Data Structure and Algorithm

Homework #5

Due: 2:00pm, Thursday, May 31, 2012

TA email: dsa1@csie.ntu.edu.tw

=== Homework submission instructions ===

- For Problem 1, submit your source code, a shell script to compile the source, and a brief documentation to the SVN server (katrina.csie.ntu.edu.tw). You should create a new folder "hw5" and put these three files in it.
- The filenames of the source code, the shell script, and the documentation file should be "arbitrage.c", "compile.sh", and "report.txt", respectively. **The shell script should compile your source codes to generate an executable binary with the filename 'arbitrage'**. You will get some penalties in your grade if your submission do not follow the naming rule.
- The documentation file should be in plain text format (.txt file). In the documentation file you should explain how your code works, and anything you would like to convey to the TAs.
- For Problem 2 to Problem 4, submit the answers through the CEIBA system (electronic copy) or to the TA in R508 (hard copy).
- Each student may only choose to submit the homework in one way; either all as hard copies or all through CEIBA except the programming assignment. If you submit your homework partially in one way and partially in the other way, you might only get the score of the part submitted as hard copies or the part submitted through CEIBA (the part that the TA chooses).
- If you choose to submit the answers of the writing problems through CEIBA, please combine the answers of all writing problems into only ONE file in the doc/docx or pdf format; otherwise, you might only get the score of one of the files (the one that the TA chooses). In addition, **do NOT forget to specify your name and student ID** in your submitted file.
- Discussions with others are encouraged. However, you should write down your solutions by your own words. In addition, you have to cite the sources you consulted from or people you discussed with on the first page of your solution to that problem. The TA can deduct up to 100% of the score for any problem without the citation.
- No late submission of the homework will be given any score (for that portion).

Problem 1. (30%) **Arbitrage** is the use of discrepancies in currency exchange rates to transform one unit of a currency into more than one unit of the same currency. For example, suppose that

1 U.S. dollar buys 49 Indian rupees, 1 Indian rupee buys 2 Japanese yen, and 1 Japanese yen buys 0.0107 U.S. dollars. Then, by converting currencies, a trader can start with 1 U.S. dollar and buy $49 \times 2 \times 0.0107 = 1.0486$ U.S. dollars, thus turning a profit of 4.86 percent. Suppose that we are given n currencies $c_1, c_2, ..., c_n$ and an $n \times n$ table R of exchange rates, such that one unit of currency c_i buys $R[i, j]$ units of currency c_j. We would like to find a sequence of currencies $\langle c_{i_1}, c_{i_2}, ..., c_{i_k} \rangle$ such that $R[i_1, i_2] \cdot R[i_2, i_3] \cdots R[i_{k-1}, i_k] \cdot R[i_k, i_1]$ is more than 1.01, where $k \leq n$. If $\{R[i_1, i_2] \cdot R[i_2, i_3] \cdots R[i_{k-1}, i_k] \cdot R[i_k, i_1]\} > 1.01$, we call this set of currencies *profitable*. To result in successful arbitrage, a sequence of exchanges must begin and end with the same currency, but any starting currency may be considered.

Input:

The first line contain n, $2 \leq n \leq 50$. The next n lines represent the exchange rate table R in the following format:

$R[1,1]$ $R[1,2]$ $R[1,3] \ldots R[1,n]$

$R[2,1]$ $R[2,2]$ $R[2,3] \ldots R[2,n]$

...

$R[n,1]$ $R[n,2]$ $R[n,3] \ldots R[n,n]$

with all numbers ranging from 10^{-5} to 10^5.

Output:

For each input table you must determine whether a sequence of exchanges exists that results in a profit of more than 1 percent (0.01). If a sequence exists you must print the sequence of exchanges that results in a profit. If there is more than one sequence that results in a profit of more than 1 percent you must print a sequence of minimal length and minimal starting index, i.e., one of the sequences that uses the fewest exchanges of currencies to yield a profit. Furthermore, if there are more than one minimal length sequence that results in a profit of more than 1 percent such as 1,3,2,1 and 1,2,3,1 , we can print either one of the sequences. All profiting sequences must consist of n or fewer transactions where n is the dimension of the table giving exchange rates. You should output the profitable sequence in the following format:

$i_1, i_2, ..., i_k, i_1$

where $i_1, i_2, ..., i_k, i_1$ is the sequence of the indices of the currencies in the profitable exchange, where $i_1 \leq i_j$, for $j = 2, ..., k$. If no profitable sequence that results in a profit of more than 1 percent (0.01) can be found within n transactions, output "Not profitable". All numbers in this output line are separated by comma.

Sample Input 1:

```
3
1 0.6 0.8
1.6 1 1.2
```

```
1.2 0.7 1
```

Sample Output 1:

```
Not profitable
```

Sample Input 2:

```
3
1 0.05 0.21
16 1 4
5.1 0.09 1
```

Sample Output 2:

```
1,3,1
```

Please write a program to solve this problem. Please also submit a report in which you give a clear description of your code.

Problem 2. Shortest Path Algorithms (22%)

1. (6%) Use Dijkstra's Algorithm to determine the shortest-path distance from vertex **a** to all other vertices in Figure 1. Please create a table similar to the one on page 9 of the lecture note "graph 3" to show how the algorithm is executed.

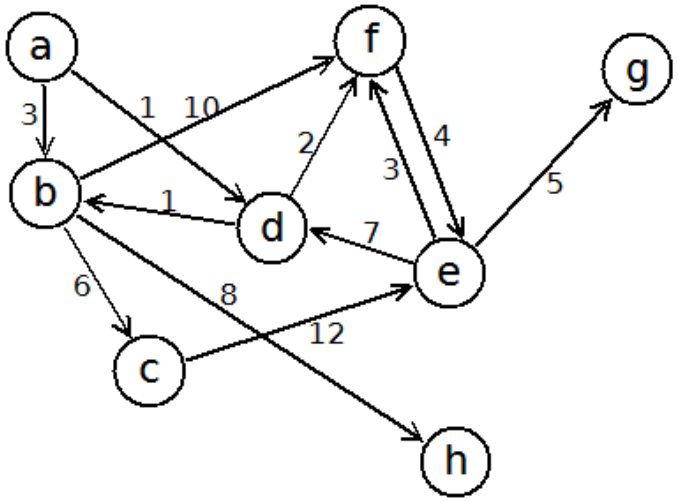

Figure 1: Determine the shortest-path distance by Dijkstra's Algorithm

2. (6%) Use Bellman-Ford Algorithm to determine the shortest-path distance from vertex **a** to all other vertices in Figure 2. Please create a table similar to the one on page 19 of the lecture note "graph 3" to show how the algorithm is executed.

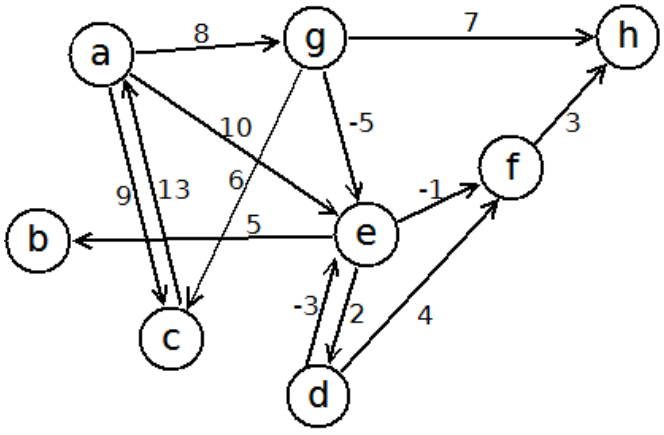

Figure 2: Determine the shortest-path distance by Bellman-Ford Algorithm

3. (5%) Explain why, in a directed graph G, we CANNOT use Dijkstra's Algorithm to find the longest simple path from one vertex to another. A simple path is a path in which each vertex appears at most once.

4. (5%) If we restrict the graph G in 3. to a Directed Acyclic Graph (DAG), can Dijkstras's Algorithm be applied to find the longest path? Why or why not?

Problem 3. More on Graph (16%)

1. (8%) Give a linear-time algorithm to examine whether an undirected and connected graph $G(V, E)$ contains any cycle. You do not have to list the vertices that the cycle is composed of. What is the time complexity of your algorithm? Note that if your algorithm is written in C code or pseudo code, you should explain it in detail.

2. (8%) Given a directed and weighted graph $G(V, E)$ with ONE negative cycle, describe an efficient algorithm to find the vertices in this cycle. Note that if your algorithm is written in C code or pseudo code, you should explain it in detail.

Problem 4. Sorting (32%)

1. (4%) Write down the sequence of the list $(5, 3, 2, 7, 8, 6, 1, 9, 4)$ at the end of each iteration when performing an insertion sort.

2. (4%) Use quick sort to sort the list $(5, 3, 2, 7, 8, 6, 1, 9, 4)$. Write down the sequence of the list after each iteration and mark the parts are processed in that iteration and are smaller and larger than the pivot, respectively. In this problem, you are asked to pick the left-most element as the pivot.

3. (4%) Give an example input with which quick sort may run in $O(n^2)$, where n is the number of numbers in the input list to be sorted .

4. (4%) Write down the sequence of the list $(5, 3, 2, 7, 8, 6, 1, 9, 4)$ at the end of each phase when performing a merge sort.

5. (4%) Insert the following list $(5, 3, 2, 7, 8, 6, 1, 9, 4)$ into an initially empty max-heap in the given order and then use heap sort to sort this list. You have to show the content of the max-heap after inserting all the elements as well as the content of the max-heap at the end of each extraction (after the maximum value of the max-heap is returned and deleted).

6. (4%) The following pseudo code is the algorithm of counting sort. A is a given list and k is the range of all the elements in A, for example, if $A = \{2, 12, 10\}$, $k = 12$. B is an initially empty list for holding the sorted elements of A. (X[i] indicates the i-th element of a list X.)

```
COUNTING-SORT(A, B, k)
for i = 0 to k
  do  C[i] = 0
for j = 1 to length[A]
  do  C[A[j]] = C[A[j]] + 1
// C[i] now contains the number of elements equal to i.
for i = 1 to k
  do  C[i] = C[i] + C[i - 1]
// C[i] now contains the number of elements less than or equal to i.
for j = length[A] downto 1
  do  B[C[A[j]]] = A[j]
      C[A[j]] = C[A[j]] - 1
```

   Use counting sort to sort the list $(5, 3, 2, 7, 8, 6, 1, 9, 4)$. You have to show the content of B and C at the end of each iteration of the fourth for loop.

7. (4%) Use radix sort to sort the following list $(501, 939, 1137, 2345, 666, 34, 218)$. Set the base $r = 10$ for radix sort and use the counting sort algorithm to sort the base elements of each pass of radix sort. Write down the sequence of the list at the end of each pass of radix sort.

8. (4%) Continuing the previous problem, please use n, k, and r to analyze the time complexity of radix sort with the case in which we only use counting sort to sort the given list $(501, 939, 1137, 2345, 666, 34, 218)$. Then compare the computational cost between these two algorithms for solving this problem.