IE531: Algorithms for Data Analytics
Spring, 2018
Homework 1: Review of Linear Algebra, Probability & Statistics and Computing
Zhenye Na(zna2)

1. **Linear Algebra:** Consider the following set of linear equations involving six equations in five unknowns $x_1, x_2, \ldots, x_5$.

$$\underbrace{\begin{pmatrix} 2 & 0 & 0 & 6 & 2 \\ 1 & -1 & -1 & 4 & 0 \\ 2 & -2 & 2 & 4 & 0 \\ 2 & -2 & 0 & 6 & 0 \\ -4 & 4 & -8 & -4 & 0 \\ 4 & 0 & -2 & 14 & 4 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} 2 \\ 7 \\ 18 \\ 16 \\ -40 \\ 2 \end{pmatrix}}_{\mathbf{y}}$$

(a) Show that there a solution to these equations.

```
>> A = [2 0 0 6 2; 1 -1 -1 4 0; 2 -2 2 4 0; 2 -2 0 6 0; -4 4 -8 -4 0; 4 0 -2 14 4];
>> y = [2; 7; 18; 16; -40; 2];
>> x = A \ y;
Warning: Rank deficient, rank = 3, tol =  2.368291e-14.
>> x

x =

         0
         0
    3.6667
    2.6667
   -7.0000
```

(b) Present a general formula for the set of all solutions that satisfy these equations.
The Reduced Row Echelon Form of Matrix (A | y) is as follows:

```
Reduced_A =

     1     0     0     3     1     1
     0     1     0     0     1    -7
     0     0     1    -1     0     1
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

So the general formula for the solutions will be:

$x^T = [8-3x+y, y, 1+x, x, -7-y]^T$

(c) Verify the general formula using MATLAB.
Define x=0 and y=1, using MATLAB to calculate the matrix multiplication of Ax, the result is:

2
7
18
16
-40
2

So this is the general formula for this equation.

(d) Show that

$$\begin{pmatrix} -7\lambda_1 - 4\lambda_2 + 8 \\ -7\lambda_1 - 7\lambda_2 \\ 1 - \lambda_2 \\ -\lambda_2 \\ -7 + 7\lambda_1 + 7\lambda_2 \end{pmatrix} \tag{1}$$

and

$$\begin{pmatrix} a \\ -7a - 7b \\ -\frac{8}{3}a - \frac{7}{3}b + \frac{11}{3} \\ -\frac{8}{3}a - \frac{7}{3}b + \frac{8}{3} \\ -7 + 7a + 7b \end{pmatrix}. \tag{2}$$

are solutions to the equation Ax = y identified earlier.

According to the Reduced Row Echelon Form of A | y, we can extract three equations from it:

$$x_1 + 3x_4 + x_5 = 1$$
$$x_2 + x_5 = -7$$
$$x_3 - x_4 = 1$$

Applying both of the solutions to Ax=y, and we can get that both (1) and (2) are correct solutions to the matrix multiplication.

(e) Present a cogent argument in support of the claim that even though the solution identified by equations (1) and (2) look different, they are in effect the same.

$$\lambda_1 = \frac{-1}{7}, \qquad \lambda_2 = 0$$
$$a = 9, \qquad b = \frac{-19}{7}$$

(f) Show that if add the constraints

$$\begin{aligned} 7x_4 + x_5 &= 7 \\ x_2 + 2x_5 &= 7, \end{aligned}$$

in addition to those already in the requirement that Ax = y as described in problem 1, there can be only one solution to the combined set of equations/constraints.

```
>> AA = [2 0 0 6 2; 1 -1 -1 4 0; 2 -2 2 4 0; 2 -2 0 6 0; -4 4 -8 -4 0; 4 0 -2 14 4; 0 0 0 7 1; 0 1 0 0 2];
>> yy = [2; 7; 18; 16; -40; 2;7;7];
>> xx = AA \ yy;
>> rank(AA)

ans =

     5

>> xx

xx =

  -10.0000
  -21.0000
   -0.0000
   -1.0000
   14.0000
```

Because the rank of (A | y) is 5, which is the number of unknown variables. There is only one solution to this equation.

2. Generating RVs using the Inverse Transform Technique: You may want to use the various C++ code samples on Compass (along with some graphing/plotting software) to verify/experiment-with things before you put your answers down for this problem.

(a) The triangular distribution has a probability density function

$$f(x) = \begin{cases} x & 0 \le x \le 1 \\ 2 - x & 1 < x \le 2 \\ 0 & \text{otherwise.} \end{cases} \qquad (3)$$

In three/four sentences describe how you can generate i.i.d. r.v's that are distributed according to equation 3.

First, you should find a formula for the quantile function F_X^{-1}. Then, you need generate a uniform random number u. Via Inverse Transform Method, we can return the random number $x = F_X^{-1}(u)$

$$f(x) = \begin{cases} x & (0 \le x \le 1) \\ 2 - x & (1 \le x \le 2) \\ 0 & otherwise \end{cases}$$

$$F(x) = \begin{cases} \frac{x^2}{2} & (0 \le x \le 1) \\ 1 - \frac{(2-x)^2}{2} & (1 \le x \le 2) \\ 0 & (x < 0) \\ 1 & (x > 2) \end{cases}$$

(b) What is the i.i.d. distribution generated by repeated calls to the code shown below?

```
double blah1( )
{
  (double) x = get_uniform();
  if (x <= 0.5)
                return (1 + √(2x));
  else
       return (1 − √(2 − 2x));
}
```

$$F(x) = \begin{cases} 1 & (x \ge 2) \\ \frac{1}{2}(x-1)^2 & (1 \le x \le 2) \\ 1 - \frac{1}{2}(x-1)^2 & (0 \le x \le 1) \\ 0 & (x \le 0) \end{cases}$$

(c) What is the i.i.d. distribution generated by repeated calls to the code shown below?

```
double blah2( )
{
  (double) x = get_uniform();
  if (x <= 0.5)
                return (2 − √(1 − 2x));
  else
       return ( √(2x − 1));
}
```

$$F(x) = \begin{cases} 1 & (x \ge 2) \\ 1 - \frac{1}{2}(2-x)^2 & (1 \le x \le 2) \\ \frac{1}{2}x^2 & (0 \le x \le 1) \\ 0 & (x \le 0) \end{cases}$$

3. General Programming Concepts:

(a) Consider the two versions of the code shown in figure 1. Sup- pose I type the string "kablooey" as input, give me the reasoning behind the output each of the programs generate 1 . This is a straightforward exercise in programming.

```
#include <iostream>
#include <cmath>
using namespace std ;
void f()
{
    char c;
    cin.get(c);
    // this line reads character "c"
    if ('\n' == c)
        // '\n' is "return/newline"
        return;
    cout << c;
    f();
}
int main()
{
    f();
    cout << endl;
}
```

(a) Problem 3a(I)

```
#include <iostream>
#include <cmath>
using namespace std ;
void f()
{
    char c;
    cin.get(c);
    // this line reads character "c"
    if ('\n' == c)
        // '\n' is "return/newline"
        return;
    f();
    cout << c;
}
int main()
{
    f();
    cout << endl;
}
```

(b) Problem 3a(II)

Figure 1: (Two versions of the) Code for problem 3a.

Version 1 (LHS):
It will return the same string as input 'kablooey'. The reason is that even there is recursion in the program. It will output first then re-call the function again. So it will return every character in the input string first, then use the rest of characters to call f() again.

Version 2 (RHS):
It will return the reverse order of the input string – 'yeoolbak'. This function is different from the LHS one. Because it calls itself first so it will kind of calling a stack to save each character then when it is done, output each character in the stack. So it will return the reverse order of the input string.

(b) The greatest common divisor (GCD) of two positive numbers m and n is the largest number k that divides m and n. That is, *m/k* and *n/k* leaves no remainder. Consider the C++ code shown in figure 2. Using the fact that the GCD of two numbers does not change if the smaller number is subtracted from the larger number, show that the recursive funtion gcd(m, n) in the code shown in figure 2 implements the GCD computation.
(Note: Please do not give me an illustrative example as a "proof" of that the code does what it is supposed to do. I am looking for a rigorous attempt at showing the correctness of the code for any pair of numbers.)

```
#include <iostream>
using namespace std;
long gcd(long m, long n)
{
        if (m == n)
                return n;
        else if (m < n)
                return gcd(m,n-m);
        else
                return gcd(m-n, n);
}
int main()
{
        cout << gcd(259, 111) << endl;
}
```

Figure 2: Problem 3b.

Proof:
The case n = m is clear.

Every time, *n-m* (or *m-n*) can be written like *n-qm*. Assume that at least one of *n* and *m* is nonzero. Now ***gcd(n, m)*** divides *n*, *m* and $n - qm$, hence

$$gcd(n, m) \leq gcd(m, n - qm).$$

On the other hand, **gcd(m, n − qm)** divides *m*, $n - qm$ and $n = (n - qm) + qm$. so

$$gcd(m, n - qm) \leq gcd(n, m).$$

So, in every recursion, it will give the right answer because $gcd(m, n - qm) = gcd(n, m)$ at the end.

(c) What is the output of the C++ code shown in figure 3. Make sure you give me your reasoning for your answer.

```
#include <iostream>
using namespace std;
void print_integer (int num)
{
    putchar (num % 10 + 'A');
        if (num / 10)
                print_integer(num / 10);
}
int main()
{
        print_integer (1234);cout << endl;
}
```

Figure 3: Problem 3c.

What is the output of the above program? What is the reasoning behind the output that this program generates?

Output: 'EDCB'

Reason:
It recursively call the function print_integer() and every time it is devided by 10. So argument in print_integer() function is devided by 10 every time and leaves the digit behind point.
num % 10 + 'A' means that extract the last digit in num and add it to the ASCII number of character 'A' so we will get B, C, D, E, F and G etc..

(d) My review of computing contains the Frame-Stewart solution to the k-peg version of the Tower of Hanoi problem. Consider the following solution to the 4-peg version of the problem.

Alternate Solution to 4-Peg Tower of Hanoi Problem

1: *int* $k = \lfloor \sqrt{2 * n} \rfloor$;
2: Move_Using_Four_Pegs ($n - k$, source, intermediate_1, intermediate_2, destination);
3: Move_Using_Four_Pegs ($k - 1$, source, intermediate_2, destination, intermediate_1);
4: cout << "Move the top disk from peg " << source << " to peg " << destination << endl;
5: Move_Using_Four_Pegs ($k - 1$, intermediate_2, destination, intermediate_1, source);
6: Move_Using_Four_Pegs ($n - k$, intermediate_1, destination, intermediate_2, source);

Comment on the correctness of this algorithm. Will this work?

Code performance result as below:

no_of_disks = 3
Move the top disk from pegA to pegB
Move the top disk from pegA to pegC
Move the top disk from pegA to pegD
Move the top disk from pegC to pegD
Move the top disk from pegB to pegD

no_of_disks = 4
Move the top disk from pegA to pegD
Move the top disk from pegA to pegB
Move the top disk from pegD to pegB
Move the top disk from pegA to pegC
Move the top disk from pegA to pegD
Move the top disk from pegC to pegD
Move the top disk from pegB to pegA
Move the top disk from pegB to pegD
Move the top disk from pegA to pegD -> illegal here

no_of_disks = 5
Move the top disk from pegA to pegD
Move the top disk from pegA to pegB
Move the top disk from pegD to pegB
Move the top disk from pegA to pegB
Move the top disk from pegA to pegC
Move the top disk from pegB to pegC
Move the top disk from pegA to pegD
Move the top disk from pegC to pegA

Move the top disk from pegC to pegD
Move the top disk from pegA to pegD
Move the top disk from pegB to pegA
Move the top disk from pegB to pegD
Move the top disk from pegA to pegD

no_of_disks = 6
Move the top disk from pegA to pegC
Move the top disk from pegA to pegD
Move the top disk from pegA to pegB
Move the top disk from pegD to pegB
Move the top disk from pegC to pegB
Move the top disk from pegA to pegB
Move the top disk from pegA to pegC
Move the top disk from pegB to pegC
Move the top disk from pegA to pegD
Move the top disk from pegC to pegA
Move the top disk from pegC to pegD
Move the top disk from pegA to pegD
Move the top disk from pegB to pegC
Move the top disk from pegB to pegA
Move the top disk from pegB to pegD
Move the top disk from pegA to pegD
Move the top disk from pegC to pegD

no_of_disks = 7
Move the top disk from pegA to pegB
Move the top disk from pegA to pegC
Move the top disk from pegB to pegC
Move the top disk from pegA to pegD
Move the top disk from pegA to pegB
Move the top disk from pegD to pegB
Move the top disk from pegC to pegA
Move the top disk from pegC to pegB
Move the top disk from pegA to pegB
Move the top disk from pegA to pegB
Move the top disk from pegA to pegC
Move the top disk from pegB to pegC
Move the top disk from pegA to pegD
Move the top disk from pegC to pegA
Move the top disk from pegC to pegD
Move the top disk from pegA to pegD
Move the top disk from pegB to pegD
Move the top disk from pegB to pegC

Move the top disk from pegD to pegC
Move the top disk from pegB to pegA
Move the top disk from pegB to pegD
Move the top disk from pegA to pegD
Move the top disk from pegC to pegB
Move the top disk from pegC to pegD
Move the top disk from pegB to pegD

no_of_disks = 8
Move the top disk from pegA to pegB
Move the top disk from pegA to pegC
Move the top disk from pegB to pegC
Move the top disk from pegA to pegD
Move the top disk from pegA to pegB
Move the top disk from pegD to pegB
Move the top disk from pegC to pegA
Move the top disk from pegC to pegB
Move the top disk from pegA to pegB
Move the top disk from pegA to pegD
Move the top disk from pegA to pegB
Move the top disk from pegA to pegC
Move the top disk from pegB to pegC
Move the top disk from pegD to pegC
Move the top disk from pegA to pegD
Move the top disk from pegC to pegB
Move the top disk from pegC to pegA
Move the top disk from pegC to pegD
Move the top disk from pegA to pegD
Move the top disk from pegB to pegD
Move the top disk from pegB to pegD
Move the top disk from pegB to pegC
Move the top disk from pegD to pegC
Move the top disk from pegB to pegA
Move the top disk from pegB to pegD
Move the top disk from pegA to pegD
Move the top disk from pegC to pegB
Move the top disk from pegC to pegD
Move the top disk from pegB to pegD

no_of_disks = 9
Move the top disk from pegA to pegB
Move the top disk from pegA to pegC
Move the top disk from pegB to pegC
Move the top disk from pegA to pegC

Move the top disk from pegA to pegD
Move the top disk from pegC to pegD
Move the top disk from pegA to pegB
Move the top disk from pegD to pegA
Move the top disk from pegD to pegB
Move the top disk from pegA to pegB
Move the top disk from pegC to pegA
Move the top disk from pegC to pegB
Move the top disk from pegA to pegB
Move the top disk from pegA to pegD
Move the top disk from pegA to pegB
Move the top disk from pegA to pegC
Move the top disk from pegB to pegC
Move the top disk from pegD to pegC
Move the top disk from pegA to pegD
Move the top disk from pegC to pegB
Move the top disk from pegC to pegA
Move the top disk from pegC to pegD
Move the top disk from pegA to pegD
Move the top disk from pegB to pegD
Move the top disk from pegB to pegD
Move the top disk from pegB to pegC
Move the top disk from pegD to pegC
Move the top disk from pegB to pegC
Move the top disk from pegB to pegA
Move the top disk from pegC to pegA
Move the top disk from pegB to pegD
Move the top disk from pegA to pegB
Move the top disk from pegA to pegD
Move the top disk from pegB to pegD
Move the top disk from pegC to pegB
Move the top disk from pegC to pegD
Move the top disk from pegB to pegD

no_of_disks = 10
Move the top disk from pegA to pegD
Move the top disk from pegA to pegB
Move the top disk from pegA to pegC
Move the top disk from pegB to pegC
Move the top disk from pegD to pegC
Move the top disk from pegA to pegC
Move the top disk from pegA to pegD
Move the top disk from pegC to pegD
Move the top disk from pegA to pegB

Move the top disk from pegD to pegA
Move the top disk from pegD to pegB
Move the top disk from pegA to pegB
Move the top disk from pegC to pegD
Move the top disk from pegC to pegA
Move the top disk from pegC to pegB
Move the top disk from pegA to pegB
Move the top disk from pegD to pegB
Move the top disk from pegA to pegD
Move the top disk from pegA to pegB
Move the top disk from pegA to pegC
Move the top disk from pegB to pegC
Move the top disk from pegD to pegC
Move the top disk from pegA to pegD
Move the top disk from pegC to pegB
Move the top disk from pegC to pegA
Move the top disk from pegC to pegD
Move the top disk from pegA to pegD
Move the top disk from pegB to pegD
Move the top disk from pegB to pegA
Move the top disk from pegB to pegD
Move the top disk from pegB to pegC
Move the top disk from pegD to pegC
Move the top disk from pegA to pegC
Move the top disk from pegB to pegC
Move the top disk from pegB to pegA
Move the top disk from pegC to pegA
Move the top disk from pegB to pegD
Move the top disk from pegA to pegB
Move the top disk from pegA to pegD
Move the top disk from pegB to pegD
Move the top disk from pegC to pegA
Move the top disk from pegC to pegB
Move the top disk from pegC to pegD
Move the top disk from pegB to pegD
Move the top disk from pegA to pegD

There exists some illegal move from peg to peg (large disk on small disk), so this solution is not correct.