CMPS 201
Midterm 2 Review
Solutions to Selected Problems

Study problems 2 and 3 on the midterm 1 review sheet, as well as the posted solutions to homework assignments 4, 5 and 6.

1. Recall the $n^{th}$ harmonic number was defined to be $H_n = \sum_{k=1}^{n} \left( \frac{1}{k} \right) = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-1} + \frac{1}{n}$. Use induction to prove that

$$\sum_{k=1}^{n} kH_k = \frac{1}{2} n(n + 1)H_n - \frac{1}{4} n(n - 1)$$

for all $n \geq 1$. (Hint: Use the fact that $H_n = H_{n-1} + \frac{1}{n}$, and note this was a previous hw problem.)

**Solution:** See the solution to problem 1 on hw assignment 5.

2. Recall that the average case runtime $t(n)$ of the $i^{th}$ selection problem satisfies the recurrence

$$t(n) = (n - 1) + \left( \frac{n - 1}{n^2} \right) \sum_{q=1}^{n-1} t(q)$$

Use this recurrence to prove that $t(n) = \Theta(n)$. (Hint: See problem 3 on the midterm 1 review sheet.)

**Solution:**
First observe from the above recurrence that $t(n) \geq n = 1 = \Omega(n)$, so that $t(n) = \Omega(n)$. In the solution to problem 3 from the Midterm 1 review sheet it was shown that $t(n) \leq 2n = O(n)$, whence $t(n) = O(n)$. It follows that $t(n) = \Theta(n)$. ■

3. Suppose we are given 4 gold bars (labeled 1, 2, 3, 4), one of which may be counterfeit: gold-plated tin (lighter than gold) or gold-plated lead (heavier than gold). Again the problem is to determine which bar, if any, is counterfeit and what it is made of. The only tool at your disposal is a balance scale, each use of which produces one of three outcomes: tilt left, balance, or tilt right.

a. Use a decision tree argument to prove that at least 2 weighings must be performed (in worst case) by any algorithm that solves this problem. Carefully enumerate the set of possible verdicts.

**Solution:**
Each probe of the input data has one of $k = 3$ possible outcomes (tilt left, balance, tilt right). There are 9 possible verdicts, namely \{1L, 2L, 3L, 4L, 1H, 2H, 3H, 4H, AllGold\}, where for instance 2L means bar 2 is light and therefore tin, 3H means bar 3 is heavy and therefore lead, and AllGold means that there is no counterfeit bar. Any decision tree representing an algorithm for this problem must therefore be a ternary tree with at least 9 leaves. By a theorem proved in class, the height of such a tree must satisfy $h \geq \lceil \log_3(9) \rceil = 2$. Therefore no algorithm for this problem can perform fewer than 2 weighings in worst case. ■
b. Determine an algorithm that solves this problem using 3 weighings (in worst case). Express your algorithm as a decision tree.

Solution:
There are many correct algorithms for this problem, two of which are presented below as decision trees. Each internal node represents a particular set of bars on the left and right side of the balance. A node labeled for instance $a:b:c:d$ would mean to place bars $a$ and $b$ on the left side of the balance, and place bars $c$ and $d$ on the right. The left child is taken if the balance tilts left, the right child is taken if the balance tilts right and the middle child is taken if the balance remains still, indicating equal weight on both sides.

Observe that both trees have height 3, while the first tree has average height $\frac{1 \cdot 1 + 8 \cdot 3}{9} = \frac{25}{9} = 2.78 \ldots$ and the second tree has average height $\frac{5 \cdot 2 + 4 \cdot 3}{9} = \frac{22}{9} = 2.44 \ldots$. Therefore both algorithms have the same worst case run time, but the second has an advantage in average case run time. This average case analysis assumes of course that all verdicts are equally likely, which may not be the case. If one believes for instance that a counterfeit bar is highly unlikely, then the first algorithm would be preferable over the second.

c. Find an adversary argument that proves 3 weighings are necessary (in worst case), and therefore the algorithm you found in (b) is best possible. (Hint: study the adversary argument for the min-max problem discussed in class to gain some insight into this problem. Further hint: put some marks on the 4 bars and design an adversary strategy that, on each weighing, removes the fewest possible marks, then show that if the balance scale is only used 2 times, not enough marks will be removed.)

Note: This problem is to long for the midterm, but it will probably a future homework problem.
4. Let \( B = b_1 b_2 \ldots b_n \) be a bit string of length \( n \). Consider the following problem: determine whether or not \( B \) contains 3 consecutive 1's, i.e. whether \( B \) contains the substring "111". Consider algorithms that solve this problem whose only allowable operation is to peek at a bit.

a. Suppose \( n = 4 \). Obviously 4 peeks are sufficient. Give an adversary argument showing that in general, 4 peeks are also necessary. (Hint: this problem is similar to problem 4b on hw6, and has a similar solution.)

Note: This problem is short enough to be on the exam. It was discussed in general terms on pages 4 and 5 of the lecture notes from 11-15-18, in which a valid adversary strategy was given. It remains to prove that the adversary strategy will defeat any algorithm that performs at most 3 peeks. You can construct an abstract proof that the strategy works. Another valid solution is to give a simple exhaustive proof, i.e. enumerate all 2^4 possible orders of 3 peeks at the 4 bits \( \{b_1, b_2, b_3, b_4\} \) and show that in each case, the unpeeked bit is needed to establish whether or not \( b \) contains "111".

b. Suppose \( n \geq 5 \). Give an adversary argument showing that \( 4 \cdot \lfloor n/5 \rfloor \) peeks are necessary. (Hint: divide \( B \) into \( \lfloor n/5 \rfloor \) 4-bit blocks separated by 1-bit gaps between them. Thus bits 1-4 form the first block, and bit 5 is the first gap. Bits 6-9 form the next block and bit 10 is the next gap, etc.. Any leftover bits form a separate block. Now run the adversary from part (a) on each of the 4-bit blocks.)

Solution:
Following the hint, compute the integer quotient and remainder of \( n \) by 5. Set the remainder bits to 0, along with the "gap" bits, i.e. those bits whose index is a multiple of 5. What remains are \( \lceil n/5 \rceil \) blocks, each of length 4. We illustrate below in the case \( n = 17 \), where there are \( \lceil 17/5 \rceil = 3 \) blocks, 3 gaps and 2 remainder bits.

\[
\begin{array}{ccccccccccccccc}
b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 & b_9 & b_{10} & b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & b_{16} & b_{17} \\
\hline
- & - & - & - & 0 & - & - & - & 0 & - & - & - & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

In general, we create \( \lceil n/5 \rceil \) independent copies of the 4-bit adversary from part (a), and use those copies to form the following adversary, to be run against any algorithm for this problem. Daemon's strategy: any peek of a gap bit or of a remainder bit is answered with 0. One copy of the 4-bit adversary is assigned to each block, and the first three peeks in any block are answered by that adversary. If the algorithm peeks at all 4 bits of some block, then the final peek is answered with 0. Thus any such block (which we shall call complete) is confirmed by the adversary to not contain "111".

Now suppose the algorithm halts and returns an answer ("yes" or "no") after doing fewer than \( 4 \cdot \lceil n/5 \rceil \) bit peeks. Then at least one of the \( \lceil n/5 \rceil \) blocks has not had all of its 4 bits probed. We shall call such a block incomplete. Since all bits outside of a block are answered 0, the only place where "111" could possibly reside would be in an incomplete block. But the correctness of the 4-bit adversary from part (a) implies that the daemon's sequence of answers is consistent with an incomplete block containing, and with not containing the substring "111". Therefore no algorithm doing fewer than \( 4 \cdot \lceil n/5 \rceil \) bit peeks can be correct. ■