CS 381 – FALL 2019

Week 16.2, Wed, December 4th

PSOs This Week: Review for Final Exam
Practice Final Released Today
No class on Friday

Reminder: Course Evaluation

- Please let me know what you liked and what could be improved
  - **http://www.purdue.edu/idp/courseevaluations/CE_Students.html**
  - **“NP is too hard”**
- Closes December 8th at 11:59PM
- Feedback is anonymous
- 1 Point Bonus on final exam for sending proof of completion (screenshot)
  - E-mail: jblocki@purdue.edu
  - Must use subject line: ``CS381 Evaluation Bonus”

Final Exam Logistics

- **Time:** Thursday, December 12th from 7-9PM (2 Hours)
- **Location:** STEW 130
- One (Double Sided) Page of Handwritten Notes
  - No calculators, smartphones, laptops etc…
- **Content:**
  - Heavier emphasis on recent topics (since Midterm 2)
    - Network Flow, Max-Flow Min-Cut, Reductions, P, NP, coNP, NP-Completeness
  - Cumulative: (roughly) half of the exam will focus on prior topics
    - D&C, Greedy, DP, Graph Algorithms etc...

Material to review/work through/understand

- Material covered in class
  - your notes, slides, Piazza notes
- 7 Assignments and posted solutions
- Clicker questions
- Midterms and practice midterm questions
- Practice Final Exam
- PSO Practice Problems

Analysis of Algorithms (1)

- Analyzing the asymptotic performance of an algorithm
- Deterministic worst-case analysis
- Complexity classes (from O(1) to exponential)
- Recursion and recurrence relations
  - Solving: Master Theorem, Unrolling, Recursion Trees
- Arguing correctness of an algorithm
  - Induction, Swapping, Case Analysis (e.g., DP Recurrences) etc…
- Fundamental algorithm design techniques
  - Divide and Conquer, Greedy, Dynamic Programming, Reductions
- Effective use of data structures
- Graph algorithms and graph explorations
  - DFS, BFS, Top Sort, Shortest paths, mspt, max flow (min-cut)

Analysis of Algorithms (2)

- Network Flow, Max-Flow Min-Cut
- Polynomial Time Reductions
- Decision vs Search (Self-Reductions)
- Classes NP, P, and NP-complete
- Making NP-completeness reductions
- Dealing with NP-completeness

About the exam

- Read through the questions and start with ones you feel most comfortable with
- Questions are not arranged in order of difficulty
- Don’t spend too much time on a single question
- Don’t give multiple answers. Make it clear what your final answer is.
- Yes/no is **not** an answer.
- Unless we explicitly say “no explanation,” we expect a brief/precise explanation (no detailed code)
  - Running Time, Correctness
- If there is no running time given for a question, determining the best one is part of the problem.
- We can only grade what is written, not what you were thinking

Which relationships are **true**?

1. $n! = O(n2^n)$
2. $2^{3\log n} = \Theta(n)$
3. $n! = O((n+1)!)$
4. $n^{0.9} = O(\frac{n}{\log n})$

A. 3
B. 1 and 2
C. 3 and 4
D. 2 and 3
E. All are false

T(n) = 4T(n/2) + 4n

with T(1)=1 and n a power of 4.

Its solution is …

A. $O(n^{1/2})$
B. $O(n)$
C. $O(n \log n)$
D. $O(n^2)$
E. $O(n^4)$

Master Theorem:

a=4, b=2, c=1

$\log_b a > c \rightarrow O\left(n^{\log_b a}\right)$

A and B are two decision problems.

- Alice shows that both A and B are in class NP.
- Bob shows that problem B is NP-complete.
- Charlie shows that $A \leq_{poly} B$.

Which of the following claims **can be concluded**?

A. A polynomial time solution for problem A implies a polynomial time solution for problem B.
B. A polynomial time solution for problem A implies P=NP.
C. Problem A is NP-complete.
D. Problem B is NP-Complete.
E. None of the above.

Which of the problems listed below **can be verified** in polynomial time?

P1: Verify that a given graph G has a clique of size k

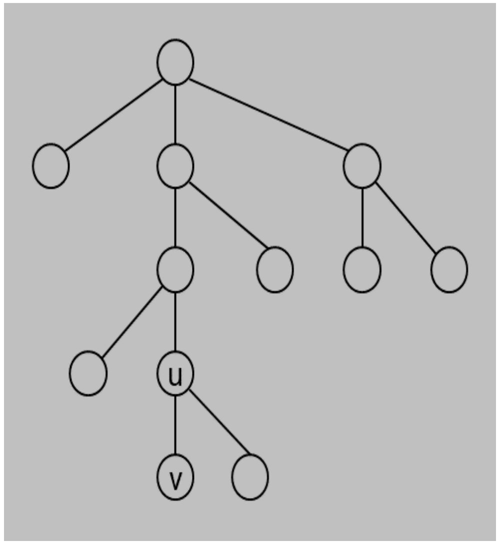
P2: Verify that a given graph G contains a simple path of length n-1

P3: Verify that a given number is not prime.

A. P1
B. P2
C. P3
D. All

Independent Set on Trees

Independent set on trees. Given a tree, find a maximum cardinality subset of nodes such that no two share an edge.



Hint. A tree on at least two nodes has at least two leaf nodes.

degree = 1

Greedy Algorithm:

0) Initialize Indep Set $S := \{\}$

1) While G is not empty

   a) find a leaf node v

   b) update $S := S \cup \{v\}$ (add v to S)

   c) update G:=G-v-N(v) (delete v and neighbors of v)

2) Return S

Claim: S is an independent set

Proof: whenever add v then we remove all incident nodes from G

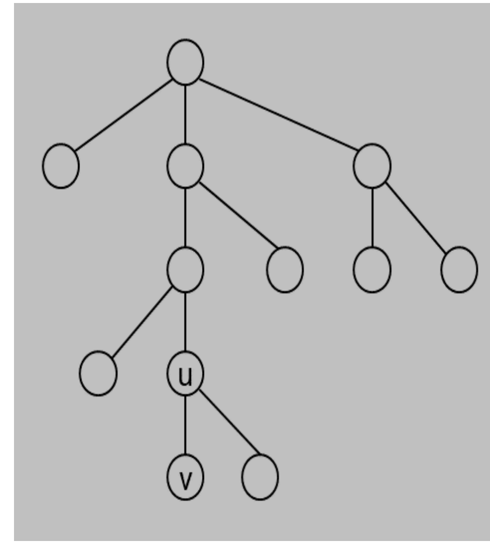
Independent Set on Trees

Greedy Algorithm:

0) Initialize Indep Set $S \coloneqq \{\}$

1) While G is not empty
   a) find a leaf node v
   b) update $S := S \cup \{v\}$ (add v to S)
   c) update G:=G-v-N(v) (delete v and neighbors of v)

2) Return S



Claim: S is maximum cardinality independent set

Proof: Let $v_1,\ldots,v_k$ be nodes in S and suppose (for contradiction) $S^* = w_1, \ldots, w_{k^*}$ is a larger maximum cardinality independent set ---

Tiebreak: maximize match with S i.e., $w_1, \ldots, w_r = v_1, \ldots, v_r$ for maximum r.

We have $v_{r+1} \notin S^*$, but v_{r+1} is incident to some node w_j in S^*

(otherwise we can simply add v_{r+1} to S^*)

Swap: $S' = \{v_{r+1}\} \cup \left(S^* \backslash \{w_j\}\right)$

Observation: S' is an independent set (since v_{r+1} is leaf in $G - \bigcup_{i \le r}(v_i \cup N(v_i))$)

(Contradicts choice of S^*)

G is a directed, weighted graph representing a flow network.
All edge weights are **unique**.
The maximum flow one can push from s to t is M.

The flow over the edges for achieving M is always unique.

A. True
B. False

G is an undirected graph.
You need to determine whether G contains two vertex disjoint cliques of size 4?

What class does the problem belong to?
Give the most precise class.

A. P
B. NP-Complete
C. NP

Which problems are in P?

1. 2-SAT
2. 3-SAT
3. Longest path in a dag
4. Hamiltonian path in a graph with at most 4n edges
5. Vertex cover in tree
6. Partition problem on n elements having identical value

A. 1 and 3
B. 3 and 4
C. 1, 3, 4 and 5
D. 1, 3, 5, and 6
E. All but 3-SAT

Every problem in class NP can be solved in exponential time.

A. True
B. False
C. True for most, unknown for some

3SAT: Decision vs Search

Suppose that we have an oracle O which solves the decision version of the 3SAT problem i.e., $\mathrm{O}(\varphi) = 1$ if φ is satisfiable otherwise $\mathrm{O}(\varphi) = 0$.

Develop an algorithm to find a satisfying assignment after making polynomially many queries to O.

Set $\varphi_0 = \varphi$
If $\mathrm{O}(\varphi_0) = 0$ **print** "No Satisfying Assignment" and **QUIT**
For (i=1 to n)
    If $\mathrm{O}(\varphi_{i-1} \wedge (x_i \vee x_i \vee x_i)) = 1$
        $\varphi_i := \varphi_{i-1} \wedge (x_i \vee x_i \vee x_i)$
        print "$x_i = 1$"
    Else
        $\varphi_i := \varphi_{i-1} \wedge (\overline{x_i} \vee \overline{x_i} \vee \overline{x_i})$
        print "$x_i = 0$"

Running Time?

Correctness?