Algorithm Design & Analysis

Assignment-4 (Basic Algorithms & Divide-and-Conquer Strategy)

1. A boy stands in front of a flight of n stairs. In one jump, the boy can cover one, two or three steps. In how many ways can the boy cross all the steps? Call it $C(n)$.

For example, if $n = 4$, then all the possibilities for the boy are (1,1,1,1), (1,1,2), (1,2,1), (1,3), (2,1,1), (2,2) and (3,1). Therefore, $C(4) = 7$.

Part 1

Frame a recurrence relation for $C(n)$, and make a recursive implementation by writing a recursive function.

Part 2

Make an efficient (linear-time and constant-space in n) iterative implementation by writing a non-recursive function.

Part 3

Suppose you want to compute $C(n, m)$ which stands for the number of ways the boy can cross n steps in exactly m jumps. Derive a recurrence relation for $C(n, m)$, and write a recursive function for it.

Part 4

Make an efficient iterative function to compute $C(n, m)$. You are permitted to use only one local array of size $n + 1$, and some constant number of local variables.

The main() function

- Read n from the user. (Take n as larger than 37.)
- Run the function of Part 1 on n.
- Run the function of Part 2 on n.
- Run the function of Part 3 on n, m for all m in [0, n]. Report the sum of all these return values.
- Run the function of Part 4 on n, m for all m in [0, n]. Report the sum of all these return values.

Sample Output

```
n = 16
   +++ Any number of jumps...

      Recursive function returns count = 10609
      Iterative function returns count = 10609

   +++ Fixed number of jumps...

      Recursive function returns count =      0 for m =  0
      Recursive function returns count =      0 for m =  1
      Recursive function returns count =      0 for m =  2
      Recursive function returns count =      0 for m =  3
      Recursive function returns count =      0 for m =  4
      Recursive function returns count =      0 for m =  5
      Recursive function returns count =     21 for m =  6
      Recursive function returns count =    266 for m =  7
      Recursive function returns count =   1107 for m =  8
      Recursive function returns count =   2304 for m =  9
      Recursive function returns count =   2850 for m = 10
      Recursive function returns count =   2277 for m = 11
      Recursive function returns count =   1221 for m = 12
      Recursive function returns count =    442 for m = 13
      Recursive function returns count =    105 for m = 14
      Recursive function returns count =     15 for m = 15
      Recursive function returns count =      1 for m = 16
      ---------------------------------------------
      Total number of possibilities     =    10609

      Iterative function returns count =      0 for m =  0
      Iterative function returns count =      0 for m =  1
      Iterative function returns count =      0 for m =  2
      Iterative function returns count =      0 for m =  3
      Iterative function returns count =      0 for m =  4
      Iterative function returns count =      0 for m =  5
      Iterative function returns count =     21 for m =  6
      Iterative function returns count =    266 for m =  7
      Iterative function returns count =   1107 for m =  8
      Iterative function returns count =   2304 for m =  9
      Iterative function returns count =   2850 for m = 10
      Iterative function returns count =   2277 for m = 11
      Iterative function returns count =   1221 for m = 12
      Iterative function returns count =    442 for m = 13
      Iterative function returns count =    105 for m = 14
      Iterative function returns count =     15 for m = 15
      Iterative function returns count =      1 for m = 16
      ---------------------------------------------
      Total number of possibilities     =    10609
```

2. You are provided two queues, $Q1$ and $Q2$, and one stack S. You are allowed to dequeue from $Q1$ and to enqueue in $Q2$. You are allowed to both push into and pop from the stack S.

   Consider that the following array 1 2 3 4 5 6 is already enqueued in $Q1$. If following set of operations are performed then $Q2$ will contain a stack permutation:

   - $enqueue(Q2, dequeue(Q1))$
   - $push(S, dequeue(Q1))$
   - $enqueue(Q2, dequeue(Q1))$
   - $push(S, dequeue(Q1))$
   - $enqueue(Q2, pop(S))$
   - $enqueue(Q2, pop(S))$
   - $enqueue(Q2, dequeue(Q1))$
   - $enqueue(Q2, dequeue(Q1))$

   The queue $Q2$ will contain the following: 1 3 4 2 5 6

   This is an example of stack permutation. A stack permutation is a ordering of numbers from 1 to n that can be obtained from the initial ordering $1, 2, \ldots n$ by a sequence of stack operations as described above. To clarify this, note that 1 5 3 4 2 6 is not a stack permutation. This is because to enqueue 5 into $Q2$ after 1 we would have to push 2 3 4 into the stack which would then be output in the order 4 3 2, not in the order 3 4 2.

   Your task is to write a program, that will take a n numbers and the final permutation of numbers as input and outputs if it is a stack permutation or not. Your program should also display the sequence of operations that formed the permutation.

3. You are provided a $n \times n$ matrix, where every row and column is sorted in increasing order. Given a key k, your task is to determine if this key is present in the matrix or not in minimum possible time.