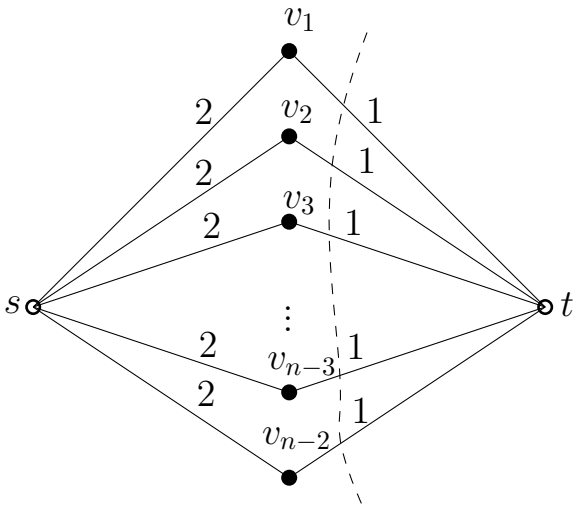
CSC473W18: Homework Assignment #1: Solutions

Question 1. (10 marks) The graph G is as shown in the following figure:



In addition to s and t, we have vertices $v_1, \ldots, v_{n-2}$, and each of them is connected to s and to t. There are no other edges. All edges incident on s have weight 2, and all other edges incident on t have weight 1. The unique s-t minimum cut is induced by $S^* = \{s, v_1, \ldots, v_{n-2}\}$ and $\bar{S}^* = \{t\}$.

When one of the two edges incident on v_i is contracted, the other one is removed immediately, so only one of the two edges can be contracted. If (s, v_i) is contracted, then in the final cut $E(S, \bar{S})$ output by the algorithm we have $v_i \in S$. If (v_i, t) is contracted, we have $v_i \in \bar{S}$. Because the two edges incident on v_i have total weight 3 for every i, we have the following equivalent way of sampling an edge to contract in every round: we pick uniformly at random one vertex v_i which has not been contracted into s or t yet, and then with probability $\frac{2}{3}$ we pick (s, v_i), and contract v_i into s, and with probability $\frac{1}{3}$ we pick (v_i, t), and we contract v_i into t. The algorithm finds the unique minimum $s - t$ cut if it contracts all v_i into s, and because these decisions are independent, this happens with probability $\left(\frac{2}{3}\right)^{n-2} < 2^{-0.58(n-2)}$.

Question 2. (10 marks) The function $h = h_{i,\tau}$ is defined by an index $i \in \{1, \ldots, d\}$ and a threshold $\tau \in \{0, \ldots, N-1\}$. We define $h_{i,\tau}(x)$ to be equal to 1 if $x_i \leq \tau$, and 0 otherwise. We pick a random $h_{i,\tau}$ by picking both i and τ uniformly at random, respectively from $\{1, \ldots, d\}$ and $\{0, \ldots, N-1\}$.

Let us fix some $x, y \in \{0, \ldots, N-1\}^d$. Notice that for any choice of i, $h_{i,\tau}(x) \neq h_{i,\tau}(y)$ exactly when

$$\min\{x_i, y_i\} \leq \tau < \max\{x_i, y_i\}.$$

This happens for $|x_i - y_i|$ of the N possible choices of τ. Summing over the d options for i, we have

$$\mathbb{P}(h_{i,\tau}(x) \neq h_{i,\tau}(y)) = \frac{1}{d} \sum_{i=1}^{d} \frac{|x_i - y_i|}{N} = \frac{d_M(x, y)}{dN},$$

as we wanted.

conditional probability

Question 3. (20 marks)

a. Consider a triangle. Its adjacency matrix A and its square A^2, and cube A^3 are:

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \qquad A^2 = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \qquad A^3 = \begin{pmatrix} 2 & 3 & 3 \\ 3 & 2 & 3 \\ 3 & 3 & 2 \end{pmatrix}$$

So, if the statement were true, there would be a simple path of length 3 between each pair of vertices in the graph, but in fact there are no such paths.

b. Let us define an $n \times n$ matrix B, whose rows and columns correspond to the vertices of G. We define $b_{uv} = 1$ if $(u, v) \in E$ and $\pi(u) \leq \pi(v)$. If either of these conditions is false, we define $b_{uv} = 0$. This graph can be constructed from π and the adjacency matrix of G in time $O(n^2)$. Let us now define a length ℓ simple path in G with vertices $v_1, \ldots, v_\ell$ to be π-increasing if for any $i < j$, $\pi(v_i) < \pi(v_j)$. We claim that there is a π-increasing path in G of length exactly ℓ from u to v if and only if $(B^\ell)_{uv} > 0$. This follows easily by induction. The base case $\ell = 1$ is obvious from the definition of B. For the induction step, observe that

$$(B^\ell)_{uv} = (B^{\ell-1}B)_{uv} = \sum_{w \in V} (B^{\ell-1})_{uw} b_{wv}.$$

Since all entries of $B^{\ell-1}$ and B are non-negative, the sum on the right hand side is greater than 0 if and only if there exists a vertex $w \in V$ such $(B^{\ell-1})_{uw} > 0$ and $b_{wv} > 0$. By the induction hypothesis, $(B^{\ell-1})_{uw} > 0$ if and only if there is a π-increasing path of length $\ell - 1$ from u to w. By definition, $b_{wv} > 0$ if and only if there is an edge from w to v, and $\pi(w) < \pi(v)$. These conditions hold simultaneously for some w if and only there is a π-increasing path of length ℓ from u to v.

Observe that G has a π-increasing k-cycle if and only if there exist two vertices u and v such that there is a π-increasing path from u to v of length $k - 1$, and $(u, v) \in E$. This suggests the following algorithm. We first compute the matrix $C = B^{k-1}$. This can be done using k matrix multiplications in time $O(kn^\omega)$, or using fast exponentiation in time $O(\log(k)n^\omega)$. Then we go over all entries of C, and whenever $c_{uv} > 0$, we check if $(u, v) \in E$ using the adjacency matrix representation of G. This check takes time $O(n^2)$. If there are u and v such that $c_{uv} > 0$ and $(u, v) \in E$, then G has a π-increasing k-cycle. The total running time is dominated by computing C, and is $O(kn^\omega)$, or $O(\log(k)n^\omega)$ using fast exponentiation.

c. We first give an algorithm which succeeds with probability $\geq \frac{1}{k!}$. The algorithm picks a uniformly random permutation π, and runs the algorithm from the previous subproblem. If the algorithm finds that there is a π-increasing k-cycle, we output "cycle found", otherwise we output "no cycle found." Since a graph without a simple k-cycle will not contain a π-increasing k-cycle for any π, we never output "cycle found" if G does not contain a simple k-cycle. Assume then that G contains a simple k-cycle with vertices $v_1, \ldots, v_k$. This cycle is π-increasing if $\pi(v_1) < \ldots < \pi(v_k)$. Since π was a uniform permutation, the ordering of $\pi(v_1), \ldots, \pi(v_k)$ is equally likely to be any of the $k!$ possible orderings, and exactly one of them ensures $\pi(v_1) < \ldots < \pi(v_k)$. So, the probability that the cycle is π-increasing is at least $\frac{1}{k!}$. Because the algorithm will output "cycle found" when there is a π-increasing k-cycle in G, it succeeds with probability at least $\frac{1}{k!}$.

To get an algorithm with a constant probability of success, we simply repeat the algorithm we just described $ck!$ times, for a large enough constant c. Each repetition is with an independent random choice of π. If any of these runs of the algorithm succeeds in finding a cycle, we output "cycle found", otherwise we output "no cycle found." Once again, we cannot output "no cycle found" if G does not contain a simple k-cycle. If G does contain such a cycle, then each run of the algorithm succeeds in finding it with probability at least $\frac{1}{k!}$, so the probability that all runs fail is

$$1 - \left(1 - \frac{1}{k!}\right)^{ck!} \geq 1 - e^{-c} \geq \frac{2}{3}$$

for $c \geq \ln(3)$.