Estructuras de Datos y Algoritmos 1: Taller ACM

Fecha
17 de Noviembre del 2017

Alberto Antonio Restrepo Velasquez

Isabel Cristina Lopez Giraldo

Contents

Problem 1

441 Lotto

In the German Lotto you have to select 6 numbers from the set 1,2,...,49. A popular strategy to play Lotto - although it doesn't increase your chance of winning — is to select a subset S containing k (k¿6) of these 49 numbers, and then play several games with choosing numbers only from S. For example, for k = 8 and S = 1, 2, 3, 5, 8, 13, 21, 34 there are 28 possible games: [1,2,3,5,8,13], [1,2,3,5,8,21], [1,2,3,5,8,34], [1,2,3,5,13,21], ..., [3,5,8,13,21,34]. Your job is to write a program that reads in the number k and the set S and then prints all possible games choosing numbers only from S.

Input

The input file will contain one or more test cases. Each test case consists of one line containing several integers separated from each other by spaces. The first integer on the line will be the number k (6¡k¡13). Then k integers, specifying the set S, will follow in ascending order. Input will be terminated by a value of zero (0) for k

Output

For each test case, print all possible games, each game on one line. The numbers of each game have to be sorted in ascending order and separated from each other by exactly one space. The games themselves have to be sorted lexicographically, that means sorted by the lowest number first, then by the second lowest and so on, as demonstrated in the sample output below. The test cases have to be separated from each other by exactly one blank line. Do not put a blank line after the last test case.

Sample Input

```
7 1 2 3 4 5 6 7
8 1 2 3 5 8 13 21 34
0
```

Sample Output

```
1 2 3 4 5 6
1 2 3 4 5 7
1 2 3 4 6 7
1 2 3 5 6 7
1 2 4 5 6 7
1 3 4 5 6 7
2 3 4 5 6 7

1 2 3 5 8 13
1 2 3 5 8 21
1 2 3 5 8 34
1 2 3 5 13 21
1 2 3 5 13 34
1 2 3 5 21 34
1 2 3 8 13 21
1 2 3 8 13 34
1 2 3 8 21 34
1 2 3 13 21 34
1 2 5 8 13 21
```

Problem 1 continued on next page...

1 2 5 8 13 34
1 2 5 8 21 34
1 2 5 13 21 34
1 2 8 13 21 34
1 3 5 8 13 21
1 3 5 8 13 34
1 3 5 8 21 34
1 3 5 13 21 34
1 3 8 13 21 34
1 5 8 13 21 34
2 3 5 8 13 21
2 3 5 8 13 34
2 3 5 8 21 34
2 3 5 13 21 34
2 3 8 13 21 34
2 5 8 13 21 34
3 5 8 13 21 34

Análisis

Input

k: Cantidad de números que se va a ingresar.

$$6 < k < 13 \tag{1}$$

s: set de números

$$s\,\{1, 2...49\} \tag{2}$$

Output

Posibilidades de conjuntos de 6 números organizados de manera ascendente y estos mismos conjuntos organizados de menor a mayor

Metodología

Los métodos que se van a implementar para darle solución a Lotto, es el uso de listas, y listas de listas, para almacenar los valores.

Para abordar este problema es necesario analizar varios aspectos, uno de ellos es la cantidad de elementos en la salida, los cuales dependen directamente del número de elementos que se van a insertar.
Con el fin de hallar la cantidad de sets que debe imprimirse en la salida, se deduce la siguiente fórmula

$$\frac{k!}{(k-6)\quad 6!} \tag{3}$$

Donde:
k!: Total de formas posibles del conjunto
6!: Total de formas posibles de 6 elementos
(k-6): Número de 6! que se puede tener

es decir para k= 7,8 y 9 el número total de combinaciones son las siguientes:

$$\frac{7!}{(7-6)\quad 6!} = 7 \tag{4}$$

$$\frac{8!}{(8-6)\quad 6!} = 28 \tag{5}$$

$$\frac{9!}{(9-6)\quad 6!} = 168 \tag{6}$$

Otro de los aspectos a tener en cuenta es como se deben ir imprimiendo los elementos, debido a que deben estar de manera ascedente, por tal razón se analiza el caso más simple que es cuando k=7 y s=1,2,3,4,5,6,7

En la figura 1 se ilustra como se deben ir tomando los elementos en cada una de las iteraciones, el color azul es la posición del elemento en una lista y el color verde es un contador, que debe reiniciarse cada 6 iteraciones y aumentar la posición en una unidad.

Después de entender el problema, se analizan los pasos a seguir que debe tener el programa, con el fin

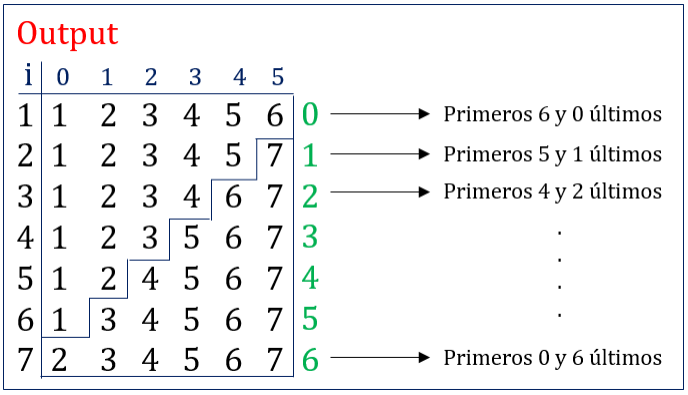

Figure 1: Output k=7

de hacer un seudocodigo, para implementarlo en un lenguaje de programación. Ver figura 2

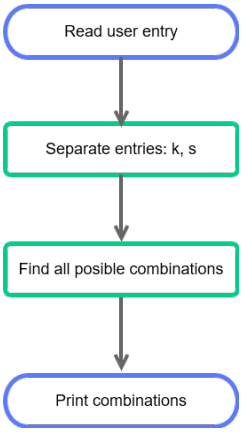

Figure 2: Pasos seudocódigo

Por lo dicho anteriormente, se procede a realizar el seudocodigo implementando uno que sirva para cualquier caso y que sea detallado para su fácil implementación.

Listing 1: Seudocódigo refinado

```
combinations(set)
# Funci n que obtiene las posibles combinaciones
if length(set)>0 then
    residuo = combinaciones(set(1:ultimo))
    return  + set(0) + cada valor que se encuentre en residuo
return set

end if


main():

entrada= lista de listas

    if(linea =! vacio) then
        if linea=0 then
            salir
        else
            k=entero de lista(0)

            if(6<k<13)

                # Lo siguiente es porque si empieza por 1 es porque k es
                # un numero de dos cifras lo que quiere decir que s empiza en
                # la posici n 3

                if(linea(0)=1) then
                    temporal=line(3:ultimo)
                else
                   temporal=line(1:ultimo)

                end if

                index=0
                insertar= falso

                tama oEntreda=len(entrada)

                while(index<tama oEntreda)

                    tama o=length(entrada(index))

                    if(k<=tama o) then
                        insert=verdadero

                        entrada.insert(index,temporal)

                    index=index + 1
```

```
                        end if

                    if(insert=falso) then

                        entrada=insertar(temporal)

                    end if
                end if
            end if
        end if


    entrada=entrada(1:ultimo)

    cont=1

    for Linea contenida en entrada
        comb=combinaciones(Linea)
        resultado=[[]]

        for item contenido en comb

            if(length(item)=6) then
                resultado=insertar(item)
            end if

        end for

        i=length(resultado)-1

        while i > 0

            print resultado(i)
            i=i-1

        cont=cont+1

        end while

    end for
```

Código Python 3.5.1

A continuación se muestra la implementación del codigo con Python 3.5.1, esta versión debido a que es la que aceptaba el juez online.

Listing 2: Código

```
#Isabel Critina Lpez
# Estructura de datos y algoritmos I
# PROBLEM ID: 441-lOTTO

#INPUT: One line containing sevesal integers separted from each other by spaces.
# The frist integer will be k and then k integers specifying s.
```

```
# k(Number of s elements 6<k<13)
#s(Set of numbers)

#Output: All posible games combinations in ascending order, each game on one line.

def combinations(s):
    if len(s)>0:
        rest = combinations(s[1:])
        return rest + [[s[0]] + x for x in rest ]
    return [[]];

import sys

def main():

    entrada=[[]]

    for line in sys.stdin:
        line=line.replace("\n","")
        if(len(line)>0):
            if line=='0':
                break
            else:
                k=(int)(line.strip().split(" ")[0])

                if(k>6 and k<13):

                    if(line[0]=='1'):
                        temp=line[3:].strip().split(" ")
                    else:
                       temp=line[1:].strip().split(" ")

                    index=0
                    insert=False

                    tamEntreda=len(entrada)
                    while(index<tamEntreda):
                        tam=len(entrada[index])
                        if(k<=tam):
                            insert=True
                            entrada.insert(index,temp)
                        index+=1

                    if(insert==False):
                        entrada.append(temp)

    entrada=entrada[1:]

    cont=1

    for InputLine in entrada:
        a=combinations(InputLine)
```

```
        result=[[]]
        for item in a:
            if(len(item)==6):
                result.append(item)

        i=len(result)-1
        while i > 0:

            sys.stdout.write(' '.join(str(x) for x in result[i]))
            sys.stdout.write('\n')
            i-=1

        cont+=1
        if(cont<=len(entrada)):
            sys.stdout.write('\n')

if __name__ == "__main__":
    main()
```

UVA Juez online

Respuesta aceptada del juez online

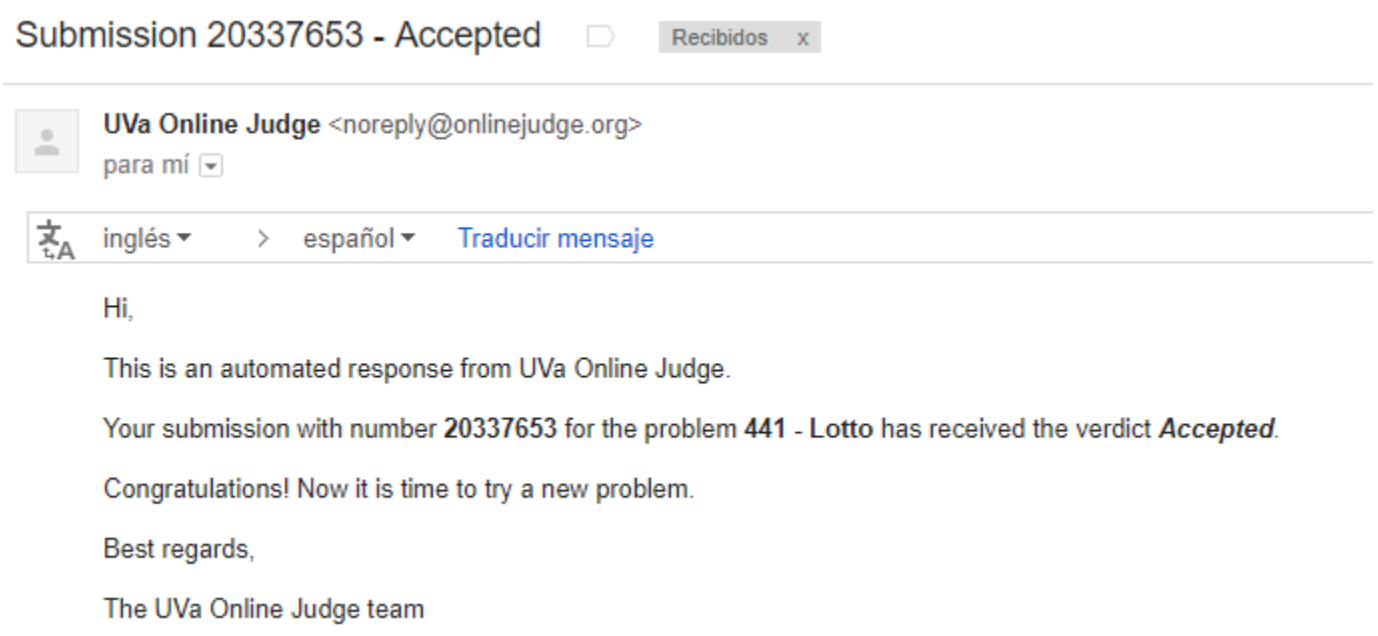

Figure 3: Respuesta UVA