浙江大学 2006－2007 学年秋季学期

《高级数据结构与算法分析》课程期末考试试卷

开课学院：软件学院、计算机学院、竺可桢学院 ，考试形式：**闭卷**，允许带_ 无 入场

考试时间：_2007_年_1_月_16_日， 所需时间：_120_分钟

考生姓名：_____学号：_____专业：_____教师：_____

题序	一	二	三	四	总 分
得分					
评卷人					

NOTE: Please write your answers on the answer sheet.
注意：请将答案填写在答题纸上。

I. Please fill in the blanks (There could be multiple answers for one blank). (24 points) ***Note:* Zero point *for a blank selection since there is at least one answer for each problem.***

(1) The time complexity for finding an element in a splay tree with N nodes is C (or b – not worst case) . (2 points)
a. O(1) b. O(log N) c. O(N) d. O(N log N)

(2) The depth of an AVL tree with 32 nodes is at least b (2 points) and at most c .(2 points)
a. 5 b. 6 c. 7 d. 8

(3) When solving the bin packing problem with the best-fit method, arranging the remaining capacities of the bins in b (2 points) can reduce the time complexity.
a. a linear sequential storage b. an AVL tree
c. a min-heap d. an ordered linear storage

(4) A binomial queue of size 53 can be represented by the following binomial queues d . (2 points)
a. B_0 B_1 B_2 B_3 B_4 B_5 b. B_0 B_1 B_3 B_5
c. B_1 B_3 B_5 B_6 d. B_0 B_2 B_4 B_5

(5) A leftist tree with r nodes on the right path must have a nodes.(2 points)
a. at least 2^r-1 b. at most 2^r-1 c. at least 2^{r-1} d. at most 2^{r-1}

(6) For a binomial queue, (b) d take(s) a constant time on average. (2 points)

a. merging b. find-min c. delete-min d. insertion

(7) Which of the following problems are not known to be run in polynomial time: b d .(2 points)

a. Euler circuit problem b. Hamilton cycle problem

c. Single-source unweighted shortest-path problem

d. Single-source unweighted longest-path problem

(8) An amortized time bound is b d . (2 points)

a. >= worst-case time bound b. >= average-case time bound

c. <= best-case time bound d. <= worst-case time bound

(9) To solve a problem with input size N by divide and conquer algorithms, if the conquer step takes O(N) extra work to form the solution from the sub-solutions, then among the following four dividing methods, b (2 points) is the best one while a (2 points) is the worst one.

a. divide into 4 sub-problems of equal complexity N/3

b. divide into 3 sub-problems of equal complexity N/4

c. divide into 4 sub-problems of equal complexity N/4

d. divide into 3 sub-problems of equal complexity N/3

(10) The fastest algorithm for constructing an optimal binary search tree is the c . algorithm. (2 points)

a. greedy b. divide and conquer c. dynamic programming d. backtracking

II. Given the function descriptions of the following two (pseudo-code) programs, please fill in the blank lines. (15 points) 每个 3 分

(1) The function is to merge two equal-sized binomial trees T1 and T2. (6 points)

```
BinTree  CombineTrees( BinTree T1, BinTree T2 )
{
    if ( T1->Element > T2->Element )
        return CombineTrees( T2, T1 );
    T2->NextSibling = ① T1->LeftChild;
    ② T1->LeftChild = T2;
    return T1;
}
```

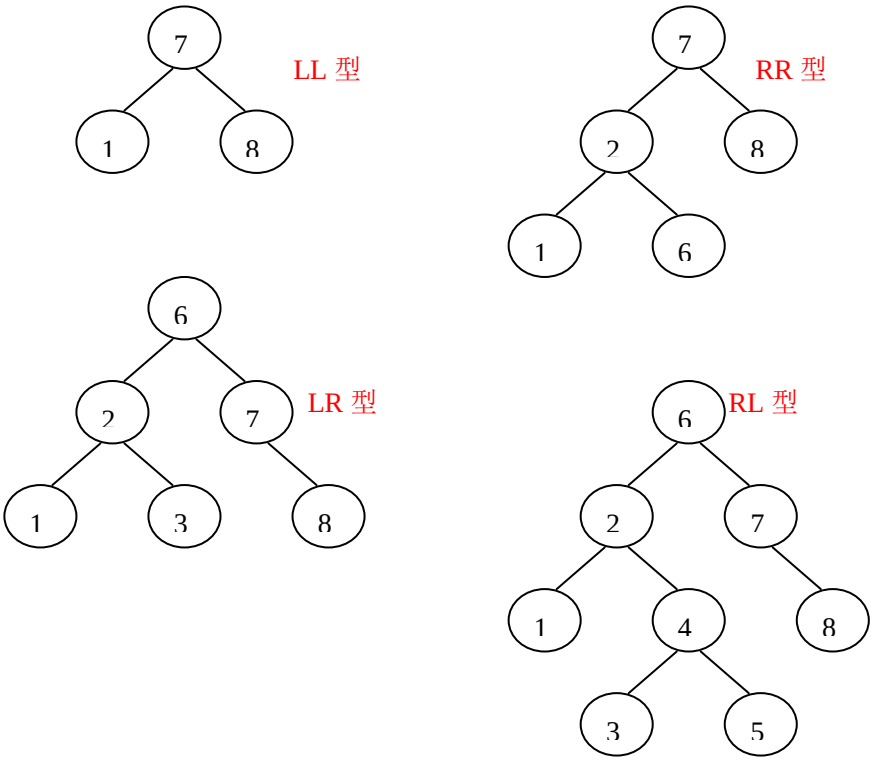
注意：不叫 LeftChild 也算对。

(2) The function is to find the shortest path between all pairs of vertices v_i and v_j. A[ ] contains the adjacency matrix with A[i][i] = 0, D[ ] contains the values of the shortest path, Path[ ] keeps the record of the shortest path, and N is the number of vertices. (9 points)

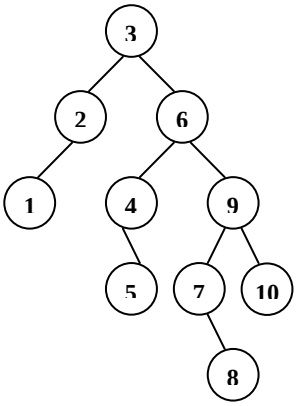
```
void AllPairs( TwoDimArray A, TwoDimArray D, TwoDimArray Path, int N )
{   int  i, j, k;
    for ( i = 0; i < N; i++ )
        for( j = 0; j < N; j++ ) {
          D[ i ][ j ] = A[ i ][ j ];
          Path[ i ][ j ] = -1;
        }
    for( k = 0; k < N; k++ )
        for( i = 0; i < N; i++ )
          for( j = 0; j < N; j++ )
             if( ① D[ i ][ k ] + D[ k ][ j ] < D[ i ][ j ] ) {
                  D[ i ][ j ] = ② D[ i ][ k ] + D[ k ][ j ];
                  ③ Path[ i ][ j ] = k;
              }
}
```

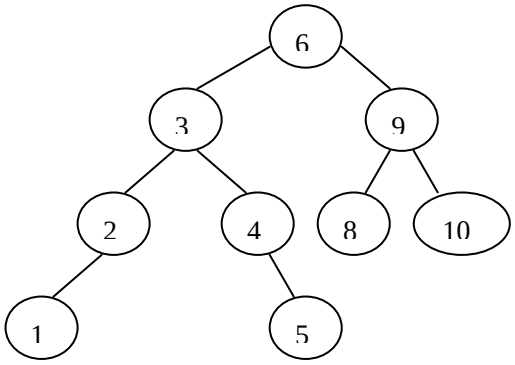
III. Please write or draw your answers for the following problems on the answer sheet. (46 points)

(1) Please draw the results of *rotations* when inserting {8, 7, 1, 2, 6, 3, 5, 4} into an initially empty AVL tree. Please specify the type of each rotation (LL, LR, RL, or RR) (8 points)



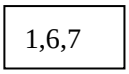
(2) Please draw the result of deleting 7 from the given splay tree. (5 points)

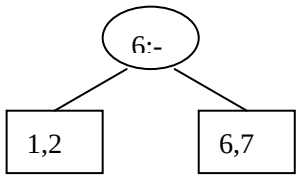


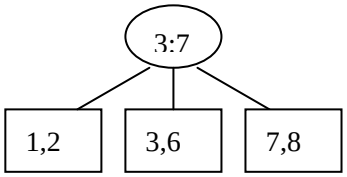


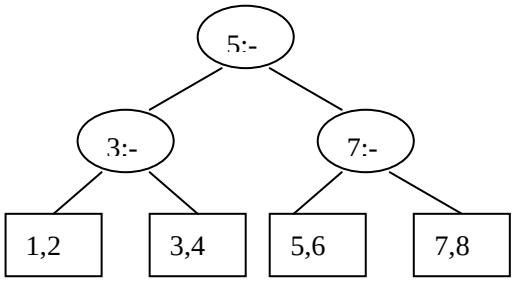
(3) Insert the keys: 6, 7, 1, 2, 8, 3, 5, 4 into an initially empty 2-3 tree. Please draw the resulting tree after each splitting (6 points) and the tree after deleting 8 (2 points).

a.

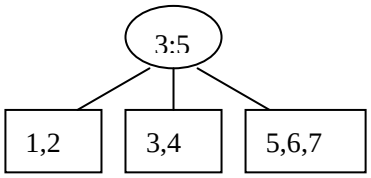




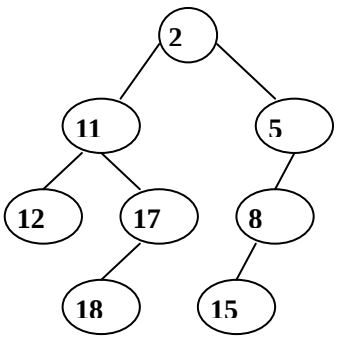


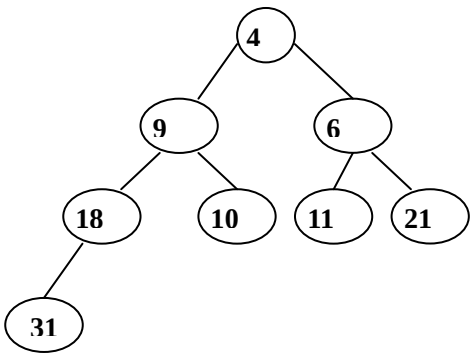


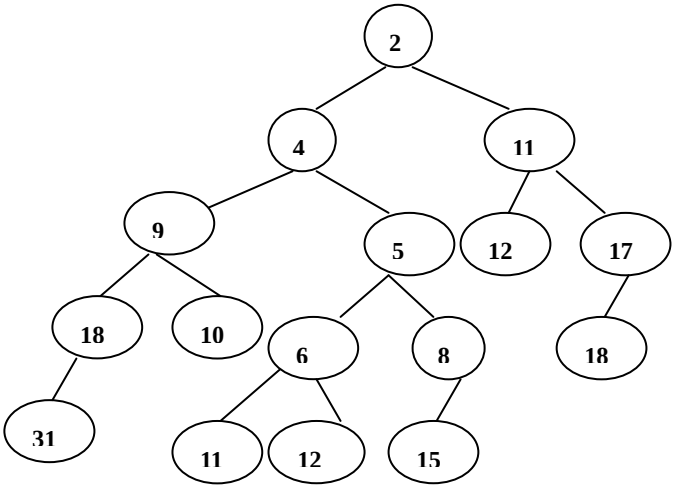
b.



(4) Please draw the result of merging the two given leftist heaps. (7 points)







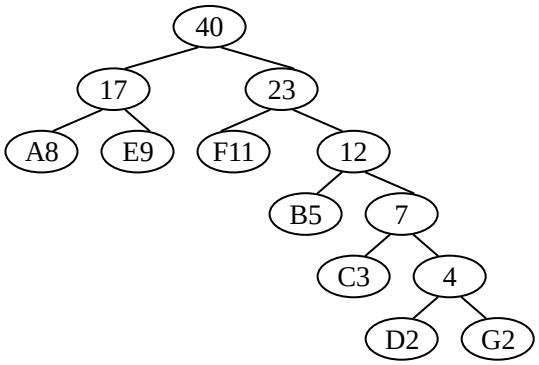
(5) Given a text containing 7 characters in the following frequencies:

character	A	B	C	D	E	F	G
frequency	8	5	3	2	9	11	2

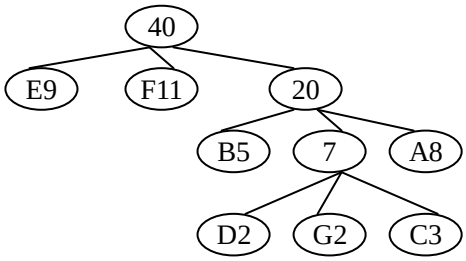
a) If the text should be encoded in binary (0 or 1), what is the minimum length of the encoded text? Please draw the corresponding Huffman tree. (4 points)

b) If the text should be encoded in ternary (三进制，0 、1、2), what is the minimum length of the encoded text? Please draw the corresponding Huffman tree. (4 points)

a) minimum length :103

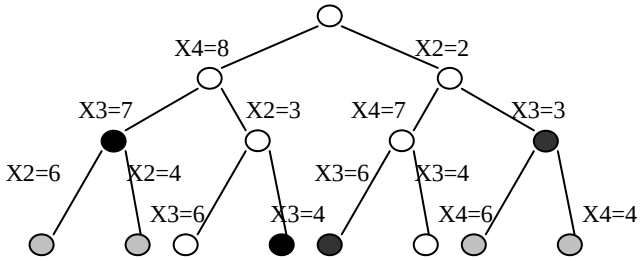


b) minimum length :67



(6) The turnpike reconstruction problem is to reconstruct a point set from distances between every pair of points. Given a set of distances {2, 2, 3, 3, 4, 5, 6, 7, 8, 10}, there are 5 corresponding points. Assume that X1 is at 0 and X5 is at 10, please mark the visited nodes black in the given search tree with backtracking method, and give all the solution sets (notice that the solution is not unique). (10 points)

a. 0, 3, 6, 8, 10
b. 0, 2, 4, 7, 10



IV. The genes for building particular proteins evolve with time, but the functional regions must remain consistent in order to work correctly. By finding the longest common subsequence of the same gene in different species, we learn what has been conserved over time. Given two strings of genes with lengths M and N, please write an algorithm to find the longest common subsequence with time complexity no more than O(M*N). (15 points)

Let L[i][j] be the string of the longest common subsequence of $A_i = \{ a_1, \ldots, a_i \}$ and $B_j = \{ b_1, \ldots, b_j \}$, then we have the following recurrence equations:

$$L[i][j] = 0 \quad \text{if } i = 0 \text{ or } j = 0 \qquad (1)$$
$$= L[i-1][j-1] + a_i \quad \text{if } i, j > 0 \text{ and } a_i = b_j \qquad (2)$$
$$= \max\{ L[i-1][j], L[i][j-1] \} \quad \text{if } i, j > 0 \text{ and } a_i \neq b_j \qquad (3)$$

```
int   LCSLength( ElementType A[ ],   int M,   ElementType B[ ],   int N )
{
    TwoDimArray L[M+1][N+1];
    int   i, j;   /* index of A and B */

    for ( i = 1; i <= M; i ++ )   L[ i ][ 0 ] =  空;
    for ( j = 1; j <= N; j ++ )   L[ 0 ][ j ] =  空;

    for ( i = 1; i <= M; i ++ )
        for ( j = 1; j <= N; j ++ )
            if ( A[ i ] == B[ j ] )
                L[ i ][ j ] = L[ i – 1 ][ j – 1 ] + A[ i ]; /* equation 2 */
            else
                L[ i ][ j ] = Max( L[ i – 1 ][ j ], L[ i ][ j – 1 ] ); /* equation 3 */

    return   L[M][N];
}
```