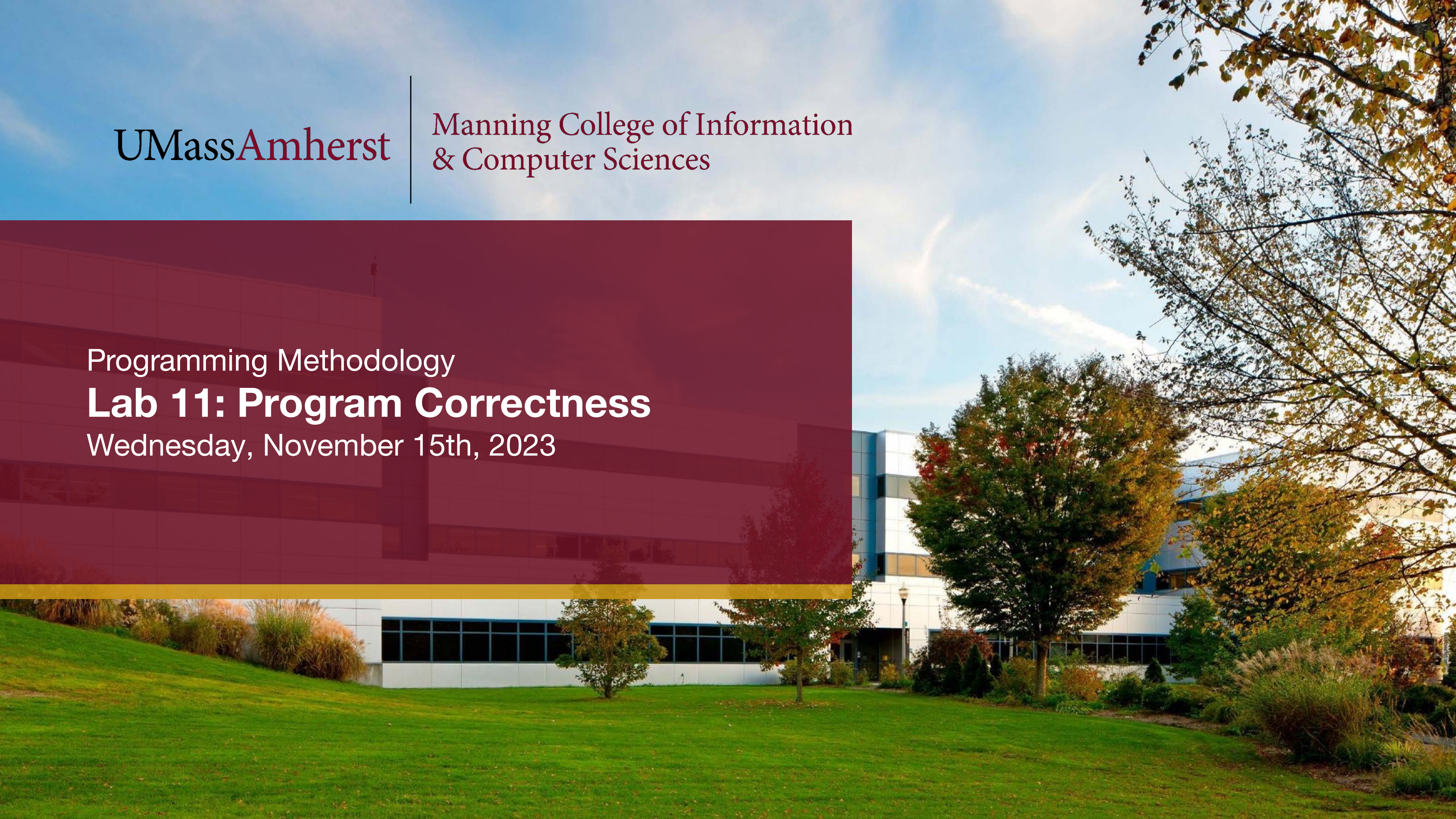
UMassAmherst | Manning College of Information & Computer Sciences

Programming Methodology

Lab 11: Program Correctness

Wednesday, November 15th, 2023



UMassAmherst | Manning College of Information & Computer Sciences

Weekly Lab Agenda

- Go over reminders/goals
- Review past material
- Work in groups of 2-3 to solve a few exercises
    - Lab leaders will assign new groups this week
- Discussion leaders will walk around and answer questions
- Solutions to exercises will be reviewed as a class
- Attendance taken at the end

UMassAmherst | Manning College of Information & Computer Sciences

Reminders

- You should be working on your group project.
- Have a planned timeline to avoid last minute rush!
- Reach out to us if there is any issues in your team.
    - Private post on campuswire or
    - Email to mkuechen@umass.edu

UMassAmherst | Manning College of Information & Computer Sciences

Today’s Goals

- Practice working on program correctness.

Binary Search Invariant

- Recall the case study on binary search we saw in class.
- What will be the invariants in this case:
    - a[lo-1] < x && x < a[hi+1]   // x is not outside searched interval (we don't miss it)
    - (lo === 0 || a[lo-1] < x) && (hi + 1 === a.length || x < a[hi+1]) // ensure indices are valid:

Question 1

UMassAmherst | Manning College of Information & Computer Sciences

The code is for binary search checking middle first.

This code has a bug, which can be identified by observing the invariant.

Use the rules to check if the invariant is maintained, or if there is some path through the code that fails to re-establish it.

```
function bsearchmid (a, target) {
        let lo = 0; let hi = a.length - 1;
        // (lo === 0 || a[lo-1] < target) && (hi + 1 === a.length || target < a[hi+1])
        while (lo <= hi) {
                let mid = lo + Math.floor((hi-lo)/2);
                if (a[mid] == target) {
                        return mid;
                }
                else if (a[mid] < target) {
                        lo = mid;
                } else {
                        hi = mid;
                }
        }
        return -1;
}
```

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
function bsearchmid (a, target) { // assumes a.length > 0 && forall i: 1 <= i < a.length ==> a[i-1] < a[i]
        let lo = 0; let hi = a.length - 1;
        // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1]) -- target is not outside interval
        while (lo <= hi) {
                let mid = lo + Math.floor((hi-lo)/2);
                if (a[mid] == target){  -- target is equal to mid
                        return mid;
                }
                else if (a[mid] < target){
                        lo = mid;
                }
                else {
                        hi = mid;
                }
        }
        return -1;
}
```

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
function bsearchmid (a, target) { // assumes a.length > 0 && forall i: 1 <= i < a.length ==> a[i-1] < a[i]
        let lo = 0; let hi = a.length - 1;
        // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1]) -- target is not outside interval
        while (lo <= hi) {
                let mid = lo + Math.floor((hi-lo)/2);
                if (a[mid] == target){
                        return mid;
                }
                else if (a[mid] < target){  -- target is towards the right
                //  (lo === 0 || a[mid] < target) && (hi === a.length - 1 || target < a[hi+1])
                        lo = mid;
                }
                else {
                        hi = mid;
                }
        }
        return -1;
}
```

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
function bsearchmid (a, target) { // assumes a.length > 0 && forall i: 1 <= i < a.length ==> a[i-1] < a[i]
    let lo = 0; let hi = a.length - 1;
    // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1]) -- target is not outside interval
    while (lo <= hi) {
        let mid = lo + Math.floor((hi-lo)/2);
        if (a[mid] == target){
            return mid;
        }
        else if (a[mid] < target){  -- target is towards the right
        //  (lo === 0 || a[mid] < target) && (hi === a.length - 1 || target < a[hi+1])
            lo = mid;
        // (lo === 0 || a[lo] < target) && (hi === a.length - 1 || target < a[hi+1]) -- substitute assignment
        }
        else {
            hi = mid;
        }
    }
    return -1;
}
```

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
function bsearchmid (a, target) { // assumes a.length > 0 && forall i: 1 <= i < a.length ==> a[i-1] < a[i]
        let lo = 0; let hi = a.length - 1;
        // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1]) -- target is not outside interval
        while (lo <= hi) {
                let mid = lo + Math.floor((hi-lo)/2);
                if (a[mid] == target){
                        return mid;
                }
                else if (a[mid] < target){
                        lo = mid;
                }
                else {    -- negation of if condition // a[mid] > target
                // (lo === 0 || a[lo - 1] < target) && (hi === a.length - 1 || target < a[mid])
                        hi = mid;
                }
        }
        return -1;
}
```

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
function bsearchmid (a, target) { // assumes a.length > 0 && forall i: 1 <= i < a.length ==> a[i-1] < a[i]
        let lo = 0; let hi = a.length - 1;
        // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1]) -- target is not outside interval
        while (lo <= hi) {
                let mid = lo + Math.floor((hi-lo)/2);
                if (a[mid] == target){
                        return mid;
                }
                else if (a[mid] < target){
                        lo = mid;
                }
                else {    -- remaining case, a[mid] > target
                // (lo === 0 || a[lo - 1] < target) && (hi === a.length - 1 || target < a[mid])
                        hi = mid;
                // (lo === 0 || a[lo - 1] < target) && (hi === a.length - 1 || target < a[hi])
                }
        }
        return -1;
}
```

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
function bsearchmid (a, target) { // assumes a.length > 0 && forall i: 1 <= i < a.length ==> a[i-1] < a[i]
    let lo = 0; let hi = a.length - 1;
    // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1]) -- target is not outside interval
    while (lo <= hi) {
        let mid = lo + Math.floor((hi-lo)/2);
        if (a[mid] == target){
            return mid;
        }
        else if (a[mid] < target){
            lo = mid;
        // (lo === 0 || a[lo] < target) && (hi === a.length - 1 || target < a[hi+1]) -- substitute assignment
        }
        else {
            hi = mid;
        // (lo === 0 || a[lo - 1] < target) && (hi === a.length - 1 || target < a[hi])
        }
        // (lo === 0 || a[lo - 1] < target) && (hi === a.length - 1 || target < a[hi])  ---- invariant not reestablished , there is an error
    } // lo === hi
    return -1; // no element found that meets the target
}
```

Solution continued

UMassAmherst | Manning College of Information & Computer Sciences

```
function bsearchmid (a, target) { // assumes a.length > 0 && forall i: 1 <= i < a.length ==> a[i-1] < a[i]
        let lo = 0; let hi = a.length - 1;
        // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1]) -- target is not outside interval
        while (lo <= hi) {
                let mid = lo + Math.floor((hi-lo)/2);
                if (a[mid] == target){
                        return mid;
                }
                else if (a[mid] < target){  -- target is towards the right
                //  (lo === 0 || a[mid] < target) && (hi === a.length - 1 || target < a[hi+1])
                        lo = mid + 1
                }
                else {
                        hi = mid;
                }
        }

        return -1;
}
```

Solution continued

UMassAmherst | Manning College of Information & Computer Sciences

```
function bsearchmid (a, target) { // assumes a.length > 0 && forall i: 1 <= i < a.length ==> a[i-1] < a[i]
        let lo = 0; let hi = a.length - 1;
        // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1]) -- target is not outside interval
        while (lo <= hi) {
                let mid = lo + Math.floor((hi-lo)/2);
                if (a[mid] == target){
                        return mid;
                }
                else if (a[mid] < target){  -- target is towards the right
                //  (lo === 0 || a[mid] < target) && (hi === a.length - 1 || target < a[hi+1])
                        lo = mid + 1
                // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1]) -- substitute assignment
                }
                else {
                        hi = mid;
                }
        }

        return -1;
}
```

Solution continued

UMassAmherst | Manning College of Information & Computer Sciences

```
function bsearchmid (a, target) { // assumes a.length > 0 && forall i: 1 <= i < a.length ==> a[i-1] < a[i]
          let lo = 0; let hi = a.length - 1;
          // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1]) -- target is not outside interval
          while (lo <= hi) {
                    let mid = lo + Math.floor((hi-lo)/2);
                    if (a[mid] == target){
                              return mid;
                    }
                    else if (a[mid] < target){
                              lo = mid + 1
                    }
                    else {    -- negation of if condition // a[mid] > target
                    // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[mid])
                              hi = mid - 1
                    }

          }

          return -1;
}
```

Solution continued

UMassAmherst | Manning College of Information & Computer Sciences

```
function bsearchmid (a, target) { // assumes a.length > 0 && forall i: 1 <= i < a.length ==> a[i-1] < a[i]
        let lo = 0; let hi = a.length - 1;
        // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1]) -- target is not outside interval
        while (lo <= hi) {
                let mid = lo + Math.floor((hi-lo)/2);
                if (a[mid] == target){
                        return mid
                }
                else if (a[mid] < target){
                        lo = mid + 1
                }
                else {    -- negation of if condition // a[mid] > target
                // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[mid])
                        hi = mid - 1
                // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1])
                }

        }

        return -1
}
```

Solution continued

UMassAmherst | Manning College of Information & Computer Sciences

```
function bsearchmid (a, target) { // assumes a.length > 0 && forall i: 1 <= i < a.length ==> a[i-1] < a[i]
        let lo = 0; let hi = a.length - 1;
        // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1]) -- target is not outside interval
        while (lo <= hi) {
                let mid = lo + Math.floor((hi-lo)/2);
                if (a[mid] == target){
                        return mid;
                }
                else if (a[mid] < target){
                        lo = mid + 1
                // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1]) -- substitute assignment
                }
                else {
                        hi = mid - 1
                // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1])
                }
                // (lo === 0 || a[lo-1] < target) && (hi === a.length - 1 || target < a[hi+1])  ---- invariant reestablished
        } // lo === hi

        return -1; // no element found that meets the target
}
```

Question 2

Please write your solution to question 2 on paper. You will submit your work to gradescope before we review the solution, and we'll grade your work to give you feedback.

UMassAmherst | Manning College of Information & Computer Sciences

Question 2

What is the largest value A for which the loop below terminates with n=20?

Use an invariant to justify your answer.

#Fall 2020 Q7

```
let n: number = 0;

let s: number = A;

while (s <= 100) {

          if (n % 2 > 0) {

                    s += n; }

         n = n + 1;

}
```

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
let n: number = 0;
let s: number = A;
 // s = A + [n/2]ˆ2 // [n/2] is integer part, Math.floor(n/2)
while (s <= 100) {
          // s = A + [n/2]ˆ2
          if (n % 2 > 0) {

                    s += n;

          } else {

          }

n = n + 1;


}
```

The loop adds odd numbers. Our invariant is that s is A (initial value) plus the sum of odd numbers less than n.

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
let n: number = 0;
let s: number = A;
 // s = A + [n/2]ˆ2 // [n/2] is integer part, Math.floor(n/2)
while (s <= 100) {
        // s = A + [n/2]ˆ2
        if (n % 2 > 0) {
                // s = A + [n/2]ˆ2 && n = 2*[n/2] + 1
                s += n;

        } else {

        }

n = n + 1;


}
```

The loop adds odd numbers. Our invariant is that s is A (initial value) plus the sum of odd numbers less than n.

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
let n: number = 0;
let s: number = A;
 // s = A + [n/2]ˆ2 // [n/2] is integer part, Math.floor(n/2)
while (s <= 100) {
        // s = A + [n/2]ˆ2
        if (n % 2 > 0) {
                // s = A + [n/2]ˆ2 && n = 2*[n/2] + 1
                s += n;
                // s = A + ([n/2] + 1)ˆ2 && n = 2*[n/2] + 1
        } else {

        }

n = n + 1;


}
```

We can see inductively that 1 + . . . + 2k −1 = k^2 , since k^2 + 2k + 1 = (k + 1)^2, (therefore, s + n = A + [n/2]^2 + 2*[n/2] + 1 = A + ([n/2] + 1) ^ 2) ; this let us handle the update s += n.

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
let n: number = 0;
let s: number = A;
 // s = A + [n/2]ˆ2 // [n/2] is integer part, Math.floor(n/2)
while (s <= 100) {
        // s = A + [n/2]ˆ2
        if (n % 2 > 0) {

                s += n;

        } else {
                // s = A + ([n/2])ˆ2 && n = 2*[n/2]
        }

n = n + 1;


}
```

For the else block since n is even, n = 2*[n/2]

Solution

```
let n: number = 0;
let s: number = A;
 // s = A + [n/2]ˆ2 // [n/2] is integer part, Math.floor(n/2)
while (s <= 100) {
        // s = A + [n/2]ˆ2
        if (n % 2 > 0) {

                s += n;
                // s = A + ([n/2] + 1)ˆ2 && n = 2*[n/2] + 1
        } else {
                // s = A + ([n/2])ˆ2 && n = 2*[n/2]
        }
// s = A + [(n+1)/2]ˆ2 (common for both branches)
n = n + 1;


}
```

Using ⌊n/2⌋ lets us treat even/odd cases and find a common form after the if statement.

Odd case
[n/2] = (n - 1)/2
([n/2] + 1)^2= [(n - 1)/2 + 1]^2
= [(n + 1)/2]^2

Even case
[n/2] = n/2
[n/2]^2 = [(n + 1)/2]^2, because floor

UMassAmherst | Manning College of Information & Computer Sciences

Solution

```
let n: number = 0;
let s: number = A;
 // s = A + [n/2]ˆ2 // [n/2] is integer part, Math.floor(n/2)
while (s <= 100) {
        // s = A + [n/2]ˆ2
        if (n % 2 > 0) {

                s += n;

        } else {

        }
// s = A + [(n+1)/2]ˆ2 (common for both branches)
n = n + 1;
// s = A + [n/2]ˆ2

}
```

Since this involves ⌊(n+1)/2⌋, the assignment rule for n= n + 1 restores the invariant.

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
let n: number = 0;
let s: number = A;
 // s = A + [n/2]ˆ2 // [n/2] is integer part, Math.floor(n/2)
while (s <= 100) {
          // s = A + [n/2]ˆ2
          if (n % 2 > 0) {

                    s += n;

          } else {

          }

n = n + 1;
// s = A + [n/2]ˆ2 // s = A + [20/2]ˆ2 && s > 100 (exit with n=20) ==> A
> 0
} // A + [19/2]ˆ2 <= 100 (enters loop with n=19) ==> A <= 19
```

To find A, we impose both s > 100 for n=20 (loop exit) and s <= 100 for n=19 (execution stays in the loop before); the latter gives us A = 19 as largest value.