Computational complexity that matters & Other models of computation

Monday, April 27, 2009

1

Summary from last week

- There exists no program solving the **Halting problem**
- There exists an infinite number of **unsolvable problems**
  - Fortunately, most problems we need to solve are solvable
- Being solvable does not mean not mean practically solvable
- Many optimization problems arising in applications take too much time to be solved exactly

Monday, April 27, 2009

2

Computational Complexity

- Programs written were based on given algorithms
  - often used libraries
  - did not formally analyze programs with respect to their efficiency
  - did run-time comparisons
  - Some studies on the impact of the input size as well as the choice of the algorithm/data structure used

Computational Complexity of a Problem

Refers to the time needed to solve the problem in terms of its input size n, independent of the machine

- For a graph, the input size is the number of nodes and the number of edges
- For detecting percolation in a grid, it is the size of the grid
- For the demon algorithm, it is the number of particles to simulate
  - Parameters driving the simulation are relevant as well
  - Exact simulation is not feasible

Measuring complexity

- Measured as **O(f(n))** – (big-Oh-notation)
  - Time is bounded by c*f(n) for some constant c
  - Time is proportional to f(n) (ignoring a constant)
- **Examples**
  - Binary search in a sorted list: $O(\log_2 n)$
  - Searching in an unsorted list: O(n)
  - Finding the connected components of a graph: proportional to the number of node and edges
  - MCL Clustering: bounded by $O(n^3)$ for a graph with n nodes; usually faster

5

Monday, April 27, 2009

Performing T(n) steps when executing 1 billion steps/second

n	**T(n) = n**	**$T(n) = n^2$**	**$T(n) = n^3$**	**$T(n) = 2^n$**
5	0.005 microsec	0.03 microsec	0.13 microsec	0.03 microsec
10	0.1 microsec	0.1 microsec	1 microsec	1 microsec
20	0.02 microsec	0.4 microsec	8 microsec	1 millisec
50	0.05 microsec	2.5 microsec	125 microsec	13 days
100	0.1 microsec	10 microsec	1 millisec	4×10^{13} years

6

Monday, April 27, 2009

Detecting percolation in a grid of size NxN

Wavefront and recursion functions gave somewhat different performance, but have the same asymptotic worst-case complexity.

What bounds the minimum and maximum number of steps the algorithms take (proportional to N)?

	Min	Max
A.	N	N
B.	N^2	N^2
C.	**N**	**N^2**
D.	$\log_2 N$	N

Dictionaries support operations on key-value pairs.

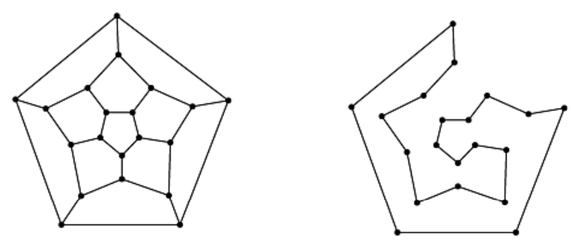
Which operations are not provided and would be inefficient?

1. Update the value associated with a given key
2. Delete the entry with the minimum key
3. Determine that a given key is in the dictionary
4. Determine the next largest key in the dictionary

A. None
B. 4
C. 1, 2, and 3
D. 2 and 4

Is there a Hamiltonian Cycle?

Given an n node graph, does there exist a cycle that visits all nodes and every node exactly once?



Why do we care to know?

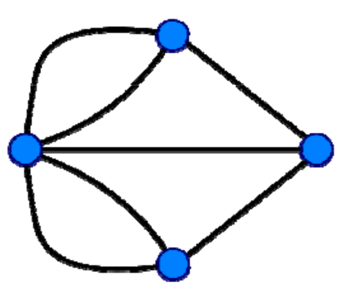
It is a basic graph property allowing one to visit all nodes in a certain way

9

Monday, April 27, 2009

Isn't it similar to the Euler Tour?

- **Euler Tour/Cycle**
  - Travel on **every edge** exactly one; nodes can be visited more than once
  - Efficient solutions exist



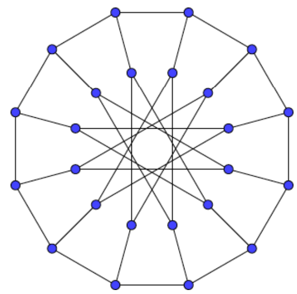
A connected, undirected graph contains an Euler tour exist if and only if ever node has even degree.

10

Monday, April 27, 2009

Hamiltonian Cycle

- Visit **every node** exactly once
- Easy to solve in exponential time (consider all permutations of the nodes, each corresponding to a cycle)
- No polynomial=efficient time solution is known



11

Monday, April 27, 2009

Formula Satisfiability

Given a Boolean formula, does there exist a true/false assignment of the variables making the formula true?

(A∨B∨¬C) ∧(A∨¬B∨D) ∧(C∨¬D)∧(¬A∨B ∨D) ∧(¬A∨¬C)

Satisfied by A=true, C=false, D=false, B=true

12

Monday, April 27, 2009

Formula Satisfiability

Given a Boolean formula, does there exist a true/false assignment of the variables making the formula true?

$(\mathbf{A}\vee\mathbf{B}\vee\neg\mathbf{C}) \wedge(\mathbf{A}\vee\neg B\vee D) \wedge(C\vee\neg\mathbf{D})\wedge(\neg A\vee\mathbf{B} \vee D) \wedge(\neg A\vee\neg\mathbf{C})$

Satisfied by A=true, C=false, D=false, B=true

- Easy to solve by trying all possible true/false assignments
- This gives exponential time
- No polynomial time solution is known
- Many practical problems can be phrased in terms of Boolean formulas

13 Monday, April 27, 2009

What do Formula Satisfiability and Hamiltonian Cycle have in common?

A fair amount with respect to their complexity.

One characteristic:

For both, we can *verify* a given solution easily.

- Given a cycle, we can easily verify if it is a Hamiltonian cycle
- Given a true-false assignment, we can easily verify if it satisfies the formula

14 Monday, April 27, 2009

Classes P and NP

P = set of all problems that can be solved in polynomial time

NP= set of all problems whose solution can be verified in polynomial time

Open question: Is P=NP?

- one the seven Millennium Problems of the Clay Mathematics Institute
- $1 Million prize for each
- http://www.claymath.org/millennium/

15

Monday, April 27, 2009

Hamiltonian Cycle and Satisfiability

- Have no known polynomial time solution
- Have straightforward exponential time solutions
- A given solution can be verified efficiently

and …

If someone comes up with a polynomial time solution for Hamiltonial Cycle or Satisfiability, then

- P=NP
- They get $1 Million

Problem having this consequence are called **NP-complete**

16

Monday, April 27, 2009

How to deal with problems for which no fast solutions exist?

- Thousands of interesting and relevant problems fall into this category – in all disciplines
- They are solved
  - Using heuristic algorithms that generate good, but not optimal solutions
  - Some problems have heuristics that work quite well, others don't
  - Probabilistic methods (Monte Carlo, Simulated annealing)
  - Looking for special cases that can be solved efficiently

17 Monday, April 27, 2009