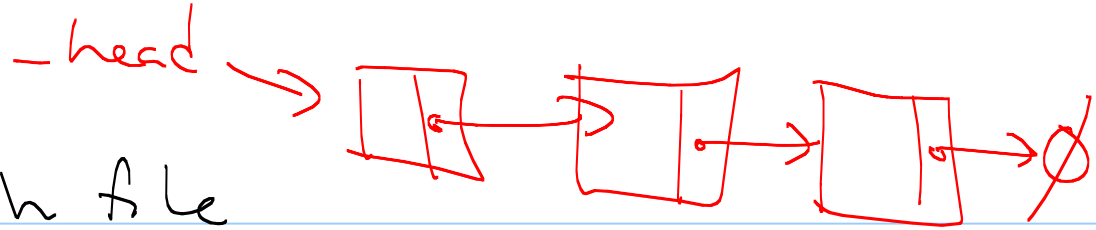
CS180 - Asymptotics

Note Title 9/22/2011

Announcements

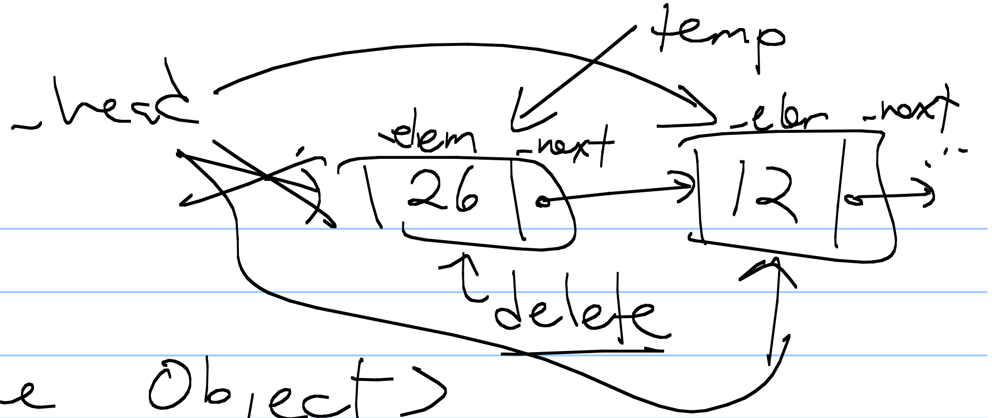
- Lecture tomorrow
- HW due Monday
- Next program will be up Saturday

~~Ch 3.2 (?)~~



Functions — .h file

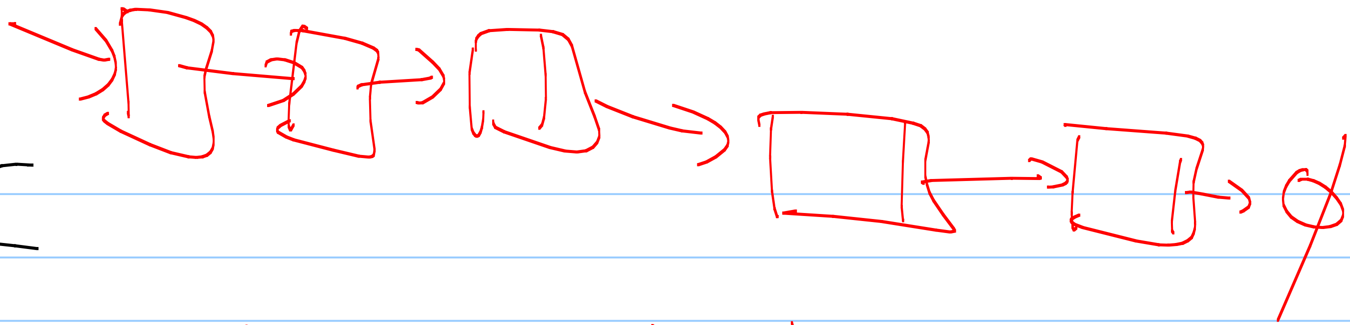
```
public:
    SLinkedList();
    ~SLinkedList();
    bool empty() const;
    const Object& front() const;
    void addFront(const Object& e);
    void removeFront();
};
```



removeFront

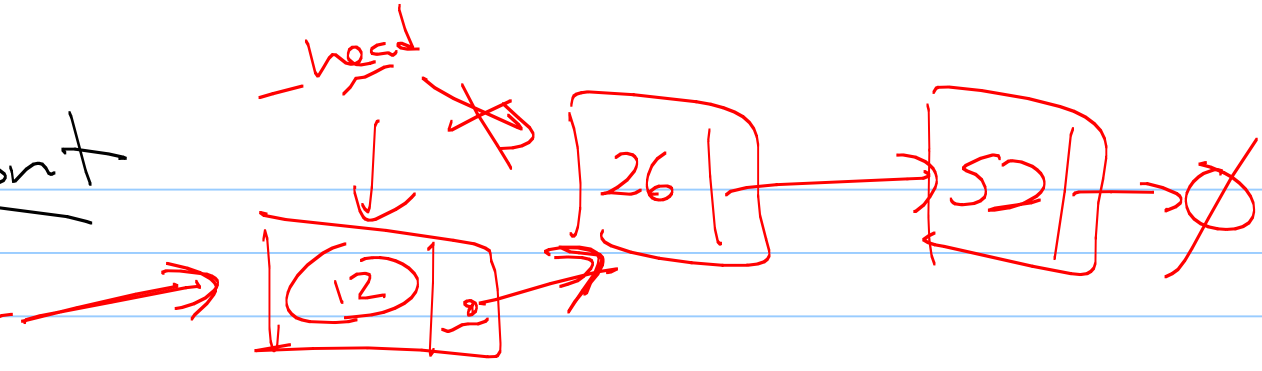
```
template <typename Object>
void    SLinkedList<Object>:: removeFront()    {
        SNode<Object> *   temp;
        temp = _head;
        _head = _head -> _next;
        delete temp;
}
```

Destructor



```
template <typename Object>
SLinkedList<Object>::~SLinkedList() {
    while (!empty()) {
        removeFront();
    }
}
```

Add Front



12

```
template <typename Object>
void SLinkedList<Object>::addFront(const Object& data){
    SNode<Object>* newguy = new SNode<Object>;

    newguy->_elem = data;
    newguy->_next = _head;
    _head = newguy;
}
```

Algorithm Analysis (Ch. 4)

How do we compare two programs?

- memory usage
- speed
- ~~interface~~
- ~~features~~
- benchmarks
- ~~cost~~

Speed

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

Examples:

- hardware
- other software on system
- OS →
- language

- input

Primitive Operations

As a way to compare algorithms in a generic way, we instead count <u>primitive operations</u>.

of 2 #s
↳ addition, subtraction, memory access, return, mult + div

In addition, we (generally) only analyze the <u>worst</u> possible running time.

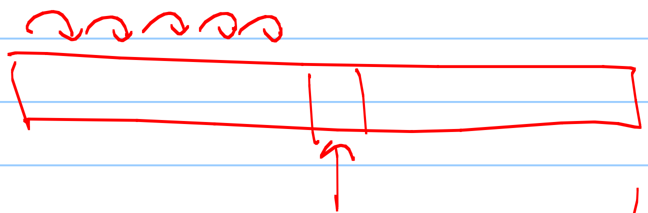
Why? guaranteeing a minimum performance

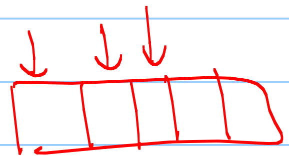
Comparing

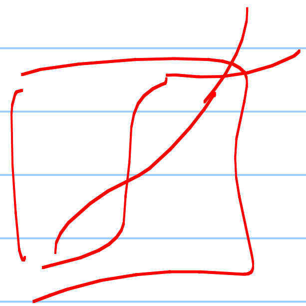
OK, so we have the worst case # of operations – usually a function of n.

How to compare?

Binary search $\sim \log_2 n$ versus linear search ← checks every element







$4 n \log_2 n$ operations

$2 n \log_2 n + 1{,}265 n$

Big-O

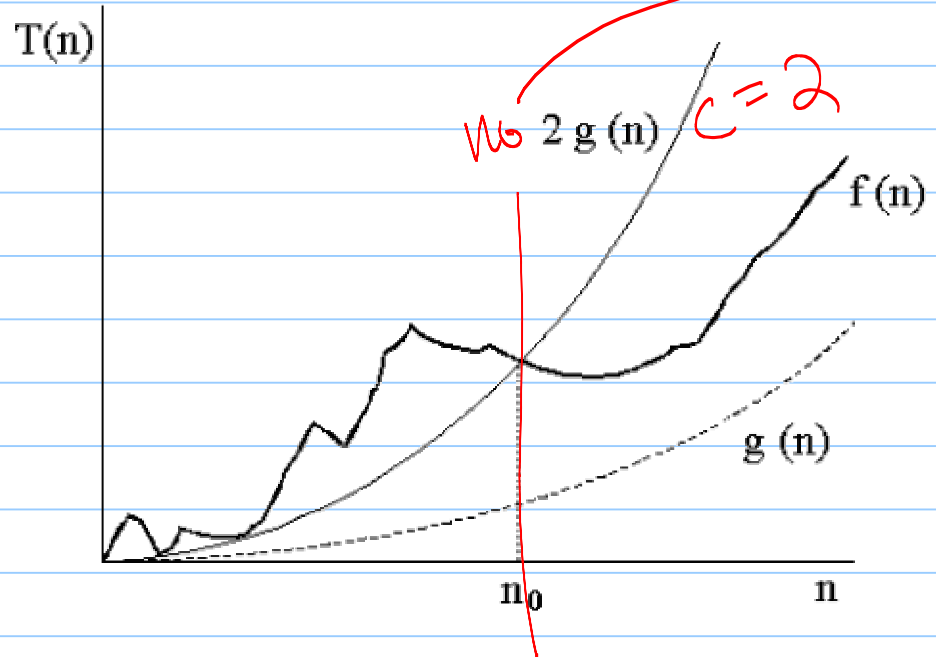
$f(n) \le g(n)$

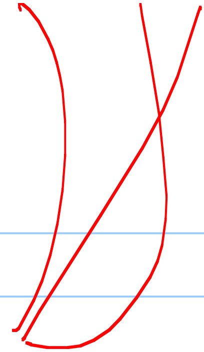
for all

We say $f(n)$ is $O(g(n))$ if $\forall n > n_0$,

$\exists\ c > 0$ such that $f(n) \le c \cdot g(n)$.

there exists





Ex: $5n$ is $O(n^2)$

$\forall n \geq \overset{n_0}{1}$, $5n \leq \overset{c}{5}n^2$

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

$5n \leq \overset{c}{5}n$

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

$16n^2 + 52 \leq 16n^2 + 52n^2 = \overset{c}{68}n^2$

$\Rightarrow$ $\underbrace{a_n x^n + a_{n-1} x^{n-1} + \dots + a_0 \text{ is } O(x^n)}$

$\log_2 a + \log_2 b = \log_2(a\,b)$

$\log_2 x^c = c\log_2 x$

Functions we will use

1. $O(1)$ - constant time
2. $O(\log n)$ - logarthmic time
   - ↳ Binary search
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