Breakout Groups Week 3

# Part One: Discussion

As a group discuss the questions below and document your agreed-upon answers. Complete these questions before moving to the other sections.

## Optimal Path

What is the optimal path to get from room 142 in the 916 building to the entrance of the 600 S. Michigan Ave building? Don't forget that the current entrance is actually not on Michigan - it's around the corner. Write the instructions for a mobile AI that understands common terms such as right, left, door, north, south, etc.

Fill in your group’s answer here:

Go towards room door

If door closed, then open door

Turn and start walking towards the exit of the building

If door closed, then open door

Once on Wabash turn till facing North

Once facing north go forward

If at 4-way road and sign has hand wait

Go if sign at 4-way road has walking sign

Once reaching E. Harrison St Stop

Once on E. Harrison St turn till you are facing east

Once facing east go forward

While still on E. Harrison St stop when reaching entrance to building 600 S. Michigan Ave

If door closed, then open door

Go into building

## Iconic Snowman Elements

What are the essential components of an iconic snowman (such as "Frosty the Snowman ")? Make a list of all the elements.

Snow

1 large snowball

1 medium snowball

1 small snowball

Carrot

Buttons

Sticks

Top Hat

Scarf

# Part Two: Algorithm Creation

Create a set of algorithms based on the discussions you had in part one. These shouldn't be in C# or any other programming language - write them as pseudocode.

## Optimal Path: Room 142 to 600 S. Michigan Ave

Convert your instructions into pseudocode.

Go()

{

-Exit room

-Turn right()

-Walk strai()

-Turn right()

-Walk strai()

-Turn Left()

-Walk strai()

For (int i = 0 ; i < 3; i++)

{

While(Walksafe() == false)  
 {stay()}

-Walk strai()

}

Turn right()

While(walksafe() == false)  
 {stay()}

-Walk strai()

While(walksafe() == false)  
 {stay()}

-Walk strai()

-Turn Right()

-Walk strai()

If(entrance\_on\_left==True)

{

-enter building  
 }

}

Walk strai()

{

Stop at intersection;  
 }

Walksafe()

{

If(walksign == on)

Return True;

{

Continue  
 }

}

Stay()

{  
 }

Turn Right()

{  
 }

Turn Left()

{  
 }

## Snowman Drawing

Write out the steps to draw a snowman. This should be written as if someone has never seen a snowman before, yet by following your steps they will have an accurate rendering of a snowman when finished. Convert the steps into pseudocode.

Draw one big circle

Draw a medium circle above the big one

Draw a small circle above the middle one

Draw sticks coming out of the middle circle

Draw a hat on top of the small circle

Draw a scarf between the medium and small circle

Draw a carrot in the middle of the small circle

Draw buttons in the starting at the center of middle circle

StartDrawClass()

{

DrawLargeSnowBall() {}

DrawmediumSnowBall()

{

PlaceButtonOnEachThird(for loop (3 times))

PlaceStickOnBothSideshoisonally()

}

PlaceScarf;

DrawsmallSnowBall()

{

PlaceCarrot()  
}

PlaceTopHat()

}

# Part Three: Algorithm Analysis

Analyze your two algorithms using big O notation. What are the worst and best runtimes?

Optimal Path

* Worst runtime: O(N2)
* Best runtime: O(1)

Snowman drawing

* Worst runtime: O(N2)
* Best runtime: O(N)

Helpful resources:

* [Asymptotic Notation](https://canvas.colum.edu/courses/6807/pages/2-dot-5-asymptotic-notation?module_item_id=850680)
* [The Idea Behind Big O Notation](https://canvas.colum.edu/courses/6807/pages/2-dot-6-the-idea-behind-big-o-notation?module_item_id=850676)

# Part Four: Advanced Challenge

If you have finished the first three parts of this assignment, here is a challenge.

On a table there are 24 Matryoshka doll sets. Each doll set has a different number of nested dolls within it with the highest nested amount being 50, and the smallest nested amount being 3.

**Task 1:** Create an algorithm that will count how many dolls there are total. What are the worst and best runtimes?

Pseudocode algorithm:

Int dollAmount = 0

Bool doll = false

If doll == true

dollAmount++

print(dollAmount)

Worst runtime: O(N2)

Best runtime: O(N)

**Task 2:** Create an algorithm that will line the dolls up from the one with the smallest nested amount to the largest. What are the worst and best runtimes?

Pseudocode algorithm:

Doll dolls

Doll sorted\_dolls

For int i; i < doll\_amount; i++

If dolls[i].nested\_amount > sorted\_dolls[i].nested\_amount

sorted\_dolls[i+1] = dolls[i]

Worst runtime: O(N2)

Best runtime: O(N)

**Task 3:** One of the dolls has a crack in it. Create an algorithm that searches through all the dolls until it finds the crack and then stops. What are the worst and best runtimes?

Pseudocode algorithm:

Int BrokenDollCount(Doll[24][?])

{

Bool Count = 0;

Bool Broken = 0;

Bool nestednext = true;

Int next = 0;

For(int I = 0; I < 24; I++)

{  
 while(nestednext == true)

{

if(Dolls[i][next] == broken)

{

Broken++;

}

Count++;

if(Dolls[i][next + 1] == null)

{

nestednext = false;

}

}

Next = 0;

Nestednext = false;

}

Return Count, Broken

}

Worst runtime: O(N)

Best runtime: O(1)

If you complete all three tasks, compare your analysis of tasks 1 and 3. Can you make any conclusions based on what you learned?

 Most of the Best Runtime tend to be O(N) while the worst are O(N2) in the algorithm.