# CSC 421: Algorithm Design & Analysis

# Spring 2014

## Complexity & lower bounds

- brute force
- decision trees
- adversary arguments
- problem reduction



# Lower bounds

## when studying a problem, may wish to establish a lower bound on efficiency

- binary search is O(log N) – can we do better?
- merge/quick/heap sorts are O(N log N) – can we do better?

## establishing a lower bound can tell us

- when a particular algorithm is as good as possible
- when the problem is intractable (by showing that best possible algorithm is BAD)

## methods for establishing lower bounds:

- brute force
- information-theoretic arguments (decision trees)
- adversary arguments
- problem reduction

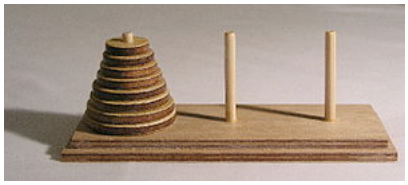
## Brute force arguments

sometimes, a problem-specific approach works

example: polynomial evaluation $p(n) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0$

- requires $\Omega(N)$ steps, since each coefficient must be processed

example: Towers of Hanoi puzzle



- can prove, by induction, that moving a tower of size N requires $\Omega(2^N)$ steps

3

## Information-theoretic arguments

can sometimes establish a lower bound based on the amount of information the solution must produce

example: guess a randomly selected number between 1 and N

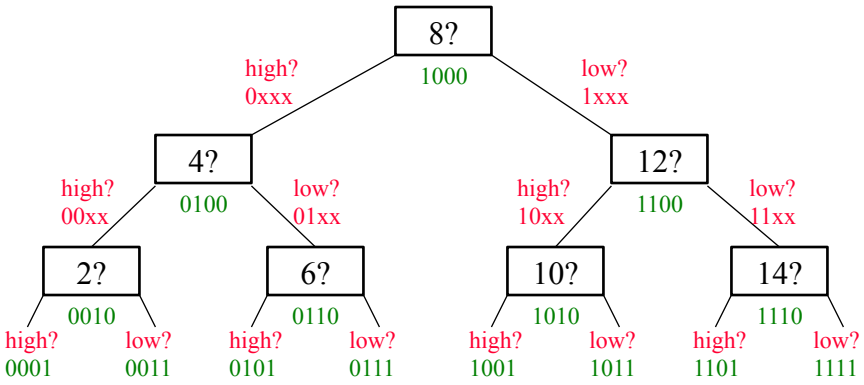
- with possible responses of "correct", "too low", or "too high"
- the amount of uncertainty is $\lceil \log_2 N \rceil$, the number of bits needed to specify the selected largest number
  e.g., N = 127 → 7 bits
- each answer to a question yields at most 1 bit of information
  if guess of 64 yields "too high," then 1st bit must be a 0 → 0xxxxxx
  if next guess of 32 yields "too low,", then 2nd bit must be 1 → 01xxxxx
  if next guess of 48 yields "too low," then 3rd bit must be 1 → 011xxxx
  . . .
- thus, $\lceil \log_2 N \rceil$ is a lower bound on the number of questions

4

## Decision trees

a useful structure for information-theoretic arguments is a *decision tree*

example: guessing a number between 1 and 15



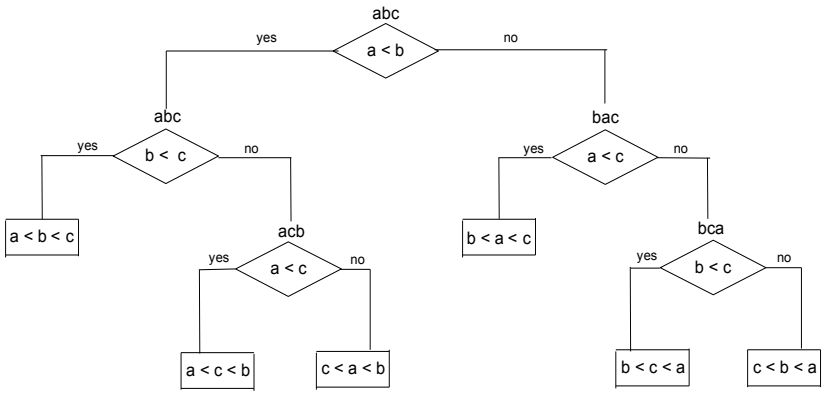

5

## Decision trees

in general, a *decision tree* is a model of algorithms involving comparisons

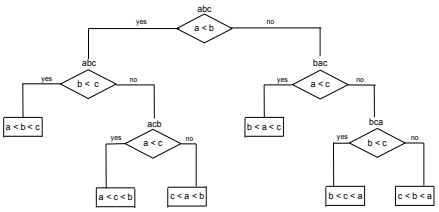
- internal nodes represent comparisons
- leaves represent outcomes

e.g., decision tree for 3-element insertion sort:




6

## Decision trees & sorting




note that any comparison-based sorting algorithm can be represented by a decision tree

- number of leaves (outcomes) ≥ N!
- height of binary tree with N! leaves ≥ $\lceil \log_2 N! \rceil$
- therefore, the minimum number of comparisons required by any comparison-based sorting algorithm ≥ $\lceil \log_2 N! \rceil$
- since $\lceil \log_2 N! \rceil \approx N \log_2 N$ (*proof not shown*), the lower bound Ω(N log N) is tight

thus, merge/quick/heap sorts are as good as it gets

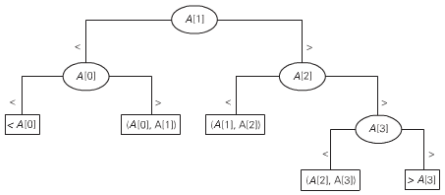
7

## Decision trees & searching

similarly, we can use a decision tree to show that binary search is as good as it gets (assuming the list is sorted)

decision tree for binary search of 4-element list:

- internal nodes are found elements
- leaves are ranges if not found




- number of leaves (ranges where not found) = N + 1
- height of binary tree with N+1 leaves ≥ $\lceil \log_2 (N+1) \rceil$
- therefore, the minimum number of comparisons required by any comparison-based searching algorithm ≥ $\lceil \log_2 (N+1) \rceil$
- lower bound Ω(log N) is tight

8

## Adversary arguments

using an *adversary argument*, you repeatedly adjust the input to make an algorithm work the hardest

example: dishonest hangman

- adversary always puts the word in a larger of the subset generated by last guess
- for a given dictionary, can determine a lower bound on guesses

example: merging two sorted lists of size N (as in merge sort)

- adversary makes it so that no list "runs out" of values (e.g., $a_i < b_j$ iff $i < j$)
- forces 2N-1 comparisons to produce $b_1 < a_1 < b_2 < a_2 < \ldots < b_N < a_N$

9

## Problem reduction

*problem reduction* uses a transform & conquer approach

- if we can show that problem *P* is at least as hard as problem *Q*, then a lower bound for *Q* is also a lower bound for *P*.

in general, to prove lower bound for P:

1. find problem *Q* with a known lower bound
2. reduce that problem to problem P
   i.e., show that can solve Q by solving an instance of P
3. then P is at least as hard as Q, so same lower bound applies

example: prove that multiplication (of N-bit numbers) is Ω(N)

1. squaring an N-bit number is known to be Ω(N)
2. can reduce squaring to multiplication: $x^2 = x * x$
3. then multiplication is at least as hard as squaring, so also Ω(N)
   (proof by contradiction) assume multiplication can be solved in O(X) where X < N
   then could square a number by multiplying it with itself → O(X) < O(N) CONTRADICTION

10

## Problem reduction example

CLOSEST NUMBERS (CN) PROBLEM: given N numbers, find the two closest numbers

1. consider the ELEMENT UNIQUENESS (EU) problem
   - given a list of N numbers, determine if all are unique (no dupes)
   - this problem has been shown to have a lower bound of Ω(N log N)

2. can reduce EU to CN
   consider an instance of EU: given numbers $e_1$, …, $e_N$, determine if all are unique
   - find the two closest numbers (this is an instance of CN)
   - if the distance between them is > 0, then $e_1$, …, $e_N$ are unique

3. this shows that CN is at least as hard as EU
   - can solve an instance of EU by performing a transformation & solving CN
   - since transformation is O(N), CN must also have a lower-bound of Ω(N log N)
     (proof by contradiction) assume CN could be solved in O(X) where X < N log N
     then, could solve EU by transforming & solving CN → O(N) +O(X) < O(N log N)
     CONTRADICTION

11

## Another example

CLOSEST POINTS (CP) PROBLEM: given N points in the plane, find the two closest points

1. consider the CLOSEST NUMBER (CN) problem
   - we just showed that CN has a lower bound of Ω(N log N)

2. can reduce CN to CP
   consider an instance of CN: given numbers $e_1$, …, $e_N$, determine closest numbers
   - from these N numbers, construct N points: ($e_1$, 0), …, ($e_N$, 0)
   - find the two closest points (this is an instance of CP)
   - if ($e_i$, 0) and ($e_j$, 0) are closest points, then $e_i$ and $e_j$ are closest numbers

3. this shows that CP is at least as hard as CN
   - can solve an instance of CN by performing a transformation & solving CP
   - since transformation is O(N), CP must also have a lower-bound of Ω(N log N)
     (proof by contradiction) assume CP could be solved in O(X) where X < N log N
     then, could solve EU by transforming & solving CP → O(N) +O(X) < O(N log N)
     this contradicts what we know about EU, so CP must be Ω(N log N)

12

## Tightness

note: if an algorithm is Ω(N log N), then it is also Ω(N)

are the Ω(N log N) lower bounds tight for CLOSEST NUMBERS and CLOSEST POINTS problems?

- can you devise O(N log N) algorithm for CLOSEST NUMBERS?
- can you devise O(N log N) algorithm for CLOSEST POINTS?

13