Intro

Comparison Count. Pick an instance characteristic .... n

`For insertion sort, pick n= number of elements to be sorted.`

Step Count A step is an amount of computing that does not depend on the instance characteristic n

steps per execution is not always 0 or 1. For example

x = sum (a,n)

Algorithimic imporvement is more useful than hardware imporvement

Performance Measurement

Uses: * Determine practicality of algorithm * Predict run time on large instance * Compare 2 algorithms that have different asymptotic complexity O(n) and O(n^2)

Limitations of Analysis

- Does not account for constant factors
- Constant factor may dominate. (1000n vs n^2)

Modern computers have a hierarchical memory organization with different access time for memory at different levels. Main memory can be 100x slower than the cache. analysis does not account for this

Need:

- Worst case data
- Best case data

Timing in C++

```
double clocksPerMillis =
    double(CLOCKS_PER_SEC) / 1000;
// clock ticks per millisecond

clock_t startTime = clock();

// Code to be timed comes here
// Need to do this while clock() - startTime < 1000

double elapsedMillis = (click() - startTime)
```

This could be an issue when the amount of time is 1.5 clock ticks or something like that.

Clock accuracy, assume 100 clicks. Repeat work many times to bring total time to be >= 1000 ticks. This makes the accuracy +- 10%. A very short process could register as 0 ticks.

When measuring sorting, must use same unsorted array each time

UNIX `time MyProgram` takes into account busy cores doing other tasks

Data Structures

Design: In every activity, there are a number of possible solutions. We must pick one when implementing software. Sometime the solution is not optimum for one solution. Design is the process of selecting one element in a viable solution space.

Data Object: Set or collection of instances. Days of the week or A list of numbers. Could be related or not, doesn't matter

The relationships are usually specified by specifying operations on one or more instances. Data Structures are a set of data that you do operations to.

Data Structure Specification

- Language independent
  - Abstract Data Type
  - Just English telling what each method of the class will do
- C++
  - Abstract Class
  - Purely virtual

Linear (or Ordered) lists

Things one after the other

(e0, e1, e2. . . .e(n-1))

- ei denotes a list element
- n >= 0 is finite
- List size is n
- e0 is the zero'th (or front) element in the list

Example:

- (Jack, Jill, Bob)

Operations

- Determine list size
- Get(given Index)
- IndexOf(element) - Items could be repeated, however, which would yield an incorrect index. -1 is returned if the item is not found
- Erase(Index)
- Insert(Index, Element)- nice to have the arguments go in natural english order "Insert into slot 5 'a' "
  - Use a for loop to shift everything over to the right and then input the argument

Extending a C++ class - (From an already created C++ pure virtual class)

```
template<class T>
class arrayList: public linearList<T> {
    //Code for all abstract methods of a linearList must come here
}
```

Linear List Array Representation

Use a one-dimensional array element[ ]

L = (a,b,c,d,e)

Store element i of list in element[i]

Representation of the list include:

- Mapping from right to left
- Mapping that skips every other position
- Wrap around mapping
  - Useful for Queues and Stacks

Potential Problems

We do not know how many elements will be in the list, therefore we could run out of allocated memory. Therefore, we must pick an initial length and dynamically increase as needed. Use another variable to store current length of the array

Solution:

- First create a new and larger array

```
T * new Array = new T[15];
// There will be a pointer to the new array
```

- Now copy elements from old array to new one
- Finally, delete old array and rename new array

```
delete[] element;
element = new Array;
arraylength = 15;
```

How Big should the new array be? At least 1 more than current array length. Cost of increasing array length when array is full is theta(old length). Cost of n insert operations done on an initially empty linear list increases by theta(n^2).