CS 341, Fall 2020 A. Lubiw, A. Storjohann

# EXAM

Released: Thursday December 10, 8 AM.
DUE: Wednesday December 16, 11:59 PM.

## IMPORTANT INSTRUCTIONS

**Start each question on a new page and answer the questions in order. If you handwrite, make it legible!**

For an **algorithm** you must include: a description in English; pseudocode (unless the algorithm is really simple); correctness proof; and runtime analysis. Always try for the fastest algorithm you can.

For an **NP-completeness proof** you must include:
(i) a proof that the problem lies in NP
(ii) a statement (with $\leq_P$) about which problem you will reduce to which
(iii) your many-one reduction
(iv) a proof of correctness
(v) a justification that your reduction takes polynomial time (in most cases, this can be brief).



You may assume that the following problems are NP-complete: 3-SAT, Clique, Independent Set, Vertex Cover, Hamiltonian cycle/path (both directed and undirected), Subset Sum.





## YOU MUST HANDWRITE AND SIGN THE FOLLOWING STATEMENT

(e.g., write it on paper, take a photo, and include that with your Crowdmark submission)

I promise, on my honour, that the work on this exam is my own. Apart from the course materials and private Piazza questions, I have not given or received any assistance on this exam.

1. **Short Answer** [15 marks]

   (a) [3 marks] Let $G = (V, E)$ be an undirected graph with $k$ connected components. Given a new edge $(u, v)$ for $u, v \in V$, how may connected components can $G' = (V, E \cup \{(u, v)\})$ have? Briefly justify your answer.

   (b) [3 marks] True or False: If a problem in NP is shown to be in P, then P=NP. Briefly justify your answer.

   (c) [3 marks] Consider the following problem.

   Given a program $P$ does it output 1 when run on no input? (YES or NO)

   FACT: this problem is undecidable.

   What is wrong with the following algorithm for the problem: Scan the code and see if there is a statement of the form "PRINT(1)" (in the appropriate syntax for the language). If there is, then output YES; otherwise output NO.

   (d) [3 marks] Give a very short program that takes no input and does not halt.

   (e) [3 marks] What is wrong with the following argument that the prime factors of a number $n$ can be found in polynomial time?

   **Input:** a number $n$.
   **If** $n$ is prime **then** output $n$ **else**
   $i := 2$
   **while** $i < n$ **do**
   **if** $i$ divides $n$ and $i$ is prime **then** output $i$
   $i := i + 1$

   This algorithm is correct. Since there is a polynomial time algorithm to test if a number is prime, the algorithm runs in polynomial time.

2. [10 marks] **Divide-and-Conquer.** In a binary computer, integers are stored using their binary encoding. For example, the binary encoding of integers of a single decimal digit are given by the following table $\texttt{binary}[0 \ldots 9]$:

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\texttt{binary}[x]$ | 0 | 1 | 10 | 11 | 100 | 101 | 110 | 111 | 1000 | 1001 |

When a program reads in a decimal integer from a file it needs to convert from the decimal encoding to the binary encoding. This question is about an efficient algorithm to do the conversion. Assume we have functions $\texttt{Add}(a, b)$ and $\texttt{Multiply}(a, b)$ that take as input two binary numbers $a$ and $b$ and return their sum and product as binary numbers, respectively. $\texttt{Add}(a, b)$ runs in time $O(\ell)$ and $\texttt{Multiply}(a, b)$ runs in time $O(\ell^{\log_2 3})$ where $\ell$ is a bound on the number of bits in $a$ and $b$.

(a) [4 marks] We begin with a helper function $\texttt{PowOfTenToBin}(n)$ that takes as input a number $n$ and outputs the binary representation of $10^n$. Assume that $n$ is a power of 2.

```
PowOfTenToBin(n)
  if n = 1 then
    return 1010
  else
    a := ???
    return Multiply(a, a)
```





i. [2 marks] Fill in the missing details, that is, complete the line $a :=$ ???.

ii. [1 mark] Give a recurrence relation for the running time $T(n)$ of the algorithm.

iii. [1 mark] Give a big-oh expression for the run time. You may use the Master Theorem.



(b) [6 marks] Now we want to convert any nonnegative integer $x$ with $n$ decimal digits (where $n$ is a power of 2) to binary. The integer $x$ is given as an array $A[1 \ldots n]$ that contains the decimal digits of $x$ in order from most significant to least significant. For example, $A[1 \ldots 4] = [0, 1, 5, 3]$ corresponds to the integer $x = 153$. The following algorithm does the conversion:

```
DecToBin(A, n)
  if n = 1 then
    return binary[A[1]]
  else
    A_L[1 ... n/2] := A[1 ... n/2]
    A_R[1 ... n/2] := A[n/2 + 1 ... n]
    return ???
```

i. [4 marks] Fill in the missing details, that is, complete the line **return** ???.

ii. [1 mark] Give a recurrence relation for the running time $T(n)$ of the algorithm. Note that the lookup in the table $\texttt{binary}$ has cost $O(1)$ time.

iii. [1 mark] Give a big-oh expression for the run time. You may use the Master Theorem.

3. [10 marks] **Greedy.** Suppose you are given the positions of $n$ geese along a line, $g_i \in \mathbb{R}, i = 1, \ldots, n$, and the positions of $n$ goose nests along the same line, $h_i \in \mathbb{R}, i = 1, \ldots, n$. The goal is to assign each goose to a nest, with only one goose per nest, so that you minimize the maximum distance that any goose must travel to get to its nest. The positions of the geese and the nests are not in sorted order, but the goose positions are distinct and the nest positions are distinct.

   First consider the following greedy algorithm: Find a pair $g_i, h_j$ that minimizes $|g_i - h_j|$. Assign goose $i$ to nest $j$, and recurse on the remaining $n - 1$ geese and nests.

   (a) [2 marks] Give an example to show that this greedy algorithm does not always produce an optimal solution.

   Next, consider the following algorithm: Sort the goose positions and the nest positions to get lists $g_{i_1} \leq g_{i_2} \leq \cdots \leq g_{i_n}$ and $h_{j_1} \leq h_{j_2} \leq \cdots \leq h_{j_n}$. Assign goose $i_k$ to nest $j_k$ for $k = 1..n$.

   To prove that this algorithm is correct, we need the following Lemma.

   **Lemma.** Consider an assignment $A$ of geese to nests in which two geese are assigned to nests out of order, i.e., there are two geese $i_a$ and $i_b$ with $a < b$ and two nests $j_k$ and $j_\ell$ with $k < \ell$ such that goose $i_a$ is assigned to nest $j_\ell$ and goose $i_b$ is assigned to nest $j_k$. Let $A'$ be the assignment that is the same as $A$ except that goose $i_a$ is assigned to nest $j_k$ and goose $i_b$ is assigned to nest $j_\ell$. Then $M(A') \leq M(A)$ where $M()$ is the maximum distance that a goose must travel to its assigned nest.

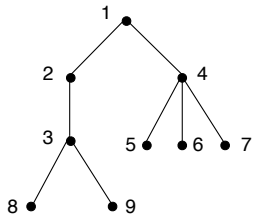
   You do not need to prove this lemma.



   (b) [8 marks] Using the lemma, give an exchange proof to show that the algorithm is correct.

4. [10 marks] **Dynamic Programming.** This question is about finding the minimum size of a vertex cover in a tree. Suppose the tree has vertices $1, 2, \ldots, n$ ordered in such a way that if $j$ is a child of $i$ then $j > i$. In particular, the root of the tree is vertex 1. Suppose we have, for each $i$, a list $D[i]$ of the children of vertex $i$.

   (a) [2 marks] Show that this tree has a vertex cover of size 3:




   There is a dynamic programming algorithm to find the minimum size vertex cover of a tree. For each $i$, let $T_i$ be the subtree that is rooted at vertex $i$ and contains all the descendants of vertex $i$. (For example, in the above tree, $T_2$ consists of vertices $2, 3, 8, 9$.) For each $i$ we find three values:

   $C^{\text{in}}[i]$, the minimum size of a vertex cover of $T_i$ that contains $i$
   $C^{\text{out}}[i]$, the minimum size of a vertex cover of $T_i$ that does not contain $i$
   $C[i]$, the minimum size of a vertex cover of $T_i$

   We solve these subproblems in order $i = n, n-1, \ldots, 1$.



   (b) [4 marks] Fill in the following high-level pseudocode. You may use expressions like $\sum_{j \in D[i]} \cdots$ instead of explicitly writing the loop.



   **for** $i = n$ **downto** 1 **do**
   **if** $i$ has no children **then** $C^{\text{in}}[i] := 1$; $C^{\text{out}}[i] := 0$; $C[i] := 0$; **else**

   $C^{\text{in}}[i] :=$

   $C^{\text{out}}[i] :=$

   $C[i] :=$

   (c) [2 marks] Prove that your algorithm correctly finds the minimum size of a vertex cover.

   (d) [2 marks] Analyze the runtime of the algorithm. Be sure to include the time to compute any expression of the form $\sum_{j \in D[i]}$.

5. [10 marks] **Graph Algorithms.** Let $G = (V, E)$ be an undirected graph with $n$ vertices and $m$ edges, and let $T \subseteq E$ be a spanning tree of $G$. A *swap* operation on $T$ replaces one edge of $T$ by an edge of $E$ such that the result is again a spanning tree. Give an algorithm that takes as input a graph $G$ and two spanning trees $T_1$ and $T_2$ and finds a sequence of swaps to convert $T_1$ to $T_2$ such that the number of swaps in the sequence is the minimum possible.

   **Note:** This is a harder question. Your answer will be marked based on the runtime of your algorithm and the elegance of your answer. If you are short of time, or having difficulties, we recommend skipping this question.

6. [15 marks] **NP-completeness**

(a) [7 marks] Prove that the following Scenic Path problem is NP-complete:

**Scenic Path**
**Input:** An undirected graph $G = (V, E)$, and number $k$, and for every edge $e \in E$, a *scenic value* $s(e) \in \mathbb{R}$.
**Output:** Is there a simple path in $G$ such that the sum of the scenic values of the edges in the path is $\geq k$? (YES or NO)

(b) [8 marks] Consider the following SAT variant:

**SAT-$\frac{2}{3}$**
**Input:** A SAT formula in Conjunctive Normal Form, with $n$ variables and $m$ clauses.
**Output:** Is there a truth-value assignment to the variables that satisfies at least $\frac{2}{3}$ of the clauses (i.e., that satisfies $\geq \frac{2}{3}m$ clauses)? (YES or NO)

i. [1 mark] What is the output for $(x_1 \vee x_2) \wedge (x_3) \wedge (\neg x_3)$?

ii. [7 marks] Prove that SAT-$\frac{2}{3}$ is NP-complete.