G22.3520: Honors Analysis of Algorithms

Problem Set 5

Partly typeset by Srihari Narayanan

Problem 1

Solve [Kleinberg-Tardos Ch. 13, Prob. 1, pg. 782].

Solution: Let $G(V, E)$ be our graph, $n := |V|$, $m := |E|$. We simply pick one of the 3 colors for each vertex uniformly at random, taking time $O(n)$. To see correctness, let M be the random variable representing the number of edges satisfied, and let M_e be the indicator variable for the event that edge e is satisfied. Clearly, we have $M = \sum_{e\in E} M_e$ and $\mathbb{E}[M] = \sum_{e\in E} \mathbb{E}[M_e]$ by linearity of expectation, and by a simple calculation the probability that a particular edge is satisfied is $\frac{2}{3}$. We thus have $\mathbb{E}[M] = \frac{2}{3}m \geq \frac{2}{3}c^*$, as desired. $\square$

The deterministic version of this algorithm will be left as an exercise; the most efficient solution runs in time $O(n + m)$.

Problem 2

Solve [Kleinberg-Tardos Ch. 13, Prob. 10, pg. 789].

Solution: First put the bids in decreasing order $b_1, \dots, b_n$. Let B be the random variable representing the number of updates to b^*, and let B_k be the indicator variable for the event that b_k results in an update to b^*. Clearly, we have $B = \sum_{k=1}^{n} B_k$ and $\mathbb{E}[B] = \sum_{k=1}^{n} \mathbb{E}[B_k]$ by linearity of expectation. We now realize that the event that b_k results in an update to b^* is exactly the event that bids $b_1, \dots, b_{k-1}$ occur after b_k. Considering the relative ordering of these k bids, we then realize this event has probability $\frac{1}{k}$ by symmetry. (Formally, first note that every permutation has equal probability, and then permute the labels of bids $b_1, \dots, b_k$ around.) We thus have $\mathbb{E}[B_k] = \frac{1}{k}$ and $\mathbb{E}[B] = \sum_{k=1}^{n} \frac{1}{k}$. $\square$

Problem 3

Solve [Kleinberg-Tardos Ch. 13, Prob. 11, pg. 789–90].

Solution:

(a) Let M be the random variable representing the number of machines with no job assigned to them, and let M_i be the indicator variable for the event that no job is assigned to machine i. We clearly have $M = \sum_{i=1}^{k} M_i$ and $N(k) = \mathbb{E}[M] = \sum_{i=1}^{k} \mathbb{E}[M_i]$ by linearity of expectation. The probability that a particular job j is assigned to a machine other than i is $1 - \frac{1}{k}$, and these events are mutually independent over j, so the event that no job is assigned to machine i is $\left(1 - \frac{1}{k}\right)^k$. We thus have $\mathbb{E}[M_i] = \left(1 - \frac{1}{k}\right)^k$, $N(k) = \mathbb{E}[M] = k\left(1 - \frac{1}{k}\right)^k$, and, by a well-known limit, $\lim\limits_{k\to\infty} \frac{N(k)}{k} = \lim\limits_{k\to\infty} \left(1 - \frac{1}{k}\right)^k = \frac{1}{e}$. $\square$

(b) Let J be the random variable representing the number of rejected jobs. It turns out that $J = M$, where M was defined in (a), which we see by realizing that, since the number of jobs assigned is equal to the number of machines, the number of rejected jobs must be equal to the number of idle machines because we have an equal total number of jobs and machines. Thus, by (a), we have $\lim\limits_{k\to\infty} \frac{R(k)}{k} = \frac{1}{e}$. □

(c) We consider the expected number of jobs rejected by machine i, $R_{2,i}(k)$. By linearity of expectation and symmetry, we have $R_2(k) = kR_{2,i}(k)$. For $j \geq 1$, the probability that machine i rejects exactly j jobs is the probability that exactly $j+2$ jobs are assigned to machine i, which is $\binom{k}{j+2}\left(\frac{1}{k}\right)^{j+2}\left(1-\frac{1}{k}\right)^{n-j-2}$. Since j can run from 1 to $k-2$, we have that

$$\begin{aligned}
R_{2,i}(k) &= \sum_{j=1}^{k-2} j\binom{k}{j+2}\left(\frac{1}{k}\right)^{j+2}\left(1-\frac{1}{k}\right)^{n-j-2}\\
&= \sum_{j=3}^{k}(j-2)\binom{k}{j}\left(\frac{1}{k}\right)^{j}\left(1-\frac{1}{k}\right)^{n-j}\\
&= \left(\sum_{j=3}^{k} j\binom{k}{j}\left(\frac{1}{k}\right)^{j}\left(1-\frac{1}{k}\right)^{n-j}\right) - \left(\sum_{j=3}^{k} 2\binom{k}{j}\left(\frac{1}{k}\right)^{j}\left(1-\frac{1}{k}\right)^{n-j}\right)\\
&= \left[\underbrace{\left(\sum_{j=0}^{k} j\binom{k}{j}\left(\frac{1}{k}\right)^{j}\left(1-\frac{1}{k}\right)^{n-j}\right)}_{*} \cancel{-2\binom{k}{2}\left(\frac{1}{k}\right)^{2}\left(1-\frac{1}{k}\right)^{k-2}} - \cancel{k}\left(\frac{1}{\cancel{k}}\right)\left(1-\frac{1}{k}\right)^{k-1}\right]\\
&\quad - \left[\underbrace{\left(\sum_{j=0}^{k} 2\binom{k}{j}\left(\frac{1}{k}\right)^{j}\left(1-\frac{1}{k}\right)^{n-j}\right)}_{**} \cancel{-2\binom{k}{2}\left(\frac{1}{k}\right)^{2}\left(1-\frac{1}{k}\right)^{k-2}} - 2\cancel{k}\left(\frac{1}{\cancel{k}}\right)\left(1-\frac{1}{k}\right)^{k-1} - 2\left(1-\frac{1}{k}\right)
\end{aligned}$$

where we have relabelled indices, split the sum in two, and otherwise rearranged terms. We now realize that * is exactly the expected value of a binomial distribution, which is the number of trials times success probability, or $k \cdot \frac{1}{k} = 1$. Similarly, ** is exactly twice the sum of probabilities over the binomial distribution, or 2. Simplifying from here and combining like terms gives

$$\frac{R_2(k)}{k} = R_{2,i}(k) = \left(1-\frac{1}{k}\right)^{k-1} + 2\left(1-\frac{1}{k}\right)^{k} - 1$$

which converges to $\frac{3}{e} - 1$ as $k \to \infty$. □

Problem 4

Solve Problem 5 from the 2019 final exam.

Solution:

1. The probability that a particular ball b is placed in bin B_j is $\frac{1}{n}$. $\mathcal{E}_{j,T}$ is the event that the k balls in T are placed in B_j, independently of each other, so we have $\mathbf{Pr}[\mathcal{E}_{j,T}] = \frac{1}{n^k}$. □

2. We claim that $\mathcal{E}_j = \bigcup_{T \subseteq \{b_i\}_{i=1}^n, |T|=k} \mathcal{E}_{j,T}$. The $\supseteq$ direction is immediate since any event $\mathcal{E}_{j,T}$ guarantees B_j has at least k balls. For the $\subseteq$ direction, suppose B_j has at least k balls, and let k of these balls be $T' = \{b_{i_\ell}\}_{\ell=1}^k$. This is exactly the event $\mathcal{E}_{j,T'}$, proving equality. $\square$

3. The union bound gives us

$$\mathbf{Pr}[\mathcal{E}_j] \leq \sum_{T \subseteq \{b_i\}_{i=1}^n, |T|=k} \mathbf{Pr}[\mathcal{E}_{j,T}] = \sum_{T \subseteq \{b_i\}_{i=1}^n, |T|=k} \frac{1}{n^k} = \frac{\binom{n}{k}}{n^k}$$

where the last equality comes from the fact that there are $\binom{n}{k}$ sets of size k. $\square$

4. The answer is $k^* = \Theta\left(\frac{\log n}{\log \log n}\right)$. We will leave the formal proof that this k^* is the best possible (up to a constant factor) as an exercise. (Hint: Use Chebyshev's inequality to upper bound the probability that no bin has greater than $c_2 \frac{\log n}{\log \log n}$ balls, for some appropriate constant c_2.)

We specifically let $k^* = \frac{4 \log n}{\log \log n}$. (Any constant slightly greater than 2 would suffice, but we take it to be 4 for simplicity.) To see that this k^* suffices, we upper bound

$$\mathbf{Pr}[\mathcal{E}_j] \leq \frac{\binom{n}{k^*}}{n^{k^*}} \leq \frac{1}{\cancel{n^{k^*}}} \left(\frac{\cancel{n^{k^*}}}{k!} \right) \leq \frac{1}{{k^*}^{k^*/2}}$$

so it remains to show ${k^*}^{k^*/2} \geq 100n$, which we do for n sufficiently large. First, we clearly have that $k^* \gtrsim 100\sqrt{\log n}$, so we apply this to see

$$ {k^*}^{k^*/2} \gtrsim \left(100\sqrt{\log n}\right)^{2 \log n / \log \log n} \geq 100(2^{\cancel{\frac{1}{2}} \cancel{\log \log n}})^{\cancel{2} \log n / \cancel{\log \log n}} = 100(2^{\log n}) = 100n$$

where we pulled the constant 100 out of the exponent and expressed $\log n$ as $2^{\log \log n}$. $\square$

5. Let $\mathcal{E}$ be the event that some bin has at least k^* balls. We have that $\mathcal{E} = \bigcup_{j=1}^n \mathcal{E}_j$, with the $\supseteq$ direction being clear, and the $\subseteq$ direction following by letting $B_{j'}$ be the bin with at least k^* balls, which is exactly the event $\mathcal{E}_{j'}$. By union bound, we have that

$$\mathbf{Pr}[\mathcal{E}] \leq \sum_{j=1}^n \mathbf{Pr}[\mathcal{E}_j] \leq \sum_{j=1}^n \frac{0.01}{n} = 0.01$$

We realize that $\overline{\mathcal{E}}$, the complement of $\mathcal{E}$, is the event that all bins have less than k^* balls. We thus have $\mathbf{Pr}[\overline{\mathcal{E}}] = 1 - \mathbf{Pr}[\mathcal{E}] \geq 1 - 0.01 = 0.99$, as desired. $\square$

Problem 5

In this problem, we explore the notion of *oracle reducibility.* If A is a language, then a *Turing machine with oracle A* is a Turing machine with a "magical" subroutine that decides membership in A. In other words, the subroutine, when given a string w, tells the machine whether or not $w \in A$. Let

$$\text{HALT}_{\text{TM}} = \{\langle M, x \rangle \mid M \text{ is a Turing machine that halts on } x\}.$$

Show that there is a Turing machine with oracle HALT_{TM} that decides the following problem with only *two* questions to the oracle: Given three (machine, input) pairs $\langle M_1, x_1\rangle, \langle M_2, x_2\rangle, \langle M_3, x_3\rangle$, decide for each pair whether the Turing machine halts on the corresponding input.

Note: This is trivial if one allows *three* questions. Just ask the oracle whether $\langle M_i, x_i\rangle \in \text{HALT}_{\text{TM}}$ for $i = 1, 2, 3$.

Solution: Suppose that exactly k of the machines $\{M_1, M_2, M_3\}$ halt on corresponding inputs. The idea is to figure out the value of k. Once we know k, we can decide which ones halt: simply simulate the three machines on corresponding inputs simultaneously (one step of every machine at a time), and halt when exactly k of them halt. Since we know k, we are guaranteed to halt.

This is how we figure out the value of k. First construct a machine M that simulates the three machines on corresponding inputs simultaneously and halts if at least two of them halt. Now decide whether M halts by asking the oracle.
(Case 1): If M halts, we know that $k \geq 2$. By running the machines simultaneously again, we figure out which two of the machines halt. Then by asking the oracle, we can figure out whether the third machine halts.
(Case 2): If M does not halt, we know that $k \leq 1$. Construct a machine M' that simulates the three machines on corresponding inputs simultaneously and halts if at least one of them halts. Decide whether M' halts by asking the oracle. If M' halts, then we know that $k = 1$. If M' does not halt, we know that $k = 0$.

Problem 6

Show that the collection of Turing-recognizable languages is closed under the operation of (a) union (b) concatenation (c) star and (d) intersection. What about complementation operation ?

Solution:

(a) Let L_1 and L_2 be two Turing-recognizable languages, and let M_1 and M_2 be TMs that recognizes L_1 and L_2 respectively. We construct a TM M that recognizes $L_1 \cup L_2$: On input w,

1. Run M_1 and M_2 on w "in parallel". That is, in each step, M runs one step of M_1 and one step of M_2.
2. If either of M_1 and M_2 accepts, output ACCEPT. If both reject, output REJECT.

If M_1 or M_2 accepts w, then M will halt and accept w since M_1 and M_2 are run in parallel and an accepting TM will halt and accept w in a finite number of steps. If both M_1 and M_2 reject w, then M will reject w. If neither M_1 nor M_2 accepts w and one of them loops on w, then M will loop on w. Thus $L(M) = L_1 \cup L_2$, and Turing-recognizable languages are closed under union.

(b) Let L_1 and L_2 be two Turing-recognizable languages, and let M_1 and M_2 be TMs that recognizes L_1 and L_2 respectively. We construct a NTM N that recognizes $L_1 \circ L_2$: On input w,

1. Nondeterministically split w into two parts $w = xy$.
2. Run M_1 on x. If it rejects, halt output REJECT. If it accepts, go to Step 3.
3. Run M_2 on w. If it accepts, output ACCEPT. If it rejects, output REJECT.

If $w \in L_1 \circ L_2$, then there is a way to split w into two parts $w = xy$ such that $x \in L_1$ and $y \in L_2$, thus, M_1 halts and accepts x, and M_2 halts and accepts y. Hence at least one branch of N will accept w. On the other hand, if $w \notin L_1 \circ L_2$, then for every possible splitting $w = xy$, $x \notin L_1$ or $y \notin L_2$, so M_1 does not accept x or M_2 does not accept y. Thus, all branch of N will reject w. Therefore, $L(N) = L_1 \circ L_2$, and Turing-recognizable languages are closed under concatenation.

(c) Let L be a Turing-recognizable languages, and let M be a TM that recognizes L. We construct a TM M' that recognizes L^*: On input w,

1. Nondeterministically choose an integer k and split w into k parts $w = w_1 w_2 \cdots w_k$.
2. Run M on each w_i. If M accepts all of $w_1, \ldots, w_k$, output ACCEPT. If M rejects one of $w_1, \ldots, w_k$, output REJECT.

If $w \in L^*$, then there is a splitting of $w = w_1 w_2 \ldots w_k$ such that $w_i \in L$ for each i, and thus at least one branch of M' will accept w. On the other hand, if $w \notin L^*$, then for every possible splitting $w = w_1 w_2 \ldots w_k$, $w_i \notin L$ for at least one i, thus all branches of M' will reject w. Therefore, $L(M') = L^*$, and Turing-recognizable languages are closed under the star operation.

(d) Let L_1 and L_2 be two Turing-recognizable languages, and let M_1 and M_2 be TMs that recognizes L_1 and L_2 respectively. We construct a TM M that recognizes $L_1 \cap L_2$: On input w,

1. Run M_1 on w. If it rejects, halt and output REJECT. If it accepts, go to Step 2.
2. Run M_2 on w. If it accepts, output ACCEPT. If it rejects, output REJECT.

Clearly $L(M) = L_1 \cap L_2$, and thus Turing-recognizable languages are closed under intersection.

(e) Not closed under complementation. Suppose it were true that whenever L is Turing-recognizable, so is $\overline{L}$. So whenever L is Turing-recognizable, then both $L, \overline{L}$ would be Turing-recognizable, and hence L would be decidable (as proved in class). This is a contradiction since there exists a language that is Turing-recognizable but not decidable, e.g. the language A_{TM}.