

Las Vegas Quicksort

- Quicksort:
 - picks target[left] as pivot
 - worst case: $O(n^2)$
 - average case: $O(n \log n)$
 - if array already sorted:
 - worst case!
- Let's pick the pivot randomly
 - target[randomInteger(left, right)]
 - worst case expected time: $O(n \log n)$
- Robin Hood effect
 - slow performance gets equally distributed among instances
 - can still be slow if you are unlucky
 - useful if you are using non-random input

© 2005-06. 324-1-132

Las Vegas Universal Hashing

- Hashing
 - worst case: $O(n)$
 - average case over random inputs: $O(1)$
- Use Robin Hood effect again
 - worst-case expected time: $O(1)$
- Idea:
 - have a set of hash functions
 - pick one randomly
 - assumption: each hash function gives good results on some inputs
 - $\text{hash}(x) = \text{hash}(y)$ happens only for $1/n$ hash functions
 - n size of hashtable
- Example of a universal set of hash functions:
 - $\text{hash}_{ij}(x) = ((i*x + j) \bmod p) \bmod n$
 - i, j : random integers $< p$
 - p prime number $>$ input size

© 2005-06. 324-1-133

Computational Complexity

- Question:
 - what is the fastest possible algorithm for a given problem?
- Example:
 - is it possible to sort faster than $O(n \log n)$?
 - we know that one can do at least as good as $O(n \log n)$
 - are there problems that can't be done in less than $O(2^n)$?
- Theoretical
 - related to theory of computation
 - question: what can one compute?
- Algorithmics gives upper bound of complexity
 - $O(n \log n)$ for sorting
- Complexity theory gives lower bound of complexity
 - ? for sorting

© 2005-06. 324-1-134

Complexity: Introductory Example

- Game of 20 questions
 - I choose a number between 1 and 1 million
 - can you find it with 20 yes-no questions?
- Algorithmics
 - find one algorithm (upper bound)
- Complexity theory
 - find lower bound
 - what is the smallest number of questions needed?
- Decision tree
 - nodes: questions
 - edges: yes or no
 - binary

© 2005-06. 324-1-135

Complexity of Sorting

- Comparison-based sorting only
 - pigeonhole sorting is faster
 - but usable only in special circumstances
- Decision tree
 - root: unsorted array
 - each leaf: a different sorted array
 - each decision tree is a sorting algorithm
- Number of leaves?
- Height?

© 2005-06. 324-1-136

Complexity of Sorting

- Lower bound: ceiling($\lg n!$)
 - $\lg n! \in \Theta(n \log n)$
- Complexity of comparison-based sorting: $O(n \log n)$
 - but not always possible to reach the lower bound
- Heapsort
 - number of comparisons less than $2 * \text{ceiling}(\lg n!)$
- Mergesort
 - number of comparisons close to lower bound
 - but requires more moving of items...

© 2005-06. 324-1-137

Adversary Arguments

- Decision trees don't always help
- Adversary (demon, devil's advocate)
 - reads input to algorithm
 - gives always that of all possible input elements that causes worst case
 - if algorithm is done, it tries to find an input element that proves algorithm wrong
- Example: finding maximum of an array
 - obvious algorithm: $n-1$ comparisons
 - decision tree: $\lceil \lg n \rceil$ is lower bound
 - adversary argument:
 - assume you have an algorithm that finds answer in $n-2$ (or fewer) comparisons
 - there is at least on input element that never lost a comparison and is not the solution
 - decide that this is going to be maximum
 - result: $n-1$ comparisons complexity

© 2005-06. 324-1-138

Linear Reduction

- Comparing the complexity classes of algorithms
 - often hard to determine precise complexity
- Idea
 - transform problem A into problem B
 - with only a constant factor of effort
 - if A can be reduced to B and vice versa, then they have the same complexity
- Example

```
int square( int x ) {
    return multiply( x, x );
}

int multiply( int x, int y ) {
    return ( square( x+ y ) - square( x - y ) ) / 4;
}
```

 - hence: multiply and square are in same complexity class
 - which class? - unknown

© 2005-06. 324-1-139

Complexity Classes P and NP

- Class P
 - all problems with polynomial algorithms
 - in $O(n^k)$ (upper bound)
 - k: any positive integer
 - examples: $O(n)$, $O(n \log n)$, $O(n^2)$, $O(n^4)$
 - considered “efficient”
- Class NP
 - “non-deterministically polynomial”
 - all problems whose solutions can be verified in polynomial time
 - non-deterministic:
 - in each step, you pick all possible paths
 - without time penalty
 - at the end, you disregard the ones that didn't work
 - complexity: $O(2^{kn})$ as upper bound

© 2005-06. 324-1-140

P versus NP

- Question: what is the relation between P and NP?
 - obvious: $P \subseteq NP$
- Big question:
 - $P = NP$????
 - no one knows...
 - but it is unlikely
 - many NP-problems are known
 - for which no P algorithm is known
- Example: Hamiltonian cycles
 - start with a node in a graph
 - visit every node exactly once
 - return to start node
 - hard to find (NP)
 - easy to verify (P)

© 2005-06. 324-1-141

Polynomial Reductions

- Transform problem A into problem B
 - using only polynomial-cost operations
- A is reduceable to B
 - means: A is easier or as easy as B
- Result
 - if A is P, and B can be polynomially reduced to A
 - B is in P, too
 - if A is in NP, and B can be polynomially reduced to A
 - B is in NP, too

© 2005-06. 324-1-142

NP-Completeness

- A problem is NP-complete
 - if it is in NP
 - and all other problems in NP can be polynomially reduced to it
- Consequence:
 - if any NP-complete problem is in P, then:
 - $P = NP$
- Method
 - prove one problem to be NP-complete
 - prove others by polynomially reducing from it

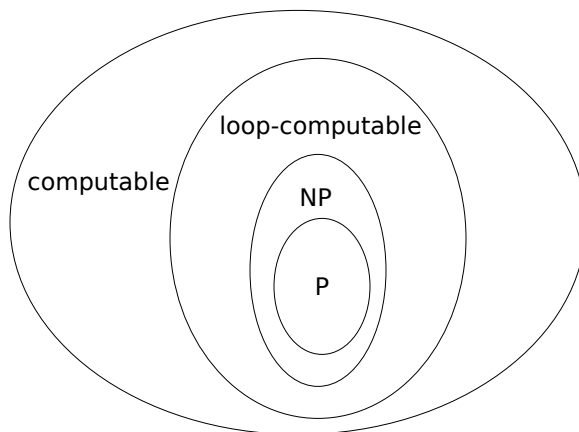
© 2005-06. 324-1-143

Examples of NP-Complete Problems

- SAT
 - usually used as the “parent” proof
 - Theorem by Cook
 - satisfiability of a boolean expression
- 3SAT
 - boolean expressions with terms of size 3
 - note: 2SAT is in P!
- 3COL
 - can a graph be 3-colored?
 - no adjacent nodes may have the same color
 - no more than three colors used
- HAM
 - Hamiltonian cycle
- TSP
 - Traveling Salesman; distances between cities
 - Find cheapest Hamiltonian cycle

© 2005-06. 324-1-144

Complexity Classes



© 2005-06. 324-1-145