Algorithms
Practice Midterm

Total time is 180 minutes and the total score is 100 points (plus 5 points bonus). You are allowed to use your textbook, your own lecture notes, and your own homework. Do not use computer or other materials. Please show all your work because partial credits could be given. Read all the problems and solve them in the order that makes the most progress. Don't spend too much time on a single problem.

Problem 1 (15 points, 5 points each) <Recurrence Complexity>
Find the asymptotic bounds for the following recurrence. Assume the base cases are constant time.

A. $T(n) = \begin{cases} \Theta(1), & \text{for n} \le 1 \\ T(\lceil c_1 n \rceil) + T(\lceil c_2 n \rceil) + T(\lceil c_3 n \rceil) + c_4 n, & \text{for n>1} \end{cases}$, where c_1, c_2, c_3, c_4, are positive constants and $c_1+c_2+c_3 = 1$. Please use the recursion tree to find the asymptotic upper bound O.

B. $T(n) = \begin{cases} \Theta(1), & \text{for n} \le 1 \\ 3T(\lceil n/3 \rceil) + n/7, & \text{for n>1} \end{cases}$ Please use the master theorem. What kind of bound is it: O, Ω, or Θ?

C. $T(n) = \begin{cases} \Theta(1), & \text{for n} \le 1 \\ 8T(\lceil n/2 \rceil) + 6n, & \text{for n>1} \end{cases}$ Use substitution method to find the asymptotic tight bound Θ.

Problem 2 (15 points, 5 points each) <Min-Heap>

A min-heap (A) is very similar to the max-heap except that $A[\text{parent}(i)] \le A[i]$ for every node i. Please use the corresponding heap algorithm in the textbook to answer the following questions.

A. Initially, array $A[i]$ = 90, 60, 70, 100, 10, 50, 40, 20, for i = 1… 8. Suppose we modify the BUILD-MAX-HEAP function in our textbook to produce a new function, BUILD-MIN-HEAP. Please run the BUILD-MIN-HEAP(A) algorithm and draw every step in the form of binary trees. (similar to the figures in our lecture notes and textbooks)

B. Suppose we modify the HEAP-INCREASE-KEY function in our textbook to produce a new function, HEAP-DECREASE-KEY. Show the min-heap after calling HEAP-DECREASE-KEY(A, i=7 ,key=30). Draw every step in the form of binary trees. (similar to the figures in our lecture notes and textbooks)

C. (continued from B) What value is returned by calling HEAP-EXTRACT-MIN(A)? Draw every step in the process of HEAP-EXTRACT-MIN(A). Please draw the tree similar to the figures in our lecture notes and textbooks. What value is returned.

Problem 3 (15 points, 5points each) <Red-Black Tree>

Suppose we insert eight numbers: 90, 60, 70, 100, 10, 50, 40, and 20, into an initially empty red-black tree.

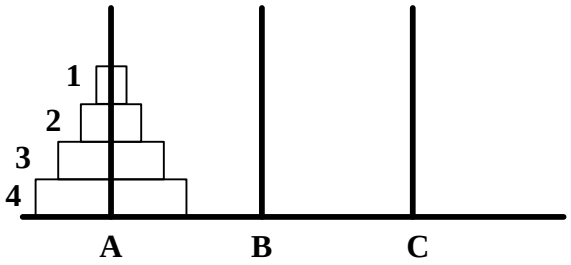
A. Draw red-black trees after each insertion, totally 8 trees. Please show the case number that you use in RB-INSERT-FIXUP.

B. (cont'd from A) Show the in-order tree walk of this tree.

C. (cont'd from A) Show the red-black trees when we delete 60. After that, show the tree when we delete 70. Show all the steps that you use in RB-DELETE-FIXUP.

Problem 4 (15 points, 5 points each) <Tower of Hanoi>

The tower of Hanoi is a famous game, invented by the French mathematician Edouard Lucas in 1883. You are required to move *n* disks from the *source pole* to the *destination pole*, with the help of one additional *spare pole*. The rules are simple: you can move only one disk at a time. A larger disk cannot sit on top of a smaller disk. The following figure shows an example of 4 disks and 3 poles.



A. Please complete the following table to move 4 disks from pole A to pole C. (hint total 15 steps)

step	disk	from	to
1	1	A	B
2	2	A	C
3	1	B	C
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

B. Please finish the following recursive algorithm.

```
Hanoi_Tower (count, source, destination, spare)
    if (count == 1)
        move disk from source to destination
    else
        Hanoi_Tower ( _____, _____, _____, _____ )
        Hanoi_Tower (   1   , _____, _____, _____ )
        Hanoi_Tower ( _____, _____, _____, _____ )
```

C. Please analyze the complexity of this algorithm.

Problem 5 (20 points) <Sorting>

In a singly linked list *L*, every object *x* has two attributes: *x.next* and *x.key*. If there is a object next to *x*, then *x.next* points to the next object; otherwise, *x.next*=NIL. If the list is non-empty, *L.head* points to the first object in the list; otherwise, *L.head=NIL*. The LIST-SORT algorithm is implemented using FUNCTION-A and FUNCTION-B.

```
LIST-SORT(L)
   L' =FUNCTION-A(L)
   LIST-SORT(L)
   LIST-SORT(L')
   L=FUNCTION-B(L, L')
```

```
FUNCTION-A(L)
   a=L.head
   b=a.next.next
   while (b ≠NIL)
      a=a.next
      b=b.next
      if (b ≠NIL)
         b=b.next
   let L' be a new empty list
   L'.head=a.next
   a.next = NIL
   return L'
```

```
FUNCTION-B(L, L')
   if (L'.head=NIL)   return L
   if (L.head=NIL)   return L'
   a=L.head;     b=L'.head
   let p be a new empty object
   if (a.key < b.key)
      p=a;   a=a.next
   else
      L.head=b;   p=b;   b=b.next
   while (a≠NIL and b≠NIL)
      if (a.key < b.key)
         p.next=a;   p=a;   a=a.next
      else
         p.next=b;   p=b;   b=b.next
   if (a = NIL)
      _______
   else
      _______
   return L
```

A. What is the function of FUNCTION-A? Please explain the while loop. (2 points)
B. Analyze the complexity of FUNCTION-A. (2 points)
C. What is the function of FUNCTION-B? (2 points)
D. Please fill in the blanks in FUNCTION-B. (2 points)
E. Analyze the complexity of FUNCTION-B (2 points)
F. Analyze the complexity of LIST-SORT. Please write a recursive function *T(n)*=? (3 points)
G. Is this a in-place sorter? Is this a stable sorter? (2 points)

Problem 6 (15 points) <Smart Robot>

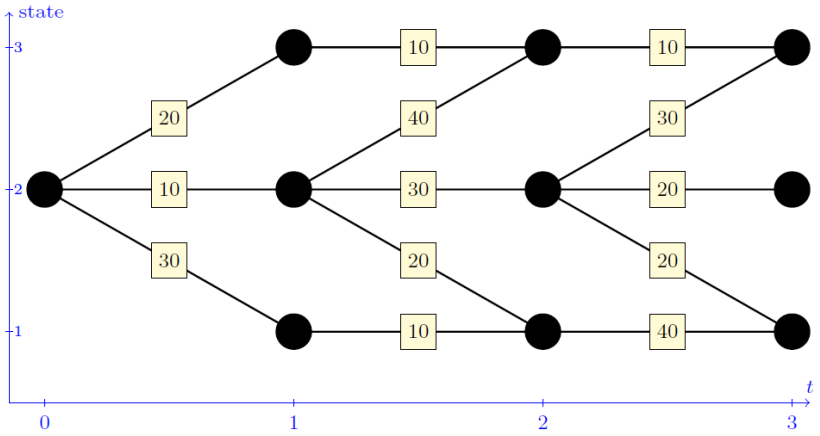
Suppose that you designed a robot, named *OGLA*, that moves from time *t*=0 to *t*=3. *OGLA* can only move from left to right (from time *t* to *t+1*). *OGLA* cannot move from right to left. At each time, OLGA can only stay in either one of three states: 1 , 2, and 3. An edge in the graph from *t* to *t+1* is a path that *OLGA* can travel. If there is no edge, *OLGA* cannot travel along that path. The numbers on the edge represents the energy needed to travel along the edge. For example, if *OGLA* takes the middle path, it will need 10+30+20= 60 energy. You want *OGLA* to use as little total energy as possible.

A. Suppose *OGLA* uses the greedy algorithm. That means, it always chooses the edge with the lowest energy from time *t* to *t+1*. What is the minimum energy needed for this graph? (2 points)

B. You want to improve *OGLA* using dynamic programming. What is the minimum energy needed for this graph? Suppose *E(s, t)* is the minimum energy at time *t* and state *s*. Please show a table of *E(s,t)* to explain your answer. (5 points)

C. Prove or disprove your answer in part **B** is optimal. Hint: show this problem have the optimal substructure property so we can solve it using dynamic programming. (3 points)

D. Given an arbitrary graph with *T* time steps and *S* states, what is the time complexity of this dynamic programming? For example, our graph *T*=3 and *S*=3. Please note the edge connection can be arbitrary. You can assume infinite edge weight if there is no connection. (5 points)



Problem 7 (10+3 points)

Your classmate, Jerry, invented a new sort algorithm. The pseudo code goes like this.

```
TRIPLE-SORT(L)
  Finish = false
  while not Finish
    Finish = true
    for i=0 to size(L)-2
      if L[i+2] < L[i]
        Reverse the order of L[i]~L[i+2]
        Finish = false
```

A. Please analyze the complexity of this algorithm. Please note that you should specify O and Θ correctly. (5 points)

B. Please show an example where Jerry's sorter can produce incorrect outputs. Please explain your answer. (5 points)

C. Please write an algorithm with complexity O(N lg N) to check whether a sequence of numbers can be sorted correctly by Jerry's algorithm. N is the length of the sequence (3 bonus)

Problem 8 (BONUS 2 points) <Beat your teacher!>

請具一個演算法生活化例子.

Grading:

2 points: your answer is innovative and correct.

1 point: your answer is not innovative but correct. Or innovative but incorrect