# Sample Solution to Problem Set 2

## 1. (3 + 3 + 4 = 10 points) Digital signatures

Problem 1.45 parts (b), (c), and (d) of text.

**(b) Answer:**

The correctness of digital signatures follows in the same way as the correctness of RSA. We repeat here for completeness.

In RSA, the modulus $n = pq$ is a product of two primes $p$ and $q$. The public key $e$ and private key $d$ satisfy

$$ed \equiv 1 \; mod \; (p-1)(q-1)$$

Therefore, there exists an integer $h$, such that

$$ed - 1 \equiv h(p-1)(q-1)$$

We can then continue to calculate

$$(m^e)^d \equiv m^{ed} \equiv m^{(ed-1)}m \equiv m^{h(p-1)(q-1)}m \equiv 1^{h(q-1)}m \equiv m \pmod{p},$$

where the last two steps follows from Fermat's little theorem, which says that if $p$ is prime and $p$ does not divide $a$ then

$$a^{p-1} \equiv 1 \; mod \; p$$

Similarly, we obtain for $q$

$$(m^e)^d \equiv m^{ed} \equiv m^{(ed-1)}m \equiv m^{h(p-1)(q-1)}m \equiv 1^{h(p-1)}m \equiv m \pmod{q}.$$

If $p$ and $q$ are coprime, $a \equiv b \pmod{p}$ and $a \equiv b \pmod{q}$ then the Chinese remainder theorem implies $a \equiv b \pmod{pq}$. Hence,

$$(m^e)^d \equiv m \pmod{pq}.$$

**(c) Answer:**

1. Choose two distinct prime numbers, such as $p = 61$ and $q = 53$.
2. Compute $n = pq$. giving $n = 61 \cdot 53 = 3233$.
3. Compute $\phi(n) = (p-1)(q-1)$ giving $\phi(3233) = (61-1)(53-1) = 3120$.

4. Choose $e$ coprime to 3120. Choosing a prime number for $e$ leaves us only to check that $e$ is not a divisor of 3120. Let $e = 17$.
5. Compute $d$, the modular multiplicative inverse of $e \pmod{\phi(n)}$ yielding $d = 2753$.
6. The public key is $(n = 3233, e = 17)$. The private key is $(n = 3233, d = 2753)$.

For instance, in order to encrypt $m = 65$, we calculate $c = 65^{17}( \pmod{3233}) = 2790$.

To decrypt $c = 2790$, we calculate m $= 2790^{2753}( \pmod{3233}) = 65$.

**(d) Answer:**

We can represent $n$ as follows:

$$\begin{aligned} n &= 391 \\ &= 17 \times 13 \end{aligned}$$

so Alice should raise her message to the exponent

$$\begin{aligned} d &= 17^{-1} \; mod \; (16 \times 22) \\ &= 145 \end{aligned}$$

**2. (10 points) Optimal location of store**

You are a new player in the mobile phone business and are planning to open a physical store in Manhattan, to rival Apple's well-established store there. You want to determine an optimal location to place the store. Manhattan is organized as a grid, and a simple model is to view it as an $m \times n$ grid consisting of points $(i, j)$, $1 \leq i \leq m$ and $1 \leq j \leq n$. Each of the $mn$ points constitutes both a potential location for the store as well as a neighborhood of homes. The distance between two points $(a, b)$ and $(c, d)$ in Manhattan – $D((a, b), (c, d))$ – is simply $|a - c| + |b - d|$. For example, the distance between $(5, 3)$ and $(2, 9) = |5 - 2| + |3 - 9| = 9$.

After a detailed survey, you have found out which of the $mn$ neighborhoods are really potential customers – so you have a set $C$ of potential customers:

$$C = \{(i, j) : 1 \leq i \leq m, 1 \leq j \leq n, \text{ neighborhood at } (i, j) \text{ is a potential customer}\}.$$

For example, the set $C$ could be $\{(1, 4), (1, 5), (1, 8), (2, 4), (3, 5), (6, 7), (2, 9)\}$. Since you have only one store to build, you would like to locate it at a point that minimizes the total distance from all the potential customers. That is, you want to find $(x, y)$ in the grid that minimizes

$$\sum_{(i,j) \in C} D((i, j), (x, y)).$$

Give an efficient algorithm that takes as input the $m \times n$ grid and the set $C$, and returns the optimal store location $(x, y)$.

**Answer:** We show that an optimal location for the store is $(x^*, y^*)$, where $x^*$ is a median of $x_1, \ldots, x_n$ and $y^*$ is a median of $y_1, \ldots, y_n$. We renumber the $x_i$'s and $y_i$'s in sorted order. Let $x_m = x^*$ be the median of the $x_i$'s and $y_p = y^*$ be the median of the $y_i$'s. We now show that

$$\sum_{i=1}^{n} |x_i - x_m| \leq \sum_{i=1}^{n} |x_i - x|,$$

for any $x$.

We first consider the case when $x < x_m$. In this case, we have

$$\begin{aligned}\sum_{i=1}^{n}(|x_i - x_m| - |x_i - x|) &\leq \sum_{x_i<x_m} |x_m - x| - \sum_{x_i \geq x_m} |x_m - x| \\ &= |x_m - x| \left( \sum_{x_i<x_m} - \sum_{x_i \geq x_m} \right) \\ &= |x_m - x| \left( (2 \sum_{x_i<x_m} 1) - 1 \right) \\ &\leq 0.\end{aligned}$$

Similarly, if $x > x_m$, we have

$$\begin{aligned}\sum_{i=1}^{n}(|x_i - x_m| - |x_i - x|) &\leq - \sum_{x_i \leq x_m} |x_m - x| + \sum_{x_i > x_m} |x_m - x| \\ &= |x_m - x| \left( (2 \sum_{x_i>x_m} 1) - 1 \right) \\ &\leq 0.\end{aligned}$$

By the same argument, we also have

$$\sum_{i=1}^{n} |y_i - y_p| \leq \sum_{i=1}^{n} |y_i - y|,$$

for any $y$. This shows that $(x_m, y_p)$ is an optimal location for the store.

Both the coordinates $x_m$ and $y_p$ can be found using the linear time median algorithm we studied in class. Therefore, the optimal store location can be found in linear time.

**3. (10 points) Optimizing a stock sale, in hindsight**

You are working for an investment firm that prides on its analysis of the past to maximize future revenues. One of the problems you repeatedly face is the following. You know the price of a particular stock for each of the last $n$ days; let us number these days 1, 2, ..., $n$. Suppose during this time period, you wanted to buy 100 shares on some day and sell all the shares on some later day (within the time period). Then, which day would you have bought and which day would you have sold? For instance, suppose that $n$ was 10, and the stock price data you had was the following.

$$10, 11, 13, 16, 12, 8, 3, 7, 10, 9.$$

Your optimal strategy would have been to buy on day 7 and sell on day 9.

Give an efficient (polynomial-time) algorithm that solves the above problem. Analyze the worst-case running time of the algorithm.

**Answer:** Suppose we denote the prices of $n$ days in an array $A$. We need to find the indices $i$ and $j$ such that $j > i$ and $A[j] - A[i]$ is maximum.

Here is Algorithm1. We can calculate $A[j] - A[i]$ for all pairs $(i, j)$ with $j > i$, and pick the pair that yields the maximum difference. The number of pairs is $n(n-1)/2 = \Theta(n^2)$, so this will take time $\Theta(n^2)$.

Can we do better? Consider the following divide-and-conquer algorithm, which we call Algorithm2.

- If $A$ has size 1, then return "no solution".
- If $A$ has size 2, then return the pair $(1, 2)$.
- Divide the array into two equal halves $A_1$ (the first half) and $A_2$ (the second half).
- $(i_1, j_1) =$ Algorithm2$(A_1)$; $(i_2, j_2) =$ Algorithm2$(A_2)$. Find the index $m$ in $A_1$ holding the minimum value in $A_1$, and and the index $M$ in $A_2$ holding the maximum value.
- Find $(i, j)$ in $\{(i_1, j_1), (|A_1| + i_2, |A_1| + j_2), (m, |A_1| + M)\}$ that maximizes $A[j] - A[i]$.

Algorithm2 is a divide and conquer algorithm. Its correctness follows from the fact that the optimal pair is either entirely in the left half, or entirely in the second half, or split between the first and second halves. Each of the three cases is handled in the algorithm. For the running time, note that we have a Mergesort-like recurrence.

$$T(n) = 2T(n/2) + Theta(n) = \Theta(n \log n).$$

Can we do even better? In linear time? Spending as much time as sorting for finding an optimal pair seems overkill. Consider an optimal pair $(i, j)$. It is clear that $i$ should be an index that holds the minimum value in $A[1 \ldots j-1]$. So, if we knew, for each $j$, the $i$ that holds the minimum value in $A[1 \ldots j-1]$, then we can simply compute $A[j] - A[i]$ and take the maximum, over all $j$.

Here is a linear-time algorithm Algorithm3.

$m \leftarrow 0; A[0] \leftarrow \infty$.
**for** $i$ from 1 to $n-1$ **do**
  $B[i] \leftarrow m$
  **if** $A[i] < A[m]$ **then**
    $m \leftarrow i$
  **end if**
**end for**
Return $(B[j], j)$ where $j$ minimizes $A[j] - A[B[j]]$.

The total running time is $O(n)$ since the algorithm consists of only two linear scans of the array $A$.

### 4. (3 + 4 + 3 = 10 points) Finding good gadgets using pairwise testing

You have $n$ supposedly identical gadgets that in principle can test each other. For instance, you can connect two gadgets A and B, each of which then reports whether the other gadget is good or bad, but the answer of a bad gadget cannot be trusted. Thus, there are four possible outcomes of such a pairwise test.

| Gadget A says | Gadget B says | Conclusion |
|---|---|---|
| B is good | A is good | both are good, or both are bad |
| B is good | A is bad | at least one is bad |
| B is bad | A is good | at least one is bad |
| B is bad | A is bad | at least one is bad |

You are told that more than $n/2$ of the gadgets are good, and you are asked to determine all of the good gadgets.

**(a)** Consider the problem of finding a single good gadget. Show that $\lfloor n/2 \rfloor$ pairwise tests are sufficient to reduce the problem to one of nearly half the size.

**(b)** Using part (a), give a divide-and-conquer algorithm to find a single good chip using $\Theta(n)$ pairwise tests. Give and solve the recurrence that describes the number of tests.

**(c)** Using part (c), show that all the good chips can be identified using $\Theta(n)$ pairwise tests.

(**Note:** This problem was taken from Cormen-Leiserson-Rivest-Stein.)

**Answer of part (a):** We will prove the desired claim under the assumption that the number of good and bad gadgets are the same. If the number of bad gadgets is greater than the number of good gadgets, then this makes the task of finding a good gadget only harder.

Now let $G$ be the set of good gadgets and $B$ be the set of bad gadgets. Suppose each bad gadget reports a good gadget as bad and a bad gadget as good. Then we have the following property: (i) all the pairwise tests within $G$ report good-good, (ii) all pairwise tests within $B$ report good-good, and (iii) all pairwise tests involving one gadget in $G$ and the other in $B$ report bad-bad. This implies that after doing all the pairwise tests, the professor will not be able to distinguish the set $G$ from $B$ since their behavior is exactly the same, and so is their number. Thus, even though the professor may be able to identify that all of the gadgets in $G$ are either all good or all bad and the gadgets in $B$ are all good or all bad, he or she will not be able to identify a single good gadget.

**Answer of part (b):** Let $S$ be the given set of $n$ gadgets. Let $g$ be the number of good gadgets and $b$ be the number of bad gadgets. We group the given set $S$ gadgets into pairs. If $n$ is odd, then there may be an unpaired gadget. Thus, we have $\lfloor n/2 \rfloor$ pairs of two gadgets each. We load each pair into the jig and perform the pairwise test.

We will construct a set $S'$ of at most $\lceil n/2 \rceil$ gadgets such that at least half of them are good. We go through the results of the pairwise tests. If the test result is good-bad, then we throw out the gadget that tested bad and add to $S'$ the gadget that tested good. If the test result is good-good, then we add to $S'$ any one of the gadgets and throw the other. If the test result is bad-bad, then we throw out both the gadgets. Since, at most gadget from each pair is included in $S'$, it is clear that the number of gadgets in $S'$ is at most $\lfloor n/2 \rfloor$. Finally, if the number of gadgets thus included is an even number, then we add the unpaired gadget to $S'$. We thus have $|S'| \leq \lceil n/2 \rceil$.

Let $gg$, $bb$, and $gb$ be the number of pairwise tests involving two good gadgets, two bad gadgets, and one good gadget and a bad gadget, respectively. (Note that the terms "good" and "bad" in the preceding sentence refer to the actual quality of the gadget, not the outcome of the test.) Let $x$ and $y$ denote the number of good gadgets and bad gadgets, respectively, that were picked during the pairwise tests. When two good gadgets pair together, the output of the test is $GG$; therefore

at least one of the gadgets is included in $S'$. If a good gadget and a bad gadget pair together, then the result is either a $GB$ or a $BB$. In the former case, only the good gadget is picked. In the latter case, both gadgets are discarded. Finally, when two bad gadgets are paired together, at most one of the gadgets is selected. Thus, we have $x \geq gg$ and $y \leq bb$.

Since the number of good gadgets is at least $n/2$, we have $gg > bb$ if $n$ is even or if $n$ is odd and the unpaired gadget is bad. If $n$ is odd and the unpaired gadget is good, then $gg \geq bb$. We thus have $x \geq y$. We still have to account for the unpaired gadget, which arises only if $n$ is odd. Recall that we pick the unpaired gadget only if the number of gadgets selected after the pairwise tests is even. If $x + y$ is odd, then since $x \geq y$, it follows that $x > y$ and we are done in this case. If $x + y$ is even, then we have two cases: either $x = y$ or $x \geq y + 2$. The former case can only arise if $x = gg = bb = y$. And $gg = bb$ can only happen if the unpaired gadget is good, which makes the number of good gadgets in $S'$ one more than the number of bad gadgets. Finally, if $x \geq y + 2$, then adding another gadget still leaves the good gadgets with a clear majority. This completes the proof.

**Answer of part (c):** We first identify a good gadget by using the procedure of part (b) recursively. The recursion for this procedure is given by

$$T(n) = T(\lceil n/2 \rceil) + \lfloor n/2 \rfloor.$$

Ignoring the ceiling, we obtain the recurrence $T(n) \leq T(n/2) + n/2$, which can be shown to be $O(n)$ using the recursion tree method, as we have done several times in class.

Once we have identified a good gadget, we test the gadget with each of the other $n - 1$ gadgets and identify their quality. This procedure takes $n - 1 = \Theta(n)$ tests. Thus the total number of pairwise tests is $\Theta(n)$.