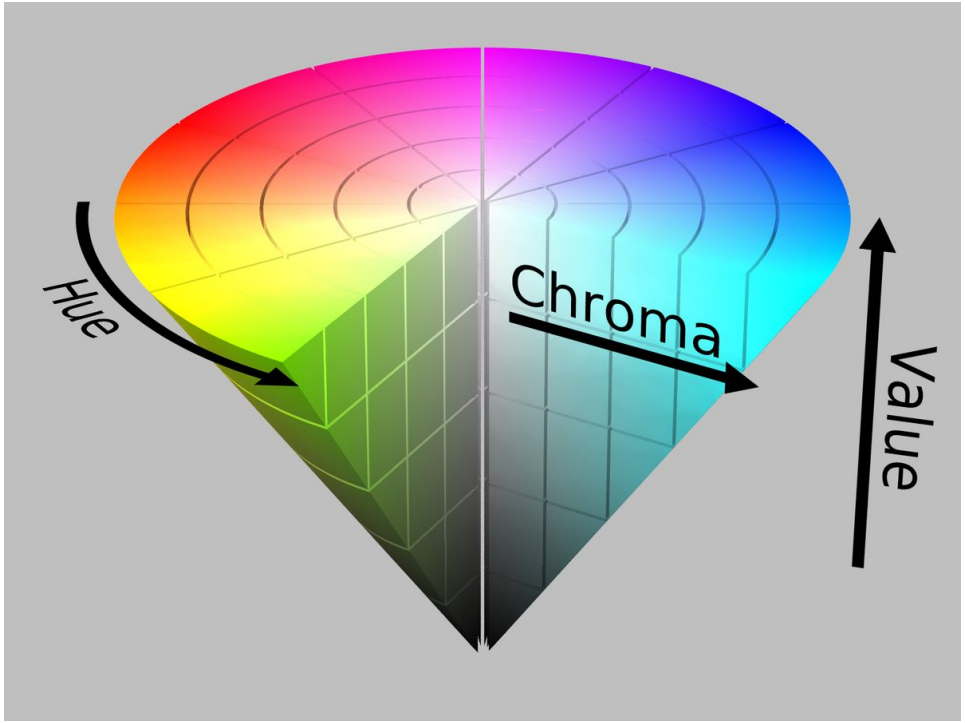
Intro to Parallel and Concurrent Programming

Matrix Multiply
Coarsening

WashU CSE 2301
Prof. Dennis Cosgrove
#04: Tue, Sep 09, 2025

Hue, Saturation (Chroma), Value (Brightness)



```
public class SequentialImageFilter implements ImageFilter {
    @Override
    public HsvImage apply(HsvImage src, PixelFilter pixelFilter) {
        HsvImage dst = new DefaultHsvImage(src.width(), src.height());
        for (int y = 0; y < src.height(); ++y) {
            for (int x = 0; x < src.width(); ++x) {
                Optional<HsvColor> srcPixelOptional = src.colorAtPixel(x, y);
                if (srcPixelOptional.isPresent()) {
                    HsvColor srcPixel = srcPixelOptional.get();
                    HsvColor dstPixel = pixelFilter.apply(srcPixel);
                    dst.setColorAtPixel(x, y, dstPixel);
                } else {
                    throw new Error();
                }
            }
        }
        return dst;
    }
}
```

DesaturateAll

```
public class DesaturateAllPixelFilter implements PixelFilter {
    @Override
    public HsvColor apply(HsvColor src) {
        return new HsvColor(src.hue(), 0.0, src.value());
    }
}
```

DesaturateAll https://newstudents.wustl.edu/

```
public class DesaturateAllPixelFilter implements PixelFilter {
    @Override
    public HsvColor apply(HsvColor src) {
        return new HsvColor(src.hue(), 0.0, src.value());
    }
}
```

DesaturateNonSkinTone newstudents.wustl.edu/

```
public class DesaturateNonSkinTonePixelFilter implements PixelFil
    private static boolean isInRange(double min, double x, double
        return min < x && x < max;
    }

    private static boolean isSkin(HsvColor src) {
        return (src.hue() < 0.1 || src.hue() > 0.9) &&
                isInRange(0.1, src.saturation(), 0.7);
    }

    @Override
    public HsvColor apply(HsvColor src) {
        return new HsvColor(  src.hue(),
                              isSkin(src) ? src.saturation() : 0.0,
                              src.value());
    }
}
```

Race Condition Image Batch Exercise

S&Q: Are the iteratable and range to be used in a future lecture? Since seems a bit unrelated right now

this is fair

in my experience, students need practice with OOP

I have chosen to require everyone to build classes that will be useful for our parallel programs

Powers Of 2 Iterable

- Experience with Iterable/Iterator
- Will use late in semester (for example: Scan)

Fork Loop Warm Up Echoes

```
public class Echoes {
    public static List<String> toEchoes(List<String> texts) {
        throw new NotYetImplementedException();
    }
}

List<String> texts = Arrays.asList("hello", "is there anybody here?", "echo");
List<String> echoes = Echoes.toEchoes(texts);
for (String echo : echoes) {
    System.out.println(echo);
}
```

Fork Loop Warm Up SquareRoots

```
public class SquareRoots {
   public static Double[] toSquareRoots(Double[] values) {
       throw new NotYetImplementedException();
   }
}
```

```
Double[] values = { 1.0, 4.0, 9.0, 16.0 };
Double[] sqrts = SquareRoots.toSquareRoots(values);
for (int i = 0; i < values.length; ++i) {
   System.out.println(sqrts[i] + " " + values[i]);
}
```

ForkLook Images Exercise

Image Filters

pixel filters: all
- ☑ hueSettingPixelFilterOf(_) only
- ☑ hueAdjustingPixelFilterOf(_) only
- ☑ saturationAdjustingPixelFilterOf(_) only
- ☑ valueAdjustingPixelFilterOf(_) only

images: all
- ☑ WASHU_BEAR only
- ☑ MCKELVEY_HALL only
- ☑ JANUARY_110 only

hueSettingPixelFilterOf(0.50)

src

== hueSettingPixelFilterOf(0.50) ==>

SequentialImageFilter

ParallelTaskPerRowImageFilter

src

== hueSettingPixelFilterOf(0.50) ==>

SequentialImageFilter

ParallelTaskPerRowImageFilter

src

== hueSettingPixelFilterOf(0.50) ==>

SequentialImageFilter

ParallelTaskPerRowImageFilter

hueAdjustingPixelFilterOf(0.50)

src

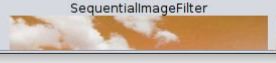
== hueAdjustingPixelFilterOf(0.50) ==>

SequentialImageFilter

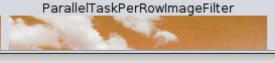
ParallelTaskPerRowImageFilter

src

SequentialImageFilter



ParallelTaskPerRowImageFilter



Code To Investigate

- PixelFilters
- SequentialImageFilter
- DefaultImmutableHsvImage, ImmutableHsvImage, HsvImage

```java
public class PixelFilters {
    private static class HueSettingPixelFilter implements PixelFilter {
        private final double hue;
        public HueSettingPixelFilter(double hue) {
            this.hue = hue;
        }
        @Override
        public HsvColor apply(HsvColor src) {
            double saturation = src.saturation();
            double value = src.value();
            return new HsvColor(hue, saturation, value);
        }
    }
    public static PixelFilter hueSettingPixelFilterOf(double hue) {
        // return new instance of named class HueSettingPixelFilter
        return new HueSettingPixelFilter(hue);
    }
    public static PixelFilter hueAdjustingPixelFilterOf(double deltaHue) {
        // return new instance of anonymous class
        return new PixelFilter() {
            @Override
            public HsvColor apply(HsvColor src) {
                double hue = src.hue();
                double saturation = src.saturation();
                double value = src.value();
                return new HsvColor(hue + deltaHue, saturation, value);
            }
        };
    }
    public static PixelFilter saturationAdjustingPixelFilterOf(double deltaSaturation) {
        // return anonymous function (also known as a lambda) with parameter type
        return (HsvColor src) -> {
            double hue = src.hue();
            double saturation = src.saturation();
            double value = src.value();
            return new HsvColor(hue, saturation + deltaSaturation, value);
        };
    }
    public static PixelFilter valueAdjustingPixelFilterOf(double deltaValue) {
        // return anonymous function (also known as a lambda) without parameter type
        return src -> {
            double hue = src.hue();
            double saturation = src.saturation();
            double value = src.value();
            return new HsvColor(hue, saturation, value + deltaValue);
        };
    }
}
```

ImmutableHsvImage and ImmuHsvImage

```
public interface HsvImage {
   int rowCount();

   List<ImmutableHsvRow> rows();

   ImmutableHsvRow rowAt(int row);

   int columnCount();
}


@Immutable
public interface ImmutableHsvImage extends HsvImage {
}
```

DefaultImmutableHsvImage

```
public final class DefaultImmutableHsvImage implements ImmutableHsvImage {
   private final ImmutableHsvRow[] rows;
   public DefaultImmutableHsvImage(Stream<ImmutableHsvRow> rows) {
      this.rows = rows.toArray(ImmutableHsvRow[]::new);
      if (this.rows.length > 0) {
         int columnCount = this.rows[0].columnCount();
         for (int i = 0; i < this.rows.length; ++i) {
            if (columnCount != this.rows[i].columnCount()) {
               throw new IllegalArgumentException();
            }
         }
      }
   }
   public DefaultImmutableHsvImage(List<ImmutableHsvRow> rows) {
      this(rows.stream());
   }
   @Override
   public int rowCount() {
      return rows.length;
   }
   @Override
   public List<ImmutableHsvRow> rows() {
      return Arrays.asList(rows);
   }
   @Override
   public ImmutableHsvRow rowAt(int row) {
      return rows[row];
   }
   @Override
   public int columnCount() {
      return rows.length > 0 ? rows[0].columnCount() : 0;
   }
}
```

SequentialImageFilter

```
public class SequentialImageFilter implements ImageFilter {
   @Override
   public ImmutableHsvImage apply(ImmutableHsvImage src, PixelFilter pixelFilter) {
      List<ImmutableHsvRow> dstRows = new ArrayList<>(src.rowCount());
      for (ImmutableHsvRow srcRow : src.rows()) {
         List<HsvColor> dstRowPixels = new ArrayList<>(srcRow.columnCount());
         for (HsvColor srcPixel : srcRow.columnColors()) {
            HsvColor dstPixel = pixelFilter.apply(srcPixel);
            dstRowPixels.add(dstPixel);
         }
         dstRows.add(new DefaultImmutableHsvRow(dstRowPixels));
      }
      return new DefaultImmutableHsvImage(dstRows);
   }
}
```

Code To Implement: ParallelTaskPerRowImageFilter

```
public class ParallelTaskPerRowImageFilter implements ImageFilter {
    @Override
    public HsvImage apply(HsvImage src, PixelFilter pixelFilter) {
        throw new NotYetImplementedException();
    }
}
```

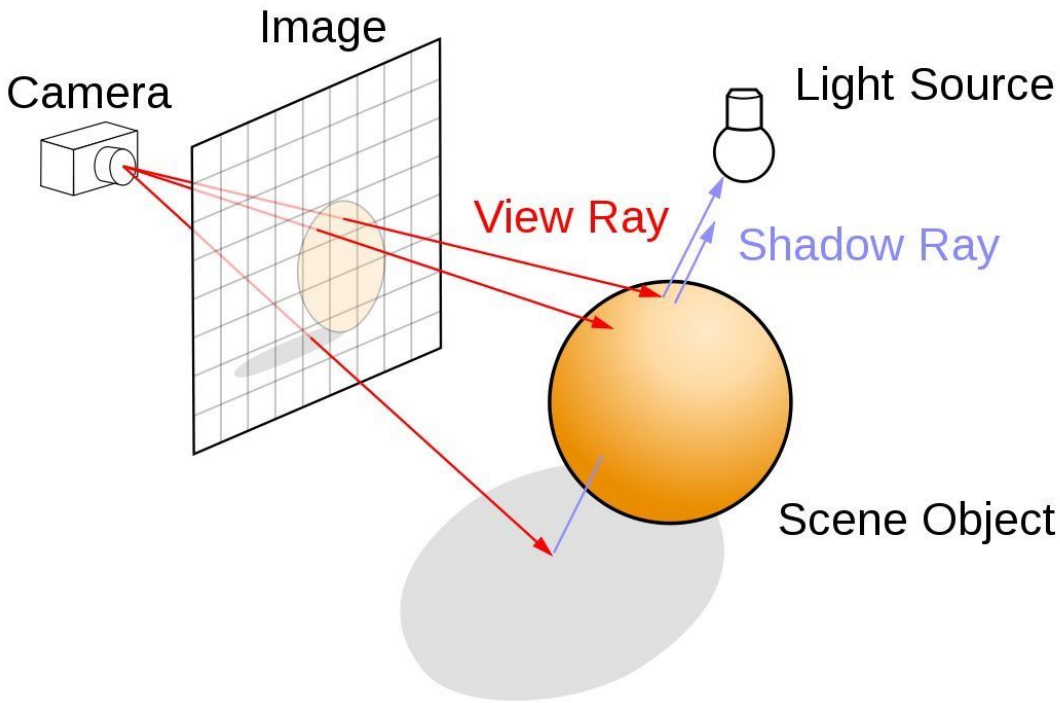
Toy Story

Ultra HD Blu-ray

- 3840 × 2160
- 60 Frames Per Second
- 81 minutes long

81x60x60 = 291600 images

Ray Tracing



You Can Never Do Better Than Log (but that is okay, because log is awesome)

- 81x60x60 = 291600 images (ImageBatch Exercise)
- 3840 × 2160 = 8294400 pixels per image (ImageFilter Exercise)

work = 291600 x 8294400 = $2.41864704 \times 10^{12}$

span = log($2.41864704 \times 10^{12}$) = ~40

ideal parallelism = $2.41864704 \times 10^{12}$ / 40 = ~50,000,000,000

Alert! ParallelTaskPerRowImageFilter

```
public class ParallelTaskPerRowImageFilter implements ImageFilter {
    @Override
    public HsvImage apply(HsvImage src, PixelFilter pixelFilter) {
        throw new NotYetImplementedException();
    }
}
```

Do NOT make a task per pixel.

Intro to Parallel and Concurrent Programming

Matrix Multiply
Coarsening

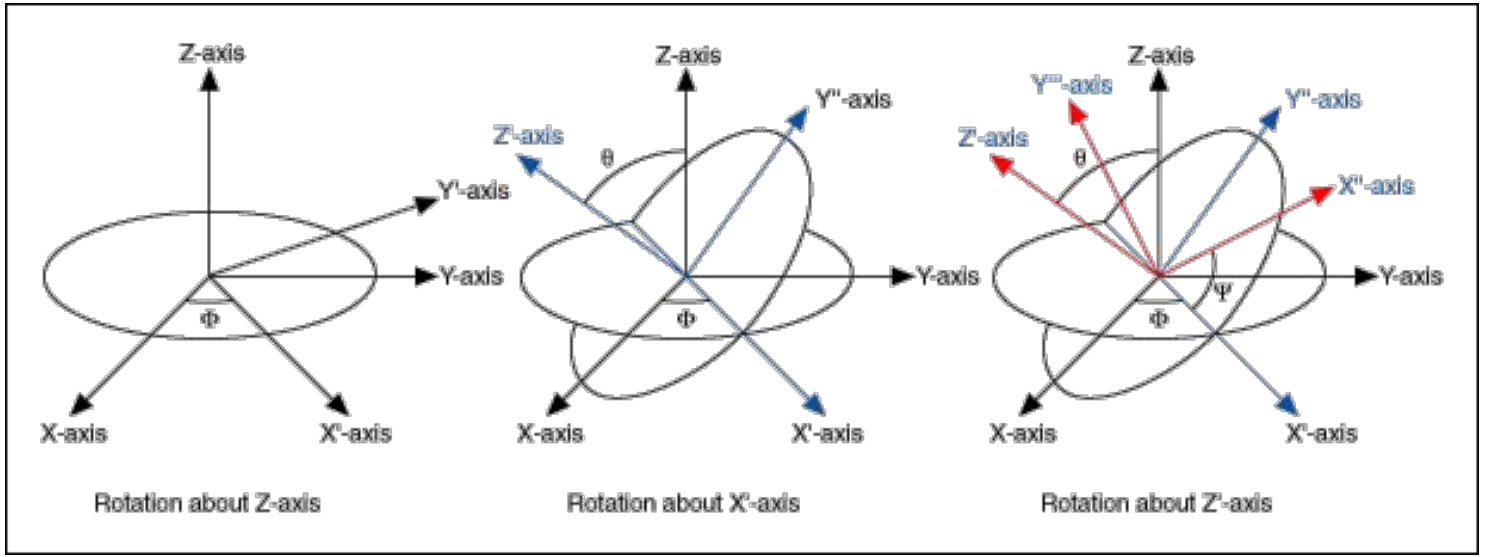
WashU CSE 2301
Prof. Dennis Cosgrove
#04: Tue, Sep 09, 2025

S&Q: Is this the main reason/purpose behind matrix multiplication?

- Linear Algebra storytime

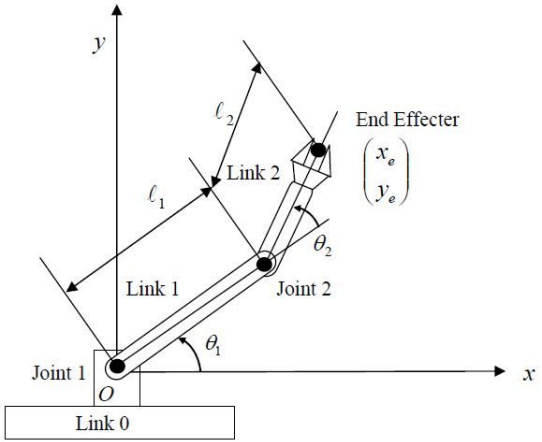
S&Q: Is this the main reason/purpose behind matrix multiplication?

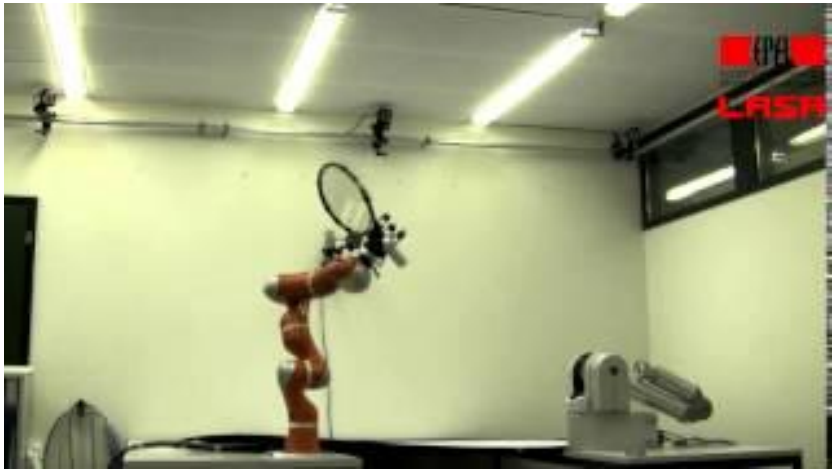
- https://vitaminac.github.io/Matrices-in-Computer-Graphics/

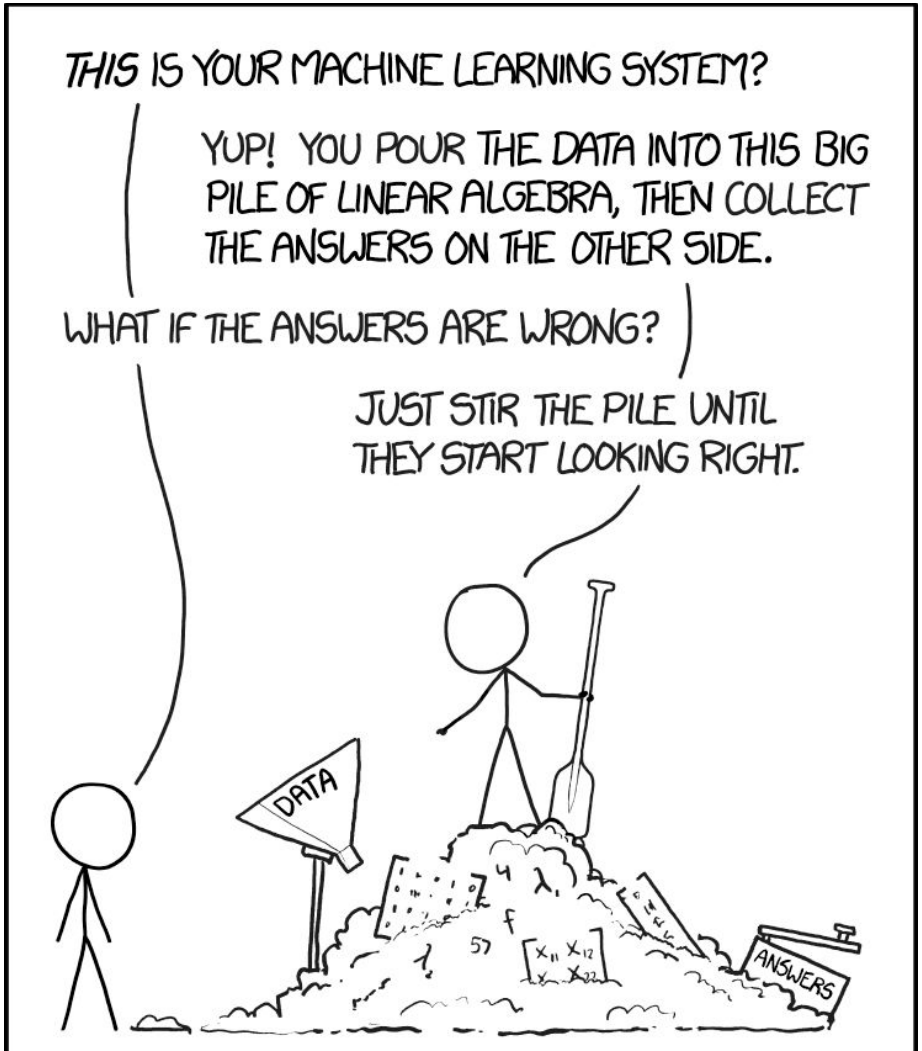


S&Q: Is this the main reason/purpose behind matrix multiplication?

- Robot Catching Objects



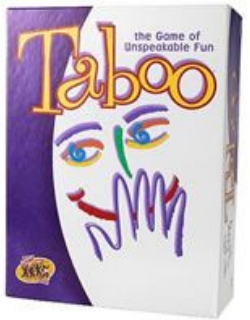



THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Worksheet

- back page info: 2X-10X tasks per processor is often reasonable
- back page clarification: how many fork calls and join calls would execute?

today's consensus builder: person who most recently played
tie breaker: biggest Sade Fan
bonus tie breaker race: last person to press tab, o, o



1000x1000, join_void_fork_loop

- Is this appropriate if you had 8 available processors? Yes / No
- What if you had 100 available processors? Yes / No

```
join_void_fork_loop(0, n, (i) -> {
      for (int j = 0; j < m; j++) {
            for (int k = 0; k < p; k++) {
                  result[i][j] += a[i][k] * b[k][j];
            }
      }
});
```

1000x1000, join_void_fork_loop X 2

- Is this appropriate if you had 8 available processors? Yes / No
- What if you had 100,000 available processors? Yes / No

```
join_void_fork_loop(0, n, (i) -> {
      join_void_fork_loop(0, m, (j) -> {
            for (int k = 0; k < p; k++) {
                  result[i][j] += a[i][k] * b[k][j];
            }
      });
});
```

1000x1000, join_void_fork_loop X 3

- Is this appropriate if you had 8 available processors? Yes / No
- What if you had 100,000,000 available processors? Yes / No

```
join_void_fork_loop(0, n, (i) -> {
      join_void_fork_loop(0, m, (j) -> {
            join_void_fork_loop(0, p, (k) -> {
                  result[i][j] += a[i][k] * b[k][j];
            });
      });
});
```

1000x1000, join_void_fork_loop X 3

- Is this appropriate if you had 8 available processors? Yes / No
- What if you had 100,000,000 available processors? Yes / No

```
seq for(0, n, (i) -> {
      seq for(0, m, (j) -> {
            join_void_fork_loop(0, p, (k) -> {
                  result[i][j] += a[i][k] * b[k][j];
            });
      });
});
```

join_void_fork_loop

- How many times will doSomething1d be invoked? _____
- How many times will fork be invoked? _____
- How many times will join be invoked? _____

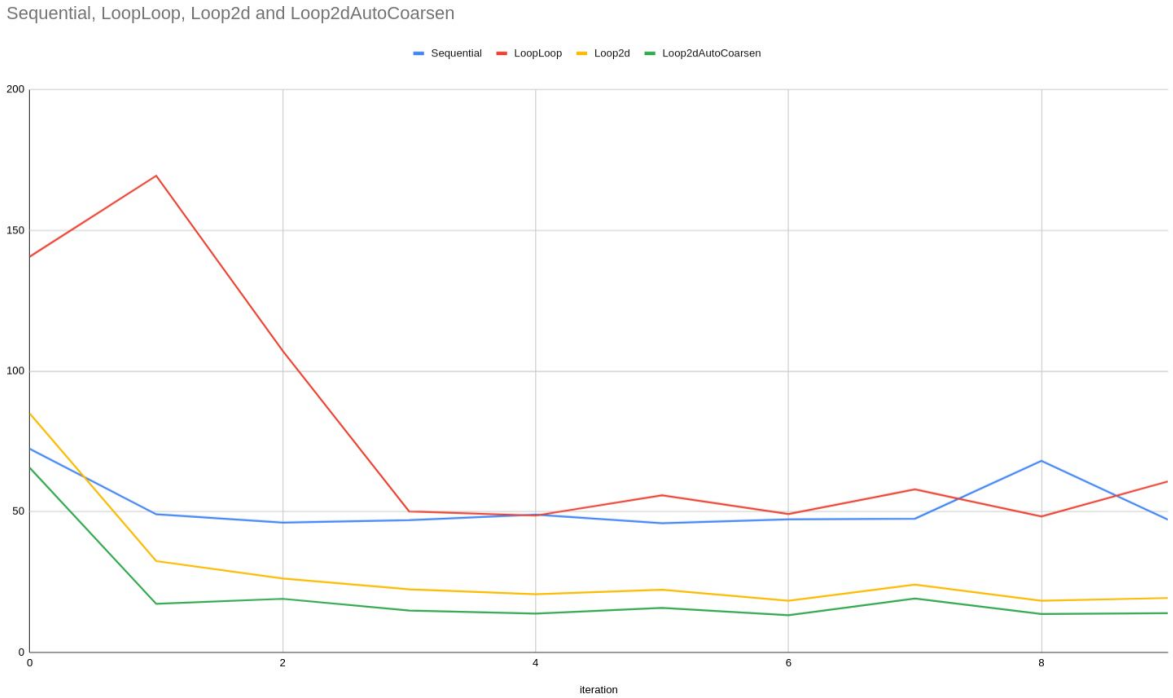
```
join_void_fork_loop(0, n, (i) -> {

      doSomething1d(i);

});
```

join_void_fork_loop, join_void_fork_loop

- How many times will doSomething2d be invoked? _____ N*M
- How many times will fork be invoked? _____ N*M + N
- How many times will join be invoked? _____

```
join_void_fork_loop(0, n, (i) -> {

      join_void_fork_loop(0, m, (j) -> {

            doSomething2d(i, j);

      });

});
```

Performance



join_void_fork_loop_2d

join_void_fork_loop_2d

```
public static void join_void_fork_loop_2d(int aMin,
                                           int aMaxExclusive,
                                           int bMin,
                                           int bMaxExclusive,
                                           TaskBiIntConsumer biConsumer)
```

Exercises: One Week Extension

Many, many policies have been discussed.

Pull, Commit, Push, Check Bitbucket.

3 Free Misses For S&Q Prep
6 Free Misses For Worksheets

You can always get a C-

ALERT: join_fork_loop returning array change

```
public static Double[] toSquareRoots(Double[] values) {
    return join_fork_loop(Double[]::new, values, (value) -> {
        return Math.sqrt(value);
    });
}

public static Double[] toSquareRoots(Double[] values) {
    return join_fork_loop((length)-> {
        return new Double[length];
    }, values, (value) -> {
        return Math.sqrt(value);
    });
}
```

S&Q: For things like for loop vs. for each loop and passing indices into join_fork_loop vs. passing an iterable, is there a significant difference in runtime, or is the difference mainly in code readability?

readability/maintainability is the primary motivation

S&Q: What sorts of criteria should we use for determining the "right tool for the right job" when some people would argue one way is better than the other and others would argue the exact opposite based on their personal preference?

- I used the for-each loop before I was a prof.
- since becoming a prof. and seeing a lot of buggy student code, I have become more passionate about evangelizing for the for-each loop.

```
for(int i=0; i<getPoints().size(); ++i) {
    getPoints().get(i).doSomething();
}
```

vs

```
for(Point pt : getPoints()) {
    pt.doSomething()
}
```

S&Q: Is there any new notation we use to denote optimization due to parallelizing programs or still just big-O?

yes. coming soon.

span/critical path length

speed up

amdahl's law

S&Q: Can we do an exercise on this?

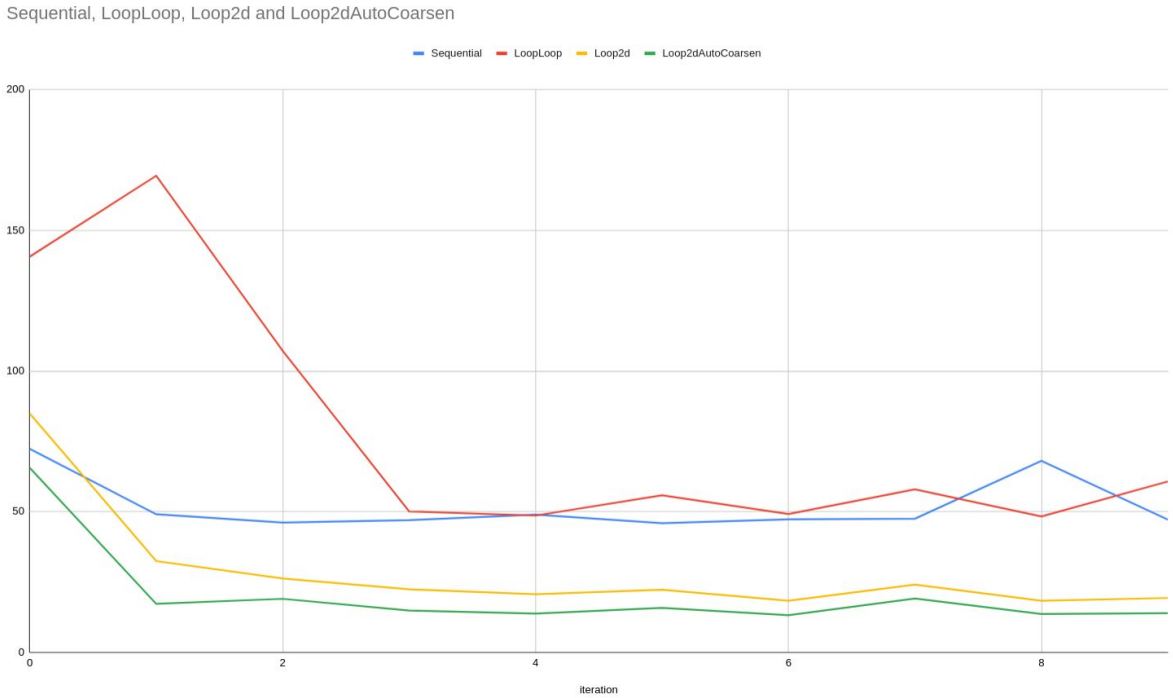
Matrix Multiply Exercise

https://classes.engineering.wustl.edu/cse231/core/index.php/MatrixMultiply

- WarmUp
  - SequentialMatrixMultiplier
- Exercise
  - ForallForallMatrixMultiplier
  - Forall2dMatrixMultiplier
  - Forall2dChunkedMatrixMultiplier

S&Q: What size does a matrix need to be for the overhead of spawning threads to be worthwhile?

Performance



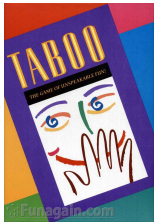
join_void_fork_loop_2d_auto_coarsen

join_void_fork_loop_2d_auto_coarsen

```
public static void join_void_fork_loop_2d_auto_coarsen(int aMin,
                                                       int aMaxExclusive,
                                                       int bMin,
                                                       int bMaxExclusive,
                                                       TaskBiIntConsumer biConsumer)
```

S&Q: For multiplying matrices, is the best way like the method in the demo? With nested for-loops?

Matrix Multiplication Storytime



http://www.greatgroupgames.com/fishbowl.htm

Phase 1) taboo without limits

Phase 2) password

Phase 3) charades

Matrix Multiplication Storytime

Massachusetts Institute of Technology

http://www.greatgroupgames.com/fishbowl.htm

Phase 1) taboo

Phase 2) password

Phase 3) charades

Iterative vs. Recursive

```
Input: matrices A and B
Let C be a new appropriate size
matrix
For i from 1 to n:
    For j from 1 to p:
        Let sum = 0
        For k from 1 to m:
            Set sum ← sum + A_ik × B_kj
        Set C_ij ← sum
Return C
```

https://en.wikipedia.org/wiki/Matrix_multiplication_algorithm

MMult(A, B, n)

1. If $n = 1$ Output $A \times B$
2. Else
3. Compute $A^{11}, B^{11}, \ldots, A^{22}, B^{22}$ % by computing $m = n/2$
4. $X_1 \leftarrow MMult(A^{11}, B^{11}, n/2)$
5. $X_2 \leftarrow MMult(A^{12}, B^{21}, n/2)$
6. $X_3 \leftarrow MMult(A^{11}, B^{12}, n/2)$
7. $X_4 \leftarrow MMult(A^{12}, B^{22}, n/2)$
8. $X_5 \leftarrow MMult(A^{21}, B^{11}, n/2)$
9. $X_6 \leftarrow MMult(A^{22}, B^{21}, n/2)$
10. $X_7 \leftarrow MMult(A^{21}, B^{12}, n/2)$
11. $X_8 \leftarrow MMult(A^{22}, B^{22}, n/2)$
12. $C^{11} \leftarrow X_1 + X_2$
13. $C^{12} \leftarrow X_3 + X_4$
14. $C^{21} \leftarrow X_5 + X_6$
15. $C^{22} \leftarrow X_7 + X_8$
16. Output C
17. End If

even more fun:
https://en.wikipedia.org/wiki/Strassen_algorithm
http://www.cs.mcgill.ca/~pnguyen/251F09/matrix-mult.pdf

Matrix Multiply Divide and Conquer Challenges

https://classes.engineering.wustl.edu/cse231/core/index.php?title=MatrixMultiply#Optional_Divide_and_Conquer_Challenges

- fun
  - SequentialDivideAndConquerMatrixMultiplier
  - ParallelDivideAndConquerMatrixMultiplier

note: not required. just for fun.

Matrix Multiply Divide and Conquer Challenges

https://classes.engineering.wustl.edu/cse231/core/index.php?title=MatrixMultiply#Optional_Divide_and_Conquer_Challenges

- fun
  - SequentialDivideAndConquerMatrixMultiplier
  - ParallelDivideAndConquerMatrixMultiplier

note: not required. just for fun.

S&Q: Is there any way to get better than a cubic algorithm for matrix multiplication?

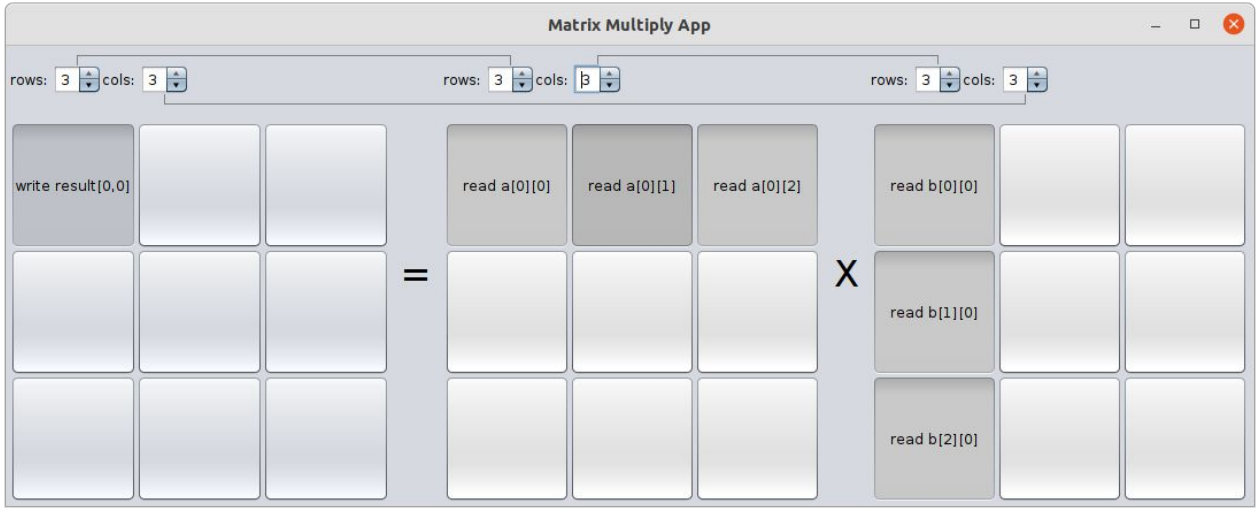
No, but you can get to 7/8ths the work: [Strassen Algorithm]

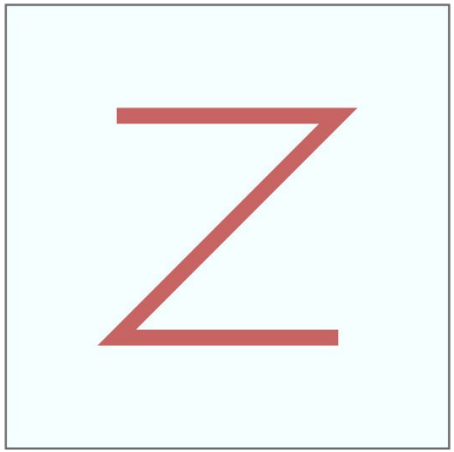
$$
\begin{aligned}
M_1 &= (A_{11} + A_{22})(B_{11} + B_{22});\\
M_2 &= (A_{21} + A_{22})B_{11};\\
M_3 &= A_{11}(B_{12} - B_{22});\\
M_4 &= A_{22}(B_{21} - B_{11});\\
M_5 &= (A_{11} + A_{12})B_{22};\\
M_6 &= (A_{21} - A_{11})(B_{11} + B_{12});\\
M_7 &= (A_{12} - A_{22})(B_{21} + B_{22}),
\end{aligned}
$$

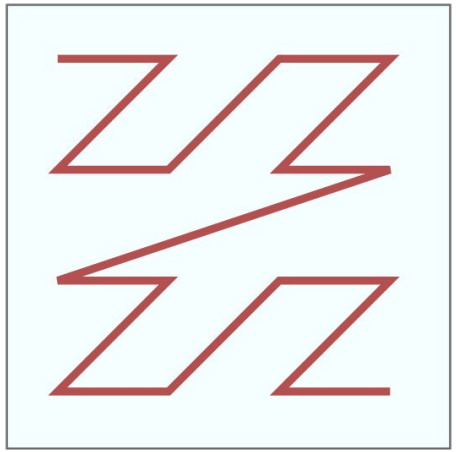
$$
\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{bmatrix}.
$$

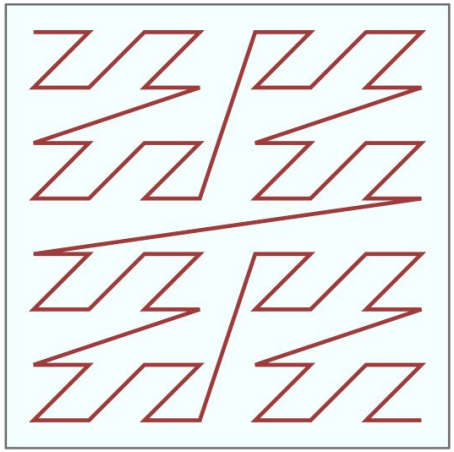
Fist to Five: Cache

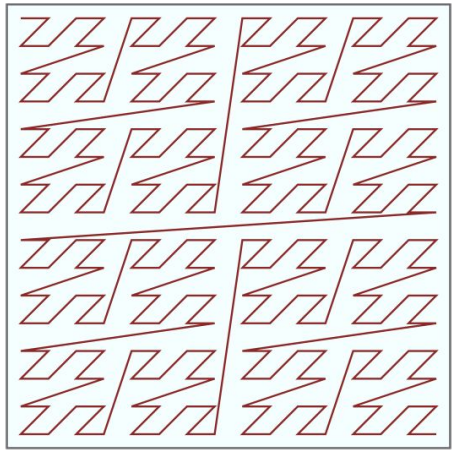
S&Q: Is it more advisable to split the tasks along the rows or the columns when using parallelism to multiply matrices ? Or is there even a better way to split the tasks?



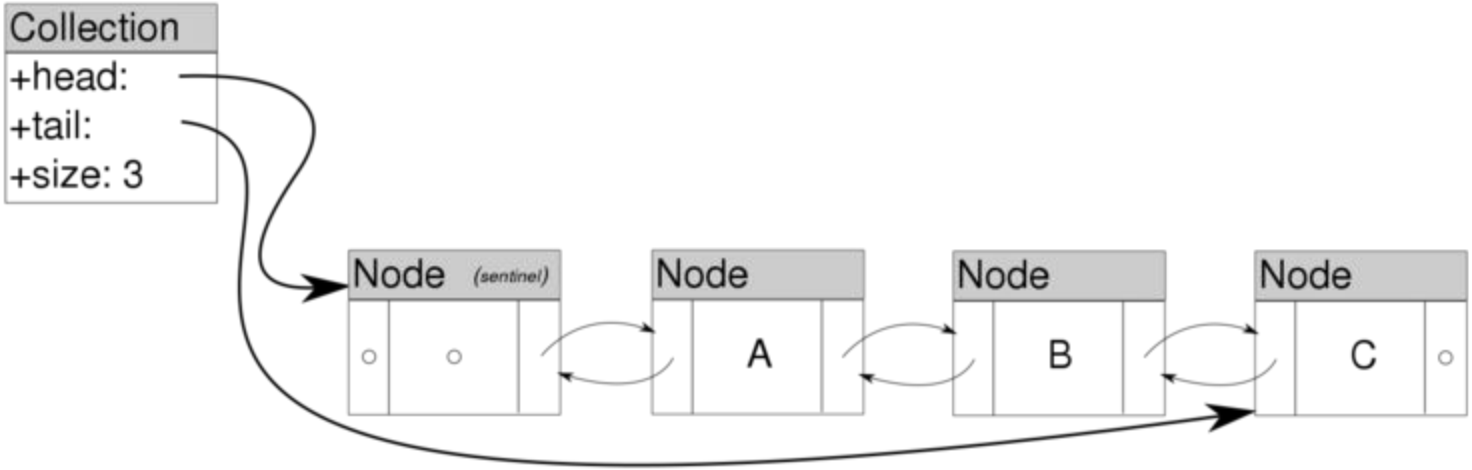




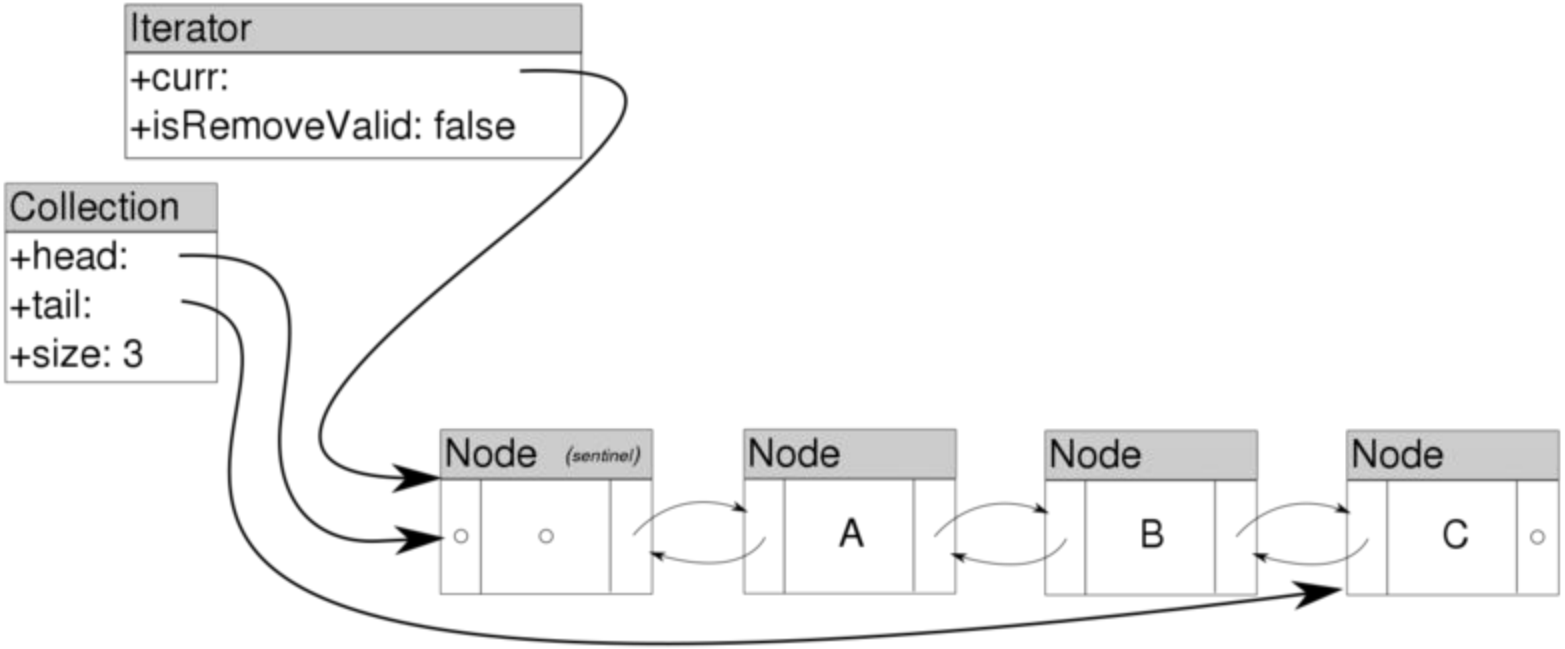


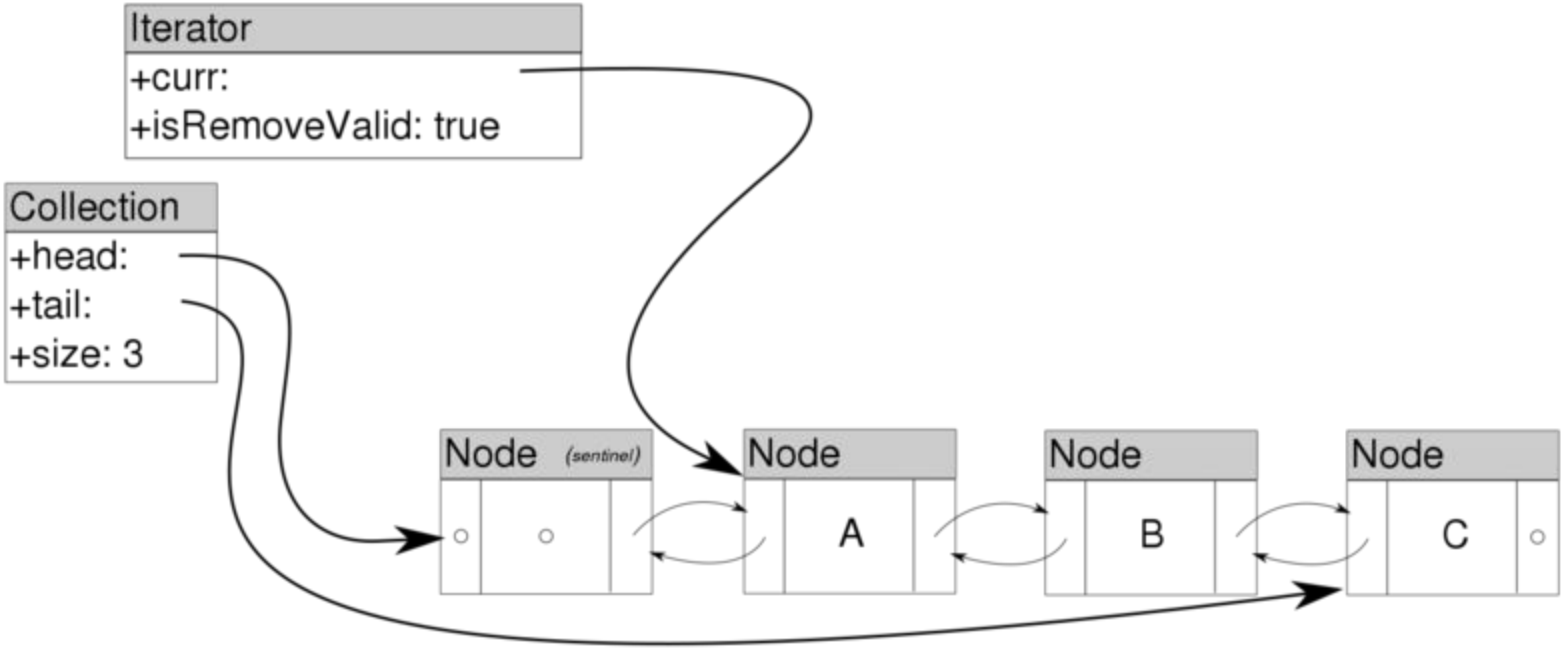


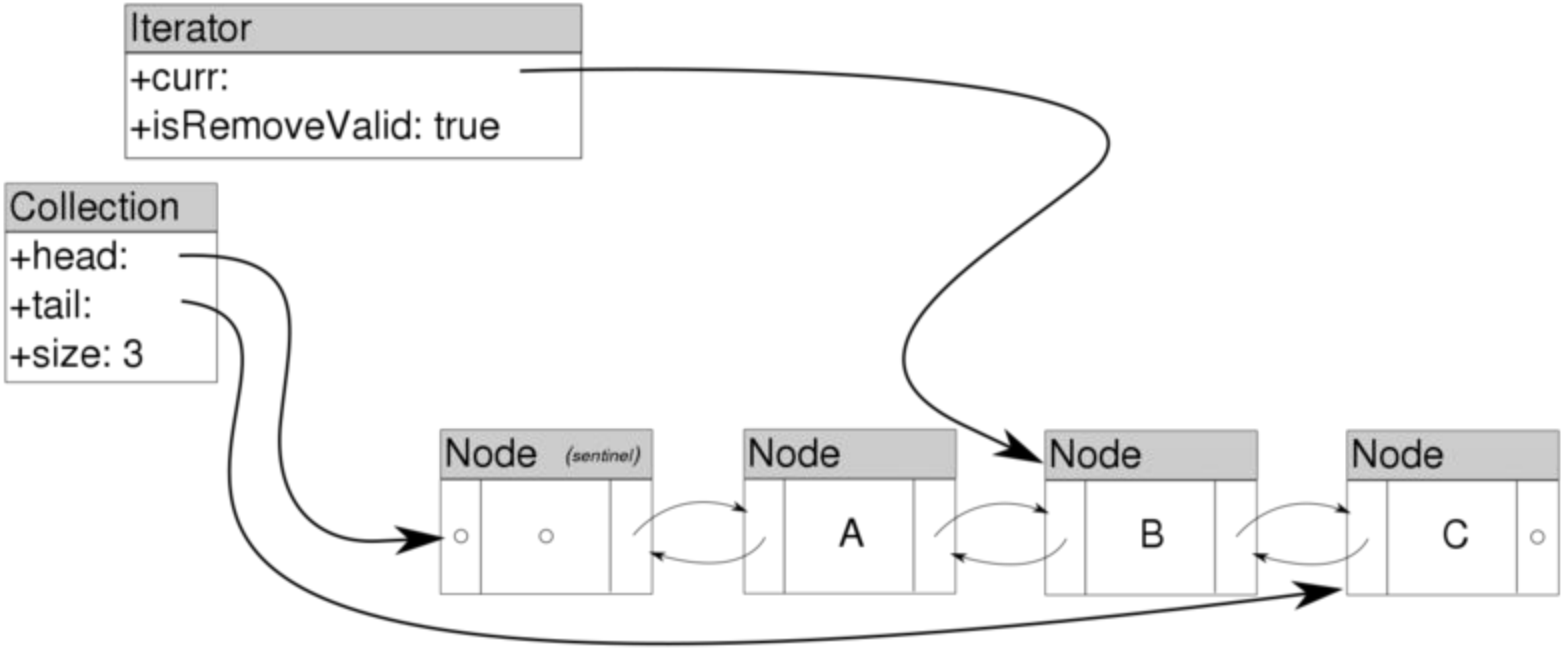
S&Q: Why is Java set up such that you must create an iterator along with an iterable? Why can't the iterable just have a way to carry out iteration itself?

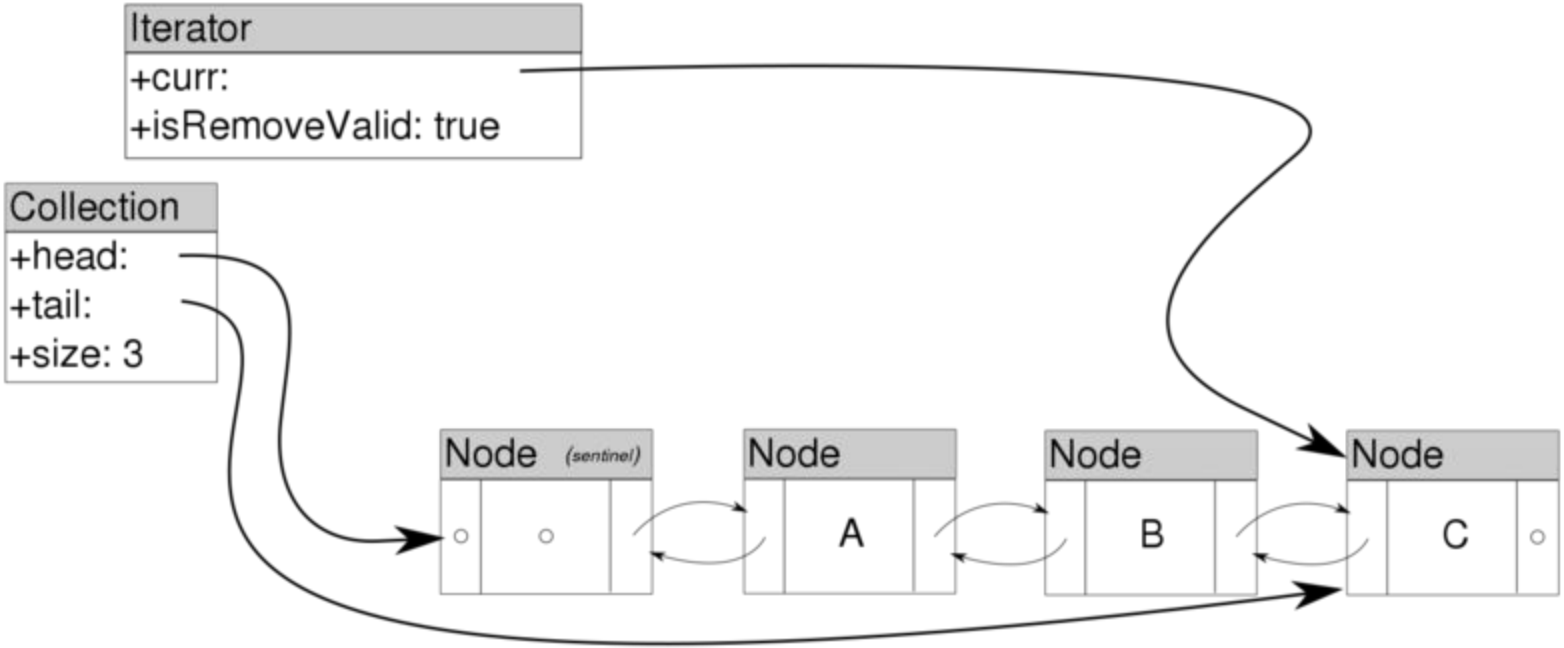

Collection
+head:
+tail:
+size: 3
Node
(sentinel)
Node
A
Node
B
Node
C

You need an instance to mutate as you march along




Iterator
+curr:
+isRemoveValid: true
Collection
+head:
+tail:
+size: 3
Node (sentinel)
Node
A
Node
B
Node
C


Iterator
+curr:
+isRemoveValid: true
Collection
+head:
+tail:
+size: 3
Node (sentinel)
Node
A
Node
B
Node
C


Iterator
+curr:
+isRemoveValid: true
Collection
+head:
+tail:
+size: 3
Node (sentinel)
Node
A
Node
B
Node
C

S&Q: When implementing the next() function in the Iterator interface, why would you specifically throw a NoSuchElementException() instead of returning null?

philosophical, I suppose (and it is the spec).

- it should never happen
- no one should call next() without first checking hasNext()

S&Q: Can you elaborate more on what parallel coarsening exactly refers to? Is it solved by allowing more sequential?

Yes. Coarsening applies when you have more parallel opportunity than processors.

S&Q: Whats the advantage of coarsening?

Better performance by reducing wasted overhead that does not yield improved parallelism on finite processors.

S&Q: At what point is coarsening necessary? Like is it okay to just fork a few times but if it gets to be too many then consider it?

Sadly, in Cosgrovistan we do not have infinite processors.

S&Q: Will coarsening cost more time since it uses less parallelism?

less parallelism, also less overhead. it is a tradeoff. Somewhere between 2X and 10X tasks per processor is where it typically crosses over.

S&Q: How would coarsening be affected when the set of processes divided would be interdependent? (for example we divide the code into 1,2,3,4;5,6,7,8 but what if 3 and 7 are interdependent?) How will the final product be affected? Will it lead to race condition or data race?

yes. if you ran code which were interdependent in parallel that would lead to a race condition.

Sorting: n^2 versus n log(n)

```
final class DualPivotQuicksort {
   private static final int MAX_INSERTION_SORT_SIZE = 44;
   private static final int MIN_PARALLEL_SORT_SIZE = 4 << 10;
```

at some point (44 apparently) the overhead of quicksort dominates and insertion sort is faster.

44 * 44 = 1936

44 * log2(44) = 44 * 5.4 = ~240

// note: read << 10 as 1024 or K

S&Q: I'm still a little confused about the iterable class and would like to go over that more in class. What exactly does it mean for something to be iterable?

S&Q: According to this video guide, a container element can be iterated without using indexing. Since string is an array of characters, can a string also be iterated without indexing like below?

```
String S = "0123456789abcdefghijklmnopqrstuvwxyz";
for (char c : S) { ... }
```

StringIterable and Iterator

```
class StringIterable implements Iterable<Character> {
    private final String s;
    public StringIterable(String s) {
        this.s = s;
    }
    @Override
    public Iterator<Character> iterator() {
        return new StringIterator(s);
    }
}
```

```
class StringIterator implements Iterator<Character> {
    private final String s;
    private int index = 0;
    public StringIterator(String s) {
        this.s = s;
    }
    @Override
    public boolean hasNext() {
        return index < s.length();
    }
    @Override
    public Character next() {
        if (hasNext()) {
            char ch = s.charAt(index);
            ++index;
            return ch;
        } else {
            throw new NoSuchElementException();
        }
    }
}
```

*** Preferred Version?

```
String s = "abc";
// String is not an Iterable
//      for(char ch : s) {
//          System.out.println(ch);
//      }
for (int i = 0; i < s.length(); ++i) {
    System.out.println(s.charAt(i));
}
System.out.println();
for (char ch : s.toCharArray()) {
    System.out.println(ch);
}
System.out.println();
for (char ch : new StringIterable(s)) {
    System.out.println(ch);
}
```

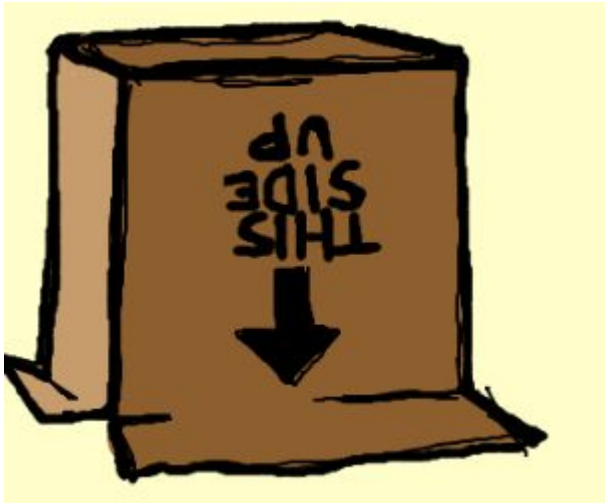
S&Q: In order to move across an iterable object, rather than using a boolean method call, could you just modify the while loop to act as the extra method?

https://docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html#forEach-java.util.function.Consumer-

S&Q: I am still very confused about the difference between a normal for loop (using i = 0; i < some_x; i++) vs a for each loop. Aren't they the same, for_each just does not include index? Because isn't runtime for both O(n)?

S&Q: Based on the "choosing the right tool for the job" video, do you think a good rule of thumb would be that if you don't need an indices for an array and just need the elements, use a for each loop, and if you do need the indices use a normal for loop?

Turn In Worksheets, Please



DoubleDeltaRange Warmup

- Experience implementing Iterable/Iterator

S&Q: Are ranges themselves not arrays of numbers? Or just a pair of numbers? What is the need for a new object known as a Range, rather than just using pairs of numbers or something?

`new Range(0, 1_000_000_000)` // do NOT create an array one billion long with the indices as each value

imagine:

```
array[0] = 0;
array[1] = 1;
```

…

S&Q: Does a range actually contain the numbers between min and max or does it just hold the 2 numbers?

Just hold the 2 numbers in the Iterable.

The Iterator will keep track of your progress between them.

S&Q: Are we able to do coarsening while still using the fork-loop?

that is the idea. slice into ranges, pair with fork_loop, sequentially process each range.

```
ranges = slice [min, maxExclusive) numProcessors * 2
Parallel.loop(ranges, (range)->{
    Sequential.loop i from [range.min to range.maxExclusive)
        process i
})
```

Human Chromosome Length

https://www.ncbi.nlm.nih.gov/grc/human/data

- every base can determine if it is an adenine independently
- therefore, all can be computed in parallel (then combined…)

*** Do you have 50 million processors?

*** Is determining if a single base is an adenine enough work to justify making a task? What about the combination in the end?

Chromosome	Total length (bp)
1	248,956,422
2	242,193,529
3	198,295,559
4	190,214,555
5	181,538,259
6	170,805,979
7	159,345,973
8	145,138,636
9	138,394,717
10	133,797,422
11	135,086,622
12	133,275,309
13	114,364,328
14	107,043,718
15	101,991,189
16	90,338,345
17	83,257,441
18	80,373,285
19	58,617,616
20	64,444,167
21	46,709,983
22	50,818,468
X	156,040,895
Y	57,227,415

Ranges Exercise

- Useful for all future coarsening exercises

S&Q: if the range cannot be evenly sliced, what will decide the length of each slices?

If you are in a group project. Say, the students at your table. Imagine there are 31 pieces of work to do. How would you split it up?

```
public class Ranges {
    public static Range[] slice(int min, int maxExclusive, int numRanges) {
        throw new NotYetImplementedException();
    }
}
```

Range

```
public final class Range implements Iterable<Integer> {
    private final int min;
    private final int maxExclusive;
    public Range(int min, int maxExclusive) {
        throw new NotYetImplementedException();
    }
    public int min() {
        throw new NotYetImplementedException();
    }
    public int maxExclusive() {
        throw new NotYetImplementedException();
    }
    @Override
    public Iterator<Integer> iterator() {
        throw new NotYetImplementedException();
    }
}
```

RangeIterator

```
/* package-private */ class RangeIterator implements Iterator<Integer> {
    /* package-private */ RangeIterator(Range range) {
        throw new NotYetImplementedException();
    }
    @Override
    public boolean hasNext() {
        throw new NotYetImplementedException();
    }
    @Override
    public Integer next() {
        throw new NotYetImplementedException();
    }
}
```

S&Q: How can we apply the N-way split coursening code to parallel programs?

Coarsening N-Way Split Nucleobase Count

```
public class CoarseningNucleobaseCounter implements NucleobaseCounter {
    private final int numRanges;
    public CoarseningNucleobaseCounter(int numRanges) {
        throw new NotYetImplementedException();
    }
    public int numRanges() {
        throw new NotYetImplementedException();
    }
    @Override
    public int count(byte[] chromosome, Nucleobase targetNucleobase) {
        throw new NotYetImplementedException();
    }
}
```

anyone??? join_void_fork_loop, join_void_fork_loop

- How many times will doSomething2d be invoked? _____
- How many times will fork be invoked? _____
- How many times will join be invoked? _____

```
join_void_fork_loop(0, n, (i) -> {

      join_void_fork_loop(0, m, (j) -> {

            doSomething2d(i, j);

      });

});
```