EECS 376: Foundations of Computer Science **Due 8:00pm, June 24**
University of Michigan, Spring 2024 # Homework 5 **(accepted until 9:59 pm, no credit after)**

This homework has 8 questions, for a total of 100 points and 0 extra-credit points.
Unless otherwise stated, each question requires *clear*, *logically correct*, and *sufficient* justification to convince the reader.
For bonus/extra-credit questions, we will provide very limited guidance in office hours and on Piazza, and we do not guarantee anything about the difficulty of these questions.
We strongly encourage you to typeset your solutions in LaTeX.
If you collaborated with someone, you must state their name(s). You must *write your own solution* for all problems and *may not use any other student's write-up.*

(0 pts) 0. **Before you start; before you submit.**

If applicable, state the name(s) and uniqname(s) of your collaborator(s).

Solution: None.

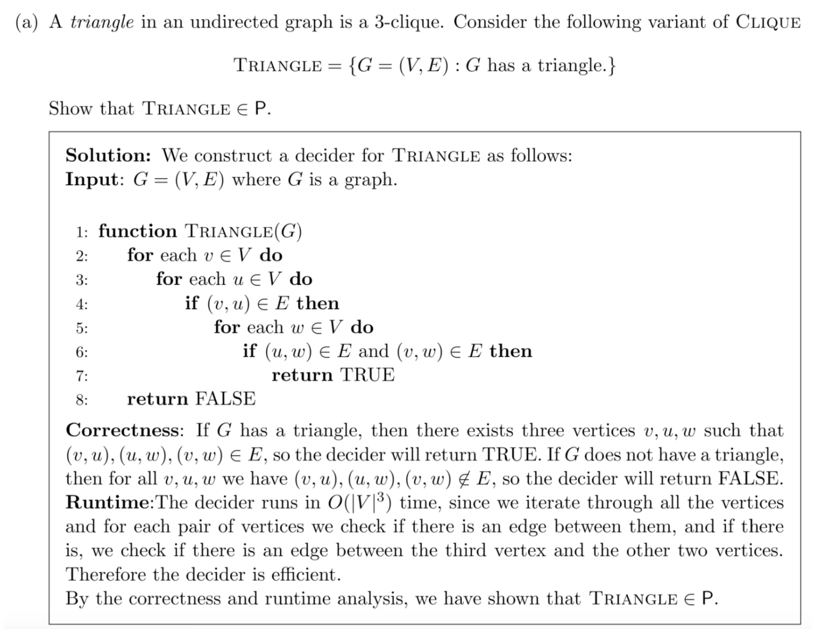
(10 pts) 1. **Self assessment.**

Carefully read and understand the posted solutions to the previous homework. Identify one part for which your own solution has the most room for improvement (e.g., has unsound reasoning, doesn't show what was required, could be significantly clearer or better organized, etc.). Copy or screenshot this solution, then in a few sentences, explain what was deficient and how it could be fixed.

(Alternatively, if you think one of your solutions is significantly *better* than the posted one, copy it here and explain why you think it is better.)

You may reference the answer key, but your answer should be in your own words.

Solution:



(a) A *triangle* in an undirected graph is a 3-clique. Consider the following variant of CLIQUE

$$\text{TRIANGLE} = \{G = (V, E) : G \text{ has a triangle.}\}$$

Show that TRIANGLE $\in$ P.

Solution: We construct a decider for TRIANGLE as follows:
Input: $G = (V, E)$ where G is a graph.

```
function TRIANGLE($G$)
    for each $v \in V$ do
        for each $u \in V$ do
            if $(v, u) \in E$ then
                for each $w \in V$ do
                    if $(u, w) \in E$ and $(v, w) \in E$ then
                        return TRUE
    return FALSE
```

Correctness: If G has a triangle, then there exists three vertices v, u, w such that $(v, u), (u, w), (v, w) \in E$, so the decider will return TRUE. If G does not have a triangle, then for all v, u, w we have $(v, u), (u, w), (v, w) \notin E$, so the decider will return FALSE.
Runtime:The decider runs in $O(|V|^3)$ time, since we iterate through all the vertices and for each pair of vertices we check if there is an edge between them, and if there is, we check if there is an edge between the third vertex and the other two vertices. Therefore the decider is efficient.
By the correctness and runtime analysis, we have shown that TRIANGLE $\in$ P.

My running time analysis on this question has problem. As we can see, the algorithm involves three layers of loops on V, but every iteration involves checking whether two vertices form an edge in E . So every iteration searchs at most the whole E, the actual running time is $O(|V|^3|E|)$.

2. **Warm-up: Multiple Choice.**

For (a) and (b), select **the one** correction option. For (c), select **all** correct options. While justification is **not required** for full credit, providing it is highly encouraged. *Note: If you are editing the LaTeX file, change* `choice` *to* `CorrectChoice` *to fill in the bubble.*

(3 pts) (a) Suppose that there is a randomized algorithm for SAT problem that always output False if there is no satisfying assignment, and output True with probability ≥ 0.6 if there is a satisfying assignment. What is the minimum number of False to guarantee that there is no satisfying assignment with at least 98% confidence? Assume that the results of each output is independent of all the others.

- ○ 6
- ○ 4
- ○ 2
- ○ 3
- ✓ **5**

Solution:

(3 pts) (b) Suppose that X is a non-negative random variable for which $\mathbb{E}[X] = 1/2$. Select the **tightest upper bound that *necessarily* holds** for $\Pr[X \geq 5/7]$.

- ○ $e^{-(3/14)^2}$
- ✓ 7/10
- ○ 5/7
- ○ 1

Solution:

(5 pts) (c) Suppose A and B are maximization problems, $A \leq_p B$ and $B \leq_p A$. Select all statements that are true?

- ○ If there is an efficient 0.9 approximation algorithm for A, then there is an efficient 0.9 approximation algorithm for B.
- ✓ **If there is an efficient algorithm for A, then there is an efficient algorithm for B.**
- ○ If there is an efficient 0.9 approximation algorithm for A, then there is an efficient α approximation algorithm for B for some $\alpha < 1$.
- ○ If there is no efficient 0.9 approximation algorithm for A, then there is no efficient 0.9 approximation algorithm for B.

√ **If there is an efficient algorithm for A, then there is an efficient 0.9 approximation algorithm for B.**

Solution:

(15 pts) 3. **Delivery Route.**

You are starting a new job as a delivery driver for WindowWalk in Ann Arbor, and they have loaned you an electric car for making deliveries. Each night, you need to charge the car so that it has sufficient power to complete a delivery route, but you don't know the exact route until the next morning. However, you do know the following:

- The route does not visit the same location more than once, except that it begins and ends at your home. We refer to such a route as a *delivery route from your home.*
- If your route takes you past a charging station, you can also charge your car on the way.

Your goal is to charge your car each night to a sufficient level so that it can make it through any delivery route that does not pass a charging station.

Formally, you model the problem as an unweighted, undirected graph G, with vertices as locations in the city and edges as connections between the locations. Given a start vertex s and a set of charging stations A, you want to find the maximal delivery route (the route with the highest number of edges) from s that does not go through any of the vertices in A. The decision problem is as follows:

$$\textsc{Delivery-Route} = \{(G = (V, E), A, s, k) : G \text{ has a delivery route from } s \text{ of length exactly } k \text{ that does } \textbf{not} \text{ go through any vertex in } A \subseteq V\}$$

Given an efficient decider for Delivery-Route, design and analyze (both correctness and efficiency) an efficient algorithm that given (G, A, s), finds the actual ordered sequence of edges that comprise a delivery route of maximal length from s that does not go through any vertex in A. *Hint:* First determine the maximal length.

Solution:
(1) Determine the Maximal Length of the Delivery Route

```
function $MaxLength(G, A, s)$
    max = 0
    for $k$ from 1 to $|E|$ do
        if Delivery-Route$(G, A, s, k)$ then
            max = k
    return max
```

(2)Find all only necessary vertices in a delivery route
We use the max returned by $MaxLength$ to find the actual ordered sequence of edges that comprise a delivery route.

```
function $FindVertices(G, A, s, k)$
    make a copy of $G$ as $G'$
    for $v$ in $V$ do
        remove $v$ from and its associated edges in $G'$ as $G_{reduced}$
        if DELIVERY-ROUTE($G_{reduced}, A, s, k$) returns true then
            remove $v$ from and its associated edges in $G'$
    return $G'$
```

(3)Find the actual ordered sequence of edges that comprise a delivery route
We use the G' returned by $FindVertices$ and the max returned by $MaxLength$ as k, as an input to the new function. $G' = (V', E')$ now contains all the necessary vertices for the delivery route. Now we delete edges to make G' only contains a cycle.

```
function $FindRoute(G' = (V', E'), A, s, k)$
    $route = [s]$
    for $e$ in $E'$ do
        remove $e$ from $E'$
        if DELIVERY-ROUTE($G', A, s, k$) returns true then
            continue
        else
            restore $e$ in $G'$
    loop V' to find a vertice $v_0$ that is connected to s    ▷ Now the $G'$ only contains a cycle, which means that every vertex in $V'$ is connected to two other vertexs in $V'$.
    currentVertex = $v_0$
    lastVertex = $s$
    loop through the $V'$ to find the next vertex in the cycle, and add next vertex into $route$, until current vertex is $s$.
    return $route$
```

(4) Correctness Justification
1. Note that k is an integer since the graph is unweighted, and is bounded by the size of E, so by iterating from 1 to $|E|$, we can find the maximal length of the delivery route using the $MaxLength$ function.

2. By the $FindVertices$ function, the returned graph G' contains all the and only the necessary vertices for the delivery route, since if a vertex is unnecessary, then $G_{reduced}, A, s, k$ returns true and it will be deleted, and since k is the maximal length, if a vertex is necessary in the route, $G_{reduced}, A, s, k$ will return false so it will not be deleted.

3. Likewise, $FindRoute$ delete all unnecessary cycles and leaves only a cycle in G', and

then find the actual ordered sequence of edges that comprise a delivery route by looping through the vertices in the cycle.
(5) Efficiency Analysis
Suppose the runtime of *DeliveryRoute* function is P which is a polynomial to the input G, then the runtime of $MaxLength$ is $O(|E|P + |V|P + |E|P + |V|) = O((|E| + |V|)P)$, which is a polynomial to the input, so the search algorithm is efficient if the *DeliveryRoute* function is efficient.

4. **Approximate f-SetCover.**

In this question, you will consider a variant of the SetCover problem where each element of the universe is in a limited number of subsets. Formally, in the f-SetCover problem, we are given a "universe" (set) U and subsets $S_1, \dots, S_n \subseteq U$ *where each universe element appears in at most f of the subsets.* The goal is to find a smallest collection of the subsets that "covers" U, i.e., an $I \subseteq \{1, \dots, n\}$ of minimum size such that $\bigcup_{i \in I} S_i = U$. We assume that $\bigcup_{i=1}^{n} S_i = U$, otherwise no solution exists.

You will analyze an approximation algorithm "f-cover" for the f-SetCover problem. The algorithm is a generalization of the "double cover" algorithm for VertexCover from lecture, and it works essentially as follows: while there is some uncovered element u in the universe, add to the cover *all* the subsets to which u belongs. The formal pseudocode is as follows. The notation $I(u) = \{i : u \in S_i\}$ for $u \in U$, that is, $I(u)$ indicates the subsets to which u belongs.

```
function $f$-COVER($U, S_1, \dots, S_n$)
    $I = C = \emptyset$                                  ▷ selected indices $I$, covered elements $C$
    while $C \neq U$ do                                   ▷ not all elements are covered
        choose an arbitrary $u \in U \setminus C$                ▷ element $u$ is not yet covered
        $I = I \cup I(u)$, $C = C \cup \bigcup_{i \in I(u)} S_i$   ▷ add all subsets $S_i$ containing $u$ to the cover
    return $I$
```

Fix some arbitrary f-SetCover instance, and let I^* denote an optimal set cover for it. Let E denote the set of elements u chosen in Step 4 during an execution of the algorithm, and let I denote the algorithm's final output.

(4 pts) (a) Prove that $I(u) \cap I(u') = \emptyset$ for every distinct $u, u' \in E$. In other words, prove that if u and u' are each selected as the uncovered element in different iterations, none of the S_i contains both u and u'.

Solution: Suppose an element u is chosen in the algorithm, then all subsets containing u are added to the set cover.
Suppose another element u' is selected in a subsequent iteration, it implies that u' was not covered previously. Thus, there is no subset that contains both u and u', implying that $I(u) \cap I(u') = \emptyset$.

(4 pts) (b) We want a lower bound on OPT $= |I^*|$. Using the previous part, prove that $|E| \leq |I^*|$.

Solution: Each element $u \in E$ chosen by the algorithm must be covered by at least one subset in any feasible solution. Since $I(u) \cap I(u') = \emptyset$ for all $u \neq u'$, each element $u \in E$ provides a unique contribution to the cover. Hence, the number of elements in E sets a lower bound on the number of subsets required in an optimal solution, implying $|E| \leq |I^*|$.

(4 pts) (c) We want an upper bound on ALG $= |I|$. Prove that $|I| \leq f \cdot |E|$, and conclude that the f-cover algorithm is an f-approximation algorithm for the f-SetCover problem.

Solution: Each element u chosen adds at most f subsets to I, as each element appears in at most f subsets. Thus, the total number of subsets added to I is at most $f \cdot |E|$. Since $|E| \leq |I^*|$, it follows that $|I| \leq f \cdot |I^*|$, denoting that the algorithm is an f-approximation algorithm.

(4 pts) (d) Prove that for every positive integer f, there is an input for which the f-cover algorithm necessarily outputs a cover that is *exactly* f times as large as an optimal one.

Solution: For each positive integer f, Consider a universe U with f elements and $n = f$ subsets, where each subset $S_i = \{u_i\}$. Each element appears in exactly one subset, so the f-cover algorithm will select all subsets when the first element is picked, resulting in $|I| = f$. This is a case where the output of f-cover is exactly f times the size of an optimal solution.

5. **Indicator variables and linearity of expectation.**

Let n be a positive integer and $0 < p < 1$. Let $G_{n,p}$ denote a random undirected graph on n vertices, constructed as follows: for each (unordered) pair of distinct vertices $u, v \in V$, include the edge (u, v) in $G_{n,p}$ with probability p, independently of all other random choices.

(5 pts) (a) Derive the expected number of edges in $G_{n,p}$.

Solution: The total number of pairs of vertices between n vertices is $\binom{n}{2}$, so the expected number of edges $\mathbb{E}[E]$ is:

$$\mathbb{E}[E] = \binom{n}{2} \cdot p = \frac{n(n-1)}{2} \cdot p$$

(5 pts) (b) Derive the expected degree of any vertex in $G_{n,p}$.

Solution: Each vertex v can potentially connect to $n-1$ other vertices, each edge formed independently with probability p. Fix a vertex $v \in V$, and let d_v denote the degree of vertex v. Consider the indicator variable X_{uv} for whether vertex $u, v \in V$ is connected. So the expected degree of vertex v is:

$$\mathbb{E}[d_v] = \sum_{u \in V - v} \mathbb{E}[X_{uv}] = (n-1) \cdot p$$

(4 pts) (c) A *triangle* in a graph is a set of three (distinct) vertices that have an edge between every pair of them. Derive the expected number of triangles in $G_{n,p}$.

Solution: The number of ways to choose three vertices from n is $\binom{n}{3}$. For each triplet of vertices (u, v, w), the probability that all three possible edges among them are present is p^3, by independency. Therefore, the expected number of triangles $\mathbb{E}[T]$ is:

$$\mathbb{E}[T] = \binom{n}{3} \cdot p^3 = \frac{n(n-1)(n-2)}{6} \cdot p^3$$

$$(n \geq 2)$$

.

(4 pts) (d) Prove that the number of triangles in $G_{n,p}$ is at least n^3p^2 with probability at most $p/6$.

Solution: By Markov's inequality: $\Pr[T \geq x] \leq \frac{\mathbb{E}[T]}{x}$.
Therefore for $x = n^3p^2$:

$$\Pr[T \geq n^3p^2] \leq \frac{\frac{n(n-1)(n-2)}{6} \cdot p^3}{n^3p^2} = \frac{(n-1)(n-2)}{6n^2} \cdot p \leq \frac{p}{6}$$

since $0 \leq (n-1)(n-2) \leq n^2$ for $n \geq 2$.

6. **Randomized max-cut.**

In this problem, all graphs are undirected and (as usual) have *no self-loops*, i.e., there is no edge from a vertex to itself. For a cut $S \subseteq V$ in a graph $G = (V, E)$, let $C(S) \subseteq E$ denote the subset of edges "crossing" the cut, i.e., those that have exactly one endpoint in S. The size of the cut is then $|C(S)|$. Consider the following randomized algorithm that outputs a cut in a given graph $G = (V, E)$.

```
1: initialize $S = \emptyset$
2: for all $v \in V$ do
3:     put $v$ into $S$ with probability $1/2$, independently of all others
4: return $S$
```

(5 pts) (a) Define suitable indicator variables and use linearity of expectation to prove that *in expectation*, the above algorithm obtains a 1/2-approximation for MaxCut. That is, the expected size of the output cut is at least half the size of a maximum cut.

Solution:
Let I_{uv} be an indicator variable that is 1 if edge $\{u, v\}$ crosses the cut S, and 0 otherwise. The edge crosses the cut if exactly one of its endpoints is in S. Each vertex

is added to S independently with probability $\frac{1}{2}$, so:

$$\Pr[I_{uv}=1]=\Pr[u\in S, v\notin S]+\Pr[u\notin S, v\in S]=\frac{1}{2}\cdot\frac{1}{2}+\frac{1}{2}\cdot\frac{1}{2}=\frac{1}{2}$$

The expected size of the cut $|C(S)|$ is:

$$\mathbb{E}[|C(S)|]=\sum_{\{u,v\}\in E}\mathbb{E}[I_{uv}]=\frac{|E|}{2}$$

Since the size of a maximum cut $|E_{max}|\le\frac{|E|}{2}$, the algorithm achieves a $\frac{1}{2}$-approximation in expectation:

$$\mathbb{E}[|C(S)|]\ge\frac{1}{2}|E_{max}|$$

(5 pts) (b) Prove that $\Pr[|C(S)|\ge(1-\varepsilon)|E|/2]\ge\frac{\varepsilon}{1+\varepsilon}$ for any $\varepsilon>0$.
Notice that for $\varepsilon=1/(2|E|)$, this is a lower bound on the probability that the number of edges crossing S is at least $|E|/2$, because the number of crossing edges is an integer.

Solution: We prove it by applying reverse Markov's inequality to the expected number of crossing edges: $C(S)\le|E|$ for all cuts S, so we can set $b=|E|$.
Then by the reverse Markov's inequality:

$$\begin{aligned}\Pr[|C(S)|\ge(1-\varepsilon)|E|/2] &\ge \frac{\mathbb{E}[|C(S)|]-(1-\varepsilon|E|/2)}{|E|-(1-\varepsilon)|E|/2}\\ &=\frac{|E|/2-(1-\varepsilon|E|/2)}{|E|-(1-\varepsilon)|E|/2}\\ &=\frac{\varepsilon|E|/2}{(1+\varepsilon)|E|/2}\\ &=\frac{\varepsilon}{1+\varepsilon}\end{aligned}$$

(5 pts) (c) As a stepping stone to the next part, we consider the following intermediate question.
Consider a probability experiment that consists of a sequence of "attempts," where each attempt succeeds with probability p, independently of all others. We keep making attempts until one succeeds, at which point the experiment terminates.
Let X denote the number of attempts until termination (including the attempt that finally succeeds). Prove that

$$\mathbb{E}[X]=p+(1-p)(1+\mathbb{E}[X])\ ,$$

and conclude that $\mathbb{E}[X]=1/p$. (The distribution of X is known as the *geometric distribution* with success probability p.)

Solution: Let X denote the number of attempts until termination (including the attempt that finally succeeds), and let each attempt succeed with probability p, independently of the others.

Consider the outcomes of the first attempt:

Case 1: Success on the First Attempt: This occurs with probability p. If the first attempt is successful, $X = 1$.

Case 2: Failure on the First Attempt: This occurs with probability $1 - p$. If the first attempt fails, the number of additional attempts required would still be expected to be $\mathbb{E}[X]$. Since this includes the failed first attempt, the total expected number of attempts would then be $1 + \mathbb{E}[X]$.

So the total expected value of X is:

$$\mathbb{E}[X] = p \cdot 1 + (1 - p) \cdot (1 + \mathbb{E}[X])$$

To solve for $\mathbb{E}[X]$, rearrange and isolate $\mathbb{E}[X]$ on one side:

$$\mathbb{E}[X] = p + 1 - p + (1 - p)\mathbb{E}[X] = 1 + (1 - p)\mathbb{E}[X]$$

$$1 - (1 - p)\mathbb{E}[X] = 1$$

$$\mathbb{E}[X] = \frac{1}{p}$$

(5 pts) (d) Suppose we repeatedly run our randomized MaxCut algorithm until we get a cut of size at least $|E|/2$. Derive an upper bound on the expected number of attempts that are needed.

Solution: Given each attempt to obtain a cut of size at least $\frac{|E|}{2}$ succeeds with a probability $\frac{1}{2}$:

$$\mathbb{E}[\text{Attempts}] = 1/p \leq \frac{1}{1/2} = 2$$

So an upper bound on the expected number of runs of the algorithm to achieve a cut of at least $\frac{|E|}{2}$ is 2.

7. **Cryptography.**

(6 pts) (a) Alice and Bob agree on the prime $p = 17$ and generator $g = 6$, and wish to establish a shared secret key k using the Diffie-Hellman protocol.

Suppose Alice privately chooses $a = 3$ and Bob privately chooses $b = 12$. What is Alice's message to Bob? What is Bob's message to Alice? What is the secret key k?

Solution:
Alice's message to Bob:

$$g^a \equiv 6^3 \equiv 2 \cdot 6 \equiv 12 \mod 17$$

So Alice's message to Bob is 12.

Bob's message to Alice:

$$g^b \equiv 6^{12} \equiv (6^3)^4 \equiv (144)^2 \equiv 8^2 \equiv 13 \mod 17$$

So Bob's message to Alice is 13.

Shared secret key k:

$$12^{12} \equiv 13^3 \equiv 4 \mod 17$$

So, the shared secret key k is 4.

(4 pts) (b) Alice uses modulus $n = 65$ and RSA public key $e = 29$. Determine a possible private key d and calculate Alice's RSA-signature for the message $m = 3$.

Solution:
$n = 65 = 5 \times 13$, $\phi(65) = (5 - 1)(13 - 1) = 48$
Finding a private key d:
Since $e = 29$, we solve $29\,d \equiv 1 \mod 48$.
Using the extended Euclidean algorithm, $d = 5$.

$$29 \cdot 5 \mod 48 = 1$$

So a private key d is 5.
RSA Signature for $m = 3$:

$$s \equiv m^d \equiv 3^5 \equiv 48 \mod 65$$

So the RSA signature s is 48.