term test info

#### CSC148 fall 2013

sorting big-oh week 9

Danny Heap heap@cs.toronto.edu BA4270 (behind elevators)

http://www.cdf.toronto.edu/~heap/148/F13/ 416-978-5899

November 6, 2013



### Outline

more big-oh

### running time analysis

```
algorithm's behaviour over large input (size n) is common way
to compare performance
  constant: c \in \mathbb{R}^+ (some positive number)
logarithmic: c \log n
     linear: cn (probably not the same c)
 quadratic: cn^2
      cubic: cn^3
exponential: c2^n
   horrible: cn^n or cn!
```



### case: $\lg n$

this is the number of times you can divide n in half before reaching 1.

- refresher:  $a^b = c$  means  $\log_a c = b$ .
- ▶ this runtime behaviour often occurs when we "divide and conquer" a problem (e.g. binary search)
- ▶ we usually assume lg n (log base 2), but the difference is only a constant:

$$2^{\log_2 n} = n = 10^{\log_{10} n} \Longrightarrow \log_2 n = \log_2 10 \times \log_{10} n$$

▶ so we just say  $\mathcal{O}(\lg n)$ .





### hierarchy

Since big-oh is an upper-bound the various classes fit into a hierarchy:

$$\mathcal{O}(1)\subseteq\mathcal{O}(\lg n)\subseteq\mathcal{O}(n)\subseteq\mathcal{O}(n^2)\subseteq\mathcal{O}(n^3)\subseteq\mathcal{O}(2^n)\subseteq\mathcal{O}(n^n)$$



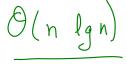
# selection sort (review?)

idea: for each position in the list, select the minimum item from that position on

### merge sort

idea: divide the list in half, (merge) sort the halves, then merge the sorted results

## quick sort



idea: choose a pivot; decide where the pivot goes with respect to the rest of the list, repeat on the partitions...

## scaling:

How well do these various sorts perform as the size of the problem (list length) increases? Time and compare.

### term test #2

- ➤ Same time as lecture, but in EX300 (A-L) and EX310 (M-Z)
- ➤ You are responsible for lecture examples, assignment 2, labs, and exercises. That's where I'll look for suitable questions.
- ► Topics include, but not limited to: binary trees, linked lists, binary search trees, recursion on trees and lists, big-oh analysis

