Lecture 3 - 1/23/2024 Last Updated 1/12/2024

---

1. Binary Search
   - Binary Search is a more efficient way to search for a particular value through a list of values. On top of taking the same inputs as linear search, an extra condition must be satisfied which is that the list of values must be sorted prior to performing the algorithm in order to obtain valid results. Here is the pseudo code version shown in lecture:

```
found = "no";
begin = 1;
end = n;
while (found == "no" and begin  end) {
   m = begin + floor((end-begin)/2)
   if(A[m] == x){
         found = "yes";
         location = m;
   }
   if(A[m]>x)
         end = m-1;
   if(A[m]<x)
  begin = m+1;
}
if (found == "no") {
   print ("Sorry, " + x + " is not on the list");
}else {
 print (x + " occurs at position " + location + " on the
list");
}
```

   - The only new component is the function floor which takes an input and brings it to the smallest whole number within the range 2.4 goes to 2 while -2.4 goes to -3

   - The simple proof for the binary search runtime is as intuitive as this, out of an N item list, how many splits do we have to make until we have 1 item remaining:

$$1 = \frac{N}{2^x}$$

$$2^x = N \rightarrow log_2(2^x) = log_2(N)$$

$$xlog_2(2) = log_2(N) \rightarrow x = log_2(N)$$

2. Selection Sort: Selection Sort visualize | Algorithms | HackerEarth
    a. For Loop
        i. For loops are like while loops except they handle the counter variable logic for us they allow us to repeat actions x times
        ii. Here is an example of a for loop you've seen in lecture:

```
for (j=i+1;j <= length(A); j++){
      if (A[locmin] > A[j])
    locmin=j;
}
```

    b. Selection Sort is a sorting algorithm in which you loop through the entire list, select the minimum and then swap it with the current element you are at in the list. The algorithm from lecture is here for your reference:

```
for (i = 1; i < length(A); i++){
     locmin=i;
     for (j=i+1;j <= length(A); j++){
       if (A[locmin] > A[j])
     locmin=j;
     }
     exchange(A[locmin], A[i]);
}
```

    c. If you are still having trouble understanding Selection sort go check out the visualization at the link above.

3. Insertion Sort: Insertion Sort visualize | Algorithms | HackerEarth
    a. Insertion Sort is a sorting algorithm in which you take your current element in the list and insert it into the proper position in the sorted portion of the list, the very first element is considered to be "sorted" for purposes of this definition. The algorithm from lecture is here for your reference:

```
for (j=2; j <= n; j ++){
      temp = A[j]
      i=j-1
     while (i > 0 and A[i] > temp){
     A[i+1]=A[i]
     i=i-1
     }
     A[i+1]=temp
}
```

b. If you are still having trouble understanding Insertion sort go check out the visualization at the link above.

Lecture 4 - 1/25/2024
---

1. Sign up for your codio account using the course token nirvana-oberon see lecture 4 ppt on courseworks for more information. Use your university email.

2. Big-O Notation and the Analysis of Algorithms
   a. We can look at the computational resources needed in order to execute an algorithm, we use a system known as Big-O to systematically rank the use of these resources for different algorithms.

   b. Selection Sort takes

$$1 + 2 + 3 + +(n-1)$$

   This is the same as the sum of the first $n-1$ numbers which equals:

$$\frac{n(n-1)}{2}$$

   This exact form is known as T(n) you can think T for total if it helps you, with Big-O we use the term with the highest power to provide an approximation for the magnitude of the runtime so for selection sort it would be O(n^2)

   c. Insertion Sort has a variable runtime depending on how the items in the list in the best case T(n) = n - 1 and in the worst case: T(n) = n(n-1)/2

   d. The Big-O for Insertion Sort is the same as the Big-O for Selection Sort, that is O(n^2)

   e. To review the 4 algorithms you should know by now: Linear Search O(n), Binary Search O(logn), Selection Sort O(n^2) and Insertion Sort O(n^2).

   f. T(n) refers to efficiency and O(n) refers to order of growth

3. Introduction to Programming in Java and the Codio Environment
    a. Basic Codio commands:
        i. ls - lists all files in current directory
        ii. Clear - clears the terminal
    b. How to Run Java Programs: You must always compile prior to execution when you've made changes to your program
        i. Compile the program: javac filename.java
        ii. Execute the program: java filename
    c. The Main Method - this is needed for all java programs in order to run
        i. public static void main(String[] args)
    d. File name and class name must match, it is convention for class names to be camel cased so instead of 'heydude' its 'HeyDude'
        i. public class FileName
    e. In order to print: System.out.println(textToPrint);
    f. There are dataTypes in Java which must proceed the variable name you write so the compiler knows how to treat it.
        i. int - for integer values
            1. int a = 2;
        ii. double - for floating point numbers
            1. double b = 2.0;
        iii. String - for sequences of characters
            1. String c = "Hello World!";