Lecture 4, 9/2/2020, CPSC 131

1. Deadlines
   1. Quiz – Today after class
      1. 30 minutes only
      2. Need to be completed by next Wednesday:   
         Recommended to do before class
      3. 2 Attempts only: waiting period of 6 hours between attempts
      4. There will be coding on it but this quiz will be multiple choice, T/F, fill in the blank (in the form of code)
      5. 10-15 questions
2. Projects
   1. Please contact Group
      1. William La(?), Nandkishor Lamtur, Patrice Jones (?)
      2. To turn it in as a group
         1. One submission for everyone. SO one person has to submit it.
         2. Please put your names on it.
   2. Deliverable Artifacts
      1. Sample\_input.txt is a testing file so you cannot edit it.
   3. DO NOT CHANGE THE HEADER FILE!!!!!!!!!!!!
   4. Deliver complete solutions
   5. Create 2 new files: Book.cpp and main.cpp
      1. Book is what the Client will use to do things
3. Lecture
   1. What you need to know
      1. C++ pointers, native arrays, dynamic memory,
      2. Binary Search, Linear Search
      3. No Data Structures (yet)
   2. What we will discuss today
      1. **Big-O Notation**
      2. How to compare different containers
   3. Key Terms
      1. **Experimental analysis**
      2. **Asymptotic Analysis**
      3. **Worst-case analysis**
         1. The most amount of time it needs to compute
         2. We calculate this so that lives won’t be lost
      4. **Big-O notation**
         1. O = Order of magnitude
            1. Is this better than that?
            2. The way we express what is found
         2. **Constant time operations O(1)**
            1. The amount of time it takes is independent of how many elements in container
            2. Time of one item = Time of many items
         3. **Linear time operations O(n)**
            1. The amount of time it takes is directly proportional to amount of stuff
            2. Time of one item < Time of 1000 items
         4. **Quadratic time operations O(n^2)**
            1. The amount of time it takes is time^(amount of items)
         5. **Logarithmic time operations O(log [base 2] n)**
            1. This is our binary search
            2. The amount of time it takes for log # of elements
   4. Comparing data structures
      1. Is doubly linked list better than singly linked list?
         1. Running time? Memory used?
         2. Usually means a trade off
      2. Two approaches to answering this question:
         1. Option 1: Experimental Analysis
            1. Go and measure the data
            2. Uses limited data to analyze
         2. Option 2: Asymptotic analysis
            1. Why is it best?

Gives us an approximation (what asymptotic means).

Allows us to categorize the algorithm.

Analysis without running any code

This one is preferred and less time and resources are demanded.

* + - * 1. Key idea:

We are interested in running time for large data sets

* + - * 1. Most important factor?

Number of elements in the data structure

Represent this by n

* + - * 1. Analysis

How does running time increase in terms of n?

* + - * 1. Example of Analysis in action

Consider a singly linked list with n elements

Singly LinkedList<int> list;  
…//insert n integers  
for (const auto & value : list)  
{   
cout<<value<<endl;  
}

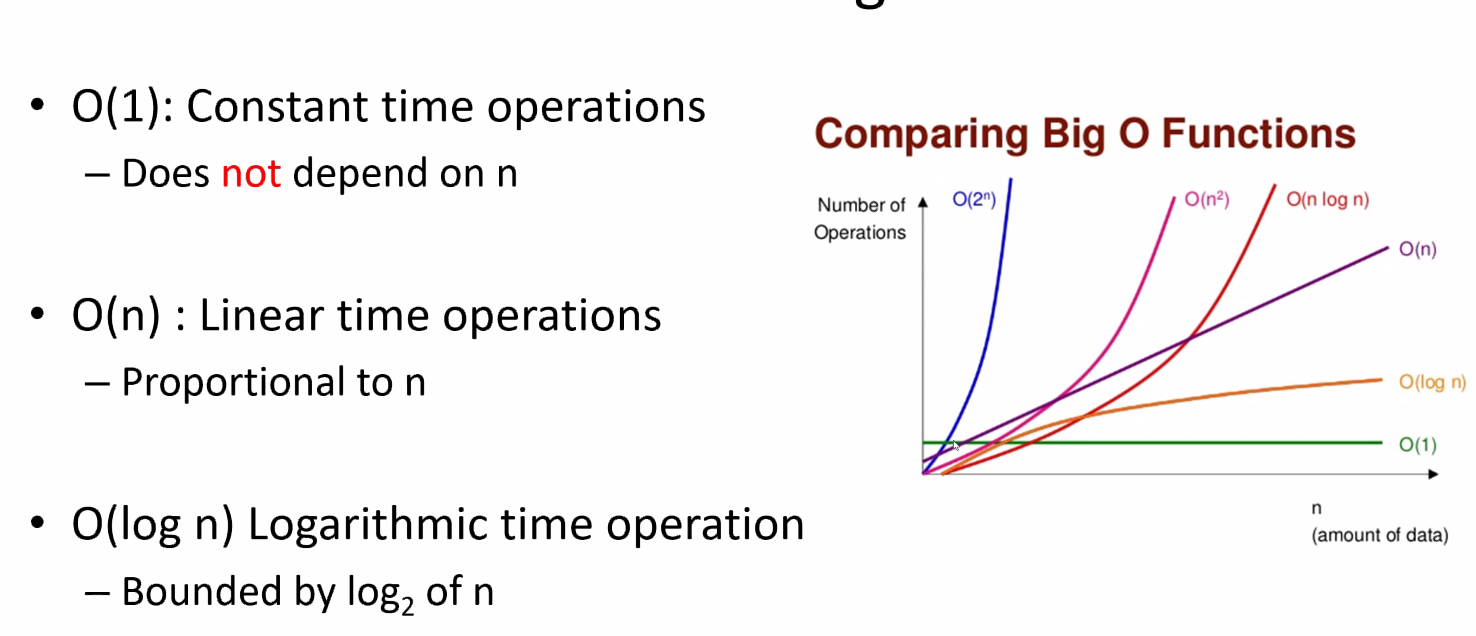
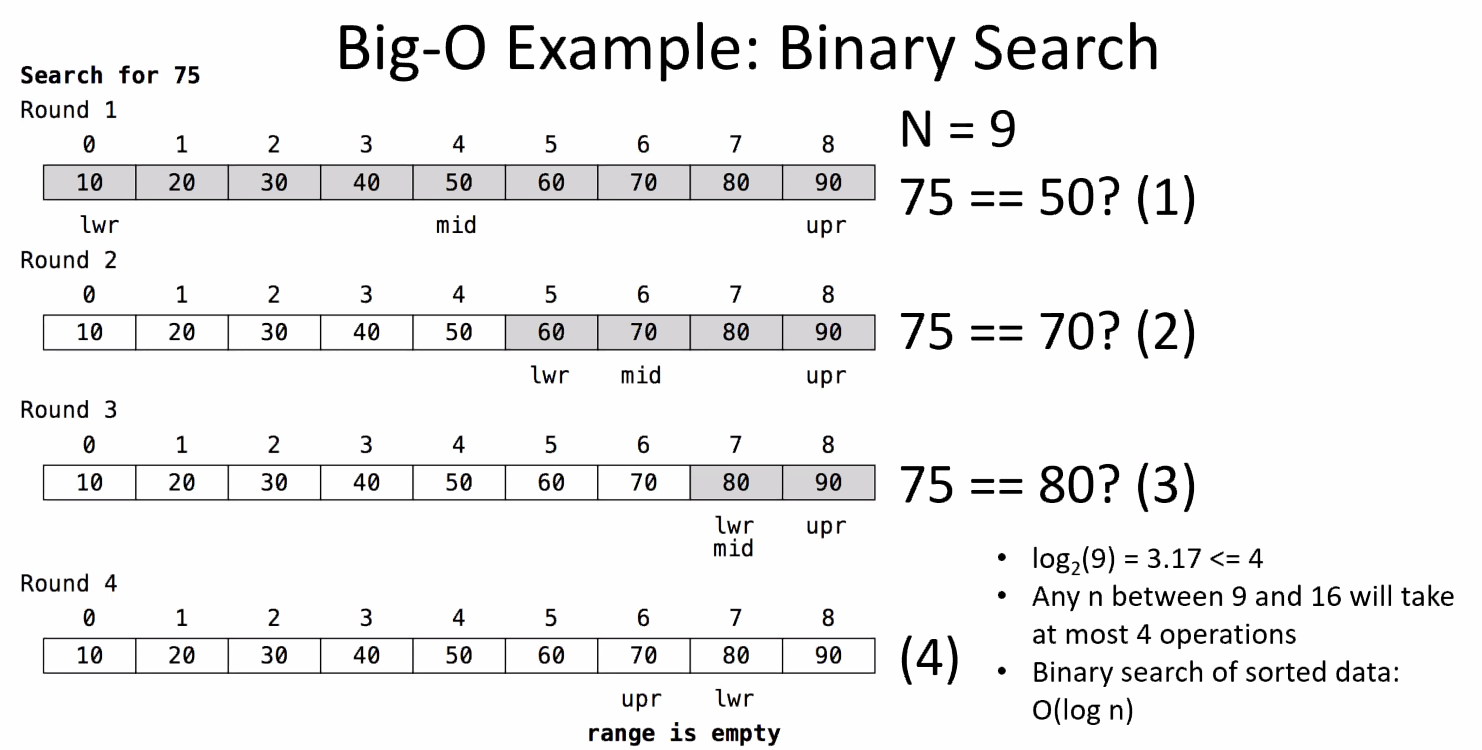
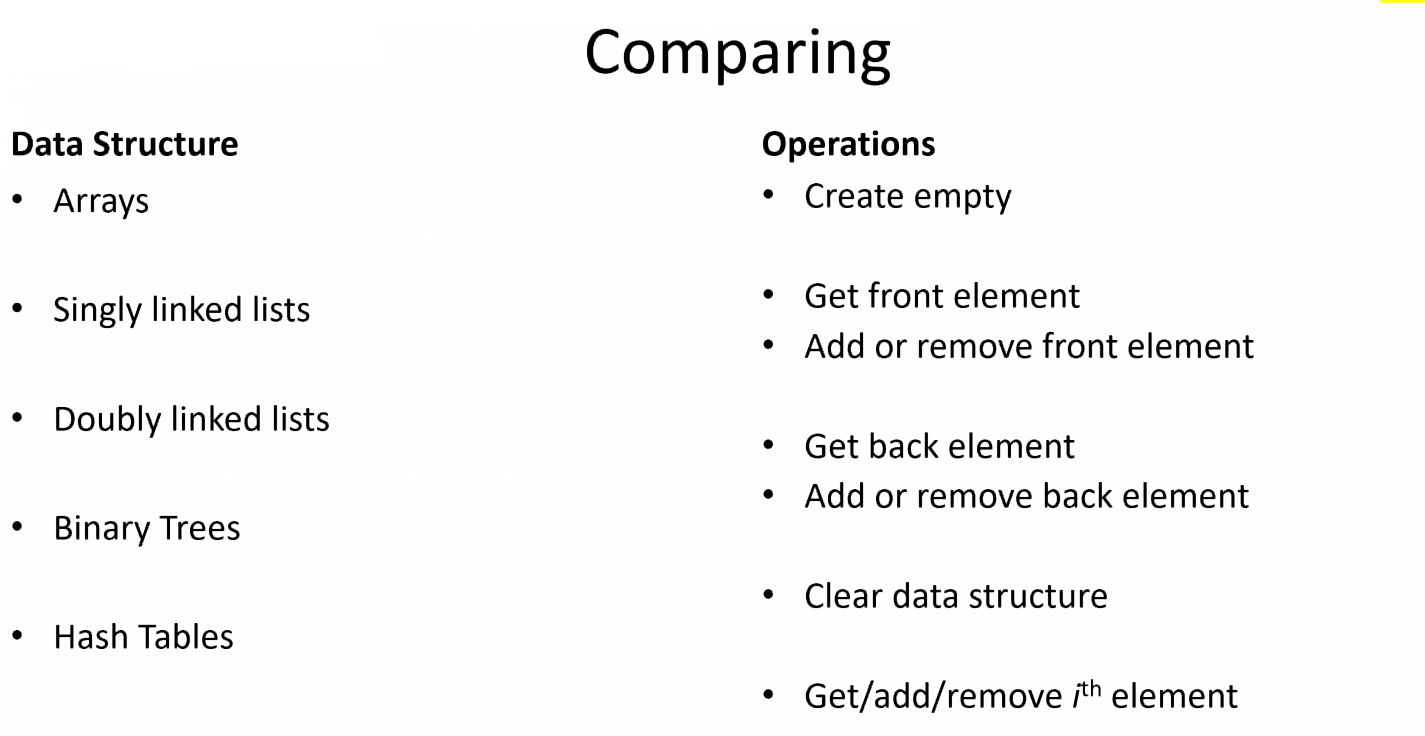
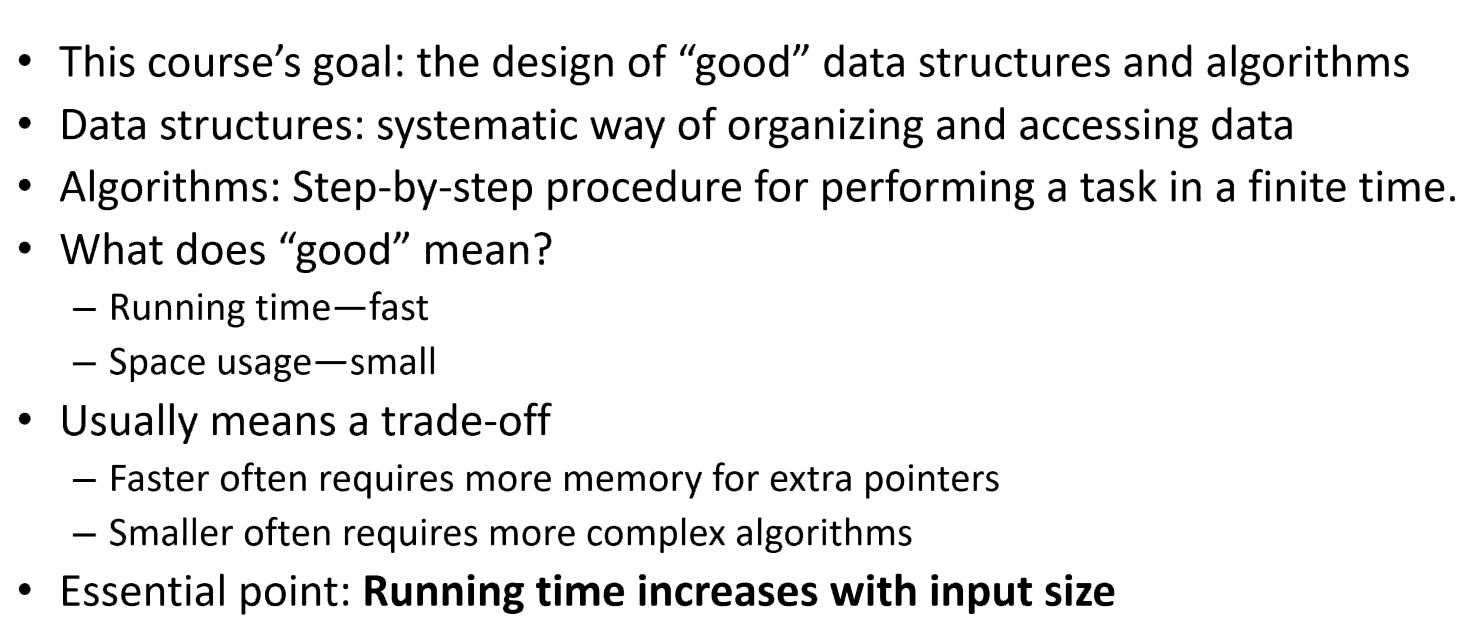
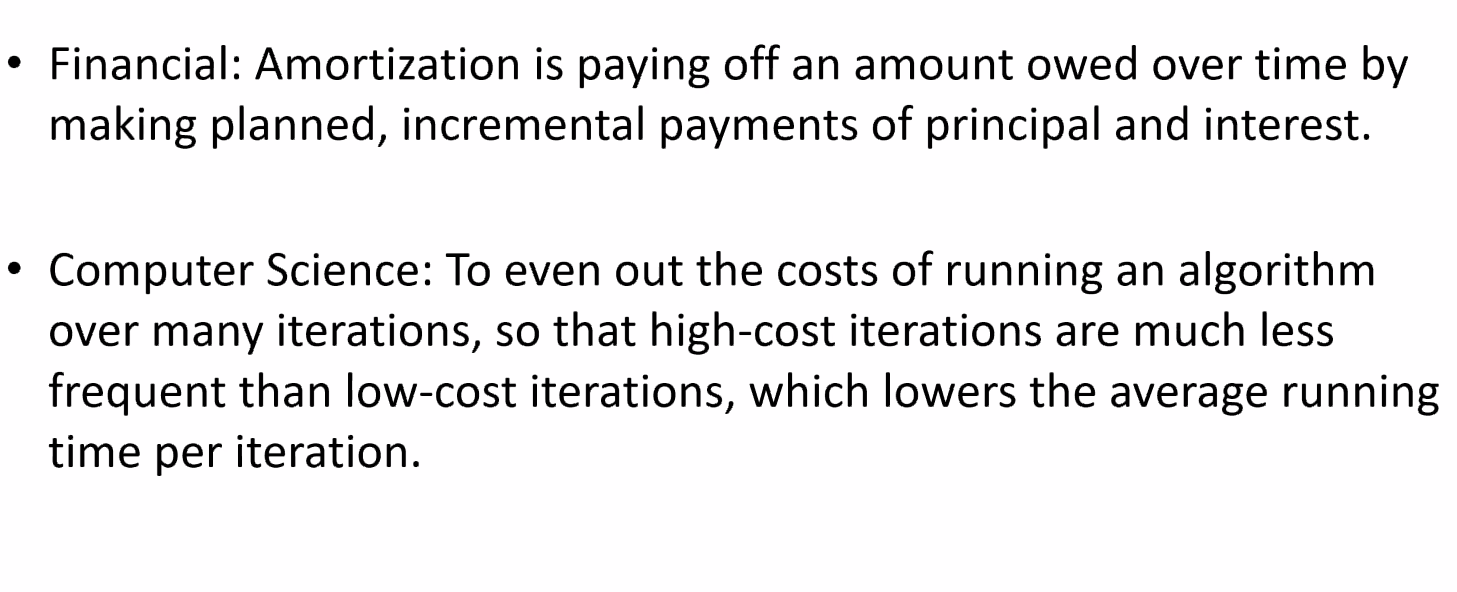
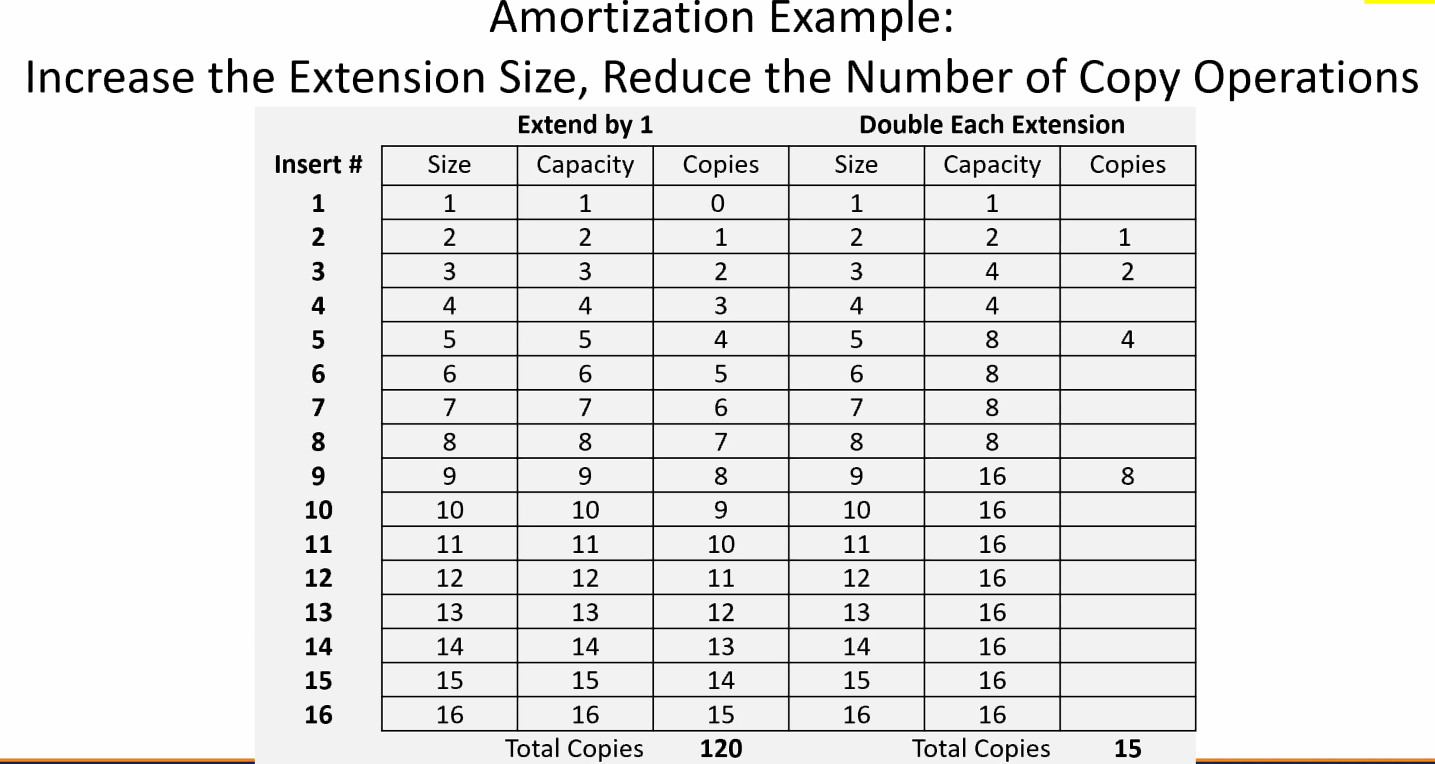
How many steps?

Too many!

Printing a linked list

Linear time: more elements, more time

Thus, Linear O(n) is used

* + 1. Common Classes of Big-O Functions
       1. Graphic  
          
          1. O(1): Constant Time is a straight line
          2. O(n): is a sloped line
          3. O(log n) is sloped but gradually curves and then that curves straightens out
          4. O(n^2) is a steep curve
       2. Big-O Example Binary Search
          1. 
          2. At most, it will take 4 checks using log2(n)
          3. The analysis time will be far shorter than O(n)
    2. Comparing
       - 1. 
         2. 
         3. Running time increases with more stuff
  1. Amortization
     1. 
     2. 
     3. In the beginning, when you double the size of vector, then you are going to amortize the worst case.

1. SI
   1. Remember, these sessions are available for you on ZOOM
   2. Holland HO
      1. MW, 11:30 – 12:45
      2. Canvas linked there