# "CS 374": Algorithms and Models of Computation, Fall 2015

# Final Exam – December 11, 2015

| | |
|---|---|
| Name: | |
| NetID: | |

| Section | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| Time | 9am | 10am | 11am | 12 noon | 1pm | 1pm | 2pm | 2pm | 3pm | 3pm |
| Room | 1304 | 1304 | 1304 | 1304 | 1304 | 1105 | 1304 | 1105 | 1304 | 1105 |

| # | 1 | 2 | 3 | 4 | 5 | 6 | Total |
|---|---|---|---|---|---|---|---|
| Score | | | | | | | |
| Max | 20 | 10 | 12 | 12 | 10 | 16 | 80 |
| Grader | | | | | | | |

---

- Please *clearly PRINT* your name and your NetID in the boxes above.
- *CIRCLE YOUR SECTION*
- This is a closed-book, closed-notes, closed-electronics exam. If you brought anything except your writing implements, put it away for the duration of the exam. In particular, you may not use *any* electronic devices.
- **Please read the entire exam before writing anything.** Please ask for clarification if any question is unclear.
- There are six problems, worth a total of 80 points. A list of **NP**-Complete problems is available on the final page. And also the rubric for dynamic programming.
- A correct algorithm is much better than a fast but incorrect algorithm. At the same time brute-force exponential time algorithms are not worth anything when there is an easy polynomial-time algorithm.
- **You have 180 minutes (3 hours).**
- If you run out of space for an answer, continue on the back of the page, or on the blank page at the end of this booklet, **but please tell us where to look.**
- **Proofs are required only if we specifically ask for them. This policy applies even for greedy algorithms.** However, giving a brief explanation may be helpful to us in understanding your algorithm and thought process.

---

# 1 Multiple Choice (20 points)

In each of the problems below, use check marks to select one or more choices as directed. Except for the first problem, each option fetches one point if it is correctly selected/not selected. Ambiguously marked options will be considered to be marked incorrectly.

If no options are selected in a problem, the problem will be considered unattempted and no points will be awarded.

1. Select one option. The tightest asymptotic upper bound for the sum $\sum_{i=1}^{n} \log^2 n$ is: [1 point]

☐ A. $O(1)$

☐ B. $O(\log^2 n)$

☐ C. $O(\log^3 n)$

☐ D. $O(n)$

☐ E. $O(n \log^2 n)$

☐ F. $O(n \log^3 n)$

☐ G. $O(n^2)$

☐ H. $O(2^n)$

2. Suppose $L_1, L_2 \subseteq \Sigma^*$ such that $L_1$ is regular and $L_2$ is not. Which of the following are necessarily TRUE? [4 points]

☐ A. $L_1 \cup L_2$ is non-regular

☐ B. $\overline{L_2}$ is non-regular

☐ C. $L_1 \cap L_2$ is regular

☐ D. $L_1 - L_2$ is non-regular

3. Select all statements that are TRUE. The class of recursively enumerable languages is: [5 points]

☐ A. closed under union

☐ B. closed under intersection

☐ C. closed under complement

☐ D. closed under Kleene star

☐ E. closed under infinite union

4. Select all statements that are TRUE. The class **NP** is: [3 points]

☐ A. closed under union

☐ B. closed under intersection

☐ C. closed under Kleene star

5. Select all statements that are TRUE. [7 points]

☐ A. If $L$ is an **NP**-Complete language and $L'$ is an **NP**-Hard language then $L \leq_P L'$.

☐ B. If $L$ is **NP**-Complete then $L \leq_P SAT$ and $SAT \leq_P L$.

☐ C. The following is considered possible (but unlikely):
$SAT \in \mathbf{P}$ but $\mathbf{P} \neq \mathbf{NP}$.

☐ D. If $L$ and $L'$ are both in **P** then $L \leq_P L'$.

☐ E. Every decidable language is in **NP**.

☐ F. Every regular language is in **P**.

☐ G. Every **NP** language is recursively enumerable.

## 2 Regular Languages (10 points)

Let $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ and $M_3 = (Q_3, \Sigma, \delta_3, s_3, F_3)$ be three DFAs. Describe formally a NFA $M = (Q, \Sigma, \delta, s, F)$ that accepts the language $(L(M_1) - L(M_2)) \cdot L(M_3)$.

**English Explanation (4 points):** Briefly explain how your construction works.

**Formal Specification (6 points)**

**States:** $Q =$

**Transition function:** $\delta$ is defined by:

**Start State:** $s$ is:

**Accepting States:** $F =$

## 3 One way streets (12 points)

The police department in the city of Urbana has made all streets one-way. The mayor contends that there is still a way to drive legally from any intersection in the city to any other intersection, but the opposition is not convinced. The city needs an algorithm to check whether the mayor's contention is indeed true.

- (5 points) Formulate this problem graph-theoretically, and describe an efficent algorithm for it.
- (7 points) Suppose it turns out that the mayor's original claim is false. Call an intersection $u$ *good* if any intersection $v$ that you can reach from $u$ has the property that $u$ can be reached from $v$. Now the mayor claims that over 95% of the intersections are good. Describe an efficient algorithm to verify her claim. Your algorithm should basically be able to find all the good intersections.

Ideally your algorithms for both parts should run in linear time. You will receive 3 points and 4 points for the two parts respectively for a polynomial-time algorithm.

## 4 Common Supersequence (12 points)

Let $X = x_1, x_2, \ldots, x_r$, $Y = y_1, y_2, \ldots, y_s$ and $Z = z_1, z_2, \ldots, z_t$ be three sequences. A common *supersequence* of $X$, $Y$ and $Z$ is another sequence to which $X$, $Y$ and $Z$ are subsequences. Suppose $X = a, b, d, c$ and $Y = b, a, b, e, d$ and $Z = b, e, d, c$. A simple common supersequence of $X$, $Y$ and $Z$ is the concatenation of $X$, $Y$ and $Z$ which is $a, b, d, c, b, a, b, e, d, b, e, d, c$ and has length 13. A shorter one is $b, a, b, e, d, c$ which has length 6. Describe an efficient algorithm to compute the *length* of the shortest common supersequence of three given sequences $X$, $Y$ and $Z$.

## 5 Balloon (10 points)

A balloon is an undirected graph on an even number of nodes, say $2n$, in which $n$ of the nodes form a cycle and the remaining $n$ vertices are connected in a "tail" that consists of a path joined to one of the nodes in the cycle. See figure below for a balloon with 8 nodes.

Given a graph $G$ and an integer $k$, the BALLOON problem asks whether or not there exists a subgraph which is a balloon that contains $2k$ nodes. Prove that BALLOON is **NP**-Complete. You need to describe a reduction and justify its correctness. The justification should be concise.

## 6 Decidability and Recognizability (16 points)

Consider the language $L_{\text{OH}} = \{z \mid M_z \text{ halts on at least one input string}\}$. Note that for $z \in L_{\text{OH}}$, it is not necessary for $M_z$ to *accept* any string; it is enough that it *halts* on (and possibly rejects) some string.

- (8 points) Show that $L_{\text{OH}}$ is recursively enumerable by describing a procedure/TM at a high-level that accepts $L_{\text{OH}}$. You don't have to prove the correctness of your procedure.
- (8 points) Show that $L_{\text{OH}}$ is not recursive (that is, undecidable). Note that Rice's theorem does not directly apply so consider a proof via a reduction. You need to briefly justify the correctness of your reduction.

This page for extra work.

## List of some useful NP-Complete Problems/Languages

- **SAT** $\{\varphi \mid \varphi$ is a boolean formula in CNF form and is satisfiable$\}$
- **3SAT** $\{\varphi \mid \varphi$ is a boolean formula in CNF form with exactly 3 literals per clause and is satisfiable$\}$
- **Circuit-SAT** $\{C \mid C$ is a boolean circuit such that there is a setting of values to the inputs to $C$ that make it evaluate to TRUE$\}$
- **Independent Set** $\{\langle G, k\rangle \mid$ Graph $G = (V, E)$ has a subset of vertices $V' \subseteq V$ of size at least $k$ such that no two vertices in $V'$ are connected by an edge$\}$
- **Vertex Cover** $\{\langle G, k\rangle \mid$ Graph $G = (V, E)$ has a subset of vertices $V' \subseteq V$ of size at most $k$ such that every edge in $E$ has at least one of its endpoints in $V'\}$
- **Clique** $\{\langle G, k\rangle \mid$ Graph $G$ has a complete subgraph of size at least $k\}$
- **3Color** $\{\langle G\rangle \mid$ The vertices of graph $G$ can be colored with 3 colors so that no two adjacent vertices share a color$\}$
- **Coloring** $\{\langle G, k\rangle \mid$ The vertices of graph $G$ can be colored with $k$ colors so that no two adjacent vertices share a color$\}$
- **Hamiltonian Cycle** $\{\langle G\rangle \mid$ Directed graph $G$ contains a directed cycle visiting each vertex exactly once$\}$
- **Undir Hamiltonian Cycle** $\{\langle G\rangle \mid$ Undirected graph $G$ contains a cycle visiting each vertex exactly once$\}$
- **Hamiltonian Path** $\{\langle G\rangle \mid$ Directed graph $G$ contains a directed path visiting each vertex exactly once$\}$
- **Undir Hamiltonian Path** $\{\langle G\rangle \mid$ Undirected graph $G$ contains a path visiting each vertex exactly once$\}$

**Rubric for all dynamic programming problems:** Thus rubric is out of 10 points. Will be adjusted slightly and appropriately if the problem is not for 10 points.

- 6 points for a correct recurrence, described either using mathematical notation or as pseudocode for a recursive algorithm.
  - + 1 point for a clear **English** description of the function you are trying to evaluate. (Otherwise, we don't even know what you're *trying* to do.) **Automatic zero if the English description is missing.**
  - + 1 point for stating how to call your function to get the final answer.
  - + 1 point for base case(s). $-\frac{1}{2}$ for one *minor* bug, like a typo or an off-by-one error.
  - + 3 points for recursive case(s). $-1$ for each *minor* bug, like a typo or an off-by-one error. **No credit for the rest of the problem if the recursive case(s) are incorrect.**
- 4 points for details of the dynamic programming algorithm
  - + 1 point for describing the memoization data structure
  - + 2 points for describing a correct evaluation order. If you use nested loops, be sure to specify the nesting order.
  - + 1 point for time analysis
- It is *not* necessary to state a space bound.
- For problems that ask for an algorithm that computes an optimal *structure*—such as a subset, partition, subsequence, or tree—an algorithm that computes only the *value* or *cost* of the optimal structure is sufficient for full credit.
- Official solutions usually include pseudocode for the final iterative dynamic programming algorithm, but this is not required for full credit. If your solution includes iterative pseudocode, you do not need to separately describe the recurrence, data structure, or evaluation order. (But you still need to describe the underlying recursive function in English.)
- Official solutions will provide target time bounds. Algorithms that are faster than this target are worth more points; slower algorithms are worth fewer points, typically by 1 or 2 points for each factor of $n$.

  Focus on obtaining a correct algorithm rather than obtaining the best time bound.