國立清華大學 104 學年度碩士班考試入學試題

系所班組別：資訊工程學系

考試科目（代碼）：基礎計算機科學（2001）

 *請在【答案卷、卡】作答

1. (5%) Among all n-digit numbers, how many of them contain the digits 2 and 7 but not the digits 0, 8, 9?

2. (5%) In how many ways can $2n$ people be divided into n pairs?

3. (7%) Let R be a transitive and reflexive relation on A. Let T be a relation on A such that (a, b) is in T if and only if both (a, b) and (b, a) are in R. Show that T is an equivalence relation.

4. (8%) Given a recursive definition of a_n: $a_1 = 1$; $a_{k+1} = 3a_k + 1$ for $k \geq 1$, please derive a close-form formula for a_n, and then prove that your formula is correct. (Hint: Write down the first six numbers (a_1, a_2, a_3, a_4, a_5, a_6) and guess the formula).

5. (8%) Prove by induction that 3^{2n}-1 is divisible by 8 for all $n \geq 1$.

6. (10%) Answer the following questions about binary trees.

   (a) (4%) Given an initially empty *min heap* H, draw the min heap after the following operations: insert 34, insert 12, insert 28, delete-min, insert 9, insert 30, insert 15, and insert 5.

   (b) (3%) Treat H as a priority queue where a key with a smaller value is of higher priority. Draw H after popping three keys out of it.

   (c) (3%) Insert the three keys popped out from H in question (b) into an initially empty binary search tree T, and then insert three other keys 45, 3, and 12. Draw T after completing these operations.

7. (6%) Answer the following questions about triangular matrix.

   (a) (3%) In a lower triangular matrix, A, with n rows, what's the total number of nonzero terms?

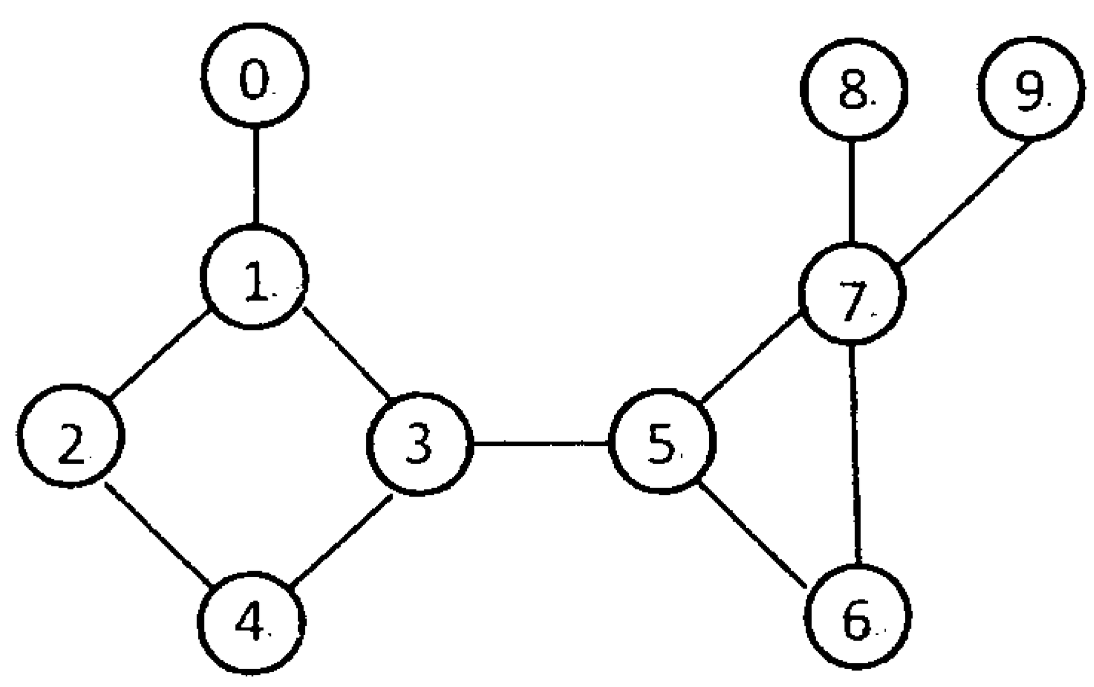
   (b) (3%) Since storing a triangular matrix as a two dimensional array wastes space, we would like to find a way to store only the nonzero terms of the triangular matrix in a one dimensional array. Find the index of $A_{i,j}$ in a one dimensional array $\boldsymbol{b}$ if we store $A_{1,1}$ at $b[0]$.

8. (8%) Consider the following graph *G* represented by an adjacency list. Assume that dfn[3] = 5 and dfn[4] = 6 after we invoke the function *dfnlow* (as shown below) with the call *dfnlow*(5, -1) being executed. Then, after the function *dfnlow*(5, -1) is invoked,

   (a) (2%) what is the value of dfn[1]?

   (b) (3%) what is the value of low[1]?

   (c) (3%) what is the value of low[2]?



The C declarations for adjacency list representation and function *dfnlow*():

```
#define MIN2 (x, y) ((x) < (y) ? (x) : (y))
#define MAX_VERTICES 100
typedef struct node *node_pointer;
typedef struct node {
        int vetex;
        struct node *link;
        };
node_pointer graph[MAX_VERTICES];
int dfn[MAX_VERTICES], low[MAX_VERTICES];
int num;

void dfnlow (int u, int v)
{
/* v is the parent of u (if any). It is assumed that all entries of all dfn[] and
low[] have been initialized to -1 and num has been initialized to 0. */
  node_pointer ptr;
  int w;
  dfn[u] = low[u] = num++;
  for (ptr = graph[u]; ptr; ptr = ptr->link) {
    w = ptr->vertex;
```

```
        if (dfn[w] < 0) {    /* w is an unvisited vertex */
           dfnlow(w, u);
           low[u] = MIN2(low[u], low[w]);
        }
        else if (w != v)
           low[u] = MIN2(low[u], dfn[w]);
     }
  }
```

9. (9%) In heap sort, functions *adjust*, *heapInitialization*, and *heapSort* (as shown below) are used. The function *adjust* starts with a binary tree whose left and right subtrees are max heaps and rearranges records so that the entire binary tree is a max heap, and the functions *heapInitialization* and *heapSort* use a series of *adjusts* to initialize the heap and perform a heap sort on a[1: n], respectively. Please analyze the complexities of:

   (a) (3%) *adjust*.

   (b) (3%) *heapInitialization*.

   (c) (3%) *heapSort*.

The C declarations for heap and functions *adjust*(), *heapInitialization*(), and *heapSort*():

```
#define SWAP (x, y, t) ((t) = (x), (x) = (y), (y) = (t))
typedef struct {
          int key
          } element;

void adjust (element a[], int root, int n)
{
   int child, rootkey;
   element temp;
   temp = a[root];
   rootkey = a[root].key;
   child = 2 * root; /*left child */
   while (child <= n) {
      if ((child < n) && (a[child].key < a[child+1].key))
         child++;
      if (rootkey > a[child].key) /* compare root and max. child */
         break;
      else {
         a[child / 2] = a[child]; /* move to parent */
```

```
        child *= 2;
      }
    }
    a[child / 2] = temp;
}

void heapInitialization (element a[], int n)
{
    int i;
    element temp;

    for (i = n/2; i > 0; i--)
      adjust (a, i, n);
}

void heapSort (element a[], int n)
{
    int i;
    element temp;

    for (i = n-1; i > 0; i--) {
      SWAP (a[1], a[i+1], temp);
      adjust (a, 1, i);
    }
}
```

10. (6%) Let n be an integer, and S be a set of integers, with range from 1 to n^2. It is known that S has at least $\sqrt{n}$ items. Explain in details how to sort S in $O(|S|)$ time.

11. (4%)
    (a). (2%) Explain why it takes at least 4 comparisons, in the worst case, to sort four distinct numbers.
    (b). (2%) Show how to sort four distinct numbers with at most 4 comparisons.

12. (7%)
    (a). (5%) Let S be a set of n positive integers, and we are interested if we can select some of the integers from S so that their sum is exactly m. Explain in details how this can be done in $O(nm)$ time.

(b). (2%) The above problem is called a subset sum problem, which is NP-hard. So far, no polynomial-time algorithms are known to solve an NP-hard problem. Explain why an O(nm)-time algorithm is not considered as a polynomial-time algorithm for the subset sum problem.

13. (6%) Testing gifted or mediocre: m students take an exam which has n questions. Gifted students get all n answers right. Mediocre students get less than $n/2$ answers right. Grade all the exams, giving all gifted students an 'A' and all mediocre students a 'C'.

Algorithm 1:

1. For each student, grade at most the first $n/2$ questions in order – stop as soon as you see a wrong answer.
2. If you've seen a wrong answer, give grade 'C'. Otherwise give grade 'A'.

Algorithm 2:

1. For each student, choose 10 questions at random and grade them.
2. If you've seen a wrong answer, give grade 'C'. Otherwise give grade 'A'.

Algorithm 3:

1. For each student, repeatedly choose a question at random and grade it, until you have graded $n/2$ correct answers or seen a wrong answer.
2. If you've seen a wrong answer, give grade 'C'. Otherwise give grade 'A'.

Explain the correctness and the running time of these three algorithms.

	Algorithm 1	Algorithm 2	Algorithm 3
Correctness	(a)	(b)	(c)
Running time	(d)	(e)	(f)

14. (6%)

(a). (3%) What is an optimal Huffman code for the set of frequencies, {1, 1, 2, 3, 5, 8}, based on the first six Fibonacci numbers?

(b). (3%) Generalize your answer to find the optimal code when the frequencies are the first n Fibonacci numbers.

15. (5%) The constrained 1-center problem: Given n planar points and a straight line L, find a smallest circle, whose center is restricted to lying on L, to cover these n points. The following lists an algorithm for solving this problem. Evaluate the time complexity, T(n), of this algorithm using the recurrence relation of T(n).

Input: n points and a straight line L: $y = y'$.

Output: The constrained 1-center on L.

Step 1. If n is no more than 2, solve this problem by a brute-force method.

Step 2. Form disjoint pairs of points (p_1, p_2), (p_3, p_4), …, (p_{n-1}, p_n). If n is odd, let the final pair be (p_n, p_1).

Step 3. For each pair of points, (p_i, p_{i+1}), find the point $x_{i,i+1}$ on L such that $d(p_i, x_{i,i+1}) = d(p_{i+1}, x_{i,i+1})$.

Step 4. Find the median of the $\lceil n/2 \rceil$ numbers of $x_{i,i+1}$'s. Denote it as x_m.

Step 5. Calculate the distance between p_i and x_m for all i. Let p_j be the point which is the farthest from x_m. Let x_j denote the projection of p_j onto L. If x_j is to the left (right) of x_m, then the optimal solution, x^*, must be to the left (right) of x_m.

Step 6. If $x^* < x_m$, for each $x_{i,i+1} > x_m$, prune the point p_i if p_i is closer to x_m than p_{i+1}; otherwise prune the point p_{i+1}. If $x^* > x_m$, for each $x_{i,i+1} < x_m$, prune the point p_i if p_i is closer to x_m than p_{i+1}; otherwise prune the point p_{i+1}.

Step 7. Go to Step 1.
