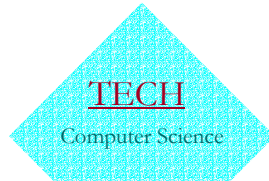


Problem: Selection

Design and Analysis: Adversary Arguments

- The selection problem
 << Ranking elements of a set in nondecreasing order, find an element rank K >>
 - Finding max and min
- Designing against an adversary
 << An algorithm playing *Information* game against an adversary >>



Design and Analysis using

Adversary arguments

- An algorithm is playing an *Information* game against an adversary.
 - The algorithm wants to get as much Information as possible in order to get as much work done as effective as possible.
 - The adversary wants to give as least Information as possible to give the algorithm *the worst case*.
 - The rule of the game is *Consistency*.
 The adversary can trick but cannot cause inconsistency in the given information.
- e.g. your algorithm needs to guess a date (a month and day) and your adversary gives yes/no answers.
 - Let's play!

Strategy for Designing against an adversary

- Assume a strong adversary!
 - the adversary will give as least information as possible
- Choose questions (or operations) as balance as possible
 - e.g. for comparison of two keys, $x > y$
 - Yes: $x > y$ and
 - No: $x \text{ not } > y$
 - should provide about the same amount information

The Selection Problem

- Find an element with rank k
 - in an array E using indexes 1 through n
 - the elements in E are assumed to be unsorted
 - where $1 \leq k \leq n$
- Finding an element with rank k is equivalent to answering the question:
 - If the array were sorted in nondecreasing order
 - which element would be in $E[k]$?
- The largest key (called max) should be $k = n$
- The smallest key (called min) should be $k = 1$
- The median key should be $k = \text{Ceiling}[n/2]$

Finding min, finding max, finding min and max

- Finding min in an unsorted array of n elements
 - require at least $n-1$ comparisons
- Finding max in an unsorted array of n elements
 - require at least $n-1$ comparisons
- Now we want to find both min and max
 - can we do better than $2(n-1)$?
 - What is the lower bound?
- Theorem: Any algorithm to find both min and max of n keys by comparison of keys must do at least $3n/2 - 2$ key comparisons in the worst case.

Proof by adversary arguments and units of information

- To *know* that a key v is max, an algorithm must know that every key other than v has lost some comparison
 - To know that a key u is min, an algorithm must know that every key other than u has won some comparison.
- If we count each win as one *unit of information*
- and each loss as one unit of information
 - Then an algorithm must have at least $2(n-1)$ units of information for finding both min and max
- We need to determine how many comparison are required (*in the worst case*) to get total $2(n-1)$ units of information.
 - The adversary to give us the *worst case* will provide as few information as possible.

The adversary strategy to give us the worst case

Status of keys x and y compared by an algorithm	Adversary response	New status	Units of new information
N, N	$x > y$	W, L	2
W, N or WL, N	$x > y$	W, L or WL, L	1
L, N	$x < y$	L, W	1
W, W	$x > y$	W, WL	1
L, L	$x > y$	WL, L	1
W, L or WL, L or W, WL	$x > y$	No change	0
WL, WL	Consistent with assigned values	No change	0

Our strategy to gain as much information

- Our algorithm can do at most $n/2$ comparisons of previously unseen keys
 - suppose for the moment that n is even
 - each of these comparisons give us 2 units of information
 - now we have n units of information
- Our algorithm need total $2(n-1) = 2n - 2$, so now we need $n - 2$ additional units of information
 - for each other comparison we gain at most one unit of information
 - so we need at least $n - 2$ additional comparisons
- In total our algorithm requires at least $n/2 + n - 2$ comparisons. For n is odd, $3n/2 - 3/2$ comparisons are needed. QED

Design an algorithm to find min and max

- Now we know the lower bound (in the worst case)
 - Can we design an algorithm to reach the lower bound?
- Exercise
 - design an algorithm to find both min and max
 - the algorithm should do at most (about) $3n/2$ comparison (in the worst case) for a problem size of n elements