Group Assignment 2

Prof. Glencora Borradaile

Due: Thursday, January 29 at 2PM

You are encouraged to work in groups of up to three students. Only one member of each group should submit the group's work to TEACH, including the **project report as a pdf** and the code that implements the algorithms. The report should have **all member's names included**. You may use any language you choose to implement your algorithms. **No questions about this assignment will be answered after the due date above.**[1]

Problem Description

The locker room of OSU's Dixon Recreation Center has N lockers that are labeled $1, 2, \ldots, N$. Each locker is locked, but be opened using its unique key. Copies of the key to each locker are in its adjacent lockers; i.e. a copy of the key to locker i is placed in locker $i+1$ and $i-1$ (the key to locker 1 is only in locker 2 and the key to locker N is only in locker $N-1$).

T tennis balls are inside T distinct lockers (and you know which of the lockers they are in). You are given keys to M of the lockers and your goal is to collect all of the tennis balls by opening the least number of lockers. And example input and solution is given below.

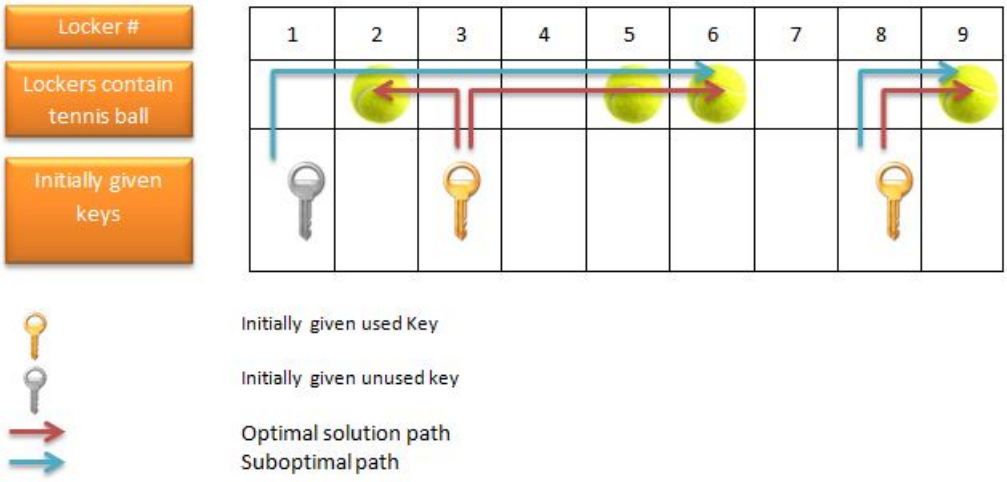

Figure 1: In one solution (blue), only the first and last key are used and to collect all of the tennis balls, lockers 1,2,3,4,5,6 and 8,9 are opened. This solution is not optimal. In an optimal solution (red), only the second and third key are used: the second key is used to open locker 3 which gives us access to the keys to locker 2 and 4. In this way, the optimal solution only opens lockers 2,3,4,5,6 and 8,9.

[1]The due date in TEACH is 24 hours late, as submissions submitted within 24 hours of the deadline above will not be penalized.

You will design, analyze and implement 2 (or 3) algorithms to solve instances of this problem correctly. You may use any language you choose to implement your algorithms. Be sure that your algorithm correctly finds the minimum number of lockers that can be opened to collect all of the tennis balls.

Testing for correctness Above all else, your algorithms should be correct. To aid in testing, a file containing test sets can be found here: `http://www.eecs.orst.edu/~glencora/cs325/dp.txt` The file has 8 test cases. Each test case starts with three integers N ($1 \leq N \leq 600$), M ($1 \leq M \leq N$), T ($1 \leq T \leq N$) that are separated by space where N is the number of lockers, M is the number of initially given keys and T is the number of tennis balls. The next line contains M numbers which represents the labels of the lockers whose keys you are given. The next line contains T numbers that are the labels of lockers that contain tennis balls. For each of these test cases, the minimum number of lockers that you can open to get all the tennis balls is also given. Sample input/output for the example in figure 1 is shown below:

Sample Input	**Sample Output**
9 3 4 1 3 8 2 5 6 9	7

You may use this test file to check that your code is correct. You should also test your code on other small hand-generated instances.

Instructions

Your three algorithms are to be based on the following ideas and should have the given *upper bounds* on their running times. Note that Algorithm 3 is for extra credit only. You can earn *full credit* without attempting Algorithm 3. For each algorithm, your report must include:

- pseudocode with sufficient explanation that we can understand your pseudocode
- an analysis (and explanation) of the running time – include *lower bounds* (Ω) on the running times as well
- solutions to the corresponding input instances

Algorithm 1: Enumeration (Brute-force algorithm) $O(N2^M)$

Hint: Guess which of the keys you are initially given you should use and then can determine (in $O(N)$ time) how best to use them.

The input instances for this algorithm are available here: `http://www.eecs.orst.edu/~glencora/cs325/dp_set1.txt`

Algorithm 2: Dynamic Programming $O(NM^2)$

Hint: Let $d[i]$ be the minimum number of lockers that must be opened between the first locker and the locker that key i opens assuming that key i is used. Start your development of a dynamic programming algorithm by figuring out how to compute $d[i]$ recursively and then compute $d[i]$ bottom up. Once you have computed $d[i]$ for all the keys you can choose from, how do you determine the final solution?

The input instances for this algorithm are available here: `http://www.eecs.orst.edu/~glencora/cs325/dp_set2.txt`

Algorithm 3 (Extra Credit): Optimal Dynamic Programming $O(T^2 + M)$

Hint: Can you efficiently decide whether there exist at least one tennis ball between i^{th} and j^{th} locker? (No questions will be answered for this algorithm, since it is extra credit.)

The input instances for this algorithm are available here: `http://www.eecs.orst.edu/~glencora/cs325/dp_set3.txt`