IEOR 8100: Randomized algortihms homwork solutions
Problem set 1

1. (Exercise 1.2) The intuitive reason why the "node-picking" approach fails is that it does not take the edge structure of the graph into account. Consider the graph $G = (U \cup V, E)$ where U and V are cliques (i.e., for all $u, u' \in U, (u, u') \in E$ and the same is true for V) and there is a single edge (u, v) between the the cliques for a particular $u \in U, v \in V$. Suppose $|U| = |V| = n$. The minimum cut of this graph is $[U, V]$ with value 1. Every other cut has value greater or equal to n. Thus, in order for the modified algorithm to find the minimum cut, at every iteration, only nodes from the same clique can be contracted, i.e., the algorithm must contract two nodes where both are elements of U or both are elements of V. Furthermore, if the algorithm is successful, the last two nodes remaining will be such that one is the contraction of the nodes in U and the other the contraction of the nodes in V. Let $\mathcal{A}$ represent this event. Let $\mathcal{B}$ denote the event that the algorithm terminates with each of the final two nodes containing the contraction of n nodes (from either set). Then,

$$\begin{aligned}
\Pr[\text{success}] &= \Pr[\mathcal{A}] \\
&= \Pr[\mathcal{A}|\mathcal{B}]\Pr[\mathcal{B}] \\
&\leq \Pr[\mathcal{A}|\mathcal{B}] \\
&= \frac{2}{\binom{2n}{n}} \qquad (*) \\
&\leq \left(\tfrac{1}{2}\right)^{n-1} \qquad \text{by Prop. B.2, 4.}
\end{aligned}$$

   To see that (*) is true, consider the algorithm as a random distribution on the partitions of the nodes into two sets. Denote the set which contains the first element the algorithm chooses the "marked" set. Given that this distribution partitioned the nodes into two sets of equal size (i.e, event $\mathcal{B}$), the probability of a node ending up in the marked or unmarked set is equally likely by the symmetry of the problem. Thus, each partition occurs with an equal probability of $\binom{2n}{n}^{-1}$, which is the probability of choosing a particular set of n balls from an urn of $2n$ balls. Since there are two partitions which result in success (U is the marked set or V is the marked set), (*) follows.

2. (Problem 1-8)

   (a) Consider the following multigraph, $G = (V, E)$. Let $V = \{s, t, v_1, \ldots, v_n\}$ and $E = E_S \cup E_T$ where $E_S = \{(s, v_i) : i = 1, \ldots, n\}$ and $E_T = \{(v_i, t) : i = 1, \ldots, n\}$ and the arcs in E_T occur twice. The minimum s-t cut consists of the node s and has value n. So there is one arc from s to each node v_i for all i, and two arcs from v_i to t for each i. Then, in order for the algorithm to terminate successfully, only arcs from E_T can be contracted. If an arc from E_s is contracted, the cut value returned will be at least $n + 1$, as neither of the two arcs from the contracted node can be chosen (as the contracted node is now the s-vertex). By this same argument, every time an edge is contracted, three arcs (including the one contracted) are no longer possible choices as they are either contracted or arcs connecting the s-vertex to the t vertex. Thus, at every iteration the probability of choosing a contraction preserving the possibility of success is $2/3$, and there are a total of n iterations. Then the probability of success is $(2/3)^n$.

   (b) The number of s-t minimum cuts can be exponential in the number of nodes of the graph. Consider the graph G from part (a) with the modfication that every edge in E_T occurs once. Then every subset including s and not including t defines a minimum cut with value n. There are 2^n such sets.

3. (implementation of Min-Cut Algorithm) Here is the "simple" method suggested in Karger and Stein (1996), which is available from Cliff's website. For the minimum cardinality minimum cut problem, it runs in $O(n^2 + m \log m)$ time and uses $O(n^2 + m)$ space. For the general weighted case the runtime becomes $O(n^2 + m \log W)$ where W is the total sum of the weights. This bound is not strongly polynomial. Karger and Stein describe an improved, although more complicated, algorithm with a strongly polynomial runtime.

   In order to implement the algorithm, there are two tasks that must be completed:

   - Randomly choosing an (multi-)edge;
   - Contracting an edge.

   To perform these tasks, the following two data structures will be maintained:

   - $\mathbf{W} = (w_{(u.v)})$, a node-node adjacency matrix where $w_{(u,v)}$ represents the number of edges between the nodes u, and v ;
   - $\mathbf{d} = (d_u)$ where d_u denotes the degree of node u.

   Note that $\mathbf{W}$ is symmetric, i.e., $w_{(u,v)} = w_{(v,u)}$, as the graph is undirected.

   The algorithm requires a subroutine that, given an integral vector $\mathbf{x} = (x_i)$, returns the number i with probability x_i. To do this, use the black box to generate a random integer, r, distributed uniformly between 1 and m. Then binary search the vector $\mathbf{y} = (y_i), i = 0, \ldots, n$, where $y_i = \sum_{j=1}^{i} x_j (y_0 = 0)$ to find an indice k such that $y_{k-1} < r \leq y_k$. The output of the subroutine will be the indice k. Label this subroutine RANDOM-BIN-SELECT and note that it requires $\log(\sum_i x_i)$ time.

   Then, to choose a random edge, call RANDOM-BIN-SELECT on the vector $\mathbf{d}$ to obtain a node u. Then call RANDOM-BIN-SELECT on the vector $(w_{u,v}), v \in V$. Label this subroutine RANDOM-EDGE and note that it requires $O(\log m)$ time, as $\sum_i d_i = 2m \geq sum_{v \in V} w_{(u,v)}, \forall u$.

   Lemma 1 RANDOM-EDGE *chooses an edge* (u, v) *with probability* $w_{(u,v)}/m$.

   Proof: Suppose the the black box generates the integer $r \in \{1, \ldots, m\}$. Then

$$\begin{aligned}
\Pr[(u,v) \text{ chosen}] &= \Pr[u \text{ chosen}] \Pr[(u,v) \text{ chosen} \text{ — } u \text{ chosen}] + \Pr[v \text{ chosen}] \Pr[(u,v) \text{ chosen} \text{ — } v \text{ chosen}], \\
&= \frac{d_u}{2m} \frac{w_{(u,v)}}{d_u} + \frac{d_v}{2m} \frac{w_{(v,u)}}{d_v}, \\
&= \frac{w_{(u,v)}}{m}.
\end{aligned}$$

   ■

   To contract an edge (u, v), the $\mathbf{d}$ and $\mathbf{W}$ matrix must be updated so that (WLOG) the u vertex attains all the edges of the v vertex, excluding any (u, v) edges. This can be accomplished via the following equations

   (1) $d_u \leftarrow d_u + d_v - 2w_{(u,v)}$
   (2) $w(u, v) \leftarrow w(v, u) \leftarrow d_v \leftarrow 0$
   - For all vertex $w \in V \setminus \{u, v\}$
     (3) $w_{(w,u)} \leftarrow w_{(u,w)} \leftarrow w_{(u,w)} + w_{(v,w)}$
     (4) $w_{(w,v)} \leftarrow w_{(v,w)} \leftarrow 0$

Equations (1), (3) and (4) change the degree of u and update $\mathbf{W}$ to reflect the additional (if any) edges from v and the removal of all (u, v) edges. Equation (2) removes all (u, v) edges from the $\mathbf{W}$ matrix and reduces the degree of v to zero. The total addition operations required for these equations is $2+n-2=n$ and the total memory operations required is $1+3+2(n-2)+2(n-2)=4n-4$. Thus, the total number of operations required to contract an edge is $O(n)$. Thus, the total time required to implement the algorithm is $O(n^2)$ and the total space, if a full matrix is used, is $O(n^2)$.

4. (Bound on the number of minimum cuts in a graph with n nodes) Note that the min-cut algorithm can be seen as a probability distribution on cuts in a particular graph. Suppose, for the sake of contradiction, that there were $k > n(n-1)/2$ minimum cuts in the graph. Since, for each of these cuts, the min-cut algorithm is guaranteed to generate it with probability at least $\frac{2}{n(n-1)}$, the total probability of generating any of these cuts is greater than one, a contradiction.