Announcements
- HW - up after class
due next Sunday
- Tutoring starts this week
Recap of arrays (As 3.1 of text)

Limits

- not very flexible
  - size is fixed at creation
  - 1 kind of data
  - inserting + moving can be difficult

Q: How would we insert an element in the middle of an array?

ex: insert (20) in sorted order

2 5 6 11 25 26 31 3

\[ 20 \]
Singly Linked List

A collection of nodes that together form a linear ordering.

Memory

Node

LAX

STL

MSP

head

head
Code

see SLinkedList.h and SLinkedList.tcc

templated
Algorithm Analysis (Ch. 4)

How do we compare two programs?

- speed
- space (or memory usage)
- maintainability
- portability
- readability
Speed

How fast an algorithm runs can be very dependent on variables in the system.

Examples:

- CPU
- RAM
- Hard drive + buses
- Network
- Language
- Design decisions
- Compiler
- Input
Primitive Operations

As a way to compare algorithms in a generic way, we instead count primitive operations.

Addition, storing a value, subtraction, multiplication, allocating space...

In addition, we (generally) only analyze the worst possible running time.

Why? Generally, worst case is what causes problems!
Comparing

OK, so we have the worst case # of operations - usually a function of \( n \).

How to compare?

Big-O notation
We say $f(n)$ is $O(g(n))$ if $\forall n > n_0$, there exists $c > 0$ such that $f(n) \leq c \cdot g(n)$.
Ex: $5n$ is $\Theta(n^2)$

Let $n_0 = 5$, $c = 1$

$5n < n^2$ for $n > 5$

Ex: $5 \cdot n$ is $O(n)$

Let $n_0 = 1$, $c = 6$

$f(n) = 5 \cdot n < c \cdot n = 6n$

Ex: $16n^2 + 52$ is $O(n^2)$

$n_0 = 52$, $c = 17$

In polynomials, largest degree matters.
Functions we will use

1. \(O(1)\) - constant time
2. \(O(\log n)\) - logarithmic time
3. \(O(n)\) - linear time
4. \(O(n \log n)\)
5. \(O(n^2)\) - quadratic time
6. \(O(n^3)\) - cubic time
7. \(O(2^n)\) - exponential time
Algorithms

Claim: Inserting an element into the first spot in an array is $O(n)$ time.

Claim: Inserting at the beginning of a list is $O(1)$ time.
Common running times

- A for loop which goes from $i=0$ to $n-1$ and reads into an array

```cpp
for (int i = 0; i < n; i++)
    cin >> array[i];
```

**Analyze:**

\[
\sum_{i=0}^{n-1} (1 + 1 + 1 + 1) = 4n = O(n)
\]

**Lazy:**

\[
\sum_{i=0}^{n-1} 1 = (1 + 1 + \ldots + 1) = n
\]
Nested For loops: find if any 2 elements are identical

```cpp
for (int i = 0; i < n; i++)
    for (int j = i + 1; j < n; j++)
        if (A[i][j] != A[j][i])
            cout << "Two items are the same" << endl;
```

Analyze:

\[
\sum_{i=0}^{n-1} \left( \sum_{j=i+1}^{n-1} 1 \right) = \sum_{i=0}^{n-1} \left( 1 + 1 + \ldots + 1 \right)
\]

\[
= \sum_{i=0}^{n-1} (n-i) = (n-1) + (n-2) + (n-3) + \ldots + 1
\]

\[
= \sum_{i=1}^{n} i = \frac{n(n+1)}{2}
\]

\[
\frac{n^2}{2} - \frac{n}{2} = \Theta(n^2)
\]