請按題目順序作答。

1. (20 points) Give asymptotically upper (big O) bounds for $T(n)$ in each of the following recurrences, where we assume that $T(n)$ is constant for small n. Make your bounds as tight as possible and **justify** your answers.
   (a) (5 points) $T(n) = T(n/3) + 1$
   (b) (5 points) $T(n) = 3T(n/3) + 1$
   (c) (5 points) $T(n) = T(n/10) + T(9n/10) + n$
   (d) (5 points) $T(n) = 2T(\sqrt{n}) + \lg n$

2. (15 points) Suppose that we have numbers between 1 and 1000 in a binary search tree and want to search for the number 400. Determine whether the following sequences could be or could not be the sequence of nodes examined and explain why or why not.
   (a) (5 points) 32 500 300 350 400
   (b) (5 points) 900 800 200 699 355 385 412 372 400
   (c) (5 points) 515 1 500 200 480 251 320 362 453 380 400

3. (15 points) A subsequence of a sequence S is obtained by deleting zero or more elements from S. Given two sequences, the longest common subsequence (LCS) problem is to find a subsequence that is common to both sequences and its length is maximized. Complete the following pseudocode for computing the length of an LCS and for delivering an LCS by filling [3A], [3B] and [3C].

Algorithm LCS_LENGTH($A = \langle a_1, a_2, \ldots, a_m \rangle$, $B = \langle b_1, b_2, \ldots, b_n \rangle$)
begin
  for $i \leftarrow 0$ **to** m **do** $len[i, 0] \leftarrow 0$
  for $j \leftarrow 1$ **to** n **do** $len[0, j] \leftarrow 0$
  for $i \leftarrow 1$ **to** m **do**
    for $j \leftarrow 1$ **to** n **do**
      if $a_i = b_j$ **then**
        $len[i, j] \leftarrow$ [3A]
        $prev[i, j] \leftarrow$ "↖"
      else if $len[i-1, j] \geq len[i, j-1]$ **then**
        $len[i, j] \leftarrow len[i-1, j]$
        $prev[i, j] \leftarrow$ "↑"
      else
        $len[i, j] \leftarrow len[i, j-1]$
        $prev[i, j] \leftarrow$ "←"
  return *len* and *prev*
end

Algorithm LCS_OUTPUT($A = \langle a_1, a_2, \ldots, a_m \rangle$, *prev*, i, j)
begin
  if $i = 0$ or $j = 0$ **then return**
  if $prev[i, j] =$ "↖" **then**
    LCS_OUTPUT([3B])
    print [3C]
  else if $prev[i, j] =$ "↑" **then** LCS_OUTPUT(A, *prev*, $i - 1$, j)
  else LCS_OUTPUT(A, *prev*, i, $j - 1$)
end

見背面

請遵守下列規定回答以下問題，違者該題不計分。

- 不要書寫任何程式碼或代碼。請以文字，圖表，數學符號陳述你要表達的概念。
- 一個問題包括兩個子題。一個子題的回答以一頁為限，並請按順序寫在完整的一頁。請先想好再動筆。如果字寫太小或是太潦草或是英文文法問題導致閱卷者無法理解你的想法，該題不計分。
- 一個問題包括兩個子題。前一個子題是觀念的陳述，佔 10 分，後一個子題是觀念的證明，佔 15 分。證明部分有倒扣，答對得 15 分，答錯倒扣 10 分，不答得 0 分。知之為知之，不知為不知，是知也。瞎掰不能解決問題，珍惜分數，遠離倒扣。

4. (25 points) We are given a network of C cities and R roads. All roads are one-way, i.e., if a road goes from city u to city v then there will be no roads going from v to u. In addition, the roads will not form any *cycle*. That is, you cannot start from a city v, go through several cities, and go back to v. Finally we assume that there is a capital that can go to all other cities in the network.

   We want to compute the *minimum* number of days for a tourist to go from the capital to each non-capital city. When a tourist encounters a city v, he will spend $c(v) > 0$ days to visit all attractions in v. When a tourist goes from city u to city v, he will spend $r(u, v) > 0$ days on the road. Now given the c function values for all cities, and r function values for all roads, please compute the *minimum* number of days for a tourist to go from the capital to each non-capital city. We use $M(v)$ to denote this minimum number of days from the capital to a city v.

   We illustrate the problem with an example of four cities A, B, C, D and four roads. Let A be the capital and $c(A) = c(B) = c(C) = c(D) = 1$, and $r(A, B) = r(A, C) = r(B, D) = 2$, and $r(C, D) = 10$. Then $M(D) = c(A) + r(A, B) + c(B) + r(B, D) + c(D) = 7$. Similarly $M(B) = M(C) = 4$.

   (a) (10 points) Please derive an *optimal* algorithm to compute the M function for all cities other than the capital.

   (b) (15 points) Please analyze the time complexity of your algorithm and prove that it is optimal.

5. (25 points) We are given a set of C classes and R classrooms and we would like to assign as many classes to classrooms as possible. Due to equipment constraint not every class can be assigned to every classroom. Instead we will be given a set M of pairs (c, r), and each pair indicates that class c can be assigned to classroom r. In addition, a class, if assigned, will be assigned to a *unique* classroom. Therefore it is possible that we cannot assign every class to a unique classroom. Our goal is to assign as many classes to classrooms as possible.

   We consider the following example. Let a lower case letter indicate a class and a positive integer indicate the index of a classroom. Let $M = \{(a, 1), (a, 2), (b, 2)\}$, which means class a can be assigned to classroom 1 and 2, but class b can only be assigned to classroom 2. Now given the set M, please find the *maximum* number of classes that can be assigned to classrooms. In the previous example the answer is 2 since we can assign class a to classroom 1 and b to classroom 2.

   We can use Ford-Fulkerson method to solve this problem. Basically the Ford-Fulkerson method initializes the flow to 0, then repeatedly finds an augmenting path p, and adjusts the capacity along the path found on the residual network, until no augmenting path can be found.

   (a) (10 points) Now assume that Ford-Fulkerson method can find the maximum flow in a network, how do we convert our problem into a network flow problem so that we can use Ford-Fulkerson method to solve it?

   (b) (15 points) Prove that Ford-Fulkerson method can correctly solve our problem by your conversion. To be more specific, is it true that *any* maximum flow will give you a correct answer for the classroom problem? Also is it true that the answer from the Ford-Fulkerson method will give you a correct answer in our classroom problem?

試題隨卷繳回