CSE 326: Data Structures Lecture #2 Analysis of Algorithms I

(And A Little More About Linked Lists)

Henry Kautz

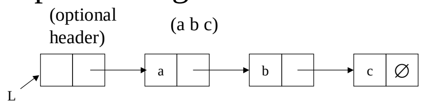
Winter 2002

Assignment #1

Goals of this assignment:

- Introduce the ADTs (abstract data types) for lists and sparse vectors, motivated by an application to information retrieval.
- Show the connection between the empirical runtime scaling of an algorithm and formal asymptotic complexity
- Gain experience with the Unix tools g++, make, gnuplot, csh, and awk.
- Learn how to use templates in C++.
  - We will use new g++ version 3.0 compiler – does templates right!

Implementing Linked Lists in C

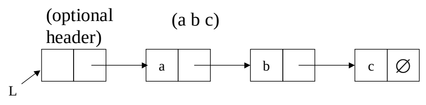


```
struct node{
    Object element;
    struct node * next; }
```

Everything else is a pointer to a node!

```
typedef stuct node * List;
typedef struct node * Position;
```

Implementing in C++



Create separate classes for

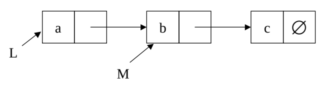
- Node
- List (contains a pointer to the first node)
- List Iterator (specifies a position in a list; basically, just a pointer to a node)

Pro: syntactically distinguishes uses of node pointers

Con: a lot of verbage! Also, is a position in a list really distinct from a list?

Structure Sharing

L = (a b c)   M = (b c)



•Important technique for conserving memory usage in large lists with repeated structure

•Used in many recursive algorithms on lists

Implementing Linked Lists Using Arrays

	1	2	3	4	5	6	7	8	9	10
Data		F	O	A	R	N		R		T
Next		3	8	6	4	-1		10		5

First = 2

“Cursor implementation” Ch 3.2.8

Can use same array to manage a second list of unused cells

List ADT

↑

Polynomial ADT

A_i is the coefficient of the x^{i-1} term:

$5 + 2x + 3x^2$	**(5 2 3)**
$7 + 8x$	**(7 8)**
$3 + x^2$	**(3 0 2)**

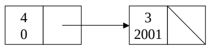
Problem?

$4 + 3x^{2001}$

```
( 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 )
```

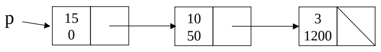
Sparse List Data Structure: $4 + 3x^{2001}$

(<4 0> <2001 3>)

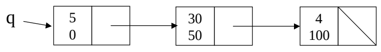


Addition of Two Polynomials?

$15+10x^{50}+3x^{1200}$



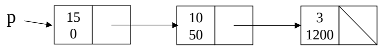
$5+30x^{50}+4x^{100}$



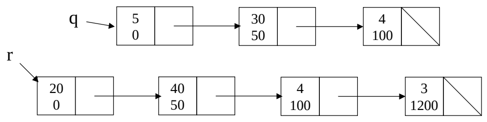
Addition of Two Polynomials

- Similar to merging two sorted lists – one pass!

$15+10x^{50}+3x^{1200}$



$5+30x^{50}+4x^{100}$



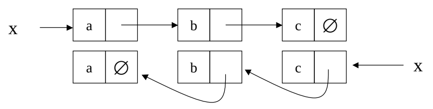
To ADT or NOT to ADT?

- Issue: when to bypass / expand List ADT?
- Using general list operations:

```
reverse(x) {
  y = new list;
  while (! x.empty()){
     /* remove 1st element from x, insert in y */
     y.insert_after_kth( x.kth(1), 0);
     x.delete_kth(1);
     }
  return y; }
```

Disadvantages?

Destructive LL Version



```
reverse(node * x) {
  last = NULL;
  while (x->next != NULL){
     tmp = x->next;
     x->next = last;
     last = x;
     x = tmp;}
  return x; }
```

Faster in practice?

Asymptotically faster?

Analysis of Algorithms

- Analysis of an algorithm gives insight into how long the program runs and how much memory it uses
  - time complexity
  - space complexity
- Why useful?
- Input size is indicated by a number *n*
  - sometimes have multiple inputs, *e.g.* m and n
- Running time is a function of *n*

  n, n^2, $n \log n$, $18 + 3n(\log n^2) + 5n^3$

Simplifying the Analysis

- Eliminate low order terms

  $4n + 5 \Rightarrow 4n$

  $0.5\ n \log n - 2n + 7 \Rightarrow 0.5\ n \log n$

  $2^n + n^3 + 3n \Rightarrow 2^n$
- Eliminate constant coefficients

  $4n \Rightarrow n$

  $0.5\ n \log n \Rightarrow n \log n$

  $\log n^2 = 2 \log n \Rightarrow \log n$

  $\log_3 n = (\log_3 2) \log n \Rightarrow \log n$

Order Notation

- BIG-O $T(n) = O(f(n))$

  Upper bound

  Exist constants c and n' such that

  $T(n) \leq c\ f(n)$ for all $n \geq n'$
- OMEGA $T(n) = \Omega(f(n))$

  Lower bound

  Exist constants c and n' such that

  $T(n) \geq c\ f(n)$ for all $n \geq n_0$
- THETA $T(n) = \theta(f(n))$

  Tight bound

  $\theta(n) = O(n) = \Omega(n)$

Examples

$n^2 + 100\ n = O(n^2)$ because

$(n^2 + 100\ n) \leq 2\ n^2$ for $n \geq 10$

$n^2 + 100\ n = \Omega(n^2)$ because

$(n^2 + 100\ n) \geq 1\ n^2$ for $n \geq 0$

Therefore:

$n^2 + 100\ n = \theta(n^2)$

Notation Gotcha

- Order notation is not symmetric; write

  $2n^2 + 4n = O(n^2)$

but never

$O(n^2) = 2n^2 + 4n$

right hand side is a crudification of the left

Likewise

$O(n^2) = O(n^3)$

$\Omega(n^3) = \Omega(n^2)$

Mini-Quiz

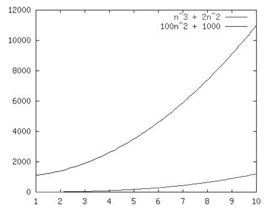
1. $5n \log n = O(n^2)$
2. $5n \log n = \Omega(n^2)$
3. $5n \log n = O(n)$
4. $5n \log n = \Omega(n)$
5. $5n \log n = \theta(n)$
6. $5n \log n = \theta(n \log n)$

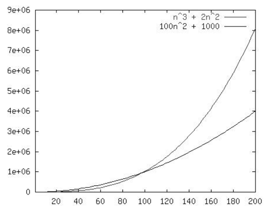
Silicon Downs

Post #1	Post #2
$n^3 + 2n^2$	$100n^2 + 1000$
$n^{0.1}$	$\log n$
$n + 100n^{0.1}$	$2n + 10 \log n$
$5n^5$	$n!$
$n^{-15}2^n/100$	$1000n^{15}$
$8^{2\log n}$	$3n^7 + 7n$

Race I

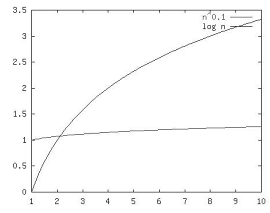
$n^3 + 2n^2$ vs. **$100n^2 + 1000$**

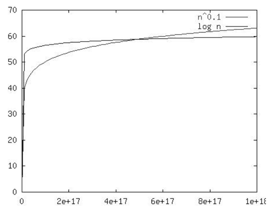




Race II

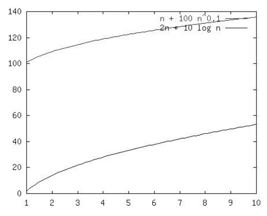
$n^{0.1}$ vs. **$\log n$**

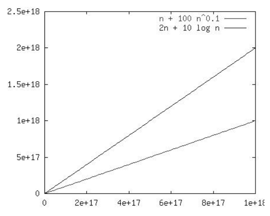




Race III

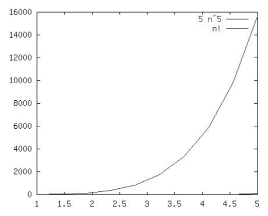
$n + 100n^{0.1}$ vs. **$2n + 10 \log n$**

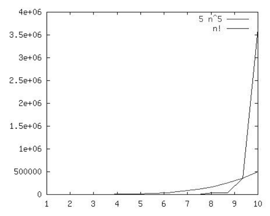




Race IV

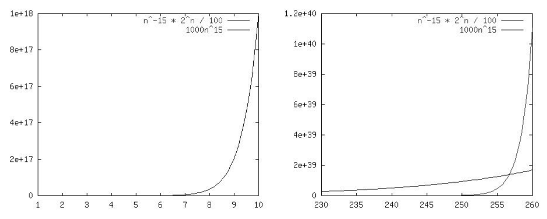
$5n^5$ vs. **$n!$**





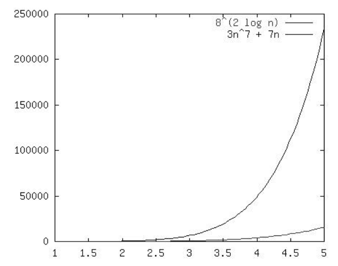
Race V

$n^{-15}2^n/100$ vs. $1000n^{15}$



Race VI

$8^{2\log(n)}$ vs. $3n^7 + 7n$



The Losers Win

Post #1	Post #2	Better algorithm!
$n^3 + 2n^2$	$100n^2 + 1000$	$O(n^2)$
$n^{0.1}$	$\log n$	$O(\log n)$
$n + 100n^{0.1}$	$2n + 10 \log n$	**TIE** $O(n)$
$5n^5$	$n!$	$O(n^5)$
$n^{-15}2^n/100$	$1000n^{15}$	$O(n^{15})$
$8^{2\log n}$	$3n^7 + 7n$	$O(n^6)$

Common Names

constant:	$O(1)$	
logarithmic:	$O(\log n)$	
linear:	$O(n)$	
log-linear:	$O(n \log n)$	
superlinear:	$O(n^{1+c})$	(c is a constant > 0)
quadratic:	$O(n^2)$	
polynomial:	$O(n^k)$	(k is a constant)
exponential:	$O(c^n)$	(c is a constant > 1)

Analyzing Code

- C++ operations - constant time
- consecutive stmts - sum of times
- conditionals - sum of branches, condition
- loops - sum of iterations
- function calls - cost of function body
- recursive functions - solve recursive equation

Above all, use your head!

Conditionals

- Conditional
  if *C* then *S₁* else *S₂*

$$\text{time} \le \text{time(C)} + \text{MAX(time(S1), time(S2))}$$

Nested Loops

```
for i = 1 to n do
  for j = 1 to n do
    sum = sum + 1
```

Nested Loops

```
for i = 1 to n do
  for j = 1 to n do
    sum = sum + 1
```

$$\sum_{i=1}^{n} \sum_{j=1}^{n} 1 = \sum_{i=1}^{n} n = n^2$$

Nested Dependent Loops

```
for i = 1 to n do
  for j = i to n do
    sum = sum + 1
```

Nested Dependent Loops

```
for i = 1 to n do
  for j = i to n do
    sum = sum + 1
```

$$\sum_{i=1}^{n} \sum_{j=1}^{n} 1 = \sum_{i=}^{n} (n-i-1) = \sum_{i=1}^{n} (n+1) - \sum_{i=1}^{n} i =$$

?

Nested Dependent Loops

```
for i = 1 to n do
  for j = i to n do
    sum = sum + 1
```

$$\sum_{i=1}^{n} \sum_{j=1}^{n} 1 = \sum_{i=}^{n} (n-i-1) = \sum_{i=1}^{n} (n+1) - \sum_{i=1}^{n} i =$$

$$n(n+1) - \frac{n(n+1)}{2} = \frac{n(n+1)}{2} = O(n^2)$$

To Do

- Finish reading Ch 1 and 2
- Start reading Ch 3
- Get started on assignment #1!