CS 466/666 Algorithm Design and Analysis **Last Updated:** Aug 6, 2018
Homework 5 **Due Date: Aug 14 (11pm), 2018**

You are allowed to discuss with others but not allowed to use any references other than the course notes. Please list your collaborators for each question. This will not affect your marks. In any case, you must write your own solutions.

There are totally 60 marks, and the full mark is 50. This homework is counted 8% of the course.

---

1. **Hitting Time**

   (10 marks) Consider a random walk on an undirected graph $G = (V, E)$ that starts at a vertex $v \in V$, and stops when it reaches s or t. Let $p(v)$ be the probability that if the random walk starts at v then it reaches s before t. Establish a connection between these probabilities and some parameters of an appropriate electrical flow problem.

2. **Cat and Mouse**

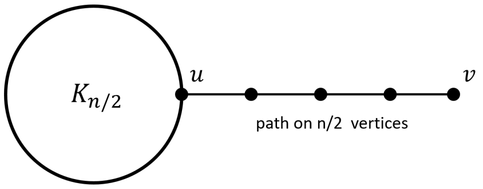
   (10 marks) A cat and a mouse each independently take a random walk on a connected, undirected, non-bipartite graph G. They start at the same time on different nodes, and each makes one transition at each time step. The cat catches the mouse if they are ever at the same node at some time step. Show an upper bound of $O(m^2 n)$ on the expected time before the cat catches the mouse, where n is the number of vertices and m is the number of edges of G.

   (Hint: Consider a Markov chain whose states are the ordered pairs (a, b), where a is the position of the cop and b is the position of the robber.)

   (Remark: You will get half the marks if you show the upper bound $O(m^2 n^2)$ instead.)

3. **Lollipop Graph**

   (10 marks) The lollipop graph on n vertices is a clique on $n/2$ vertices connected to a path on $n/2$ vertices, as shown in the figure. The node u is a part of both the clique and the path. Let v denote the other end of the path.



   (a) Show that the expected cover time of a random walk starting at v is $\Theta(n^2)$.

   (b) Show that the expected cover time of a random walk starting at u is $\Theta(n^3)$.

4. **Online Experts**

(15 marks) Consider the following weighted majority algorithm for the online expert problem. There are n experts. Initially each expert i has weight $w_i = 1$. Each day, we need to make a binary decision (say Yes or No), and we follow the weighted majority of the experts, i.e. if the total weight of the experts saying Yes is at least the total weight of the experts saying No, then we say Yes; otherwise we say No. We see the outcome at the end of each day, and we update the weight of an expert i by setting $w_i \leftarrow (1-\epsilon)w_i$ if expert i makes a mistake on that day, where $1/2 > \epsilon > 0$ is a parameter of the algorithm.

(a) Prove that the number of mistakes made by the weighted majority algorithm is at most

$$2(1+\epsilon)m + \frac{2\ln n}{\epsilon},$$

where m is the number of mistakes made by the best expert.

(b) Give an example to show that the weighted majority algorithm will make at least $2m$ mistakes.

(Hint: Consider a simple example in which there are only two experts.)

(Remark: This shows that the factor 2 in this deterministic algorithm cannot be improved, while the randomized algorithm in L16 could achieve $(1+\epsilon)m$.)

5. **Multiplicative Weights Update Method for Maximum Flow**

(15 marks) Consider the flow problem from s to t on a directed graph, where each edge has capacity one. A fractional s-t flow solution with value k is an assignment of each edge e to a fractional value $x(e)$, satisfying that

$$\sum_{e \in \delta^{\text{out}}(s)} x(e) = \sum_{e \in \delta^{\text{in}}(t)} x(e) = k,$$

$$\sum_{e \in \delta^{\text{in}}(v)} x(e) = \sum_{e \in \delta^{\text{out}}(v)} x(e) \quad \forall v \in V - \{s,t\}, \text{ and}$$

$$0 \le x(e) \le 1 \quad \forall e \in E.$$

Use the multiplicative weights update method to solve this LP by reducing the flow problem to the problem of finding shortest paths between s and t. Analyze the convergence rate and the total complexity of your algorithm to compute a flow of value $k(1-\epsilon)$ for any $\epsilon > 0$ or to report that a flow with value k does not exist.

(Remark: Any correct algorithm with a correct analysis will get most of the marks, so it is not crucial to achieve the fastest running time possible.)