國立中央大學99學年度碩士班考試入學試題卷

所別：資訊工程學系碩士班 不分組(一般生)　科目：資料結構與演算法　共 二 頁 第 一 頁

軟體工程研究所碩士班 不分組(一般生)

＊請在試卷答案卷（卡）內作答

＊本科考試禁用計算器

1. A directed graph consists of three types of vertices and one type of edge. The vertices are classified as Blue vertices, Red vertices, and Green vertices. The class for a vertex is defined as follows:

```
Class Vertex {
  Integer vertexID;   // the unique id to identify a vertex
  Color vertexType;   // could be Blue, Red, or Green

  Vertex(Integer ID, Color Type) {...}      // the constructor
  Integer getID() {return vertexID;}
  Color getVertexType() {return vertexType;}
  Void setVertexType(Color newType) {vertexType=newType;}
}
```

The programmers may use three member functions of Graph *G* as follows:

- ***Boolean addVertex(Vertex v)*: The function adds Vertex *v* in G. If *v* exists, return *false*; otherwise add *v* and return *true*;**
- ***Boolean addEdge(Vertex source, Vertex destination)*: The function adds an edge *e* from Vertex *source* to Vertex *destination* in *G*. If *e* exists, return *false*; otherwise add *e* and return *true*;**
- ***Void evolve()*:**

  Step 1: For each edge <*x*,*y*> in *G*,

  Case 1: If *x* is Red and *y* has any Color, then create a new edge <*y*,*x*> in *G* and make *y* Red

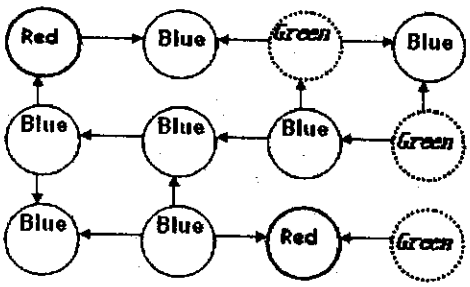
  Case 2: If *x* is Green and *y* is Green or Blue, then create a new edge <*y*,*x*> in *G* and make *y* Green

  Case 3: If *x* is Green and *y* is Red, then create a new edge <*y*,*x*> in *G*

  Step 2: Repeat Step 1 until nothing changes in *G*

1.1 Please write down the pseudocode of Class *Graph* **using either *Array* or *Vector***. (15%)

1.2 Given the following Graph *Ga*, What is **the result graph of *Ga*** after *Ga.evolve()* has been executed? (5%)



2. Let the function ***Integer size(character c, List x)*** return the total number of character c in List x. Given a list L consisting of '1', '2', and '3', **use Stack to design an algorithm** which checks if the following equation is true. (5%)

   5 * size('1', L) = 2* size('2', L) + 3* size('3',L)

3. Given the postorder traversal as ACEGFDBIH and the inorder traversal as ABCDEFGHI for a binary tree, please answer the following questions.

3.1 Can we uniquely identify the binary tree? Draw all possible binary trees with such postorder and inorder traversals. (8%)

3.2 Please also give the preorder traversal of those trees in 3.1. (2%)

4. Recursion vs. Iteration

4.1 Fill in the following code to make a recursive function which computes Fibonacci numbers. (4%)

```
unsigned long Fib(int n)
{
   if (n <= 2)
 else _____________________
    _____________________
}
```

4.2 The following code is the iterative equivalent which computes Fibonacci numbers. Which of the Fibonacci functions runs faster? (2%) Briefly explain why. (3%)

參考用

注意：背面有試題

```
unsigned long Fib(int n)
{
  unsigned long f0, f1, temp;
  if (n <= 2)
    return 1;
  else {
    for (f0 = f1 = 1, i = 3; i <= n; i++) {
      temp = f0 + f1;
      f0 = f1;
      f1 = temp;
    }
    return f1;
  }
}
```

4.3 Give **two advantages** and **two disadvantages** of recursion. (6%)

5. Given three sets of integers X, Y, and Z, and another integer k, we want to know if there exist three numbers x, y, and z, such that $x \in X$, $y \in Y$, $z \in Z$, and $x+y+z=k$. Design an algorithm to solve this problem. A naïve method by checking all possible sums of triples $(x+y+z)$ will take $\Theta(|X|\times|Y|\times|Z|)$ time. Your algorithm must be more efficient than it. Analyze the time complexity of your algorithm. (15%)

6. Given a weighted tree T with n nodes, we want to solve the all-pairs-shortest-distance problem on T. That is, we want to compute the distance between any two vertices on T. We may apply Floyd-Warshall algorithm or Dijkstra's algorithm n times to solve this problem and we need $O(n^3)$ or $O(n^2 \log n)$ time to solve this problem. As a matter of fact, the problem can be solved in $O(n^2)$ time. And such an algorithm is optimal since $O(n^2)$ time is needed for giving the output. Describe such an optimal algorithm by giving a pseudo-code. (Hint: For any two vertices u and v of T, there is one and only one path from u to v.) (10%)

7. Below is the algorithm LCS_LENGTH to find the length of the longest common subsequence of two sequences $X = \langle x_1, x_2, \ldots, x_m \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$ of m and n symbols, respectively. Note that a subsequence of X is derived by deleting 0 or more symbols (not necessarily consecutive) from X. Design an algorithm by taking LCS_LENGTH as a subroutine call to figure out the length of the longest non-decreasing subsequence of a given sequence $Z = \langle z_1, z_2, \ldots, z_k \rangle$ with the assumption that symbols have a total ordering. For example, the algorithm should return 6 for the input sequence Z=<1, 2, 2, 4, 1, 2, 3, 8>, whose longest non-decreasing subsequence is <1, 2, 2, 2, 3, 8> of length 6. (13%)

```
LCS_LENGTH(X, Y)
1  m ← length[X]
2  n ← length[Y]
3  for i ← 1 to m
4     do c[i, 0] ← 0
5  for j ← 1 to n
6     do c[0, j] ← 0
7  for i ← 1 to m do
8     for j ← 1 to n do
9       if x_i = y_j
10          then  c[i, j] ← c[i-1, j-1]+1
11          else if c[i-1, j] ≥ c[i, j-1]
12                  then c[i, j] ← c[i-1, j]
13                  else c[i, j] ← c[i, j-1]
14 return c[m, n]
```

8. Below is Dijkstra's algorithm for finding the costs of the shortest paths going from a given source node v_0 to all the other nodes in a given directed, weighted graph $G=(V, E)$, where each edge (u, v) is associated with a cost (or weight) $c(u, v)$. When some edges of the graph are negatively weighted, Dijkstra's algorithm may fail to return correct results. Show an example of a directed, weighted graph that will lead to such a failure. You should explain the reason why Dijkstra's algorithm fails for the graph. (12%)

```
Dijkstra (G, v0)
1  S ← {v0}
2  d[v0] ← 0
3  for each v ∈ V-{v0} do
4     if (v0, v) ∈ E
5        then d[v] ← c(v0, v)
6        else d[v] ← ∞
7  while S≠V do
8     choose u from V-S such that d[u] is minimum
9     add u into S
10    for each w ∈ V-S do
11       d[w] ← min (d[w], d[u]+c(u, w))
12 return d
```


注意：背面有試題