Suggested Solutions to Midterm Problems

1. Consider the geometric series: 1, 2, 4, 8, 16, ... Prove *by induction* that any positive integer can be written as a sum of distinct numbers from this series.

   Solution. The proof is by *strong induction* on n that represents an arbitrary positive integer.

   Base case ($n = 1$): the statement is obviously true, as 1 is itself in the geometric series.

   Inductive step ($n > 1$): we consider two cases separately: when n is even and when n is odd.

   Case 1: n is even. Let $n = 2 \times k$, where $k \geq 1$. By the induction hypothesis, let k be the sum of the series $s_1, s_2, \ldots, s_j$, which are distinct numbers taken from the geometric series. Then, n is the sum of $2 \times s_1, 2 \times s_2, \ldots, 2 \times s_j$, which are also distinct numbers from the geometric series.

   Case 2: n is odd. Let $n = 2 \times k + 1$, where $k \geq 1$. By the induction hypothesis, let k be the sum of the series $s_1, s_2, \ldots, s_j$, which are distinct numbers from the geometric series. Then, n is the sum of $1, 2 \times s_1, 2 \times s_2, \ldots, 2 \times s_j$, which are also distinct numbers from the geometric series. □

2. Consider a round-robin tournament among n players. In the tournament, each player plays once against all other $n-1$ players. There are no draws, i.e., for a match between A and B, the result is either A beat B or B beat A. Prove *by induction* that, after a round-robin tournament, it is always possible to arrange the n players in an order $p_1, p_2, \cdots, p_n$ such that p_1 beat p_2, p_2 beat p_3, $\cdots$, and p_{n-1} beat p_n. (Note: the "beat" relation, unlike "$\geq$", is not transitive.)

   Solution. The proof is by induction on the number n of players.

   Base case ($n = 2$): There are exactly two players, say A and B. Either A beat B, in which case we order them as A, B, or B beat A, in which case we order them as B, A.

   Induction step ($n > 2$): Pick any of the n players, say A. From the induction hypothesis, the other $n-1$ players can be ordered as $p_1, p_2, \cdots, p_{n-1}$ such that p_1 beat p_2, p_2 beat p_3, $\cdots$, and p_{n-2} beat p_{n-1}. We now exam the result of the match played between A and p_1. If A beat p_1, then we get a satisfying order $A, p_1, p_2, \cdots, p_{n-1}$. Otherwise ($p_1$ beat A), we continue to exam the result of the match played between A and p_2. If A beat p_2, then we get a satisfying order $p_1, A, p_2, \cdots, p_{n-1}$. Otherwise ($p_2$ beat A), we continue as before. We end up either with $p_1, p_2, \cdots, p_{i-1}, A, p_i, \cdots, p_{n-1}$ for some $i \leq n-1$ or eventually with $p_1, p_2, \cdots, p_{n-1}, A$ if A is beaten by every other player, in particular p_{n-1}. □

3. Consider the following variant of Euclid's algorithm for computing the greatest common divisor of two positive integers.

```
Algorithm Euclid_Simplified $(m, n)$;
begin
    // assume that $m > 0 \wedge n > 0$
    $x := m$;
    $y := n$;
    while $x \neq 0 \wedge y \neq 0$ do
        if $x < y$ then swap($x$,$y$);
        $x := x - y$;
    od
    ...
end
```

where swap(x,y) exchanges the values of x and y.

(a) To speak about the values of a variable at different times during an execution, let m', n', x', and y' denote respectively the new values of m, n, x, and y after the next iteration of the while loop (m, n, x, and y themselves denote the current values of these variables at the start of the next iteration). Please give a precise relation between m', n', x', and y' and m, n, x, and y.

Solution. From the loop body, we deduce the following relationship (assuming that the loop condition holds):

$$\begin{array}{ll} & ((x < y) \rightarrow (x' = y - x) \wedge (y' = x)) \\ \wedge & ((x \geq y) \rightarrow (x' = x - y) \wedge (y' = y)) \\ \wedge & m' = m \\ \wedge & n' = n \end{array}$$

□

(b) Prove *by induction* that the following is a loop invariant of the while loop:

$$x \geq 0 \wedge y \geq 0 \wedge (x \neq 0 \vee y \neq 0) \wedge \gcd(x, y) = \gcd(m, n).$$

Solution. Let $Inv(m, n, x, y)$ denote the assertion to be proven a loop invariant.

(1) When the flow of control reaches the loop for the first time, $x = m$ and $y = n$, with $m > 0$ and $n > 0$. Obviously, $x \geq 0$, $y \geq 0$, $x \neq 0 \vee y \neq 0$, and $\gcd(x, y) = \gcd(m, n)$ and therefore the assertion $Inv(m, n, x, y)$ holds.

(2) Assume that $Inv(m, n, x, y)$ is true at the start of the next iteration and the loop condition $(x \neq 0 \wedge y \neq 0)$ holds. We need to show that $Inv(m', n', x', y')$ also holds. There are two cases to consider: when $x < y$ and when $x \geq y$. We prove the first case; the second case can be proven similarly.

Suppose $x < y$. $x' = y - x > 0$ and hence $x' \geq 0$; also, $y' = x \geq 0$ (from the induction hypothesis). These also imply that $x' \neq 0 \vee y' \neq 0$. $\gcd(x', y') = \gcd(y - x, x) = \gcd(y, x) = \gcd(x, y)$, which from the induction hypothesis equals $\gcd(m, n) = \gcd(m', n')$, and therefore $\gcd(x', y') = \gcd(m', n')$. Therefore, $Inv(m', n', x', y')$ holds and this concludes the proof. □

4. Consider bounding summations by integrals.

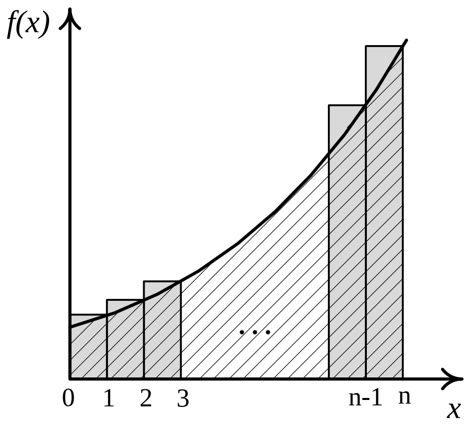
(a) If $f(x)$ is monotonically *increasing*, then

$$\int_0^n f(x)dx \leq \sum_{i=1}^{n} f(i).$$

Show that this is indeed the case.

Solution. (Jing-Jie Lin)

This is easily seen by comparing the areas (on the $R \times R$ plane) defined by the formulae on the two sides. As shown in the following diagram, the integral $\int_0^n f(x)dx$ equals the area under the curve that is shaded with thin parallel lines. The area is apparently no larger than the total area of the vertical bars which represents $\sum_{i=1}^n f(i)$.



□

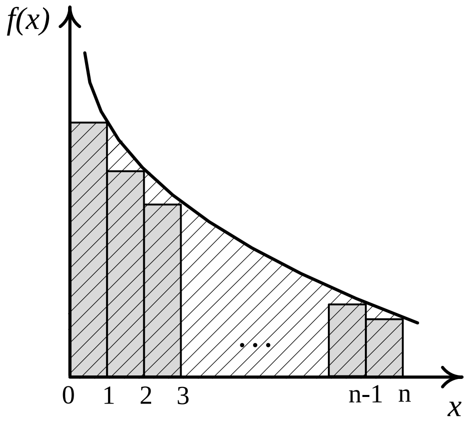
(b) If $f(x)$ is monotonically *decreasing*, then

$$\sum_{i=1}^{n} f(i) \leq f(1) + \int_1^n f(x)dx.$$

Show that this is indeed the case.

Solution. (Jing-Jie Lin)

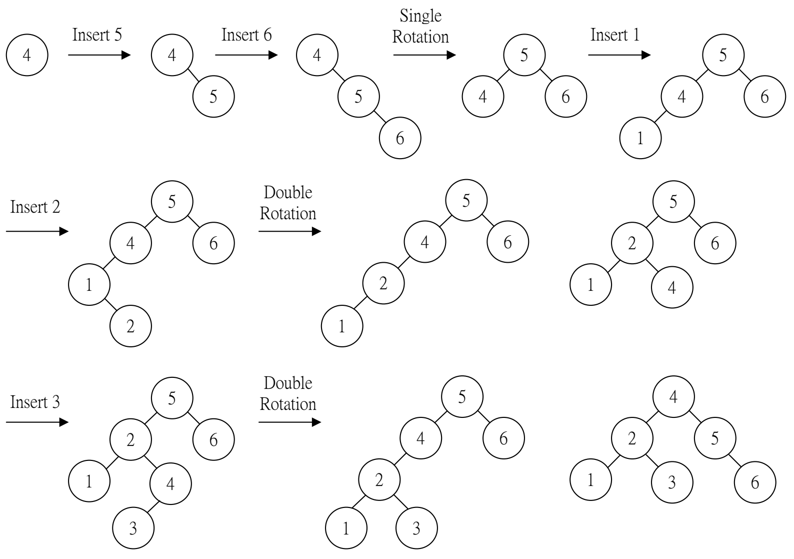
Similar to the previous solution.



□

5. Show all intermediate and the final AVL trees formed by inserting the numbers 4, 5, 6, 1, 2, and 3 (in this order) into an empty tree. Please use the following ordering convention: the key of an internal node is larger than that of its left child and smaller than that of its right child. If re-balancing operations are performed, please also show the tree before re-balancing and indicate what type of rotation is used in the re-balancing.

   Solution. (Jing-Jie Lin)



(Note: "Insert 4" is not shown.) □

6. The input is a set S with n real numbers. Design an $O(n)$ time algorithm to find a number that is *not* in the set. Prove that $\Omega(n)$ is a lower bound on the number of steps required to solve this problem.

   Solution. When there is just one number (i.e., $n = 1$) in S, it is trivial to find a real number different from the number in S, e.g., by adding one to or subtracting one from the existing number. When there are exactly two numbers, we simply take their average which will be different from both numbers, since by the definition of a set, all numbers in S are distinct. When there are more than two numbers, we proceed as follows.

   Store the first two numbers as a pair and read the remaining numbers one by one. When the next number read falls between the pair, we replace the smaller of the pair by the number just read. At the end we will have a pair of real numbers and none of the numbers in S falls between the pair. We take the average of the pair which will be different any number in S.

   We next argue for the lower bound $\Omega(n)$. This is quite straightforward, since every number in S must be read and there are n numbers. Any algorithm that skips a number may return a wrong result, as the result may happen to be equal to the number that is skipped. □

7. Let $x_1, x_2, \cdots, x_{2n-1}, x_{2n}$ be a sequence of $2n$ real numbers. Design an algorithm to partition the numbers into n pairs such that the maximum of the n sums of pair is minimized. It may be intuitively easy to get a correct solution. You must explain how the algorithm can be designed using induction.

   Solution. We first fix some notations:

   - We represent a partition of a list $x_1, x_2, \cdots, x_{2n-1}, x_{2n}$ into n pairs as a set of sets of two elements $\{\{y_1, y_2\}, \{y_3, y_4\}, \cdots, \{y_{2n-1}, y_{2n}\}\}$, where $y_1, y_2, \cdots, y_{2n-1}, y_{2n}$ is a permutation of $x_1, x_2, \cdots, x_{2n-1}, x_{2n}$.
   - For a list A of $2n$ elements, let **MinMaxPair**(A) denote some partition that meets the problem requirement for $2n$ elements.

- $A \setminus B$, where A is a list and B a set, denotes the list of elements in A but not in B. We stipulate that elements in $A \setminus B$ appear in the same order as in A.

We are given a list $X = x_1, x_2, \cdots, x_{2n-1}, x_{2n}$. If $n = 1$, i.e., there are only two elements, the solution is obvious, namely $\{\{x_1, x_2\}\}$. Now consider the cases of $n > 1$. Let y_1 denote the smallest element and y_{2n} the largest element in X. We claim that **MinMaxPair**$(X \setminus \{y_1, y_{2n}\}) \cup \{\{y_1, y_{2n}\}\}$ meets the problem requirement for $2n$ elements, i.e., the pair $\{y_1, y_{2n}\}$ is part of an optimal partition. Suppose $\{y_i, y_j\}$ is the pair with the largest sum in **MinMaxPair**$(X \setminus \{y_1, y_{2n}\})$. Pairing y_{2n} with either y_i or y_j (instead of y_1) would produce a pair whose sum is at least as large as that of $\{y_1, y_{2n}\}$ and that of any pair in **MinMaxPair**$(X - \{y_1, y_{2n}\})$. To compute **MinMaxPair**$(X \setminus \{y_1, y_{2n}\})$, we need to solve the same problem with two elements less and here we invoke the induction hypothesis.

In the above, we select and remove the smallest and the largest elements from the current list in each step. This would incur a complexity of $O(n)$ for each step, making the complexity of the whole algorithm $O(n^2)$. We can improve this by sorting the list right in the beginning before pairing up the elements. So, the algorithm can be summarized as follows.

(a) Sort the input list X to get Y.

(b) Suppose the current list $Y = y_1, y_2, \cdots, y_{2i-1}, y_{2i}$ $(i \geq 1)$. Remove and output the pair $\{y_1, y_{2i}\}$ from Y.

(c) Repeat the previpus step until Y is empty.

□

8. Apply the Quicksort algorithm to the following array. Show the contents of the array after each partition operation. If you use a different partition algorithm (from the one discussed in class), please describe it.

1	2	3	4	5	6	7	8	9	10	11	12
10	9	4	7	12	6	8	2	1	11	5	3

Solution. (Wei-Hsien Chang)

1	2	3	4	5	6	7	8	9	10	11	12
10	9	4	7	12	6	8	2	1	11	5	3
5	9	4	7	3	6	8	2	1	$\underline{10}$	11	12
3	1	4	2	$\underline{5}$	6	8	7	9	$\underline{10}$	11	12
2	1	$\underline{3}$	4	$\underline{5}$	6	8	7	9	$\underline{10}$	11	12
1	$\underline{2}$	$\underline{3}$	4	$\underline{5}$	6	8	7	9	$\underline{10}$	11	12
1	$\underline{2}$	$\underline{3}$	4	$\underline{5}$	$\underline{6}$	8	7	9	$\underline{10}$	11	12
1	$\underline{2}$	$\underline{3}$	4	$\underline{5}$	$\underline{6}$	7	$\underline{8}$	9	$\underline{10}$	11	12
1	$\underline{2}$	$\underline{3}$	4	$\underline{5}$	$\underline{6}$	7	$\underline{8}$	9	$\underline{10}$	$\underline{11}$	12

□

9. Consider rearranging the following array into a max heap using the *bottom-up* approach.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	3	5	1	2	6	14	8	11	4	10	13	15	9	12

Please show the result (i.e., the contents of the array) after a new element is added to the current collection of heaps (at the bottom) until the entire array has become a heap.

Solution. (Jui-Shun Lai)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	3	5	1	2	6	14	8	11	4	10	13	15	9	12
7	3	5	1	2	6	14	8	11	4	10	13	15	9	12
7	3	5	1	2	$\underline{15}$	14	8	11	4	10	13	$\underline{6}$	9	12
7	3	5	1	$\underline{10}$	15	14	8	11	4	$\underline{2}$	13	6	9	12
7	3	5	$\underline{11}$	10	15	14	8	$\underline{1}$	4	2	13	6	9	12
7	3	$\underline{15}$	11	10	$\underline{13}$	14	8	1	4	2	$\underline{5}$	6	9	12
7	$\underline{11}$	15	$\underline{8}$	10	13	14	$\underline{3}$	1	4	2	5	6	9	12
$\underline{15}$	11	$\underline{14}$	8	10	13	$\underline{12}$	3	1	4	2	5	6	9	$\underline{7}$

$\square$

10. Prove that the sum of the heights of all nodes in a complete binary tree with n nodes is at most $n-1$. You may assume it is known that the sum of the heights of all nodes in a *full* binary tree of height h is $2^{h+1}-h-2$. (Note: a single-node tree has height 0.)

    Solution. Let $G(n)$ denote the sum of the heights of all nodes in a complete binary tree with n nodes. For a full binary tree (a special case of complete binary trees) with $n = 2^{h+1}-1$ nodes where h is the height of the tree, we already know that $G(n) = 2^{h+1}-(h+2) = n-(h+1) \leq n-1$. With this as a basis, we prove the general case of arbitrary complete binary trees by induction on the number n (≥ 1) of nodes.

    Base case ($n=1$ or $n=2$): When $n=1$, the tree is the smallest full binary tree with one single node whose height is 0. So, $G(n) = 0 \leq 1-1 = n-1$. When $n=2$, the tree has one additional node as the left child of the root. The height of the root is 1, while that of its left child is 0. So, $G(n) = 1 \leq 2-1 = n-1$.

    Inductive step ($n > 2$): If n happens to be equal to $2^{h+1}-1$ for some $h \geq 1$, i.e., the tree is full, then we are done; note that this covers the case of $n = 3 = 2^{1+1}-1$. Otherwise, suppose $2^{h+1}-1 < n < 2^{h+2}-1$ ($h \geq 1$), i.e., the tree is a "proper" complete binary tree with height $h+1 \geq 2$. We observe that at least one of the two subtrees of the root is full, while the other is complete (possibly full). There are three cases to consider:

    Case 1: The left subtree is full with n_1 nodes and the right one is complete but not full with n_2 nodes (such that $n_1+n_2+1 = n$). In this case, both subtrees much be of height h and $n_1 = 2^{h+1}-1$. ¿From the special case of full binary trees and the induction hypothesis, $G(n_1) = 2^{h+1}-(h+2) = n_1-(h+1)$ and $G(n_2) \leq n_2-1$. $G(n) = G(n_1)+G(n_2)+(h+1) \leq (n_1-(h+1))+(n_2-1)+(h+1) = (n_1+n_2+1)-2 \leq n-1$.

    Case 2: The left subtree is full with n_1 nodes and the right one is also full with n_2 nodes. In this case, the left subtree much be of height h and $n_1 = 2^{h+1}-1$, while the right subtree much be of height $h-1$ and $n_2 = 2^h-1$. ¿From the special case of full binary trees, $G(n_1) = 2^{h+1}-(h+2) = n_1-(h+1)$ and $G(n_2) = 2^h-(h+1) = n_2-h$. $G(n) = G(n_1)+G(n_2)+(h+1) \leq (n_1-(h+1))+(n_2-h)+(h+1) = (n_1+n_2+1)-(h+1) \leq n-1$.

    Case 3: The left subtree is complete but not full with n_1 nodes and the right one is full with n_2 nodes. In this case, the left subtree much be of height h, while the right subtree much be of height $h-1$ and $n_2 = 2^h-1$. ¿From the induction hypothesis and the special case of full binary trees, $G(n_1) \leq n_1-1$ and $G(n_2) = 2^h-(h+1) = n_2-h$. $G(n) = G(n_1)+G(n_2)+(h+1) \leq (n_1-1)+(n_2-h)+(h+1) = (n_1+n_2+1)-1 = n-1$. $\square$