OIFem II Soluciones Clasificatorio

Comprando churros

Autora del problema: Blanca Huergo Muñoz

Teoría

- Bucles

Solución

Creamos una variable `suma`, donde en todo momento guardaremos la suma de los precios que ya sabemos. Iremos leyendo los precios uno a uno y sumándolos al total al leerlos. Una vez leídos todos, imprimimos la suma.

Código- C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;


int main() {
    int n;
    cin >> n;
    int suma = 0;
    for (int i = 0; i < n; i++) {
        int precio;
        cin >> precio;
        suma += precio;
    }
    cout << suma << '\n';
    return 0;
}
```

Código- Python

```python
n = int(input())
precios = list(map(int, input().split()))
print(sum(precios))
```

Código- Java

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int sum = 0;
        for (int i = 0; i < n; ++i) {
            sum += sc.nextInt();
        }
        System.out.println(sum);
        sc.close();
    }
}
```

Palabras alienígenas

Autora del problema: Blanca Huergo Muñoz

Teoría

- Solución con vector: ASCII, vectores
- Solución con mapa: Mapas, iteradores

Solución

Hay varias formas de resolver este problema, pero la idea común es contar cuántas ocurrencias hay de cada carácter e iterar en orden alfabético para ver si la palabra dada por el usuario sigue la regla o no. En este caso, hemos usado un vector de enteros para guardar la frecuencia de cada carácter.

Código- C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;


int main() {
    int t;
    cin >> t;
    while (t--) {
        string S;
        cin >> S;
        vector<int> freq(26, 0);
        for (char & c: S) {
            freq[c - 'a']++;
        }

        bool sigueLaRegla = true;
        for (int i = 1; i < 26; i++) {
            if (freq[i] > freq[i-1]) {
                sigueLaRegla = false;
                break;
            }
        }

        if (sigueLaRegla)
            cout << "SIGUE LA REGLA\n";
        else
            cout << "NO SIGUE LA REGLA\n";
    }
    return 0;
}
```

Código- Python

```python
t = int(input())
for x in range(t):
    S = input()
    chars = dict()
    for c in 'abcdefghijklmnopqrstuvwxyz':
        chars[c] = 0
    for c in S:
        chars[c] += 1
    prev = chars['a']
    sigue = True
    for c in 'bcdefghijklmnopqrstuvwxyz':
        if chars[c] > prev:
            sigue = False
            break
        prev = chars[c]
    if sigue:
        print("SIGUE LA REGLA")
    else:
        print("NO SIGUE LA REGLA")
```

Código- Java

```java
import java.io.*;
import java.util.*;
import java.util.Scanner;

public class Solution {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int t = input.nextInt();
        for (int x = 0; x < t; x++) {
            String S = input.next();
            int [] abc = new int[26];
            for (int i = 0; i < 26; i++)
                abc[i] = 0;
            for (int i = 0; i < S.length(); i++)
                abc[S.charAt(i) - 'a']++;
            boolean sigue = true;
            for (int i = 1; i < 26; i++) {
                if (abc[i] > abc[i-1]) {
                    sigue = false;
                    break;
                }
            }
            if (sigue) System.out.println("SIGUE LA REGLA");
            else System.out.println("NO SIGUE LA REGLA");
        }
        input.close();
    }
}
```

SumaBase

Autora del problema: Blanca Huergo Muñoz

Teoría

- Representación de enteros en diferentes bases
- Representación de enteros grandes con vectores
- Sumar por columnas

Solución

Para resolver este problema, la forma más sencilla era usar el método aprendido en Primaria para sumar por columnas. Guardamos los números dados en un vector cada uno y los sumamos por columnas, guardando cuánto llevamos (`carry`) cada vez que sumamos dos columnas.

Código- C++

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
    int T; cin >> T;
    while(T--){
        string A, B; int k;
        cin >> A >> B >> k;
        // Doy la vuelta a las strings para trabajar más cómodamente
        reverse(A.begin(), A.end());
        reverse(B.begin(), B.end());
        string C(max(A.size(), B.size()), '0'); // resultado
        // Aquí empezamos la suma
        int carry = 0;
        for(int i = 0; i < A.size() || i < B.size(); ++i){
            int a = (i < A.size() ? A[i] - '0' : 0);
            int b = (i < B.size() ? B[i] - '0' : 0);
            C[i] = '0' + ((a + b + carry) % k);
            if(a + b + carry >= k) carry = 1;
            else carry = 0;
        }
        if(carry == 1) C += '1';
        // Salida
        reverse(C.begin(), C.end());
        cout << C << "\n";
    }
    return 0;
}
```

Código- Python

```python
t = int(input())
for x in range(t):
    A, B, k = input().split()
    k = int(k)
    n = max(len(A), len(B))
    C = [0 for i in range(n+1)]
    carry = 0
    for i in range(1, n+1):
        c1 = 0
        if len(A) >= i:
            c1 = int(A[-i])
        c2 = 0
        if len(B) >= i:
            c2 = int(B[-i])
        S = c1+c2+carry
        C[-i] = S % k
        carry = S // k
    C[0] = carry
    if C[0] == 0 and len(C) > 1:
        C = C[1:]
    C = list(map(str, C))
    print("".join(C))
```

Código- Java

```java
import java.io.*;
import java.util.*;

public class Solution {

    static String repeat(char c, int n) {
        String s = "";
        while (n-- > 0) s += c;
        return s;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        while (t-- > 0) {
            String A = sc.next();
            String B = sc.next();
            int k = sc.nextInt();
            int n = Math.max(A.length(), B.length());
            A = repeat('0', n + 1 - A.length()) + A;
            B = repeat('0', n + 1 - B.length()) + B;
            int carry = 0;
            int[] sum = new int[n+1];
            for (int i = 0; i <= n; ++i) {
                sum[i] = carry + (A.charAt(n-i) - '0') + (B.charAt(n-i) - '0');
                carry = sum[i]/k;
                sum[i] %= k;
            }
            boolean started = false;
            for (int i = 0; i <= n; ++i) {
                int d = sum[n-i];
                if (d != 0 || started) {
                    started = true;
                    System.out.print(d);
                }
            }
            if (!started) System.out.print('0');
            System.out.print('\n');
        }
        sc.close();
    }
}
```

Bugs en raya

Autora del problema: Blanca Huergo Muñoz

Teoría

- Grafos implícitos
- BFS

Solución

Para cada tablero que nos dan, iniciamos una BFS desde el tablero vacío y tratamos de llegar a él, añadiendo fichas del tablero final al tablero intermedio, siempre siguiendo las reglas. Si encontramos alguna forma de llegar al final, sabremos que esto es posible, pero si agotamos todos los caminos sin resultado, es un tablero imposible.

Código- C++

```cpp
#include <iostream>
#include <vector>
#include <unordered_set>
#include <queue>
using namespace std;

char turno(string & tablero) {
    int relleno = 0;
    for (int i = 0; i < 9; i++) {
        if (tablero[i] != 'V')
            relleno++;
    }
    if (relleno & 1)
        return 'O';
    return 'X';
}

bool partidaAcabada(string & tablero) {
    for (int columna = 0; columna < 3; columna++) {
        // columna completa
        if (tablero[columna] != 'V' && tablero[columna] == tablero[3+columna] &&
        ↪  tablero[3+columna] == tablero[6+columna])
            return true;
    }
    for (int fila = 0; fila < 3; fila++) {
        // fila completa
        if (tablero[fila*3] != 'V' && tablero[3*fila] == tablero[3*fila+1] &&
        ↪  tablero[3*fila+1] == tablero[3*fila+2])
            return true;
    }
    // diagonal 1 completa
    if (tablero[0] != 'V' && tablero[0] == tablero[4] && tablero[4] == tablero[8])
        return true;
    // diagonal 2 completa
    if (tablero[2] != 'V' && tablero[2] == tablero[4] && tablero[4] == tablero[6])
```

```
34          return true;
35      return false;
36  }
37
38  bool posible(string & tablero) {
39      queue<string> Q;
40      Q.push("VVVVVVVVV");
41      unordered_set<string> visited;
42      visited.insert("VVVVVVVVV");
43      while(Q.size()) {
44          string tableroActual = Q.front();
45          Q.pop();
46          if (tableroActual == tablero)
47              return true;
48          if (!partidaAcabada(tableroActual)) {
49              char turnoActual = turno(tableroActual);
50              for (int i = 0; i < 9; i++) {
51                  if (tableroActual[i] != tablero[i] && tablero[i] == turnoActual) {
52                      string tableroVecino = tableroActual;
53                      tableroVecino[i] = tablero[i];
54                      visited.insert(tableroVecino);
55                      Q.push(tableroVecino);
56                  }
57              }
58          }
59      }
60      return false;
61  }
62
63  int main() {
64      int t;
65      cin >> t;
66      while(t--) {
67          string tablero = "", fila;
68          for (int i = 0; i < 3; i++) {
69              cin >> fila;
70              tablero += fila;
71          }
72          if (posible(tablero))
73              cout << "POSIBLE\n";
74          else
75              cout << "IMPOSIBLE\n";
76          cin >> fila;
77      }
78      return 0;
79  }
```

Código- Python

```python
import queue

def turno(tablero):
    relleno = 0
    for i in range(9):
        if (tablero[i] != 'V'):
            relleno += 1
    if (relleno & 1):
        return 'O'
    return 'X'

def partidaAcabada(tablero):
    for columna in range(3):
        if (tablero[columna] != 'V' and tablero[columna] == tablero[3+columna] and tablero[3+columna] == tablero[6+columna]):
            return True
    for fila in range(3):
        if (tablero[fila*3] != 'V' and tablero[3*fila] == tablero[3*fila+1] and tablero[3*fila+1] == tablero[3*fila+2]):
            return True
    if (tablero[0] != 'V' and tablero[0] == tablero[4] and tablero[4] == tablero[8]):
        return True
    if (tablero[2] != 'V' and tablero[2] == tablero[4] and tablero[4] == tablero[6]):
        return True
    return False

def posible(tablero):
    Q = queue.Queue()
    Q.put("VVVVVVVVV")
    visited = set()
    visited.add("VVVVVVVVV")
    while not Q.empty():
        tableroActual = Q.get()
        if (tableroActual == tablero):
            return True
        if (not partidaAcabada(tableroActual)):
            turnoActual = turno(tableroActual)
            for i in range(9):
                if (tableroActual[i] != tablero[i] and tablero[i] == turnoActual):
                    tableroVecino = tableroActual[:i] + tablero[i] + tableroActual[i+1:]
                    visited.add(tableroVecino)
                    Q.put(tableroVecino)
    return False

t = int(input())
for x in range(t):
    tablero = ""
    for i in range(3):
        fila = input()
        tablero += fila
```

```
    if (posible(tablero)):
        print("POSIBLE")
    else:
        print("IMPOSIBLE")
    fila = input()
```

Código- Java

```
import java.io.*;
import java.util.*;

public class Solution {
    public static boolean gana (ArrayList<ArrayList<Integer>> m, int p){
        for(int i = 0; i<3; ++i){
            if(m.get(i).get(0) == p && m.get(i).get(1) == p && m.get(i).get(2) ==
            ↪  p)
                return true;
        }
        for(int j = 0; j<3; ++j){
            if(m.get(0).get(j) == p && m.get(1).get(j) == p && m.get(2).get(j) ==
            ↪  p)
                return true;
        }
        if(m.get(0).get(0) == p && m.get(1).get(1) == p && m.get(2).get(2) == p)
        ↪  return true;
        if(m.get(0).get(2) == p && m.get(1).get(1) == p && m.get(2).get(0) == p)
        ↪  return true;
        return false;
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int t = input.nextInt();
        for (int x = 0; x < t; x++) {
            int cnt1 = 0, cnt2 = 0;
            ArrayList<ArrayList<Integer>> tablero = new
            ↪  ArrayList<ArrayList<Integer>>();
            for (int i = 0; i < 3; i++) {
                tablero.add(new ArrayList<Integer>());
                tablero.get(i).add(0);
                tablero.get(i).add(0);
                tablero.get(i).add(0);
            }
            String a;
            for(int i = 0; i < 3; ++i){
                a = input.next();
                for(int j = 0; j < 3; ++j) {
                    if(a.charAt(j) == 'X') {
                        tablero.get(i).set(j, 1);
                        ++cnt1;
                    }
                    else if(a.charAt(j) == 'O') {
                        tablero.get(i).set(j, 2);
                        ++cnt2;
```

```
                    }
                }
            }
            a = input.next();
            if(cnt2 > cnt1 || cnt1 > 1 + cnt2 // cantidad de turnos, básico
                || (gana(tablero, 1) && gana(tablero, 2)) // no ganan los dos a la vez
                || (gana(tablero, 2) && cnt1 == 1 + cnt2) // si gana el segundo, no puede poner ficha el primero
                || (gana(tablero, 1) && cnt1 == cnt2) // si gana el primero, no puede poner ficha el segundo
                )
                System.out.println("IMPOSIBLE");
            else System.out.println("POSIBLE");
        }
        input.close();
    }
}
```

Trampas por una buena causa

Autora del problema: Blanca Huergo Muñoz

Teoría

- abc

Solución

La clave para resolver este problema era darse cuenta de que k factorizaba $n!$ si todos los factores primos de k eran factores de $n!$ con sus correspondientes multiplicidades. Es decir, que si a^b factorizaba k, siendo a primo y b la mayor potencia de a que factoriza k, entonces a^b debe factorizar $n!$. Basta con comparar b con c, donde a^c es la mayor potencia de a que factoriza $n!$. Para encontrar c sin calcular $n!$, sumaremos el número múltiplos de a menores o iguales que n, el número de múltiplos de a^2 menores o iguales que n, de a^3, etc. hasta llegar a la mayor potencia de a que no supere n. Esta suma será c.

Código- C++

```cpp
#include <bits/stdc++.h>
using namespace std;

// Potencia de f más grande que divide n!
int times(int f, int n) {
    long long F = f;
    int cnt = 0;
    while (F <= n) {
        cnt += n/F;
        F *= f;
    }
    return cnt;
}

// Factorizamos k en O(sqrt k) y miraremos
// que las potencias sean menores o iguales
// a las que tiene n! para cada primo
int main() {
    int t;
    cin >> t;
    while (t--) {
        int n, k;
        cin >> n >> k;
        bool ok = true;
        for (int i = 2; i*i <= k; ++i) {
            if (k % i == 0) {
                int cnt = 0;
                while (k % i == 0) {
                    ++cnt;
                    k /= i;
                }
                ok &= times(i, n) >= cnt;
            }
        }
        if (k > 1) ok &= n >= k;
```

```
            cout << (ok ? "SI" : "NO") << endl;
        }
    }
```

Código- Python

```python
def times(f, n):
    F = f
    cnt = 0
    while F <= n:
        cnt += n/F
        F *= f
    return cnt

t = int(input())
for _ in range(t):
    n, k = map(int, input().split())
    ok = True
    i = 2
    while i*i <= k:
        if k % i == 0:
            cnt = 0
            while k % i == 0:
                k //= i
                cnt += 1
            ok &= times(i, n) >= cnt
        i += 1
    if k > 1:
        ok &= n >= k
    print("SI" if ok else "NO")
```

Código- Java

```java
import java.io.*;
import java.util.*;

public class Solution {

    static int times(int n, int i) {
        int cnt = 0;
        long p = i;
        while (p <= n) {
            cnt += n/p;
            p *= i;
        }
        return cnt;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        while (t-- > 0) {
```

```
            int n = sc.nextInt();
            int k = sc.nextInt();
            boolean ok = true;
            for (int i = 2; i <= k; ++i) {
                if (k % i == 0) {
                    int cnt = 0;
                    while (k % i == 0) {
                        k /= i;
                        ++cnt;
                    }
                    ok = ok && (times(n, i) >= cnt);
                }
            }
            if (k > 1) {
                ok = ok && (n >= k);
            }
            System.out.println(ok ? "SI" : "NO");
        }
        sc.close();
    }
}
```

Aliteración

Autora del problema: Blanca Huergo Muñoz

Teoría

- Structs
- Tries

Solución

Añadiremos las palabras del poema a un trie, guardando en cada nodo cuántas palabras lo están usando a la vez. Una vez añadidas todas las palabras, hacemos una DFS por el trie, guardando en todo momento la profundidad actual y multiplicándola por el número de palabras que usan un nodo para ver la puntuación del prefijo formado por el camino de la raíz a ese nodo. Nos quedamos la mayor puntuación.

Código- C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;

struct TrieNode {
    int freq;
    char c;
    vector<TrieNode*> children;
    TrieNode(char d) {
        c = d;
        children.assign(26, NULL);
        freq = 1;
    }
};

void insert(TrieNode * curNode, size_t ind, string & S) {
    if (ind == S.length())
        return;
    if (curNode->children[S[ind] - 'a'] != NULL) {
        curNode->children[S[ind] - 'a']->freq++;
        insert(curNode->children[S[ind] - 'a'], ind+1, S);
    } else {
        curNode->children[S[ind] - 'a'] = new TrieNode(S[ind]);
        insert(curNode->children[S[ind] - 'a'], ind+1, S);
    }
}

int puntuacion(TrieNode * curNode, int depth) {
    if (curNode->freq == 1)
        return 0;
    int ans = curNode->freq * depth;
    for (int i = 0; i < 26; i++) {
```

```
        if (curNode->children[i] != NULL)
            ans = max(ans, puntuacion(curNode->children[i], depth+1));
    }
    return ans;
}

int main() {
    int t, n;
    string S;
    cin >> t;
    while(t--) {
        cin >> n;
        TrieNode root = TrieNode('-');
        root.freq = 0;
        while(n--) {
            cin >> S;
            insert(&root, 0, S);
        }
        cout << puntuacion(&root, 0) << '\n';
    }
    return 0;
}
```

Código- Python

```
import sys
sys.setrecursionlimit(10000)
class TrieNode:
    def __init__(self, char):
        self.char = char
        self.counter = 0
        self.children = {}

class Trie(object):
    def __init__(self):
        self.root = TrieNode("")

    def insert(self, word):
        node = self.root

        for char in word:
            if char in node.children:
                node = node.children[char]
            else:
                new_node = TrieNode(char)
                node.children[char] = new_node
                node = new_node
            node.counter += 1

    def dfs(self, node, depth):
        if node.counter == 1:
            return 0

        ans = node.counter * depth
```

```
30
31          for child in node.children.values():
32              ans = max(ans, self.dfs(child, depth+1))
33
34          return ans
35
36  t = int(input())
37  for x in range(t):
38      n = int(input())
39      wordTrie = Trie()
40      for i in range(n):
41          A = input()
42          wordTrie.insert(A)
43      print(wordTrie.dfs(wordTrie.root, 0))
```

Código- Java

```
1   import java.io.*;
2   import java.util.*;
3   import java.util.Scanner;
4
5   class Node {
6       private Node child[];
7       private int cnt = 0;
8
9       Node() {
10          this.cnt = 0;
11          this.child = new Node[26];
12      }
13
14      public void insert(String s, int k) {
15          ++this.cnt;
16          if (k >= 0) {
17              char c = s.charAt(k);
18              if (this.child[c - 'a'] == null) {
19                  this.child[c - 'a'] = new Node();
20              }
21              this.child[c - 'a'].insert(s, k - 1);
22          }
23      }
24
25      public int get_best(int depth) {
26          int best = depth * this.cnt * (this.cnt > 1 ? 1 : 0);
27          for (int i = 0; i < 26; ++i) {
28              if (this.child[i] != null) {
29                  best = Math.max(best, this.child[i].get_best(depth + 1));
30              }
31          }
32          return best;
33      }
34  }
35
36  public class Solution {
37      public static void main(String[] args) {
```

```
38          Scanner sc = new Scanner(System.in);
39          int t = sc.nextInt();
40          while (t-- > 0) {
41              Node root = new Node();
42              int n;
43              n = sc.nextInt();
44              for (int i = 0; i < n; ++i) {
45                  String s = sc.next();
46                  s = new StringBuilder(s).reverse().toString();
47                  root.insert(s, s.length()-1);
48              }
49              System.out.println(root.get_best(0));
50          }
51          sc.close();
52      }
53  }
```