Suggested Solutions to HW #1

2.18 Given a set of n points in the plane such that any three of them are contained in a unit-size cycle, prove that all n points are contained in a unit-size cycle.

Solution. (Chih-Pin Tai) The proof is by induction on the number n of points.

Base case: When $n = 3$, it is obvious that "if a set of n points in the plane such that any three of them are contained in a unit-size circle, all n points are contained in a unit-size circle."

Inductive step: When $n = k + 1$, we proceed as follows. Let us consider the smallest circle that contains the $k + 1$ points. Let O denote the smallest circle. There are two cases to consider:

The first case is that there are three points on the circle and not all of them are on the same semicircle. Let X, Y, Z denote the three points respectively. We remove an arbitrary point that is not X, Y, or Z. Because X, Y, Z are still in the set of the remaining k points, the circle O is still the smallest circle that contains the remaining k points. Besides, any three of the remaining k points are contained in a unit-size circle because any three of the $k + 1$ points are contained in a unit-size circle. According to the induction hypothesis which states that "when $n = k$, if a set of n points in the plane such that any three of them are contained in a unit-size circle, all n points are contained in a unit-size circle," the remaining k points are contained in a unit-size circle, implying that circle O which contains the k+1 points is no larger than a unit-size circle. Therefore, the $k + 1$ points are contained in a unit-size circle.

The second case is that there are two points on the circle and the two points are the end points of a diameter. This case can be proved in a similar manner. □

2.21 Prove that the regions formed by a planar graph all of whose vertices have even degrees can be colored with two colors such that no two neighboring regions have the same color.

Solution. The proof is by *strong* induction on the number m of edges in the graph.

Base case: When $m = 0$ (i.e., the graph has one or more isolated vertices), there is only one region which can be colored by any of the two colors.

Inductive step: Consider a planar graph G with $m = k$ ($k \geq 1$) edges. The induction hypothesis says that "a planar graph with $< k$ edges can be colored with two colored such that no two neighboring regions have the same color".

G must contain a simple cycle (a cycle that passes through a node at most once). Remove the cycle from G to obtain a graph G' that has $< k$ edges and all of whose edges have even degrees. By the induction hypothesis, G' can be properly colored with two colors. The removed cycle, when put back, divides G' into two areas: one inside the cycle and the other outside the cycle. The cycle also divides some of the regions of G' into smaller regions (it is possible that a region be divided into more than two smaller regions). Flip the colors of the regions inside the cycle and we get a proper coloring for graph G (why this is so follows from an argument similar to that for the example of regions divided by lines in general position that we discussed in class). □

2.24 We can define anti-Gray codes in the following way. Instead of minimizing the difference between two consecutive strings, we can try to maximize it. Is it possible to design an encoding for any even value of objects such that each two consecutive strings differ by k bits (where is k is the number of bits in each string)? How about $k - 1$ bits (or $k - 2, k - 3, etc.$)? If it is possible, find an efficient construction.

Solution. (Chih-Pin Tai) It is impossible to design an encoding for an even number of objects such that each pair of two consecutive strings differ by k bits (where k is the number of bits in each string). The reason is that if we want to make each pair of two consecutive strings differ by k bits, we will find that we just can construct two-object anti-Gray codes because the third string will be the same as the first string.

Therefore, we try to make each pair of two consecutive strings differ by $k - 1$ bits. We can construct anti-Gray codes in the following way: (1) Construct Gray codes for the n objects. (2) Reverse each bit of the code for every $2i$-th object. (3) For $(2i + 1)$-th objects, add an additional bit 0 to the left of the most significant bit and, for $2i$-th objects, add an additional bit 1.

The construction is correct if each pair of two consecutive strings differ by $k - 1$ bits which is the difference we can maximize and there is no collision for all objects. To see that each pair of two consecutive strings differ by $k - 1$ bits, we consider the property

of Gray codes. Because each pair of consecutive strings differ by 1 bit in Gray codes and we reverse each bit for $2i$-th objects, each pair of consecutive strings must differ by $k-1$ bits. To see that there is no collision for all objects, we consider the p-th object in our construction. If p is odd, it is impossible that the p-th object will conflict with other $(2i+1)$-th objects because it doesn't conflict with other $(2i+1)$-th objects in Gray codes. Besides, it is impossible that the p-th object will conflict with $2i$-th objects because relative to $2i$-th objects, we add a different bit 0 to the left of the most significant bit of the p-th object. Similarly, we can prove the case when p is even. Therefore, there is no collision for all objects in our construction.

□

2.39 Design an algorithm to convert an binary number to a decimal number. The algorithm should be the opposite of algorithm *Convert_to_Binary* (see Fig. 2.6). The input is an array of bits b of length k, and the output is a number n. Prove the correctness of your algorithm by using a loop invariant.

Solution. (Modified by Jinn-Shu Chang)

Algorithm Convert_to_Decimal (b, k);
Input: b (an array of bits contains a binary number), k (the array size of b)
Output: dec (an positive integer corresponding to the decimal number of array b)
begin
    $dec := 0$;
    $i := k$;
    while $i > 0$ **do**
        $dec := dec \times 2 + b[i]$;
        $i := i - 1$;
end

Let $Inv(n, dec, k, b)$ denote the following assertion:

$n = dec \times 2^i + m$ and $i \geq 0$,
where n is the binary number represented by b, i.e.,

$$n = \begin{cases} 0 & \text{if } k = 0 \\ b[k] \times 2^{k-1} + b[k-1] \times 2^{k-2} + \cdots + b[1] \times 2^0 & \text{if } k \geq 1 \end{cases}$$

and m is the binary number represented by b from position i to 0, i.e.,

$$m = \begin{cases} 0 & \text{if } i = 0 \\ b[i] \times 2^{i-1} + b[i-1] \times 2^{i-2} + \cdots + b[1] \times 2^0 & \text{if } i \geq 1 \end{cases}$$

Claim: $Inv(n, dec, i, b)$ is a loop invariant of the while loop, assuming that n is non-negative. (The invariant is sufficient to deduce that, when the program terminates, dec stores the decimal representation of b.)

Proof: The proof is by induction on the number of times the loop is executed. More specifically, we show that (1) the assertion is true when the flow of control reaches the loop for the first time and (2) given that the assertion is true and the loop condition holds, the assertion will remain true after the next iteration (i.e., after the loop body is executed once more).

(1) When the flow of control reaches the loop for the first time, $dec = 0$ and $i = k$. With m denoting the binary number represented by b from position k to 0, $dec \times 2^i + m = 0 \times 2^k + m = 0 + n = n$ (when $i = k$, m equals n, trivially) and $i \geq 0$. Therefore, the assertion $Inv(n, dec, i, b)$ holds.

(2) Assume that $Inv(n, dec, i, b)$ is true at the start of the next iteration and the loop condition ($i > 0$) holds. Let n', dec', i', and b' denote respectively the values of n, t, i, and b after the next iteration. We need to show that $Inv(n', t', i', b')$ also holds.

¿From the loop body, we deduce the following relationship:

$$\begin{aligned} &i' = i - 1 \\ &b'[j] = b[j] \text{ for all } j \leq k \\ &dec' = dec \times 2 + b[i] \\ &m' = m - b[i] \times 2^{i-1} \\ &n' = n \text{ (the value of } n \text{ never changes)} \end{aligned}$$

Thus, we have:

$$\begin{aligned} dec' \times 2^{i'} + m' &= (dec \times 2 + b[i]) \times 2^{i-1} + m' \\ &= dec \times 2 \times 2^{i-1} + b[i] \times 2^{i-1} + m' \\ &= dec \times 2^i + b[i] \times 2^{i-1} + m - b[i] \times 2^{i-1} \\ &= dec \times 2^i + m = n = n' \end{aligned}$$

In addition, since $i > 0$ (given that the loop condition holds), $i' = i - 1 \geq 0$.Therefore, $Inv(n', t', k', b')$ holds after the next iteration. □