

Efficiency of Algorithms

- **Last time**

- Algorithm for pattern matching
- Efficiency of algorithms

- **Today**

- Efficiency of sequential search
- Data cleanup algorithms
 - Copy-over, shuffle-left, converging pointers
- Efficiency of data cleanup algorithms
- Order of magnitude $\Theta(n)$, $\Theta(n^2)$

(Time) Efficiency of an algorithm

worst case efficiency

is the *maximum* number of steps that an algorithm can take for *any* collection of data values.

Best case efficiency

is the *minimum* number of steps that an algorithm can take *any* collection of data values.

Average case efficiency

- the efficiency averaged on all possible inputs
- must assume a distribution of the input
- we normally assume uniform distribution (all keys are equally probable)

If the input has size n , efficiency will be a *function of n*

Order of Magnitude

- Worst-case of sequential search:
 - $3n+5$ comparisons
 - Are these constants accurate? Can we ignore them?
- Simplification:
 - ignore the constants, look only at the order of magnitude
 - $n, 0.5n, 2n, 4n, 3n+5, 2n+100, 0.1n+3$ are all linear
 - we say that their **order of magnitude** is n
 - $3n+5$ is order of magnitude n : $3n+5 = \Theta(n)$
 - $2n + 100$ is order of magnitude n : $2n+100 = \Theta(n)$
 - $0.1n+3$ is order of magnitude n : $0.1n+3 = \Theta(n)$
 -

Efficiency of Copy-Over

- Best case:
 - all values are zero: no copying, no extra space
- Worst-case:
 - No zero value: n elements copied, n extra space
 - Time: $\Theta(n)$
 - Extra space: n

Efficiency of Shuffle-Left

- Space:
 - no extra space (except few variables)
- Time
 - Best-case
 - No zero value:
 - no copying \implies order of $n = \Theta(n)$
 - Worst case
 - All zero values:
 - every element thus requires copying $n-1$ values one to the left
 - $n \times (n-1) = n^2 - n =$ order of $n^2 = \Theta(n^2)$ (why?)
 - Average case
 - Half of the values are zero
 - $n/2 \times (n-1) = (n^2 - n)/2 =$ order of $n^2 = \Theta(n^2)$

Efficiency of Converging Pointers Algorithm

- Space
 - No extra space used (except few variables)
- Time
 - Best-case
 - No zero value
 - No copying => order of n = $\Theta(n)$
 - Worst-case
 - All values zero:
 - One copy at each step => n-1 copies
 - order of n = $\Theta(n)$
 - Average-case
 - Half of the values are zero:
 - n/2 copies
 - order of n = $\Theta(n)$

Data Cleanup Algorithms

- Copy-Over
 - worst-case: time $\Theta(n)$, extra space n
 - best case: time $\Theta(n)$, no extra space
- Shuffle-left
 - worst-case: time $\Theta(n^2)$, no extra space
 - Best-case: time $\Theta(n)$, no extra space
- Converging pointers
 - worst-case: time $\Theta(n)$, no extra space
 - Best-case: time $\Theta(n)$, no extra space

Order of magnitude $\Theta(n^2)$

- Any algorithm that does cn^2 work for any constant c
 - $2n^2$ is order of magnitude n^2 : $2n^2 = \Theta(n^2)$
 - $.5n^2$ is order of magnitude n^2 : $.5n^2 = \Theta(n^2)$
 - $100n^2$ is order of magnitude n^2 : $100n^2 = \Theta(n^2)$

Another example

- Problem: Suppose we have n cities and the distances between cities are stored in a table, where entry $[i,j]$ stores the distance from city i to city j
 - How many distances in total?
 - An algorithm to write out these distances
 - For each row 1 through n do
 - For each column 1 through n do
 - » Print out the distance in this row and column
 - Analysis?

Comparison of $\Theta(n)$ and $\Theta(n^2)$

- $\Theta(n)$: $n, 2n+5, 0.01n, 100n, 3n+10, \dots$
- $\Theta(n^2)$: $n^2, 10n^2, 0.01n^2, n^2+3n, n^2+10, \dots$
- We do not distinguish between constants..
 - Then...why do we distinguish between n and n^2 ??
 - Compare the shapes: n^2 grows much faster than n
 - Anything that is order of magnitude n^2 will eventually be larger than anything that is of order n , no matter what the constant factors are
 - Fundamentally n^2 is more time consuming than n
 - $\Theta(n^2)$ is larger (less efficient) than $\Theta(n)$
 - $0.1n^2$ is larger than $10n$ (for large enough n)
 - $0.0001n^2$ is larger than $1000n$ (for large enough n)

The Tortoise and the Hare

Does algorithm efficiency matter??

- ...just buy a faster machine!

Example:

- Pentium Pro
 - 1GHz (10^9 instr per second), \$2000
- Cray computer
 - 10000 GHz (10^{13} instr per second), \$30million
- Run a $\Theta(n)$ algorithm on a Pentium
- Run a $\Theta(n^2)$ algorithm on a Cray
- For what values of n is the Pentium faster?
 - For $n > 10000$ the Pentium leaves the Cray in the dust..