17CS43-DAA 2nd Assignment (programming)
Last date of submission: Apr 29, 2019
Marks for assignments: 10 (5+5)

Note:

i. Your group id remains the same.
ii. You need to submit two questions.
iii. The question allocations to teams are as follows.
    a. G1 will do Q1 and Q7.
    b. G2 will do Q2, and Q8.
    c. G3 will do Q3 and Q9.
    d. G4 will do Q4 and Q10
    e. G5 will do Q5 and Q1.
    f. :
    g. :
    h. Essentially Group G#i will do Q(i %10), and Q(i+6%10). If the remainder comes to 0, this implies question number 10.
iv. There are 2 bonus marks each for doing extra questions, subject to max not exceeding total.
v. Change of assignment questions is allowed by Apr 24, 2019. The change in one assignment question will cost 2 marks. Max of two questions can be changed. On changing the question, you can not pick the question, but it will be assigned randomly.
vi. Assignment should be uploaded on server 202.62.79.36 in the lexicographically smallest USN of the group in the ha02 subdirectory of the user i.e. in the directory /home/<usn>/ha02. Submission should include the following:
    a. README file providing group details, approach taken to solve the problem, challenges faced, how did you approach to address the challenges and contribution of each team member. If you taken help from any external source e.g. geekforgeeks.com, stackexchange.com etc, provide those explicit URLs. Any plagiarism will result in 0 marks. It should also contains instructions to run the program
    b. Source code and input test data files which you used to test the data.
    c. OUTPUT.txt containing the results of your test data.
vii. Please ensure your program should not crash under any bad input or no input. Further, there should not be any hardcoding of data (i.e. magic constants) w.r.t. to your input data.

Q1. Divide and Conquer problem.

Detecting a counterfeit coin of a different weight using $\log_2 N$ comparisons. You are given a bag with $N>0$ coins and told that one of these coins is counterfeit. Further, you are told that counterfeit coin can be either lighter or heavier than genuine ones. Your task is to find whether bag contains a counterfeit coin. To aid you in the task, you have a machine that compares the weights of two sets of coins and tells you if they are of equal weight of which set is lighter than the other. For a coin create a property having one of 3 possible values of 0, 1, and 2. All the coins except 1 will have this value as 1. Choose a random coin among these N coins (and randomly

assign this value to either 0 or 2). The main code of the program does not read this property. There is a function available `compare()` which takes two inputs (two sets of coins of same count) and returns the result (write this function and use this function as library/API call in the main program. The main program does know how it is implemented) of comparison. Result=0 implies both sets have equal weights, result=-1 implies set1 is lighter than set2 and result=+1 implies that set 1 is of higher weight than set 2. Implement the sets as per your choice of your data structure e.g. lists, array, tree etc.
Test your program for a value of N between 100 and 1000 and count the number of times you invoke `compare()` method.
The input to your program is a value N.
The input to library function `compare()` is a file with 2 lines as below

Line 1: `N` # the number of coins

Line 2: `D, k` # the defective coin and its weight e.g. 11,0 implies 11th coin is defective and its coin weight is 0. If the value is 13,2, it implies that 13th coin is defective with higher weight. All other coins are of same weight, which is 1.

Output expected:

a) Defective coin number and

b) number of comparisons made.

Q2. Binary search problem (possibly in a reverse fashion).
Consider a larger array A of size `N` (e.g. `N=1000000` or even larger), of which first `M` cells are filled with certain positive integers in sorted order and the remaining array element from `M+1` to N are filled with `-1`. Initialize this array for some value of `M` taken randomly between `sqrt(N)` and `N` and thus fill this array up to `M` positions with random number in sorted order as follows

`A[1]` = random value between `1` and `sqrt(N)`

`A[2]` = random value between `A[1]` and N^2+2

:

`A[i]` = random value between `A[i-1]` and N^2+i

:

`A[M]` = random value between `A[M-1]` and N^2+M

Provide this array to your main assignment program which does not know the value of `M`, but knows the value of `N` only for the purpose of indexing the array (so that you do not array index out of bounds), other N has no significance.

Write a program that takes some input `X` and determines if `X` is present in the array in $O(\log_2 N)$ time. Use the variant of Binary search to determine and how many comparison operations you performed.

Thus, the program would have two inputs

Param1 : N (size of the array)

Param 2: X (value to be found in this array using $O(\log_2 N)$ time

Q3. MergeSort problem.

Consider a larger array A of size N (e.g. N=1000000), which contains unsorted positive integers. Once N is given, created an array of size N with random integer values. Use a variation of MergeSort where instead of dividing the input into two parts and then merge, divide the input into 3 parts and then merge. Compute the total number of sorting (compare) operations in this case and analyse its performance as compared to regular mergesort where the input is divided into 2 parts.

Input to the program: N # size of the array

Q4. Polynomial multiplication.

Given two polynomials, compute their product by using divide and conquer approach and evaluate the computed polynomial for a given value of x. The two polynomials are specified in an input file as follows

c_{11} e_{11} c_{12} e_{12} c_{13} e_{13} …. c_{1n} e_{1n}

c_{21} e_{21} c_{22} e_{22} c_{23} e_{23} …. c_{2m} e_{2m}

The values above for two polynomials should be interpreted as

$P(x) = c_{11}x\text{^}e_{11} + c_{12}x\text{^}e_{21} + \ldots + c_{1n}x\text{^}e_{1n}$, and

$Q(x) = c_{21}x\text{^}e_{21} + c_{22}x\text{^}e_{22} + \ldots + c_{2m}x\text{^}e_{2m}$

The command line arguments are as follows.

Arg 1: filename containing two polynomials

Arg 2: value of x (This is a real number).

Verify your computed answer by evaluating P(x), Q(x) and multiplying two evaluated values and compare it with the evaluated value of R(x)=P(x)*Q(x).

Q5. Divide and Conquer problem.

Computation of Fibonacci number. Consider the following matrix multiplication

$$[F_{n-1}\ \ F_n] = [F_{n-2}\ \ F_{n-1}] * \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

Where F_n is the 2^{nd} component of this matrix multiplication. Taking the value of $F_0=0$, and $F_1=1$, compute the value of F_n using Divide and Conquer approach in $O(\log_2 n)$ matrix multiplications. Please note this is much better compared to using general recursion approach which may take exponential time.

Input to the program: N

Output of the program: F_N # i.e. nth Fibonacci term, starting 0, 1, …

Q6. Machine Scheduling problem.

Consider that you are given N activities with their start time and finish time and K machines. Use the greedy algorithm to schedule maximum number of jobs on K machines. The input to the program is a file with following contents:

Line 1: K # number of machines available

Line 2: S_1, E_1 # both S1 and E1 are some positive integers.

Line 3: S_2, E_2

:

:

Line n: S_n, E_n

Implement the following two approaches.

i. Approach 1: First sort the jobs in non-decreasing order of Finish time and use the greedy approach to schedule a job with earliest finish time.

ii. Approach 2: Sort the jobs in non-increasing order of Finish time and use greedy approach to schedule the jobs with highest finish time.

Run your program on multiple such inputs and analyze which approach is better and which algorithm provides optimum result.

Input: A filename containing required data as explained above.

Q7. Knapsack Problem.

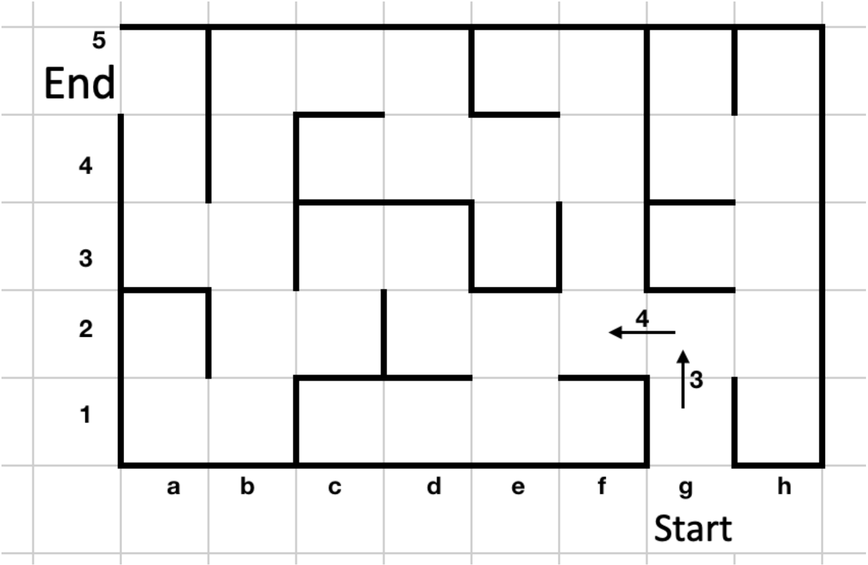
Consider that you are organizer of an event management company and need to conduct multiple events on a given day. You have N venues (V_1, V_2, …, V_n) available to you to conduct such events. At one venue, you can conduct only 1 event. For each venue, you know the number of people (P_1, P_2, …, P_n) that will come to that venue. For each event, you also know the participation fee (F_1, F_2,…, F_n). Your company management has given you a revenue target of M to collect while using minimum number of venues. You can exceed the target M but with minimum possible value while keeping the number of venues to be minimum. Write a program to assign which event should be conducted at which venue to reach the target, i.e. program should output the venue name and associated event. Also, identify if the target is not achievable, then justify why this target of collecting M money is not possible. The input to this program would be a file with following information.

Line 1: M # the target to be collected
Line 2: V_1, V_2, …, V_n # names of venues
Line 3: P_1, P_2,…, P_n # P_i is number of people that will come to the venue V_i
Line 4: E1, E2, …, Em # name of events.
Line 5: F_1, F_2,…, F_m # the participant fee for event E1, E2, …, Em.

Q8. Single Source Single destination shortest path problem

Consider the maze as shown below for an example. This maze is only as an example to explain and input data to program can be any maze.

You need to start from maze position g1(Start point) and reach maze position a5 (end point). In this example there are two possible paths to reach a5. The arrow marks from one maze position to another shows the cost of moving from one position to another. Each open position has some cost of movement associated with it. Two such examples are shown in the maze diagram. Cost from moving from g1 to g2 is 3, and cost of moving from g2 to f2 is 4. The solid line boundary between two positions indicates that one can not move between these two positions. For example, one can not moved from position c1 to c2. Similarly, from position c1, one can not move to position b1, but can move to position d1.



Using Dijkstra's algorithm, find the shortest cost path from start to end. Your program should output the path like

```
    g1, g2, f2, …
```

The input to the program is a file containing following information

Line 1: <start pos> <end pos>
Line 2: `g1,g2,3`   # cost of moving from g1 to g2
Line 3: `g2, f2, 4`
:
:
Line n: …

Hint: Represent the maze as a weighted graph.

Q9. Kruskal's algorithm.

Union Find problem using `O(1)` cost for Find and `O(log n)` cost for Union. Consider the maze as given in Q8. Your job is to find minimum cost path from "Start" to "End". Union operation will basically involve joining two sets if there is and edge from one set to another. You can use the built in function for sorting the edges. Input is same as in Q8.

Hint: Represent the maze as a weighted graph.

Q10. Kruskal's algorithm.

Union Find problem using `O(1)` cost for Union and `O(log n)` cost for Find. Consider the maze as given in Q8. Your job is to find minimum cost path from "Start" to "End". Union operation will basically involve joining two sets if there is and edge from one set to another. You can use the built in function for sorting the edges. Input is same as in Q8.

Hint: Represent the maze as a weighted graph.

← end of document →