CSE521 Homework 3 Solution

Problem 1. Let X_{ij}, $i < j$ be the indicator that the ith and jth elements are compared. Also, let α be the process described in the problem statement. Then, by linearity of expectaion and the fact that the ith and jth elements are compared at most once during the process, we have:

$$\mathbf{E}[C] = \mathbf{E}_\alpha\left[\sum_{1\leqslant i<j\leqslant n} X_{ij}\right] \tag{1}$$

$$= \sum_{1\leqslant i<j\leqslant n} \mathbf{E}_\alpha[X_{ij}] \tag{2}$$

$$= \sum_{1\leqslant i<j\leqslant n} \mathbf{Pr}_\alpha[X_{ij}=1]. \tag{3}$$

where C is the number of comparisons made.

To compute $\mathbf{Pr}_\alpha[X_{ij}=1]$, we define another process β such that $\mathbf{Pr}_\beta[X_{ij}=1] = \mathbf{Pr}_\alpha[X_{ij}=1]$, but β is easier to analyze. This process is optained by "expanding" each step of α, so that instead of choosing a pivot uniformly at random in "the current set" S, we keep picking elements uniformly at random in the original set until an element in S is picked. One can easily verify that for any $x \in S$, the probability of x being picked by this sub-process is exactly $\frac{1}{|S|}$. Thus, the probability that either i and j is picked in β is the same as that probability in α, which means $\mathbf{Pr}_\beta[X_{ij}=1] = \mathbf{Pr}_\alpha[X_{ij}=1]$.

Now we compute $\mathbf{Pr}_\beta[X_{ij}=1]$ where $i < j \leqslant k$. We say that β *determines* X_{ij} at some step if after that step, we know the value of X_{ij}. At any step of β, the probability (conditioned on the fact that β did not determine X_{ij} before this step) that $X_{ij}=1$ is $\frac{2}{n}$. Also, the probability that β still haven't determined X_{ij} after this step is the probability that the picked element is either before i or after k, which is $\frac{i-1+n-k}{n}$. Therefore, the probability that X_{ij} is determined to be 1 at some step t is $\frac{2}{n}\cdot\left(\frac{n-k+i-1}{n}\right)^{t-1}$. This gives us:

$$\mathbf{Pr}_\beta[X_{ij}=1] = \frac{2}{n}\sum_{t=1}^{\infty}\left(\frac{n-k+i-1}{n}\right)^{t-1} \tag{4}$$

$$= \frac{2}{n}\cdot\frac{1}{1-\frac{n-k+i-1}{n}} \tag{5}$$

$$= \frac{2}{n}\cdot\frac{n}{k-i+1} \tag{6}$$

$$= \frac{2}{k-i+1} \tag{7}$$

Similarly, we have

$$\mathbf{Pr}_\beta[X_{ij}=1] = \frac{2}{j-i+1} \qquad \text{for } i\leqslant k\leqslant j \tag{8}$$

$$\mathbf{Pr}_\beta[X_{ij}=1] = \frac{2}{j-k+1} \qquad \text{for } k\leqslant i< j \tag{9}$$

The rest of the proof is a matter of algebraic manipulations. Let H_m be the mth hamonic number, i.e.

$H_m = \sum_{i=1}^{m} \frac{1}{i}$. We have

$$\begin{aligned}
\mathbf{E}[C] &= \sum_{1 \leqslant i < j \leqslant n} \mathbf{Pr}_\beta [X_{ij} = 1] && (10)\\
&\leqslant \sum_{i<j\leqslant k} \mathbf{Pr}_\beta [X_{ij} = 1] + \sum_{i \leqslant k \leqslant j} \mathbf{Pr}_\beta [X_{ij} = 1] + \sum_{k \leqslant i < j} \mathbf{Pr}_\beta [X_{ij} = 1] && (11)\\
&= \sum_{i<j\leqslant k} \frac{2}{k-i+1} + \sum_{i\leqslant k\leqslant j} \frac{2}{j-i+1} + \sum_{k\leqslant i<j} \frac{2}{j-k+1} && (12)\\
&= \sum_{i\leqslant k} \frac{2(k-i)}{k-i+1} + \sum_{i\leqslant k\leqslant j} \frac{2}{j-i+1} + \sum_{k<j} \frac{2(j-k)}{j-k+1} && (13)\\
&\leqslant 2n + \sum_{i\leqslant k\leqslant j} \frac{2}{j-i+1} && (14)\\
&= 2n + \sum_{k\leqslant j} \sum_{i\leqslant k} \frac{2}{j-i+1} && (15)\\
&= 2n + 2\sum_{k<j} (H_j - H_{j-k}) + O(1) && (16)
\end{aligned}$$

Observe that for $a > b > 0$, $H_a - H_b \leqslant \ln \frac{a}{b}$ (one can prove this by showing that the function $f(t) = H_t - \ln t$ is decreasing, or by observing that $H_a - H_b$ is the approximation of $\int_b^a \frac{dx}{x}$ by sum of rectangles.) Thus, the above sum is bounded from above by

$$\begin{aligned}
2n + \sum_{k\leqslant j} \ln \frac{j}{j-k} + O(1) &= 2n + \ln \frac{\prod_{j>k} j}{\prod_{j>k}(j-k)} + O(1) && (17)\\
&= 2n + 2\ln \binom{n}{k} + O(1). && (18)\\
& && (19)
\end{aligned}$$

We can bound the last quantity using two facts which can be proved easily by induction: (i) $\binom{n}{k}$ is maximized when $k = \lfloor n/2 \rfloor$; and (ii) $\binom{n}{\lfloor n/2 \rfloor} \leqslant 2^n$ for all n. With these:

$$\begin{aligned}
2n + 2\ln \binom{n}{k} + O(1) &\leqslant 2n + 2\ln(2^n) + O(1) && (20)\\
&\leqslant 3.5n && (21)
\end{aligned}$$

Problem 2.

1. We will analyze the version of the contraction algorithm that terminates and returns the cut $(v, V\backslash v)$ whenever a vertex v of degree at most $k\ell$ is found. Let $P(i)$ denote the probability that this algorithm will eventually return an approximate cut, given that it preserves one at step i. There are two case at step i: either there is a vertex with degree at most $k\ell$, which means $P(i) = 1$, or there is no such vertex in the graph. In the second case, the number of edges in the graph is at least $\frac{k\ell(n-i)}{2}$. Thus the probability of contracting an edge of any particular min cut is at most $\frac{2}{\ell(n-i)}$. Therefore:

$$P(i) \geqslant \begin{cases} 1 & \text{if there is a node of degree at most } k\ell \\ \left(1 - \frac{2}{\ell(n-i)}\right) P(i+1) & \text{otherwise} \end{cases} \quad (22)$$

Now note that $P(0)$ is precisely the probability that this algorithm returns an approximate cut. We have:

$$P(0) \geqslant \prod_{i=0}^{n-3} \left(1 - \frac{2}{\ell(n-i)}\right) \tag{23}$$

$$= \prod_{j=3}^{n} \left(1 - \frac{2}{\ell j}\right) \tag{24}$$

$$\geqslant \prod_{j=3}^{n} \exp\left(-\frac{2/\ell}{j}\right) \tag{25}$$

$$= \exp\left(-\frac{2}{\ell} \sum_{j=3}^{n} \frac{1}{j}\right) \tag{26}$$

$$> \exp\left(-\frac{2}{\ell} \ln n\right) \tag{27}$$

$$= n^{-2/\ell}. \tag{28}$$

where (25) follows from the fact that $1 - x \geqslant e^{-x}$ for all $0 \leqslant x \leqslant 1$. The last inequality completes the proof.

2. We give an algorithm that lists all approximate st-cuts for some fixed s and t. To list all approximate cuts in the graph, we call this algorithm for all possible s and t. Furthermore, our algorithm is polynomial on nM where M is the number of its outputs.

   First, we arbitrarily order the vertex in V such that s is the first and t is the last vertices. Then, associate ach vertex i with a variable $x_i \in \{0, 1\}$. Given a set $S \subseteq V$ and a *partial assigment* α to $\{x_i : i \in S\}$, we say that a cut C yields α if $\alpha(x_i)$ is the indicator that i is on the same side with s in C. Note that each cut yields a unique *total assignment* to $\{x_i : i \in V\}$.

   We say that an assignment is *good* if some approximate cut yields it. Given an assignment α for a set S, we can check if α is good in polynomial time as follows. First, for each $i \in S$, if $\alpha(x_i) = 1$ then "identify" i and s, i.e. imagine that the edge si exists and contract it; otherwise identify i and t. Then, run a polynomial MinCut algorithm to compute the min cut of the resulting graph. If the size of this min cut is at most $k\ell$, answer "yes"; otherwise, answer "no". One can easily check the correctness of this algorithm.

   Now, given a partial assignment α to $\{x_1, x_2, \ldots x_i\}$, we construct all total assignments yielded by approximate cuts which are extensions of α by the following algorithm $\texttt{AppCuts}(\alpha)$.

   1. $S = \emptyset$.
   2. $\alpha_1 = \alpha \cup \{x_{i+1} = 1\}$.
   3. $\alpha_0 = \alpha \cup \{x_{i+1} = 0\}$.
   4. If α_1 is good: $S = S \cup \texttt{AppCuts}(\alpha_1)$.
   5. If α_2 is good: $S = S \cup \texttt{AppCuts}(\alpha_2)$.
   6. Return S.

   One can easily check that the correctness of this algorithm. Furthermore, $\texttt{AppCut}(\emptyset)$ returns all good assignments. We analyze the running time of the algorithm in the following.

   Consider the computation tree of the algorithm. At each node of this tree, the algorithm consider a partial assignment α. We name this node α and say that it's good iff α is good. Clearly, each good node besides the leaves have two children and the bad nodes are leaves of the tree.

   Furthermore, besides the root of the tree, each bad node has a good sibling; for otherwise, its parent would have been a bad node. Thus, the number of bad nodes is at most the number of good nodes.

Observe that all the good nodes lie on the paths from the root to the good leaves, which correspond to good total assignments. Since we increase the size of the partial assignment by 1 at each level, the length of each such path is at most n. Thus, the number of good nodes is at most nM. This means the size of the computation tree is polynomial on nM. Together with the fact that the computation at each node is polynomial on n, this shows that the total running time is polynomial on nM.

Finally, we will prove that M is at most a polynomial of n, thus completes the proof that our algorithm is a polynomial time one. To do so, consider the variant of the contraction algorithm where we keep contracting until at most 2ℓ vertices left, then output a random cut of the remaining graph. We claim that the probability that the algorithm returns a particular cut C is at least $n^{-2\ell}$.

Note that at any stage of the algorithm, none of the vertices has degree smaller than k, for otherwise we would have a cut of size smaller than k, contradicting the fact that k is the size of the min cut. Thus, the probability that an edge of C is contracted when there are i vertices left is at most $\frac{|C|}{ik/2} \leqslant \frac{2\ell}{i}$. Similar to above, this means the probability that C "survives" all the contractions, i.e. none of its edges is contracted is at least

$$\prod_{i=2\ell+1}^{n} \left(1 - \frac{2\ell}{i}\right) \geqslant \prod_{i=2\ell+1}^{n} \exp\left(-\frac{2\ell}{i}\right) \tag{29}$$

$$= \exp\left(-2\ell \sum_{i=2\ell+1}^{n} \frac{1}{i}\right) \tag{30}$$

$$\geqslant \exp\left(-2\ell \left(\ln n - \ln(2\ell)\right)\right) \tag{31}$$

$$= n^{-2\ell} \cdot (2\ell)^{2\ell} \tag{32}$$

Given that C survies all the contraction, the probability that C is returned after the last step is $2^{-2\ell}$, since there are $2^{2\ell}$ cuts in a graph of 2ℓ vertices. Therefore, the prbability that C is returned is at least

$$n^{-2\ell} \cdot (2\ell)^{2\ell} \cdot 2^{-2\ell} \geqslant n^{-2\ell} \tag{33}$$

Since each approximate cut is returned with probability at least $n^{-2\ell}$, the number of approximate cuts is at most $n^{2\ell}$. This completes the proof.

Problem 3. This problem is purely an excercise of calculations.

Problem 4.

1. When x is a vertex of the feasible region, n constrain must be tight, i.e. there is a $n \times n$ submatrix A_T of A and a subset b_T of n coordinates of b such that $A_T x = b$. Since A is non-degenerate, $\det(A_T) \neq 0$. Thus, $\det(A_T) \in \{-1, 1\}$. By Crammer's rule:

$$x_i = \frac{\det(A_T^i)}{\det(A_T)} \tag{34}$$

where A_T^i is the matrix obtained by replacing the ith column of A_T by b_T. Since all entries of A_T^i are integers, $\det(A_T^i)$ is an integer. Therefore, x_i's are integers.

Since an optimal solution of the integer program is a vertex of the feasible region, it is an integral solution.

2. The linear program from Maximum Matching is as follows:

$$\max \quad \sum_{e \in E} x_e \tag{35}$$

$$\text{s.t.} \quad \sum_{e \in E(v)} x_e \leqslant 1 \qquad \forall v \in V \tag{36}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{37}$$

where $E(v)$ denotes the set of edges incident to v.

3. The dual of the relaxation of the above integer linear program is:

$$\min \quad \sum_{i \in V} y_i \tag{38}$$

$$\text{s.t.} \quad y_i + y_j \geqslant 1 \qquad \forall ij \in E \tag{39}$$

$$y_i \geqslant 0 \qquad \forall i \in V \tag{40}$$

   This linear program is a natural linear program relaxation of the minimum vertex cover problem.

4. Since G is a bipartite graph, we can partition V into two subsets L and R such that any edge of E connects a vertex in L and a vertex in R. From now on, when we say edge ij, we assume that $i \in L$ and $j \in R$.

   Rewrite the constrains of the linear program for maximum matching in the form $Ax \leqslant b$ and consider any $d \times d$ submatrix B of A. We show that $\det(B) \in \{-1, 0, 1\}$ by induction on d.

   For $d = 1$, the assertion is immediate since each entry of A is either 1 or 0. Assume that the assertion holds for $d - 1$, we prove that it holds for d. There are 3 cases:

   - B contains a row of all zero's, then $\det(B) = 0$.
   - B contains a row with at most one non-zero entry $b_{k,l} = 1$ then $\det(B) = \sum_{h=1}^{d} (-1)^{k+h} b_{k,h} B^{k,h} = (-1)^{k+l} B^{k,l}$ where $B^{k,h}$ is the matrix obtained by removing the kth row and hth column of B. By the induction hypothesis, $\det(B^{k,l}) \in \{-1, 0, 1\}$, thus $\det(B) \in \{-1, 0, 1\}$.
   - each row of B contains at least 2 non-zero entries. Then the number of non-zero entries of B is at least $2d$. On the other hand, the number of non-zero entries on each column of B is at most 2. Thus, the number of non-zero entries of B is exactly $2d$ and the number of non-zero entries on each row and on each column is exactly 2.

     Let L_B and R_B be the set of rows of B corresponding to vertices in L and R respectively. Then each column must contain a 1 in a row in L_B and a 1 in a row in R_B. Hence, the number of non-zero entries in the rows in L_B and R_B are exactly d. Let u and v be the sum of all rows in L_B and R_B respectively. Then $u_i \geqslant 1$ for all i. Furthermore, $\sum_{u_i} = d$, since the number of non-zero entries in the rows in L_B is exactly d. Thus, u must be the all-1 vector. Similarly, v is the all-1 vector. This means the rows of B are linearly dependent. Hence, $\det(B) = 0$.

   Note that the coefficent matrix of the dual is A^T. Thus the stated property also holds for this matrix.

   Finally, both linear program are feasible, as $x = 0$ is a solution for the primal and $y = 1$ is a solution for the dual; and bounded, as the primal is bounded from above by $|E|$ and the dual is bounded from below by 0. Thus, both of them has integral optimal solutions.

5. We conclude that in biparty graph, the minimum vertex cover equals to the maximum matching.