Suggested Solutions to Midterm Problems

1. Given a set of $n+1$ numbers out of the first $2n$ (starting from 1) natural numbers 1, 2, 3, ..., $2n$, prove *by induction* that there are two numbers in the set, one of which divides the other.

   Solution. The proof is by induction on n.

   Base case ($n = 1$): There is only one subset of 2 ($= n + 1$) numbers from $\{1, 2\}$, which is the set $\{1, 2\}$ itself. 1 divides 2.
   Inductive step ($n = k+1 > 1$): We need to show that any selection (subset) of $k+2$ numbers from $\{1, 2, \cdots, 2k, 2k+1, 2k+2\}$ contains two numbers, one of which divides the other. If the selection includes $k+1$ numbers from $\{1, 2, \cdots, 2k\}$, then by the induction hypothesis we are done. Otherwise, the selection must contain both $2k+1$ and $2k+2$ and also include a selection S of other k numbers from $\{1, 2, \cdots, 2k\}$.
   Case one ($k+1 \in S$): $k+1$ divides $2k+2$.
   Case two ($k+1 \notin S$): If S happens to contain two numbers one of which divides the other, then we are done. Otherwise, from the induction hypothesis, S must contain a number that divides $k+1$. This is so, because (a) $k+1$ does not divide any number in $\{1, 2, \cdots, 2k\}$ and (b) $S \cup \{k+1\}$ is a selection of $k+1$ numbers from $\{1, 2, \cdots, 2k\}$ and by the induction hypothesis must contain two numbers one of which divides the other. The number that divides $k+1$ also divides $2k+2$. □

2. Below is a theorem from Manber's book:

   > For all constants $c > 0$ and $a > 1$, and for all monotonically increasing functions $f(n)$, we have $(f(n))^c = O(a^{f(n)})$.

   Prove, by using the above theorem, that $n(\log n)^2 = O(n^{1.5})$.

   Solution. If we are able to show that $(\log n)^2 = O(n^{.5})$, then $n(\log n)^2 = O(n \cdot n^{.5}) = O(n^{1.5})$.

   Applying the theorem with $f(n) = \log n$, $c = .5$, and $a = 2^{.5}$, we have $(\log n)^{.5} = O((2^{.5})^{\log n}) = O(2^{.5 \log n}) = O(2^{\log n^{.5}}) = O(n^{.5})$.
   (Note: As usual, we have assumed the base of logarithm is 2. The same result can still be obtained even if a different base is used.) □

3. Find the asymptotic behavior of the function $T(n)$ defined by the recurrence relation

   $$T(n) = T(n/2) + \sqrt{n},\ T(1) = 1.$$

You may consider only values of n that are powers of 2. (5 points)

Solution. $T(n) = O(\sqrt{n})$. We prove by induction that $T(n) \leq 4\sqrt{n}$.
Base case: $T(1) = 1 \leq 4 = 4\sqrt{1}$.
Inductive step: $T(2n) = T(n) + \sqrt{2n} \leq 4\sqrt{n} + \sqrt{2}\sqrt{n} = (4 + \sqrt{2})\sqrt{n} \leq 4\sqrt{2}\sqrt{n} = 4\sqrt{2n}$.

The problem can also be solved by applying Theorem 3.4 (Page 51) in Manber's book.

Note: How was the constant 4 determined? Suppose $T(n) \leq c\sqrt{n}$ for some c. Anticipating the proof obligation in the inductive step that $T(2n) = T(n) + \sqrt{2n} \leq c\sqrt{n} + \sqrt{2}\sqrt{n} \leq c\sqrt{2}\sqrt{n}$, we stipulate that the constant c should be $\geq \sqrt{2}/(\sqrt{2} - 1) = 3.\cdots$; 4 is just one of such constants.

But, how do we know in the first place that $T(n) = O(\sqrt{n})$ is a good guess? The guess is motivated by considering two other recurrence relations: "$T(n) = T(n/2) + 1$, $T(1) = 1$" ($T(n) = O(\log n)$) and "$T(n) = T(n/2) + n$, $T(1) = 1$" ($T(n) = O(n)$). □

4. In the towers of Hanoi puzzle, there are three pegs A, B, and C, with n (generalizing the original eight) disks of different sizes stacked in decreasing order on peg A. The objective is to transfer all the disks on peg A to peg B, moving one disk at a time (from one peg to one of the other two) and never having a larger disk stacked upon a smaller one.
(a) Give an algorithm to solve the puzzle. (5 points)

Solution.

```
Algorithm Towers_Hanoi(A,B,C,n);
begin
   if n=1 then
      pop x from A and push x to B
   else
      Towers_Hanoi(A,C,B,n-1);
      pop x from A and push x to B;
      Towers_Hanoi(C,B,A,n-1);
end;
```

□

(b) Suppose that there is an additional fourth peg D. Modify your algorithm to take advantage of the additional peg. (10 points)

Solution.

```
Algorithm Four_Towers_Hanoi(A,B,C,D,n);
begin
   if n<=2 then
```

```
      Towers_Hanoi(A,B,C,n);
   else
      Four_Towers_Hanoi(A,D,B,C,n-2);
      Towers_Hanoi(A,B,C,2);
      Four_Towers_Hanoi(D,B,C,A,n-2);
end;
```

□

5. Show all intermediate and the final AVL trees formed by inserting the numbers 9, 8, 7, 6, 5, 0, 1, 2, 3, and 4 (in this order).

   Solution. See the attached. □

6. Suppose that you are given an algorithm as a *black box* (you cannot see how it is designed) that has the following properties: If you input any sequence of real numbers and an integer k, the algorithm will answer "yes" or "no," indicating whether there is a subset of the numbers whose sum is exactly k. Show how to use this black box to find the subset whose sum is k, if it exists. You should use the black box $O(n)$ times (where n is the size of the sequence).

   Solution. Let `Find_Subset` denote the given algorithm, which takes as input an array of real numbers, the size of the array (these two together representing the sequence of real numbers), and an integer.

```
Algorithm Print_Subset(S,n,k);
begin
   if Find_Subset(S,n,k)="no" then
      print "No suitable subset"; halt;
   print "Below is a suitable subset:";
   remain := k;
   i := n;
   while remain>0 do
      thisreal := S[i];
      S[i] := 0;
      if Find_Subset(S,n,remain)="no" then
         print thisreal;
         /* readjust remain */
         if (thisreal has a fraction) then
            remain := truncate(remain-thisreal) + 1;
            S[i] := 1 - fraction(thisreal);
         else remain := remain - thisreal;
      i := i - 1;
```

□

7. Apply the quicksort algorithm to the following array. Show the result after each partition operation.

8	1	5	11	14	12	2	16	7	3	13	4	10	9	15	6

Solution.

8	1	5	11	14	12	2	16	7	3	13	4	10	9	15	6
7	1	5	6	4	3	2	(**8**)	16	12	13	14	10	9	15	11
2	1	5	6	4	3	(**7**)	**8**	16	12	13	14	10	9	15	11
1	(**2**)	5	6	4	3	**7**	**8**	16	12	13	14	10	9	15	11
1	**2**	4	3	(**5**)	6	**7**	**8**	16	12	13	14	10	9	15	11
1	**2**	3	(4)	**5**	6	**7**	**8**	16	12	13	14	10	9	15	11
1	**2**	3	4	**5**	6	**7**	**8**	11	12	13	14	10	9	15	(**16**)
1	**2**	3	4	**5**	6	**7**	**8**	10	9	(**11**)	14	13	12	15	**16**
1	**2**	3	4	**5**	6	**7**	**8**	9	(**10**)	**11**	14	13	12	15	**16**
1	**2**	3	4	**5**	6	**7**	**8**	9	**10**	**11**	12	13	(**14**)	15	**16**
1	**2**	3	4	**5**	6	**7**	**8**	9	**10**	**11**	(**12**)	13	**14**	15	**16**

□

8. (a) Give an algorithm for building a (max) heap using the bottom-up approach.

Solution. Left as an exercise. □

(b) Rearrange the following array into a heap using the algorithm obtained in (a).

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

Solution.

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

□

Show the result after each element is added to the part of array that already satisfies the heap property.

9. Draw a Huffman tree for a text that contains eight characters A, B, C, D, E, F, G, and H with frequencies 7, 2, 3, 5, 16, 10, 4, and 2, respectively.

Solution. See the attached. □

10. Compute the *next* table as in the KMP algorithm for the string *aababaabaa*. Show the details of your calculation.

Solution.

$next[1] = -1$.

$next[2] = 0$.

$next[3] = 0$: $B_{3-1} = B_2 = b$, while $B_{next[3-1]+1} = B_1 = a$; $B_{3-1} \neq B_{next[3-1]+1}$. As $next[next[3-1]+1]+1 = next[1]+1 = 0$, $next[3] = 0$.

$next[4] = 0$: $B_{4-1} = B_3 = b$, while $B_{next[4-1]+1} = B_1 = a$; $B_{4-1} \neq B_{next[4-1]+1}$. $next[1](= next[next[4-1]+1]) + 1 = 0$. So, $next[4] = 0$.

$next[5] = 1$: $B_{5-1} = B_4 = a$, while $B_{next[5-1]+1} = B_1 = a$; $B_{5-1} = B_{next[5-1]+1}$. So, $next[5] = next[5-1]+1 = 1$.

$next[6] = 0$: $B_{6-1} = B_5 = b$, while $B_{next[6-1]+1} = B_2 = a$; $B_{6-1} \neq B_{next[6-1]+1}$. $B_{next[2]+1} = B_1 = a$; $B_{6-1} \neq B_{next[2]+1}$. $next[1](= next[next[4-1]+1]) + 1 = 0$. So, $next[6] = 0$.

$next[7] = 1$: $B_{7-1} = B_6 = a$, while $B_{next[7-1]+1} = B_1 = a$; $B_{7-1} = B_{next[7-1]+1}$. So, $next[7] = 1$.

$next[8] = 2$: $B_{8-1} = B_7 = a$, while $B_{next[8-1]+1} = B_2 = a$; $B_{8-1} = B_{next[8-1]+1}$. So, $next[8] = 2$.

$next[9] = 3$: $B_{9-1} = B_8 = b$, while $B_{next[9-1]+1} = B_3 = b$; $B_{9-1} = B_{next[9-1]+1}$. So, $next[9] = 3$.

$next[10] = 4$: $B_{10-1} = B_9 = a$, while $B_{next[10-1]+1} = B_4 = a$; $B_{10-1} = B_{next[10-1]+1}$. So, $next[10] = 4$.

1	2	3	4	5	6	7	8	9	10
a	a	b	a	b	a	a	b	a	a
-1	0	1	0	1	0	1	2	3	4

□