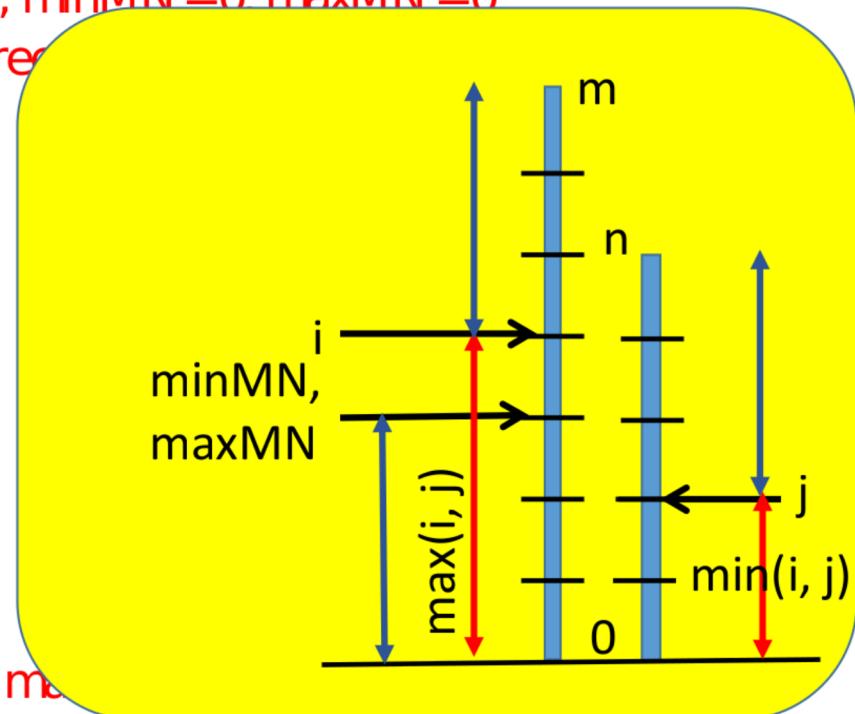
```
//
```

```
// LOOP VARIANT:
```

```
for (i = m, j = n; ((i >= 1) || (j >= 1)); i--, j--)
{ if ((i >= 1) && (j >= 1)) {
    minMN++; threeRaisedMin *= 3;
}
    maxMN++; twoRaisedMax *= 2;
}
```

```
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)
```

```
//                                twoRaisedMax =2max(m,n), threeRaisedMin =3min(m,n)
```



Reasoning About Loops

```
// PRECONDITION: integers m >=1, n >=1, minMN =0 maxMN =0
//                                twoRaisedMax =1, threeRaisedMin =1
```

```
// LOOP INVARIANT:
```

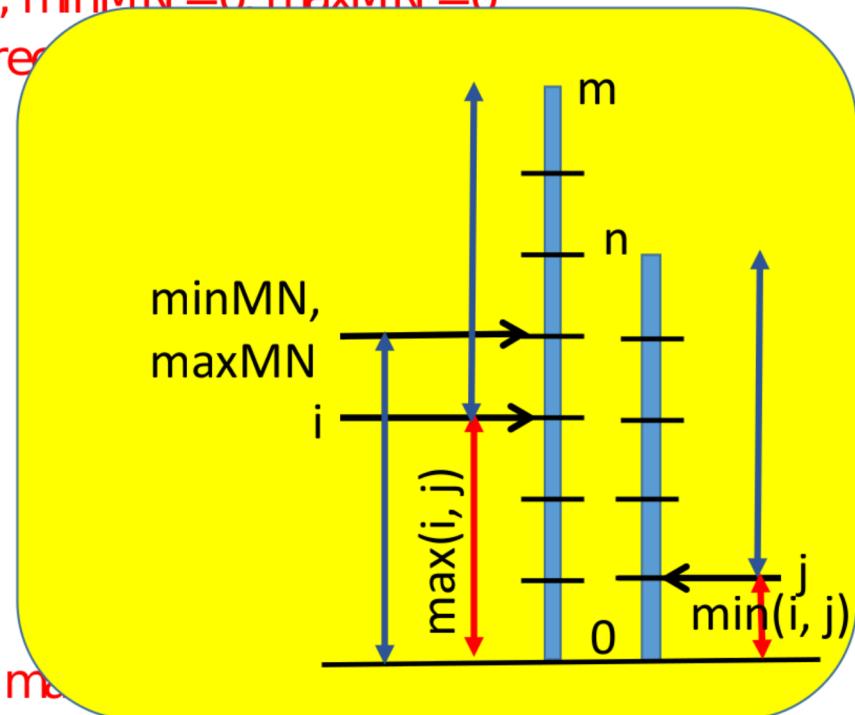
```
//
```

```
// LOOP VARIANT:
```

```
for (i = m, j = n; ((i >= 1) || (j >= 1)); i--, j--)
{ if ((i >= 1) && (j >= 1)) {
    minMN++; threeRaisedMin *= 3;
}
    maxMN++; twoRaisedMax *= 2;
}
```

```
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)
```

```
//                                twoRaisedMax = $2^{\max(m,n)}$ , threeRaisedMin = $3^{\min(m,n)}$ 
```



Reasoning About Loops

```
// PRECONDITION: integers m >=1, n >=1, minMN =0 maxMN =0
//                                twoRaisedMax =1, threeRaisedMin =1
```

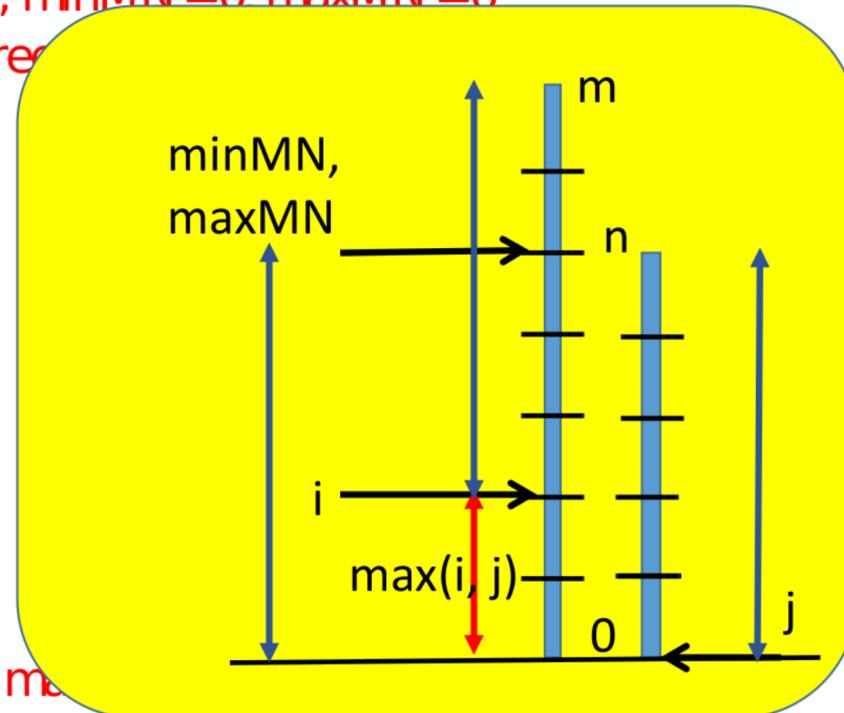
```
// LOOP INVARIANT:
```

```
//
```

```
// LOOP VARIANT:
```

```
for (i = m, j = n; ((i >= 1) || (j >= 1)); i--, j--)
{ if ((i >= 1) && (j >= 1)) {
    minMN++; threeRaisedMin *= 3;
}
    maxMN++; twoRaisedMax *= 2;
}
```

```
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)
//                                twoRaisedMax =2max(m,n), threeRaisedMin =3min(m,n)
```



Reasoning About Loops

```

// PRECONDITION: integers m >=1, n >=1, minMN =0, maxMN =0
//                      twoRaisedMax =1, threeRaisedMin =1
// LOOP INVARIANT: min(i,j) +minMN =min(m, n) when both i, j are >=0
//   When one of i or j becomes zero for first time, 0 + minMN =min(m, n)
// LOOP VARIANT:
for (i = m, j = n; ((i >= 1) || (j >= 1)); i--, j--)
{ if ((i >= 1) && (j >= 1)) {
    minMN++; threeRaisedMin *= 3;
}
    maxMN++; twoRaisedMax *= 2;
}
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)
//                  twoRaisedMax = $2^{\max(m,n)}$ , threeRaisedMin = $3^{\min(m,n)}$ 

```

Reasoning About Loops

// PRECONDITION: integers $m \geq 1, n \geq 1, \minMN = 0, \maxMN = 0$

// $\text{twoRaisedMax} = 1, \text{threeRaisedMin} = 1$

// LOOP INVARIANT: $\min(i, j) + \minMN = m$

//

// LOOP VARIANT:

for ($i = m, j = n; ((i \geq 1) \text{ || } (j \geq 1)); i--, j--$)

{ if $((i \geq 1) \&\& (j \geq 1))$ {

$\minMN++;$ $\text{threeRaisedMin} *= 3;$

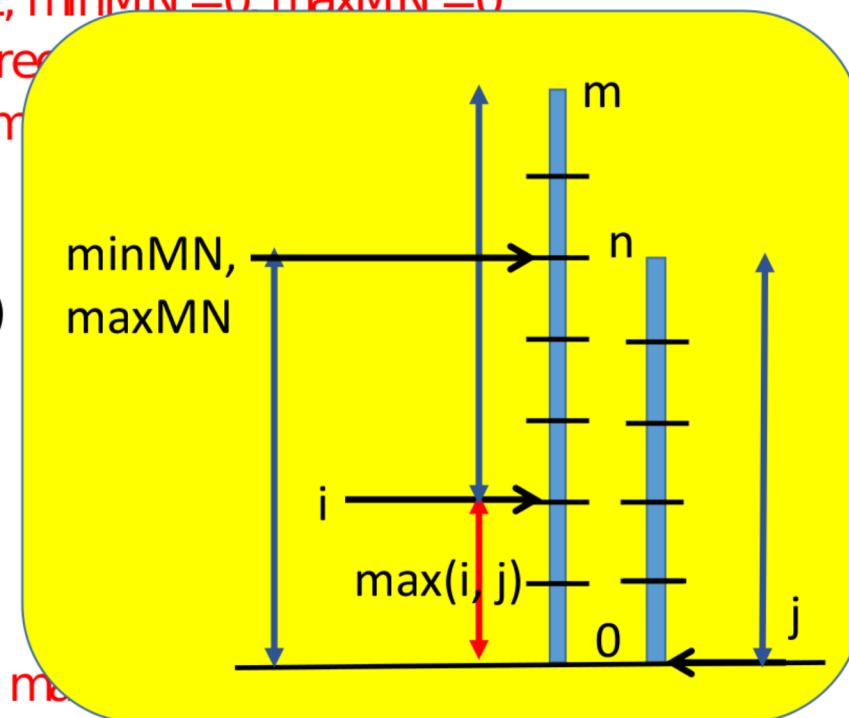
}

$\maxMN++;$ $\text{twoRaisedMax} *= 2;$

}

// POSTCONDITION: $\minMN = \min(m, n), \maxMN = \max(m, n)$

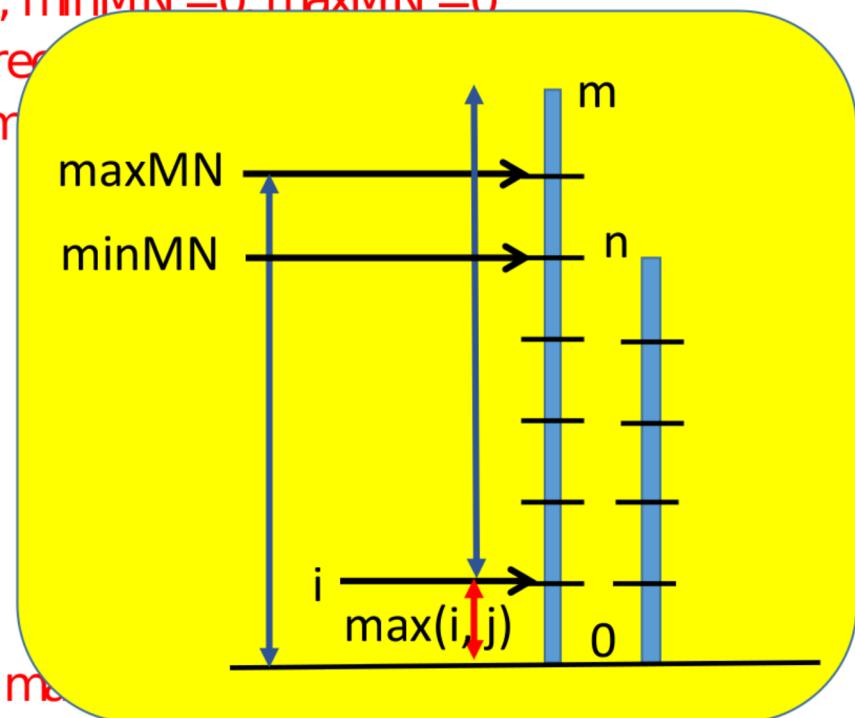
// $\text{twoRaisedMax} = 2^{\max(m,n)}, \text{threeRaisedMin} = 3^{\min(m,n)}$



Reasoning About Loops

```

// PRECONDITION: integers m >=1, n >=1, minMN =0 maxMN =0
//                                twoRaisedMax =1, threeRaisedMin =1
// LOOP INVARIANT: min(i,j) +minMN =m
//                                maxMN +maxMN =n
//                                twoRaisedMax *= 2;
//                                threeRaisedMin *= 3;
// LOOP VARIANT:
for (i = m, j = n; ((i >= 1) || (j >= 1)); i--, j--)
{ if ((i >= 1) && (j >= 1)) {
    minMN++; threeRaisedMin *= 3;
}
    maxMN++; twoRaisedMax *= 2;
}
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)
//                                twoRaisedMax =2max(m,n), threeRaisedMin =3min(m,n)
  
```

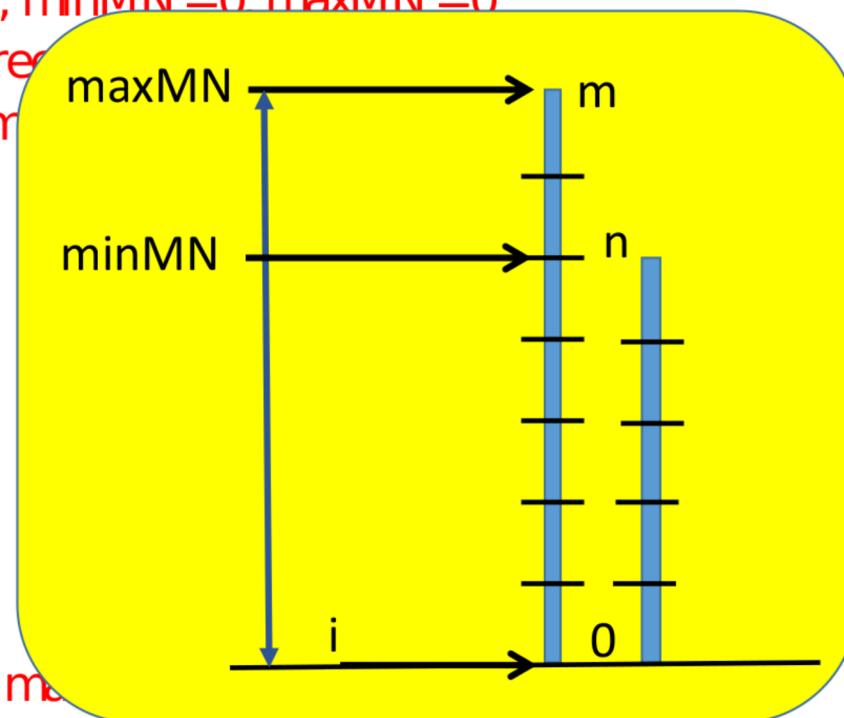


Reasoning About Loops

```

// PRECONDITION: integers m >=1, n >=1, minMN =0 maxMN =0
//                                twoRaisedMax =1, threeRaisedMin =1
// LOOP INVARIANT: min(i,j) +minMN =m, max(i,j) +maxMN =n
// 
// LOOP VARIANT:
for (i = m, j = n; ((i >= 1) || (j >= 1)); i--, j--)
{ if ((i >= 1) && (j >= 1)) {
    minMN++; threeRaisedMin *= 3;
}
    maxMN++; twoRaisedMax *= 2;
}
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)
//                                twoRaisedMax =2max(m,n), threeRaisedMin =3min(m,n)

```



Reasoning About Loops

```

// PRECONDITION: integers m >=1, n >=1, minMN =0, maxMN =0
//                      twoRaisedMax =1, threeRaisedMin =1
// LOOP INVARIANT: min(i,j) +minMN =min(m, n) when both i, j are >=0
//                      max(i,j) +maxMN =max(m, n) when at least one of i, j >=0
// When the last of i or j becomes zero, 0 + maxMN =max(m, n)
for (i = m, j = n; ((i >= 1) || (j >= 1)); i--, j--)
{ if ((i >= 1) && (j >= 1)) {
    minMN++; threeRaisedMin *= 3;
}
    maxMN++; twoRaisedMax *= 2;
}
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)
//                      twoRaisedMax = $2^{\max(m,n)}$ , threeRaisedMin = $3^{\min(m,n)}$ 

```

Reasoning About Loops

```

// PRECONDITION: integers m >=1, n >=1, minMN =0, maxMN =0
//                                twoRaisedMax =1, threeRaisedMin =1
// LOOP INVARIANT: max(0, min(i,j)) +minMN =min(m, n)
//                                max(i,j) +maxMN =max(m, n)
// Taking care of
// "when both i, j >=0"
// LOOP VARIANT:
for (i = m, j = n; ((i >= 1) || (j >= 1)); i--, j--)
{ if ((i >= 1) && (j >= 1)) {
    minMN++; threeRaisedMin *= 3;
}
    maxMN++; twoRaisedMax *= 2;
}
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)
//                                twoRaisedMax =2max(m,n), threeRaisedMin =3min(m,n)

```

Reasoning About Loops

```

// PRECONDITION: integers m >=1, n >=1, minMN =0, maxMN =0
//                      twoRaisedMax =1, threeRaisedMin =1
// LOOP INVARIANT: max(0, min(i,j)) +minMN =min(m, n), threeRaisedMin =3minMN
//                      max(i,j) +maxMN =max(m, n), twoRaisedMax =2maxMN
// LOOP VARIANT:
for (i = m, j = n; ((i >= 1) || (j >= 1)); i--, j--)
{ if ((i >= 1) && (j >= 1))
    minMN++; threeRaisedMin *= 3;
}
maxMN++; twoRaisedMax *= 2;
}
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)
//                      twoRaisedMax =2max(m,n), threeRaisedMin =3min(m,n)
  
```

Reasoning About Loops

```
// PRECONDITION: integers m >=1, n >=1, minMN =0, maxMN =0
//                                twoRaisedMax =1, threeRaisedMin =1
```

```
// LOOP INVARIANT: max(0, min(i,j)) +minMN
//                                max(i,j) +maxMN
```

```
// LOOP VARIANT: max(i, j)
```

```
for (i = m, j = n; ((i >= 1) || (j >= 1)); i--, j--)
{ if ((i >= 1) && (j >= 1)) {
    minMN++; threeRaisedMin *= 3;
}
    maxMN++; twoRaisedMax *= 2;
}
```

```
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)
```

```
//                                twoRaisedMax = $2^{\max(m,n)}$ , threeRaisedMin = $3^{\min(m,n)}$ 
```

Non-negative integer-valued expression that monotonically decreases towards 0 in every iteration

Hence, loop terminates

Reasoning About Loops

```
// PRECONDITION: integers m >=1, n >=1, minMN =0, maxMN =0  
//                      twoRaisedMax =1, threeRaisedMin =1  
// LOOP INVARIANT: max(0, min(i,j)) +minMN =min(m, n)  
//                      max(i,j) +maxMN =max(m, n)  
// LOOP VARIANT: max(i, j)
```

Obtaining right loop invariant/variant after design not always easy,
but crucial to reason about loops

Best practice: Write pre-conditions, post-conditions, loop
invariants, loop variants as comments when programming

```
// POSTCONDITION: minMN =min(m, n), maxMN =max(m, n)  
//                      twoRaisedMax =2max(m,n), threeRaisedMin =3min(m,n)
```

Summary



- Reasoning about loops in programs
- Pre-conditions, post-conditions, loop invariants and loop variants
- Importance of comments