**CS787: Advanced Algorithms** **Fall 2009**

**Homework 1 Solutions**

**Lecturer:** Shuchi Chawla **TA:** Keith Langston

## Exercises

1. Let $T$ be the collection of maximum bottleneck paths. We show that if $T$ is not a tree, then its cycle(s) can be eliminated while preserving the maximum bottleneck property. Suppose that $T$ contains a cycle $C$. Let $e$ be the edge in $C$ with minimum weight. Since $e \in T$, there exist two vertices $u$ and $v$ such that the maximum bottleneck path $P$ between $u$ and $v$ contains $e$. The bottleneck of this $u$–$v$ path is at most $c_e$. Rerouting this path along the other side of $C$ (i.e. not using $e$) results in another $u$–$v$ path. The bottleneck of this new path is at least the bottleneck of $P$, since all edges along this alternate path have weight at least $c_e$. Hence, the cycle $C$ can be eliminated by safely removing $e$ from $T$.

   The following modified Kruskal's algorithm finds $T$: Consider edges in decreasing order of weight, and pick each edge that does not form a cycle with previously included edges. Just like Kruskal's, this algorithm runs in $O(m \log n)$ time.

2. Let $G = (X, Y; E)$ be a bipartite graph. Suppose that $G$ has a perfect matching $M$. Then for each $S \subseteq X$, $|\Gamma(S)| \geq |S|$ since $\Gamma(S)$ contains all vertices in $Y$ that are matched to vertices in $S$. Similarly, $|\Gamma(T)| \geq |T|$ for all $T \subseteq Y$.

   Now assume that $G$ does not have a perfect matching and let $M$ be a maximum matching of $G$. We treat $G$ as a flow network (having invisible source $s$, sink $t$ and a few more edges) and $M$ as a flow in $G$. Let $S$ be the set of vertices in $X$ that can be reached from the source in the residual network. Let $u \in S$ and $v$ be one of $u$'s neighbors. Then $v$ must be matched, otherwise we could find an augmenting path (the path from $s$ to $u$, $(u, v)$, followed by $(v, t)$) and increase the flow. Suppose that $v$ is matched to $w$. Then $w \in S$, because (1) if $v$ is matched to $u$, then $w = u \in S$; (2) if $u \neq w$, then the edges $(u, v)$ and $(v, w)$ are in the residual network, so $w$ is also reachable from $s$. Let there be $k$ matched vertices in $S$. From the above observations, we can conclude that $|\Gamma(S)| = k < |S|$. (The last inequality holds because all unmatched vertices of $X$ are contained in $S$). This shows that Hall's conditions is violated by some $S \subseteq X$.

3. (a) Flip the coin twice. If it comes up H followed by T, output H; if it comes up T followed by H, output T; otherwise, repeat. By symmetry we can see that the probability of outputting H is indeed 1/2. The probability that two consecutive coin tosses have different outcomes is $2p(1-p)$, hence the expected number of tosses required is $2/(2p(1-p)) = 1/(p(1-p))$.

   (b) This idea is to write $p$ in binary. Let's say H corresponds to 0 and T correspond to 1. Flip coins until the generated sequence differs from the binary expansion of $p$. Output the result of the last toss. The probability of getting the "wrong" outcome at any position is 1/2, regardless of the value of $p$. Hence the expected number of tosses needed is 2.

(c) We first show that picking an integer uniformly at random from $[1, n]$ requires $O(\log n)$ tosses of a fair coin (in expectation). The basic idea is to toss a coin $\lceil \log n \rceil$ times to get a binary number of length $\lceil \log n \rceil$. With probability less than $1/2$, this number will be strictly larger than $n$, so we must try again. We need to try this at most twice (in expectation) in order to generate an integer in $[1, n]$. Therefore, the expected total number of coin tosses is less than $2 \lceil \log n \rceil$. Given such a random number generator, we can shuffle a deck of $n$ cards as follows: Generate a random number $k$ between 1 and $n$. Take out the $k$-th card from the bottom and put it on top of the deck. Decrement $n$ by 1 and repeat until $n$ reaches 0. $n$ random numbers need to be generated, hence the number of coin tosses needed is less than $2n \lceil \log n \rceil$.

**Problems**

1. (a) The reduction is as follows.
   i. Add an edge $e' = (t, s)$ to $G$. Let $G'$ be the resulting graph. Set $c_{e'} = \sum_{e \in E} f_e$ and $\ell_{e'} = -M$ where $M$ is a very large positive integer (e.g. $M = 1 + \sum_{e \in E} |\ell_e|$).
   ii. Run MCC on $G'$.
   iii. Let $f$ be the min-cost circulation of $G'$. Then, $f$ (ignoring the flow on $e'$) is a min-cost max-flow of $G$.

   We first show that $f$ is a maximum flow of $G$. Clearly, $f$ is a feasible flow of $G$, since it satisfies the capacity constraints of all edges, and flow is conserved at every vertex other than $s$ and $t$. To show maximality, we consider the cost of circulation $f$. If $f$ is not a maximum flow of $G$, then there exists an augmenting path $P$ in the residual graph $G_f$. We can augment the circulation in $G'$ by pushing flow along the cycle formed by $P$ and $e'$. Since $\ell_{e'} = -M$, the increase in the circulation cost must be negative. This contradicts our assumption that $f$ is a min-cost circulation. Therefore, $f$ is a maximum flow of $G$.

   Next, we show that $f$ is a min-cost max-flow of $G$. Let $\tilde{f}$ be any maximum flow of $G$. The cost of the flow $\tilde{f}$ is $\ell(\tilde{f}) = \sum_{e \in E} \tilde{f}_e \ell_e$. We augment $\tilde{f}$ to a circulation of $G'$ by setting $\tilde{f}_{e'} = |\tilde{f}|$. The cost of the circulation $\tilde{f}$ is $\sum_{e \in E} \tilde{f}_e \ell_e + \tilde{f}_{e'} \ell_{e'} = \ell(\tilde{f}) - |f|M$. We see that the difference between the cost of a max flow and that of the corresponding circulation is always $-|f|M$. Since $f$ is a min-cost circulation, we can conclude that $f$ has minimum cost among all max flows of $G$.

   (b) (For convenience, we allow the residual graph to contain multiple edges.) For every edge $e = (u, v) \in E$, we add one or two edges in the residual graph:
   i. if $f_e < c_e$, add an edge $e_1 = (u, v)$ to the residual graph with $c_{e_1} = c_e - f_e$ and $\ell_{e_1} = \ell_e$;
   ii. if $f_e > 0$, add an edge $e_2 = (v, u)$ to the residual graph with $c_{e_2} = f_e$ and $\ell_{e_1} = -\ell_e$.

   Intuitively, sending flow along $e_1$ corresponds to sending more flow along $e$, while sending flow along $e_2$ corresponds to taking away some flow from $e$.

(c) Let $f$ be a circulation. Let $C$ be a negative-cost cycle in the residual graph $G_f$ and $\ell_C$ be its cost. Suppose that we saturate $C$ by pushing $\delta$ units of flow into $C$. It is straightforward to verify that the augmented circulation $f+C$ has cost $\ell(f)+\delta\cdot\ell_C$. Since $\ell_C < 0$ and $\delta > 0$, the cost of the circulation always decreases after each augmentation. Assuming that all capacities are integral, the cost is decreased by at least $\min_{e\in E}|\ell_e|$ each time. Since the minimum circulation cost is finite (a lower bound is $-\sum_{e\in E} c_e|\ell_e|$), the algorithm always terminates.

(d) Let $f$ be the output of the algorithm. Suppose that a different circulation $f^*$ has a smaller cost than $f$. Consider the difference $g = f^* - f$, which is also a circulation. (Note: $g$ may not be feasible in $G$.) Observe that $g$ is feasible in $G_f$. (In case this is not obvious, consider for each edge $e$ the following two cases: (1) $f_e \geq f^*_e$; (2) $f_e < f^*_e$.) Moreover, since $\ell(f^*) < \ell(f)$, $g$ has negative cost in $G_f$. Hence, $g$ contains a negative-cost cycle, which implies that $G_f$ has a negative-cost cycle — a contradiction to the assumption that the algorithm has terminated.

2. To compute the expected running time, we divide the process of selection into $r = \log n$ rounds, where in round $i$ the size of the list under consideration is between $2^{r-i}$ and $2^{r-i+1}$, and analyze the expected running time of each round separately. Essentially what we need to do is compute the expected number of recursive calls to reduce the size of the list under consideration by half.

   Let $A$ be an unordered list of $n$ distinct elements and we use `SELECT` to find the $k$-th largest element of $A$. Suppose that at the beginning of some round $i$, the list $A_i$ under consideration has size $\ell$. Now, we call an iteration in this round "good" if the length of the list decreases by a factor of at least $3/4$ in the worst case, and "bad" otherwise. The probability that an iteration is good is at least $1/2$, because this happens whenever we pick elements ranked between $1/4$ the size of the list and $3/4$ the size of the list. Furthermore, we need only $\lceil \log 2/(\log 4/3)\rceil = 3$ good iterations to reach the next round. So the expected number of iterations in this round is at most 6. The time per iteration in this round is $O(\ell)$, so the expected total time for this round is also $O(\ell)$. By linearity of expectation, the total expected running time is the sum of the expected running times of all the rounds, which is $O(n+n/2+n/4+\cdots+2+1) = O(n)$.

   An alternate way to analyze the running time is to compute the expected length of the list at every step. If we start with a list of length $n$, then in expectation we throw out $n/2-k(n-k)/n$ items (compute this yourself!). This expression is smallest when $k = n/2$, in which case, the expected number of remaining elements is at most $3n/4$. Thus the expected size of the list decreases by a constant factor every time, and we can use this along with Markov's inequality to analyze the number of rounds.

3. (a) Let $X_i$ be a random variable that takes value 1 if the $i^{th}$ element is uniquely covered, and 0 otherwise. Let $X = \sum_{i=1}^{n} X_i$ be the number of uniquely covered elements. Then $\mathbf{E}[X_i] = \mathbf{Pr}\big[\ i^{th}$ element uniquely covered$\big] = Fp(1-p)^{F-1}$. By linearity of expectation, $\mathbf{E}[X] = \sum_{i=1}^{n} \mathbf{E}[X_i] = nFp(1-p)^{F-1}$.
   We find the value of $p$ maximizing $\mathbf{E}[X]$ by looking at the derivative of $\mathbf{E}[X]$ with respect

to $p$:

$$(\mathbf{E}[X])' = nF((1-p)^{F-1} - (F-1)p(1-p)^{F-2}) = 0 \tag{0.0.1}$$

$$\implies 1 - p = (F-1)p \tag{0.0.2}$$

$$\implies p = 1/F. \tag{0.0.3}$$

It can be verified that $p = 1/F$ is a global maximum. The maximum value of $\mathbf{E}[X]$ is $n(1-1/F)^{F-1}$.

(b) We apply the algorithm in part (a) directly (setting $p = 1/F$). For an element $x$ with frequency $F_x$, the probability that $x$ is uniquely covered is $(F_x/F)(1-1/F)^{F_x-1}$. Since $F_x \in [F/2, F]$, we have $F_x/F \geq 1/2$, and $(1-1/F)^{F_x-1} > (1-1/F)^{F_x} \geq (1-1/F)^F \geq 1/4$, therefore the probability is at least $1/8$ (here we assumed $F \geq 2$). By linearity of expectation, the expected number of uniquely covered elements is at least $n/8$. This is a constant factor approximation since an optimal solution uniquely covers at most $n$ elements.

(c) Let $S$ be the set of $n$ elements. We partition $S$ into $k = \lceil \log_2 m \rceil$ subsets $S_1, \ldots, S_k$. For $1 \leq i \leq k$, $S_i = \{x \in S \mid \text{the frequency of } x \text{ is in } (2^{i-1}, 2^i]\}$. (Note: we put elements with frequency 1 into $S_1$.) Then, for $1 \leq i \leq k$, we construct the following instance of Unique Set Cover: The set of elements is $S_i$; the subsets are the ones in the original instance, except that we remove all elements not in $S_i$ from the subsets. By taking $F = 2^i$ and running the algorithm in part(b) on this instance, we get a constant factor approximation for this instance.

Let $OPT$ be an optimal solution for the original instance. Let $U$ be the elements in $S$ uniquely covered by $OPT$. By the pigeonhole principle, there exists some $S_i$ containing at least $OPT/\log_2 m$ elements of $U$. So, the optimal solution for instance $S_i$ uniquely covers at least $OPT/\log_2 m$ elements. Suppose that the algorithm in part (b) gives a solution with size $c \cdot OPT/\log_2 m$ for some constant $c$. By picking exactly the same subsets in the original instance, we have got a $O(\log m)$-approximation.

(d) Pick a minimal collection of subsets $\mathcal{C}$ that covers $S$. Clearly, $\mathcal{C}$ contains at most $n$ subsets. We are going to discard all sets not in $\mathcal{C}$ and run the algorithm from part (c) on this collection. As in part (c), we partition $S$ into subsets according to the frequencies of the elements. However, this time we look at the frequency of an element in $\mathcal{C}$ rather than the whole collection of $m$ subsets. Thus there are only $\log n$ partitions. By the pigeonhole principle, the largest partition has at least $n/\log n$ elements. Applying the constant factor approximation algorithm in part (b) to the collection $\mathcal{C}$ with this partition of elements, the expected number of uniquely covered elements is at least $n/(8\log n)$. This gives a $O(\log n)$-approximation for Unique Set Cover.

4. We can model the game as a bipartite graph $G = (X, Y; E)$ where $E = \{(x, y) | x \in X, y \in Y,\ x \text{ and } y \text{ have appeared in the same movie }\}$. We claim that $P_1$ has a winning strategy if and only if $G$ has a perfect matching. Given this claim, a polynomial-time algorithm for determining which of the two players has a winning strategy is simply finding a maximum matching of $G$: if the matching covers all vertices, then $P_1$ has a winning strategy, otherwise

$P_0$ has a winning strategy. The proof below also shows how to compute the winning strategy in polynomial time.

The proof of the claim is as follows. Suppose that $G$ has a perfect matching $M$. Then $P_1$ has the following winning strategy: Whenever $P_0$ names an actress $x$, $P_1$ names the actor $y$ matched to $x$ in $M$. Since $y \in Y$ is matched to exactly one vertex in $M$, $P_0$ can never force $P_1$ to name the same actor twice. Hence, $P_1$ always wins using this strategy. Now suppose that $G$ does not have a perfect matching. Then by Hall's theorem, there exists a subset of actresses $S \subseteq X$ such that $|\Gamma(S)| < |S|$. We want $S$ to be a *minimal* such subset, i.e., every proper subset $S'$ of $S$ satisfies $|\Gamma(S')| \geq |S'|$. Let $S'$ be any subset of $S$ with $|S'| = |\Gamma(S)|$. Consider the subgraph $G'$ of $G$ induced by $V' = S' \cup \Gamma(S)$, i.e., we only consider the vertices $V'$ and edges with both endpoints in $V'$. Since $S$ is a minimal set violating Hall's condition, we know that $G'$ satisfies Hall's condition, and hence $G'$ has a perfect matching $M'$. Going back to $G$, we conclude that there exists a matching for $G$ in which every $y \in \Gamma(S)$ is matched to some $x \in S$. Now, we can describe $P_0$'s winning strategy: $P_0$ starts by naming an actress in $S$ that is not matched (in particular, any vertex in $S \backslash S'$). Then, $P_1$ is forced to name an actor in $\Gamma(S)$. Now, since every vertex of $\Gamma(S)$ is matched to some vertex in $S$, $P_0$ can simply copy $P_1$'s strategy in the case where a perfect matching exists and forces a win. To compute the winning strategy, note that we can use max flow to find some set $S$ with $|\Gamma(S)| < |S|$. Now pick some $S' \subseteq S$ of size $|\Gamma(S)|$. Try to find a perfect matching on the graph $G'$ induced by $S' \cup \Gamma(S)$. If one exists, we use it to construct a winning strategy as in the proof. If not, then we recurse on the graph $G'$ by reapplying Hall's theorem. Each time we find a smaller and smaller set $S$, and eventually terminate with a minimal one.

5. To prove that the given family is 2-universal we need to show the following:

   For all $x, y \in [m]$ and $x \neq y$, $\Pr_{a,b}[h_{a,b}(x) = h_{a,b}(y)] \leq 1/n$.

   Suppose that $x, y \in [m]$ and $x \neq y$. We want to count the number of pairs $(a, b)$ such that $((ax + b) \mod p) \mod n = ((ay + b) \mod p) \mod n$. Let $r = (ax + b) \mod p$ and $s = (ay + b) \mod p$. Note that $r \neq s$, otherwise $ax + b \equiv ay + b( \mod p) \implies a(x - y) \equiv 0( \mod p) \implies x = y$. As an intermediate step, we want to count the number of pairs $(r, s) \in [p] \times [p]$ satisfying $r \mod n = s \mod n$ and $r \neq s$. Fixing $r$, the number of choices for $s$ is at most $\lceil p/n \rceil - 1 = \lfloor p/n \rfloor \leq (p - 1)/n$. Thus the total number of such pairs is at most $p(p-1)/n$. Now, given such a pair $(r, s)$, we want to count the number of pairs $(a, b)$ satisfying

   $$\begin{cases} r &= (ax + b) \mod p \\ s &= (ay + b) \mod p \end{cases}.$$

   It can be verified that this system always has a unique solution with $a \neq 0$ (provided that $x \neq y$ and $r \neq s$). Hence, the number of pairs $(a, b)$ such that $((ax+b) \mod p) \mod n = ((ay+b) \mod p) \mod n$ is at most $p(p - 1)/n$. Since the family has $p(p - 1)$ hash functions, the probability that a randomly picked function maps $x$ and $y$ to the same output is at most $1/n$.

6. We first compute the expected number of rounds, then use Markov's inequality to show the high probability result we want.

Let us first see how the number of balls reduces over time in expectation. Suppose that at the beginning of some round we have $m$ balls and $n$ bins. Then the probability that any particular ball in this round is involved in a collision is $1 - (1 - 1/n)^{m-1}$. Therefore, the expected number of balls remaining after this round is $m - m(1 - 1/n)^{m-1}$. Let $m_i$ be the number of balls remaining after round $i - 1$, so $m_1 = n$. In order to simplify the above expression, we are going to consider two special cases.

(a) Case 1: $m_i \geq n/4$. The expected number of remaining balls is $m_{i+1} \leq m_i - m_i(1 - 1/n)^n \leq m_i(1 - 1/4)$ (using $n \geq 2$).

(b) Case 2: $m_i < n/4$. The expected number of remaining balls is $m_{i+1} \leq m_i - m_i(1 - (m_i - 1)/n) = m_i(m_i - 1)/n$.

Now suppose for a moment that the number of balls remaining after each round is *exactly equal* to its expectation. (Of course, this doesn't really happen.) Then, we stay in the first case for some constant number of rounds (5). In the second case, the number of balls goes down in the following manner: $n/4 \to n/16 \to n/256 \to \cdots \to n/2^{2^{i+1}} \to \cdots$. In order for the number to go down to one, we need $2^{2^{i+1}} > n$, which gives $i = O(\log \log n)$.

However, the number of remaining balls is not exactly equal to its expectation. So instead we will use the following trick and employ Markov's inequality: Call a round "good", if the number of balls remaining after this round is no more than 2 times its expectation. Then Markov's inequality tells us that any particular round is good with probability at least $1/2$. Now if we look at the series $\{m_i\}$, with $m_{i+1} \leq 2m_i(m_i - 1)/n$, we note that we still require only $O(\log \log n)$ good rounds for $m_i$ to decrease down to 1. (Note that in the bad rounds, $m_i$ may not decrease by much but at least it doesn't increase.)

How many rounds do we need to see in expectation before we have had $O(\log \log n)$ good rounds? We can apply Markov's inequality again to see that we need only $O(\log \log n)$ rounds for our failure probability to go down to any desired constant. We can get better failure probabilities by applying more advanced concentration results which we haven't discussed in class yet.