CS 6150: HW 6 – Review

Submission date: Friday, Dec 7, 2018 (11:59 PM)

This assignment has 6 questions, for a total of 60 points. Unless otherwise specified, complete and reasoned arguments will be expected for all answers.

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 questions with a lie</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>CD problem with small song lengths</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>CD problem with few distinct lengths</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Non-negativity in Markov</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Randomized rounding and constraints</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Distributed independent set</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>60</td>
<td></td>
</tr>
</tbody>
</table>
Consider the following generalization of the “20 questions” game: player A chooses a number \( z \) between 1 and \( n \), and player B asks comparison queries (i.e., questions of the form, “is the number < \( x \)?”), with the goal of finding \( z \).

If A always gives the right answer, an easy strategy for B is to do binary search. Now, consider the scenario in which A is allowed to lie precisely once throughout the game (and of course, B doesn’t know when A is lying). One strategy here is to ask each question thrice, and take the majority answer. This will take \( 3 \lceil \log n \rceil \) queries.

(a) [5] Give a strategy that only makes \( 2 \lceil \log n \rceil + 1 \) queries.

(b) [10] (Bonus) Use the following hint to develop a strategy that makes \( \leq (3/2) \lceil \log n \rceil + 4 \) queries.

[Hint: Suppose we view binary search as trying to find \( z \) bit-by-bit. Say we query as usual, and obtain 1 as the answer for the first bit. Now, consider the answer to the query for the second bit. If the answer is also 1, then we can be certain that the first bit is correct. We can thus mark it as correct and move on. Now, if the answer to the query for the second bit is 0, we query for the third bit. If that happens to be 1, again, we can be sure that the first bit was right. The problematic case is when we get 100. In this case, re-query the first bit. If the answer is 0 this time, we know that at least one of the answers was a lie, and we can simply query the bit a third time, and know that the answers in the future will all be correct. If the answer is 1 again, then we can be sure that the first two bits are both correct ...
]

(While the question seems like a puzzle, ideas like this are used in fault tolerant computing. It quickly leads to the notion of error correcting codes.)

Consider the question of writing songs to CDs we saw in HW 2. We considered the greedy algorithm: consider the songs one at a time, move through the list of CDs, and write the song to the first CD that has space for the song — assuming that CDs are all re-writeable — or create a new CD with that song.

We proved there that this algorithm is not always optimal, but the number of CDs produces is within a factor 2 of the optimal number. This problem and the next will see how to improve this approximation ratio.

Suppose that the songs we want to write are all at most 20 minutes in length (recall that the capacity of the CD is 60 minutes). In this case, prove that the greedy algorithm above uses at most \( (3/2) \text{Opt} \) CD’s, where \( \text{Opt} \) is the minimum possible number of CDs.

(It’s OK if the bound you show is \( (3/2) \text{Opt} + 1 \) or \( (3/2) \text{Opt} + 2 \).)

Suppose that the songs we wish to write have only a small number of distinct lengths/durations, say \( s_1, s_2, \ldots, s_r \). (Let \( n \) be the total number of songs.) Devise an algorithm that runs in time \( O(n^r) \), and computes the optimal number of CDs.

[Hint: first find all the possible “configurations” that can fit on a single CD. Then use dynamic programming.]

Markov’s inequality states that for a non-negative random variable \( X \), we have \( \Pr[X > t \cdot \mathbb{E}[X]] \leq 1/t \), for any \( t \geq 1 \).

The point of this exercise is to show that the non-negativity is important. Give an example of a random variable (that takes negative values), for which (a) \( \mathbb{E}[X] = 1 \), and (b) \( \Pr[X > 5] \geq 0.9 \).

Consider the vertex cover problem we saw in class. We are given an undirected graph \( G = (V, E) \), and the goal is to find an \( S \subseteq V \) of the smallest possible size, such that for every edge \( \{i, j\} \), at least one of \( i, j \) is in \( S \).
We saw how to capture this using a linear programming relaxation. We have variables $0 \leq x_i \leq 1$ for all $i \in V$, and for every edge $\{i, j\}$, we have a constraint $x_i + x_j \geq 1$. The objective is to minimize $\sum x_i$.

Suppose we solve the LP above and find the optimal solution. We saw that rounding via “thresholding at 1/2” gives a feasible solution, and chooses a set of size $S$ of size at most $2 \cdot \text{OPT}$.

(a) [2] Consider a different rounding strategy, randomized rounding. i.e., suppose we add vertex $i$ to $S$ with probability $x_i$, independently. What is the expected value of $|S|$ (in terms of the $x_i$)?

(b) [3] The problem with randomized rounding is that constraints need not all be satisfied. However, many of the constraints are satisfied. Formally, prove that the probability that an edge constraint is satisfied is $\geq 3/4$.

(c) [5] Now, suppose we play around with the probability of adding $i$ to $S$. Specifically, suppose we add with probability $\min\{1, (3/2)x_i\}$ (we are basically capping the probability at 1). In this case, prove that the probability of satisfying an edge constraint is $\geq 15/16$.

(d) [5] Suppose we solve the LP, and find that the $x_i$ values are all either $\geq 0.9$ or $\leq 0.1$. Prove that even the “thresholding at 1/2” procedure gives an approximation ratio $< 1.2$.

Question 6: Distributed independent set ................................................................. [10]

A basic problem in distributed algorithms (used in P2P networks, distributed coloring, etc.) is the following: suppose we have a graph with $n$ vertices in which every vertex has degree $\leq d$, for some parameter $d$. The goal is to find a large independent set (a set of vertices with no edges between them) in a distributed manner.

Consider the following algorithm. (1) Every vertex becomes active with probability $\frac{1}{2d}$. (2) Every active vertex queries its neighbors, and if any vertex in the neighborhood is also active, it becomes inactive. ((2) is done in parallel; thus if $i$ and $j$ are neighbors and they were both activated in step (1), they both become inactive.)

(a) [2] Let $X$ be the random variable that is the number of vertices activated in step (1). Find $\mathbb{E}[X]$.

(b) [3] Let $Y$ be the random variable that is the number of edges $\{i, j\}$ both of whose end points are activated in step (1). Find $\mathbb{E}[Y]$.

(c) [5] Prove that the number of vertices remaining active after step (2) is $\geq X - 2Y$, and thus show that the expectation of this quantity is $\geq n/4d$. 

Page 3