HW 11 (due Friday, at noon, December 7, 2018)

CS 473: Algorithms, Fall 2018 Version: **1.1**

Unlike previous homeworks, this homework should be submitted individually – it is **highly** recommended that you do not collaborate in this homework (but frankly, do whatever you want – as long as it is not outright copying/cheating[1]). If you collaborate, you need to provide all relevant details in your submission.

Note, that this homework is based on earlier semester final – it is intended for you to learn for the exam. It would be lightly graded by the TAs (0 points: not submitted, 25 points: submitted but majority of questions not answered, etc). This homework would be worth only one question (previous homeworks were each 3 questions. In particular, you should submit only a single pdf file for this homework.

Solutions for this homework would not be provided.

1 (8 PTS.) Multiple choice

For each of the questions below choose the most appropriate answer. No **IDK** credit for this question!

1.A. Consider a problem X that is NPC and a problem Y which is in P. There always exists a polynomial time reduction from X to Y.

False: ☐ **True**: ☐ **Answer depends on whether P = NP**: ☐

1.B. A graph represented using n vertices and m edges, can be encoded into a binary string with $O(m \log\log n)$ bits.

False: ☐ **True**: ☐ **Answer depends on whether P = NP**: ☐

1.C. Given a graph G. Deciding if there is an independent set X in G, such that $\mathsf{G} \setminus X$ (i.e., the graph G after we remove the vertices of X from it) has k vertices can be solved in polynomial time.

False: ☐ **True**: ☐ **Answer depends on whether P = NP**: ☐

1.D. Given a 4SAT formula F with m clauses (i.e., CNF formula where every clause has exactly four literals), there is always an assignment to its variables that satisfies at least $(31/32)m$ clauses.

False: ☐ **True**: ☐ **Answer depends on whether P = NP**: ☐

1.E. Given a graph G, with n vertices, deciding if contains a clique made out of $\lceil n^{1/3} \rceil$ vertices is NP-Complete.

False: ☐ **True**: ☐ **Answer depends on whether P = NP**: ☐

1.F. Given a bipartite graph $\mathsf{G} = (L \cup R, \mathsf{E})$, deciding if there are k vertices $v_1, \ldots, v_k \in L$ such that every vertex of R is connected to one of these k vertices can be done in polynomial time.

False: ☐ **True**: ☐ **Answer depends on whether P = NP**: ☐

1.G. Given a directed graph G with positive weights on the edges, and a number k, finding if there is a walk in G from s to t (two given vertices of G) with weight $\geq k$, can be done in polynomial time.

False: ☐ **True**: ☐ **Answer depends on whether P = NP**: ☐

[1]Just do not do anything mind boggling stupid (like submitting identical or similar solutions), OK? If you do not have time to do this homework, then do not do it.

1.H. Given an undirected graph G, computing the smallest size subset of edges $X \subseteq \mathsf{E}(\mathsf{G})$, such that any vertex in G is adjacent one of the edges of X, can be done in polynomial time.

False: ☐ **True:** ☐ **Answer depends on whether $\mathbf{P} = \mathbf{NP}$:** ☐

---

2 (16 PTS.) Short Questions.

2.A. (8 PTS.) Give a tight asymptotic bound for each of the following recurrences.

(I) (4 PTS.) $A(n) = A(n-3) + A(1) + \log n$, for $n > 2$ and $A(1) = A(2) = 1$.

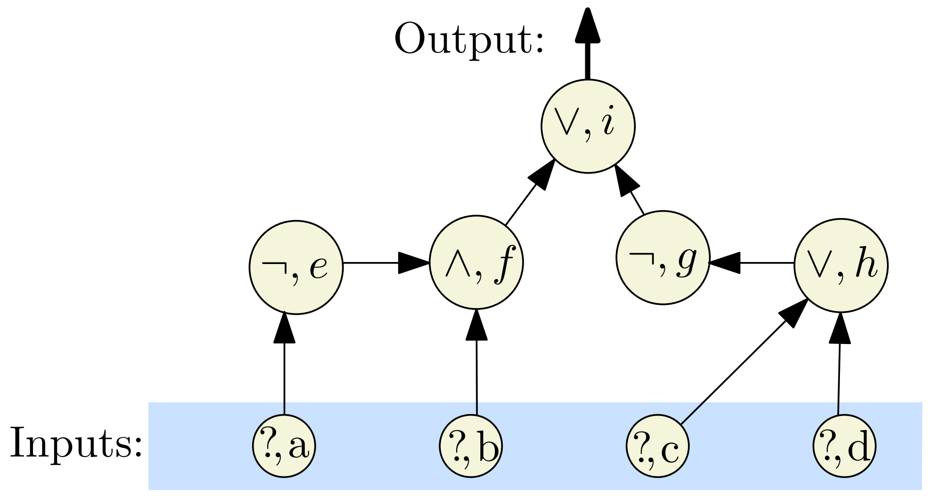
(II) (4 PTS.) $B(n) = B(\lfloor n/6 \rfloor) + B(\lfloor n/3 \rfloor) + B(\lfloor n/2 \rfloor) + n^2$, for $n > 10$ and $B(i) = 1$ for $1 \leq i \leq 10$.

2.B. (8 PTS.) Convert the following boolean circuit (i.e., an instance of Circuit-SAT) into a CNF formula (i.e., an instance of SAT) such that the resulting formula is satisfiable if and only if the circuit sat instance is satisfiable. Use $x_a, x_b, x_c, x_d, \ldots$ as the variable names for the corresponding gates in the drawing.

Useful formulas:

$$\begin{aligned}
z = \overline{x} \quad &\equiv \quad (z \vee x) \wedge (\overline{z} \vee \overline{x}).\\
z = x \vee y \quad &\equiv \quad (z \vee \overline{y}) \wedge (z \vee \overline{x}) \wedge (\overline{z} \vee x \vee y)\\
z = x \wedge y \quad &\equiv \quad (z \vee \overline{x} \vee \overline{y}) \wedge (\overline{z} \vee x) \wedge (\overline{z} \vee y)
\end{aligned}$$

(You may need additional variables.) Note, that a node $\textcircled{\scriptsize $\wedge, g$}$ in the figure below denotes an and gate, where g is its label. You need to show all the steps in deriving the CNF formula.



For no extra charge (!), we provide the following table, which might be convenient in writing down the solution to this problem.

	Equivalent CNF formula

3 (24 PTS.) Huffman encoding + randomized algorithms

3.A. (12 PTS.) Let Σ be an alphabet with n characters, and consider a prefix-free binary code for Σ, where $\sigma \in \Sigma$ is assigned the binary string $f(\sigma)$. **Prove** that $\sum_{\sigma \in \Sigma} 2^{-|f(\sigma)|} \leq 1$, where $|f(\sigma)|$ denotes the length of the binary string $f(\sigma)$.

3.B. (12 PTS.) You are given a graph G with n edges and m vertices, and an integer parameter $k > 0$. Describe an algorithm that removes in expectation at most m/k edges of G, such that the remaining graph can be colored using k colors. What is the running time of your algorithm? (Faster is better, naturally.)

4 (13 PTS.) Rank and file

4.A. (4 PTS.) The ***rank*** of a vertex v in a DAG G, is the length of the longest path in DAG that starts in v. Describe a linear time algorithm (in the number of edges and vertices of G) that computes for all the vertices in G their rank.

4.B. (3 PTS.) Prove that if two vertices $u, v \in \mathsf{V}(\mathsf{G})$ have the same rank, then the edges (u, v) and (v, u) are not in G.

4.C. (3 PTS.) Using (B), prove that in any DAG G with n vertices, for any k, either there is a path of length k, or there is a set X of $\lfloor n/k \rfloor$ vertices in G that is ***independent***; that is, there is no edge between any pair of vertices of X.

4.D. (3 PTS.) Prove, that in any sequence of n distinct real numbers $x_1, x_2, \ldots, x_n$, there are indices $i_1 < i_2 < \cdots < i_N$, for $N = \lfloor \sqrt{n} \rfloor$, such that either

(i) $x_{i_1} < x_{i_2} < \cdots < x_{i_N}$, or

(ii) $x_{i_1} > x_{i_2} > \cdots > x_{i_N}$.

Describe an algorithm, as fast as possible, for computing this subsequence. What is the running time of your algorithm?

5 (13 PTS.) Convert this graph.

An undirected graph G is given to you by an adjacency matrix M of size $n \times n$. The graph G has n vertices and m edges. Assume that you are also given a function $\mathit{isAllZero}([i..j] \times [k..l])$, which tells you in *constant time* whether the submatrix $M[i..j][k..l]$ is all zeros (i.e., $M[x][y] = 0$ for all $x = i, \ldots, j$, and $y = k, \ldots, l$).

5.A. (8 PTS.) Present an algorithm such that given a vertex v, it computes all of its neighbors in G, as fast as possible (as a function of $k = d(v)$ and n), where $d(v)$ is the number of neighbors of v. In particular, how fast is your algorithm, when v has $d(v) = \sqrt{n}$ neighbors? (Your algorithm should be faster than $O(n)$ for credit, for the case that $d(v) \ll n$.)

5.B. (5 PTS.) Present an algorithm, as fast as possible, such that given a graph G as above, it generates a linked list representation of G. How fast is your algorithm as a function of n and m. For credit, your algorithm has to be faster than $O(n^2)$, if $m \ll n^2$.

6 (13 PTS.) Set chaining.

You are given a set X of n elements, and a family of sets $\mathcal{F} = \{F_1, \ldots, F_m\}$, such that, for all i, we have that $F_i \subseteq \mathsf{X}$. A ***chain*** of sets, is an ordering $F_{\pi(1)}, F_{\pi(2)}, \ldots, F_{\pi(m)}$ of the sets of $\mathcal{F}$, such that:

(i) Every set of $\mathcal{F}$ appears exactly once in this chain.

(ii) For all $i < m$, we have $F_{\pi(i)} \cap F_{\pi(i+1)} \neq \emptyset$.

The problem is to compute such a chain if it exists.

Either prove that this problem is NP-HARD, or alternatively, provide a polynomial time algorithm, as fast as possible for solving this problem. What is the running time of your algorithm?

7 (13 pts.) Perfect, just perfect.

At the Perfect Programming Company, programmers program in pairs in order to ensure that the highest quality code is produced. The productivity of each pair of programmers is the speed of the slower programmer. Formally, there are $2n$ programmers, and programmer i has programming speed s_i, for $i = 1, \ldots, 2n$. Give an efficient algorithm for pairing them up so that the sum of the productivity of all pairs is maximized. Analyze the running time and argue the correctness of your algorithm. (Your algorithm should be as simple and as fast as possible.)

8 (13 pts.) Dominos are forever.

A ***domino*** is a rectangle partitioned into two, such that two numbers are written on both parts of the rectangle.

8.A. (5 pts.) You are given a set of m domino pieces, with the numbers written on them taken from $\{1, \ldots, n\}$. You are allowed to place two pieces of domino next to each other (horizontally) if the touching parts have the same number written on them. Consider the problem of deciding if you can place a subset of the given domino pieces horizontally one after the other, such that:

(i) the first number in the first domino piece is 1,
(ii) the last number in the last domino piece is 1,
(iii) every number between 1 and n appears exactly twice in the pieces chosen, and
(iv) two pieces are adjacent if and only if their touching parts have the same number written on them.

(Note, that you are allowed to flip a domino piece.) For example, for the following set of pieces:

$$\boxed{1 \mid 2}\quad \boxed{3 \mid 2}\quad \boxed{4 \mid 2}\quad \boxed{3 \mid 4}\quad \boxed{3 \mid 1}$$

A valid solution is:

$$\boxed{1 \mid 2}\quad \boxed{2 \mid 4}\quad \boxed{4 \mid 3}\quad \boxed{3 \mid 1}\ .$$

Either prove that this problem is NP-Complete, or provide a polynomial time algorithm for solving it.

8.B. (4 pts.) Consider the problem of deciding if you can find a subset X of the given domino pieces such that:

(I) Every piece has an odd number written on one part of it, and even number written on the other part.
(II) Every number $1, \ldots, n$ appears exactly once in X.

Either prove that this problem is NP-Complete, or provide a polynomial time algorithm for solving it.

8.C. (4 pts.) Consider the problem of deciding if you can place ***all*** the given pieces horizontally, such that any two consecutive pieces that touch each other have the same number written on the touching parts.

Either prove that this problem is NP-Complete, or provide a polynomial time algorithm for solving it.