CSC473 W18: Homework Assignment #1

Due: February 14, by midnight

January 30, 2018

Guidelines: (read fully!!)

- Your assignment solution must be submitted as a *typed* PDF document. Scanned handwritten solutions, solutions in any other format, or unreadable solutions will **not** be accepted or marked. You are encouraged to learn the LaTeX typesetting system and use it to type your solution.
- To submit this assignment, use the MarkUs system, at URL `https://markus.teach.cs.toronto.edu/csc473-2018-01`
- This is a *group assignment*. This means that you can work on this assignment with *at most one other* student. You are *strongly encouraged* to work with a partner. Both partners in the group should work on and arrive at the solution together. Both partners receive the same mark on this assignment.
- Work on all problems together. For each problem, one of you should write the solution, and one should proof-read and revise it. The first page of your submission must list the *name*, *student ID*, and *UTOR email address* of both group members. It should also list, for each problem, which group member wrote the problem, and which group member proof-read and revised it.
- You **may not** consult any other resources except: your partner; your class notes; your textbook and assigned readings. *Consulting any other resource, or collaborating with students other than your group partner, is a violation of the academic integrity policy!*
- You may use any data structure, algorithm, or theorem previously studied in class, or in one of the prerequisites of this course, by just referring to it, and without describing it. This includes any data structure, algorithm, or theorem we covered in lecture, in a tutorial, or in any of the assigned readings. Be sure to give a *precise reference* for the data structure/algorithm/result you are using.
- Unless stated otherwise, you should justify all your answers using rigorous arguments. Your solution will be marked based both on its completeness and correctness, and also on the clarity and precision of your explanation.

Question 1. (10 marks) We consider a variant of the CONTRACTION algorithm for finding a minimum $s-t$ cut in a graph. Let $G=(V,E)$ be an undirected connected graph, with two special vertices s and t. Every edge $e \in E$ is given a positive integer weight w_e. An $s-t$ minimum cut is a cut $E(S,\bar{S})$ such that $s \in S$, $t \in \bar{S}$, and the total weight across the cut $\sum_{e \in E(S,\bar{S})} w_e$ is minimized.

The algorithm STCONTRACTION first removes any edge of E between s and t. Then, in every iteration, the algorithm samples an edge e of E with probability equal to $\frac{w_e}{\sum_{e \in E} w_e}$. The edge $e=(u,v)$ is contracted as in the CONTRACTION algorithm: the nodes u and v are identified into a new supernode, and all edges between u and v are removed. Additionally, so that s and t are never contracted, the algorithm also removes all edges between the supernodes corresponding to s and t. The algorithm makes $|V|-2$ iterations, until only the supernodes corresponding to s and t remain, and it outputs the corresponding $s-t$ cut.

For every $n > 2$, give a n-node weighted graph G as above, so that G has a unique s-t minimum cut $E(S,\bar{S})$, and the probability STCONTRACTION outputs this cut is exponentially small, i.e. the probability of success is $2^{-\Omega(n)}$. Justify your answer.

Question 2. (10 marks) The Manhattan distance between two points x and y in the set $\{0,\ldots,N-1\}^d$ is defined by

$$d_M(x,y) = \sum_{i=1}^{d} |x_i - y_i|.$$

(This is called the Manhattan distance because in the $d=2$ case you imagine the set $\{0,\ldots,N-1\}^2$ as the intersections in Manhattan and $d_M(x,y)$ tells you how many blocks you need to travel to get from x to y.)

Give a random hash function $h : \{0,\ldots,N-1\}^d \to \{0,1\}$, so that

$$\mathbb{P}(h(x)=h(y)) = 1 - \frac{d_M(x,y)}{dN}$$

holds for any x and y in $\{0,\ldots,N-1\}^d$. (Just as in class, notice that x and y are not random, and the probability is only over the random choice of h.) The function should be computable in time $O(1)$ when x is given as an array of size d. Justify your answer.

Question 3. (20 marks) This question builds on the tutorial we had about finding triangles in graphs. Recall that a path in an undirected graph $G=(V,E)$ is simple if no vertices on the path are repeated. Similarly a simple k-cycle in G is sequence of *distinct* vertices $v_1,\ldots,v_k$ in V such that for any $i<k$ the edge (v_i,v_{i+1}) is in E, and also the edge (v_k,v_1) is in E, as well. Recall also that ω is the best constant so that two $n \times n$ matrices A and B can be multiplied in time $O(n^\omega)$.

a. In the tutorial we argued that if A is the adjacency matrix of an undirected graph $G=(V,E)$, then there is a simple path of length exactly 2 between u and v, $u \neq v$, if and only if $(A^2)_{uv} > 0$. Give a simple counterexample to the following statement: "There is a simple path of length exactly 3 between u and v in G if and only if $(A^3)_{uv} > 0$."

b. Let $G=(V,E)$ be an undirected graph with n nodes. Let π be a permutation of V; i.e. π assigns each vertex $v \in V$ a distinct integer in $\{1,\ldots,n\}$. A π-increasing k-cycle in G is a cycle whose vertices can be listed as $v_1,\ldots,v_k$ so that $\pi(v_i) < \pi(v_j)$ for all $i<j$. (Notice that such a cycle must be simple.) Give an algorithm that, given G and π, decides whether G has a π-increasing k-cycle in time $O(kn^\omega)$. Describe your algorithm clearly and argue why it is correct and runs in the required time.

BONUS: Give an algorithm running in time $O(\log(k)n^\omega)$.

c. Use the previous subproblem to design a Monte Carlo algorithm that decides if a given undirected graph G with n nodes has a simple k-cycle in time $O(k!kn^\omega)$. When G has a simple k-cycle, the algorithm should output "cycle found" with probability at least $\frac{2}{3}$. When G has no simple k-cycle, the algorithm should output "no cycle found". The algorithm does not need to output the actual cycle. Describe your algorithm clearly and argue why it satisfies the guarantees above and runs in the required expected time.