Soluciones OIFem 2020/21

Final Día 2

Robando chuches

Autora: Blanca Huergo

Teoría

- Bucles
- Ordenamiento

Solución

Ordenaremos la lista de chocolates y la de gominolas en orden descendente, ya que si cogemos k chocolates, querremos los que tienen las k mayores delicias. Como son pocos chocolates y gominolas, podemos probar todas las posibilidades de coger chocolates, es decir, podemos coger cualquier número entre 0 y 5 chocolates (o el máximo número de chocolates, en el caso de que haya menos de 5) y, para cada cantidad de chocolates, el máximo número de gominolas posible (10 - 2*k, o el número de gominolas si son menos).

Código

C++

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

int coger(vector<int> v, int k) {
    return v[min(k, (int) v.size() - 1)];
}

int main() {
    int n, m;
    vector<int> choc, gomi, sumaC, sumaG;
    cin >> n;
    choc = vector<int>(n);
    for (int i = 0; i < n; i++)
        cin >> choc[i];
```

```
    sort(choc.begin(), choc.end(), greater<int>());
    cin >> m;
    gomi = vector<int>(m);
    for (int i = 0; i < m; i++)
        cin >> gomi[i];
    sort(gomi.begin(), gomi.end(), greater<int>());
    sumaC = vector<int>(n+1);
    sumaC[0] = 0;
    for (int i = 1; i <= n; i++)
        sumaC[i] = sumaC[i-1] + choc[i-1];
    sumaG = vector<int>(m+1);
    sumaG[0] = 0;
    for (int i = 1; i <= m; i++)
        sumaG[i] = sumaG[i-1] + gomi[i-1];
    int ans = 0;
    for (int i = 0; i <= 5; i++)
        ans = max(ans, coger(sumaC, i) + coger(sumaG, 10 - 2*i));
    cout << ans << '\n';
    return 0;
}
```

Java

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

public class Solution {
    static int[] choc, gomi, sumaC, sumaG;
    static int cogerChoc(int k, int n) {
        return sumaC[Math.min(k, n)];
    }
    static int cogerGomi(int k, int m) {
        return sumaG[Math.min(k, m)];
    }
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int n, m;
        n = s.nextInt();
        choc = new int[n];
        sumaC = new int[n+1];
        for (int i = 0; i < n; i++)
            choc[i] = s.nextInt();
        Arrays.sort(choc);
        for (int i = 0; i < n/2; i++) {
            int temp = choc[i];
            choc[i] = choc[n-i-1];
            choc[n-i-1] = temp;
        }
        sumaC[0] = 0;
        for (int i = 1; i <= n; i++)
            sumaC[i] = sumaC[i-1] + choc[i-1];
        m = s.nextInt();
        gomi = new int[m];
        sumaG = new int[m+1];
        for (int i = 0; i < m; i++)
```

```
            gomi[i] = s.nextInt();
        Arrays.sort(gomi);
        for (int i = 0; i < m/2; i++) {
            int temp = gomi[i];
            gomi[i] = gomi[m-i-1];
            gomi[m-i-1] = temp;
        }
        sumaG[0] = 0;
        for (int i = 1; i <= m; i++)
            sumaG[i] = sumaG[i-1] + gomi[i-1];
        int ans = 0;
        for (int i = 0; i <= 5; i++)
            ans = Math.max(ans, cogerChoc(i, n) + cogerGomi(10 - 2*i, m));
        System.out.println(ans);
        s.close();
    }
}
```

Python

```
def cogerdos(k):
    if k >= len(totaldos):
        return totaldos[-1]
    else:
        return totaldos[k]

def cogeruno(k):
    if k >= len(totaluno):
        return totaluno[-1]
    else:
        return totaluno[k]


n = int(input())
dos = list(map(int, input().split()))
m = int(input())
uno = list(map(int, input().split()))

dos.sort(reverse=True)
uno.sort(reverse=True)

totaldos = [0]
totaluno = [0]

for i in range(n):
    totaldos.append(totaldos[-1] + dos[i])

for i in range(m):
    totaluno.append(totaluno[-1] + uno[i])

record = 0
for i in range(6):
    record = max(record, cogerdos(i) + cogeruno(10 - 2*i))

print(record)
```

Practicando por temas

Autor: Jacobo Vilella

Teoría

- Bucles

Solución

Este era un problema de simulación. Por lo tanto, bastaba con hacer cuidadosamente lo que decía el enunciado, pasando las instrucciones a código.

Código

C++

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main()
{
    vector<int> vi(4);
    int n;

    for (int i = 0; i < vi.size(); i++) cin >> vi[i];
    cin >> n;

    int t = -1; // Último tema practicado. Nos sirve para saber, en caso de empate de puntuaciones, qué tema tenemos que elegir. La posición -1 es inválida en el vector pero sabemos que en la primera iteración no hay empates por lo que sabemos que se modificará su valor antes de usarse
    for (int i = 0; i < n; i++) {
        // Buscamos si por la mañana hay empate de puntuación entre dos temas
        int v = -1;
        for (int j = 0; j < vi.size(); j++) {
            if (j == t || t == -1) continue;
            if (vi[j] == vi[t]) {
                // Empate a puntuaciones, cogemos el tema que no sea el último practicado
                v = vi[j];
                t = j;
                break; // Como inicialmente no puede haber empate y siempre se desempata inmediatamente solo puede haber dos temas empatados. Eso nos evita ademas que t sea el valor inicial (por ejemplo si t == 1 y vi[0] == vi[1] y no pusiéramos este break en la primera iteración t pasaría a ser 0 pero en la siguiente iteracion del mismo for() t volvería a ser 1
            }
        }
        if (v == -1) { // Si habia un empate no buscamos el de menor puntuación, porque desempatar tiene prioridad
```

```
            // Buscamos el tema con menor puntuación
            t = 0;
            v = vi[0];
            for (int j = 1; j < vi.size(); j++) {
                if (vi[j] < vi[t]) {
                    t = j;
                    v = vi[t];
                }
            }
        }
        // Sumamos al tema elegido las nuevas 5 puntuaciones del día
        for (int j = 0; j < 5; j++) {
            int p;
            cin >> p;
            vi[t] += p;
        }
    }
    for (int i = 0; i < vi.size(); i++) {
        if (i > 0) cout << " ";
        cout << vi[i];
    }
}
```

Java

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {
    public static void main(String[] args) {
        int[] nums = new int[4];
        int n;
        Scanner s = new Scanner(System.in);

        for (int i = 0; i < 4; i++)
            nums[i] = s.nextInt();

        n = s.nextInt();

        int t = -1; // Último tema practicado. Nos sirve para saber, en caso de
        ↪  empate de puntuaciones, qué tema tenemos que elegir. La posición -1 es
        ↪  inválida en el vector pero sabemos que en la primera iteración no hay
        ↪  empates por lo que sabemos que se modificará su valor antes de usarse

        for (int i = 0; i < n; i++) {
            // Buscamos si por la manana hay empate de puntuación entre dos temas
            int v = -1;
            for (int j = 0; j < 4; j++) {
                if (j == t || t == -1) continue;
                if (nums[j] == nums[t]) {
                    // Empate a puntuaciones, cogemos el tema que no sea el último
                    ↪  practicado
```

```
                    v = nums[j];
                    t = j;
                    break; // Como inicialmente no puede haber empate y siempre se desempata inmediatamente solo puede haber dos temas empatados. Eso nos evita ademas que t sea el valor inicial (por ejemplo si t == 1 y nums[0] == nums[1] y no pusiéramos este break en la primera iteracion t pasaría a ser 0 pero en la siguiente iteración del mismo for() t volvería a ser 1
                }
            }
            if (v == -1) { // Si habia un empate no buscamos el de menor puntuación, porque desempatar tiene prioridad
                // Buscamos el tema con menor puntuación
                t = 0;
                v = nums[0];
                for (int j = 1; j < 4; j++) {
                    if (nums[j] < nums[t]) {
                        t = j;
                        v = nums[t];
                    }
                }
            }
            // Sumamos al tema elegido las nuevas 5 puntuaciones del día
            for (int j = 0; j < 5; j++) {
                int p;
                p = s.nextInt();
                nums[t] += p;
            }
        }
        System.out.println(nums[0] + " " + nums[1] + " " + nums[2] + " " + nums[3]);
        s.close();
    }
}
```

Python

```
vi = list(map(int, input().split()))
n = int(input())
t = -1 # Último tema practicado. Nos sirve para saber, en caso de empate de puntuaciones, qué tema tenemos que elegir. La posición -1 es inválida en el vector pero sabemos que en la primera iteración no hay empates, por lo que sabemos que se modificará su valor antes de usarse
for i in range(n):
    # Buscamos si por la mañana hay empate de puntuación entre dos temas
    v = -1
    for j in range(len(vi)):
        if (j == t or t == -1):
            continue
        if (vi[j] == vi[t]):
            # Empate a puntuaciones, cogemos el tema que no sea el último practicado
            v = vi[j]
            t = j
```

```
            break # Como inicialmente no puede haber empate y siempre se desempata inmediatamente solo puede haber dos temas empatados. Eso nos evita ademas que t sea el valor inicial (por ejemplo si t == 1 y vi[0] == vi[1] y no pusiéramos este break en la primera iteración t pasaría a ser 0 pero en la siguiente iteración del mismo for() t volvería a ser 1
    if (v == -1):
        # Si habia un empate no buscamos el de menor puntuación, porque desempatar tiene prioridad
        # Buscamos el tema con menor puntuación
        t = 0;
        v = vi[0];
        for j in range(1, len(vi)):
            if (vi[j] < vi[t]):
                t = j
                v = vi[t]
    # Sumamos al tema elegido las nuevas 5 puntuaciones del día
    p = list(map(int, input().split()))
    vi[t] += sum(p)

print(" ".join(map(str, vi)))
```

Baq la calculadora

Autora: Blanca Huergo

Teoría

- Programación dinámica
- Búsqueda binaria

Solución

Ordenaremos la lista de números de forma ascendente. Luego, haremos otra lista; esta vez la i-ésima posición será el producto hasta el i-ésimo número en la lista original (módulo $10^9 + 7$).

Para cada pregunta, hacemos una búsqueda binaria en la lista original para encontrar la última posición aceptada e imprimimos esa posición en la lista de productos.

Código

C++

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


int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    int t, n, q, c, lo, hi, mid, k;
    cin >> t;
    vector<int> canicas;
    vector<long long int> prod;
    long long int mod = 1e9+7;
    while(t--) {
        cin >> n;
        canicas = vector<int>(n);
        prod = vector<long long int>(n+1);
        for (int i = 0; i < n; i++)
            cin >> canicas[i];
        sort(canicas.begin(), canicas.end());
        prod[0] = 1;
        for (int i = 0; i < n; i++) {
            prod[i+1] = (prod[i]*canicas[i])%mod;
        }
        cin >> q;
        while(q--) {
            cin >> c;
            lo = 0;
            hi = n-1;

```

```
            k = 0;
            while(lo <= hi) {
                mid = lo + (hi-lo)/2;
                if (canicas[mid] <= c)
                    k = lo = mid+1;
                else hi = mid-1;
            }
            cout << prod[k] << '\n';
        }
    }
    return 0;
}
```

Java

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

public class Solution {
        public static void main(String[] args) throws IOException {
            BufferedReader bufferedRead = new BufferedReader(new
            ↪  InputStreamReader(System.in));
            StringTokenizer stringTok = new
            ↪  StringTokenizer(bufferedRead.readLine());
            int t, n, q, c, lo, hi, mid, k;
            t = Integer.parseInt(stringTok.nextToken());
            int[] canicas;
            long[] prod;
            long mod = 1000000007;
            for (int caso = 0; caso < t; caso++) {
                stringTok = new StringTokenizer(bufferedRead.readLine());
                n = Integer.parseInt(stringTok.nextToken());
                canicas = new int[n];
                prod = new long[n+1];
                stringTok = new StringTokenizer(bufferedRead.readLine());
                for (int i = 0; i < n; i++)
                    canicas[i] = Integer.parseInt(stringTok.nextToken());
                Arrays.sort(canicas);
                prod[0] = 1;
                for (int i = 0; i < n; i++) {
                    prod[i+1] = (prod[i]*canicas[i])%mod;
                }
                stringTok = new StringTokenizer(bufferedRead.readLine());
                q = Integer.parseInt(stringTok.nextToken());
                stringTok = new StringTokenizer(bufferedRead.readLine());
                for (int preg = 0; preg < q; preg++) {
                    c = Integer.parseInt(stringTok.nextToken());
                    lo = 0;
                    hi = n-1;
                    k = 0;
                    while(lo <= hi) {
                        mid = lo + (hi-lo)/2;
                        if (canicas[mid] <= c)
                            k = lo = mid+1;
                        else hi = mid-1;
```

```
                    }
                    System.out.println(prod[k]);
                }
            }
        }
}
```

Python

```
t = int(input())
mod = int(1e9+7)
for kkk in range(t):
    n = int(input())
    canicas = list(map(int, input().split()))
    canicas.sort()
    prod = [1]
    for i in range(n):
        prod.append((prod[-1]*canicas[i])%mod)
    q = int(input())
    queries = list(map(int, input().split()))
    for ll in range(q):
        c = queries[ll]
        lo = 0
        hi = n-1
        k = 0
        while(lo <= hi):
            mid = lo + (hi-lo)//2
            if (canicas[mid] <= c):
                k = lo = mid+1
            else:
                hi = mid-1
        print(prod[k])
```

Copiando de la pizarra

Autor: Izan Beltrán

Teoría

- Números primos

Solución

Necesitamos encontrar, si es que existe, un primo P factor de $N+1$ y de $N!+1$. $P <= N+1$, ya que es su factor. También debe de ser factor de $(N+1)!$, pero si lo es de $N+1$, esto ya se cumple. $N!+1$ no puede ser múltiplo de ningún primo entre 2 y N por el $+1$. Esto significa que, en el caso de que P exista, debe de ser igual a $N+1$. Por lo tanto, el ejercicio consiste en comprobar si $N+1$ es primo o no.

Código

C++

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

bool primo(int n) {
    if (n <= 1 || (n % 2 == 0 && n != 2))
        return false;
    int i = 3;
    while(i*i <= n) {
        if (n % i == 0)
            return false;
        i += 2;
    }
    return true;
}

int main() {
    int t;
    cin >> t;
    while (t--) {
        int N;
        cin >> N;
        if (primo(N+1))
            cout << "SI\n";
        else
            cout << "NO\n";
    }
}
```

Java

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

public class Solution {
```

```java
    static boolean esPrimo(int n) {
        if (n <= 1 || (n % 2 == 0 && n > 2))
            return false;
        int i = 3;
        while(i*i <= n) {
            if (n % i == 0)
                return false;
            i += 2;
        }
        return true;
    }

    public static void main(String[] args) {
        int T, n;
        Scanner s = new Scanner (System.in);
        T = s.nextInt();
        for (int caso = 0; caso < T; caso++) {
            n = s.nextInt();
            if (esPrimo(n+1))
                System.out.println("SI");
            else
                System.out.println("NO");
        }
    }
}
```

Python

```python
def esPrimo(x):
    if x % 2 == 0:
        return x == 2
    i = 3
    while i*i <= x:
        if x % i == 0:
            return False
        i += 2
    return True

n = int(input())
for caso in range(n):
    k = int(input())
    if esPrimo(k+1):
        print("SI")
    else:
        print("NO")
```

Oasis de ensueño

Autor: David García Quintas

Teoría

- Programación dinámica
- DFS

Solución

Probamos todas las posibilidades de coger/no coger, ayudados por una DFS. Guardamos los resultados intermedios para evitar recalcularlos.

Código

C++

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

typedef long long ll;

int n;
vector<vector<int>> g;
vector<int> litros;
vector<ll> dpCoger, dpNoCoger; // dpCoger no significa que cojamos, solo que
↪  podemos hacerlo

ll rec(int u, int par, bool puede) {
    if (puede) {
        if (dpCoger[u] != -1)
            return dpCoger[u];
        ll ans1 = rec(u, par, false), ans2 = litros[u];
        for (int v: g[u]) {
            if (v == par)
                continue;
            ans2 += rec(v, u, false);
        }
        return dpCoger[u] = max(ans1, ans2);
    } else {
        if (dpNoCoger[u] != -1)
            return dpNoCoger[u];
        ll ans = 0;
        for (int v: g[u]) {
            if (v == par)
                continue;
            ans += max(rec(v, u, false), rec(v, u, true));
        }
        return dpNoCoger[u] = ans;
    }
```

```cpp
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    int t, u, v;
    cin >> t;
    while(t--) {
        cin >> n;
        g = vector<vector<int>>(n);
        litros = vector<int>(n);
        for (int i = 0; i < n; i++)
            cin >> litros[i];
        for (int i = 0; i < n-1; i++) {
            cin >> u >> v;
            g[u].push_back(v);
            g[v].push_back(u);
        }
        dpCoger.assign(n, -1);
        dpNoCoger.assign(n, -1);
        cout << rec(0, -1, true) << "\n";
    }
    return 0;
}
```

Java

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

public class Solution {
    static int n;
    static int[] val = new int[10000];
    static long[][] dp = new long[10000][2];
    static ArrayList<ArrayList<Integer>> g;

    static long rec() {
        Stack<Integer> nodo = new Stack<Integer>(), padre = new Stack<Integer>(),
        ↪  puedo = new Stack<Integer>();
        nodo.push(0);
        padre.push(-1);
        puedo.push(1);
        while(!nodo.isEmpty()) {
            int u = nodo.peek();
            int par = padre.peek();
            int pue = puedo.peek();
            boolean wait = false;
            for (int v: g.get(u)) {
                if (v != par && dp[v][1] == -1) {
                    nodo.push(v);
                    padre.push(u);
                    puedo.push(1);
                    wait = true;
                }
            }
```

```
            if (pue == 1) {
                for (int v: g.get(u)) {
                    if (v != par && dp[v][0] == -1) {
                        nodo.push(v);
                        padre.push(u);
                        puedo.push(0);
                        wait = true;
                    }
                }
            }
            if (!wait) {
                long ans1 = 0;
                for (int v: g.get(u)) {
                    if (v != par)
                        ans1 += dp[v][1];
                }
                if (pue == 0)
                    dp[u][0] = ans1;
                else {
                    long ans2 = val[u];
                    for (int v: g.get(u)) {
                        if (v != par)
                            ans2 += dp[v][0];
                    }
                    dp[u][1] = Math.max(ans1, ans2);
                }
                nodo.pop();
                padre.pop();
                puedo.pop();
            }
        }
        return dp[0][1];
    }

    public static void main(String[] args) {
        int T;
        Scanner s = new Scanner(System.in);
        T = s.nextInt();
        for (int caso = 0; caso < T; caso++) {
            n = s.nextInt();
            g = new ArrayList<ArrayList<Integer>>();
            for (int i = 0; i < n; i++) {
                val[i] = s.nextInt();
                g.add(new ArrayList<Integer>());
                dp[i][0] = -1;
                dp[i][1] = -1;
            }
            for (int i = 1; i < n; i++) {
                int u = s.nextInt();
                int v = s.nextInt();
                g.get(u).add(v);
                g.get(v).add(u);
            }
            rec();
            System.out.println(dp[0][1]);
```

```
        }
        s.close();
    }
}
```

Python

```
import sys
sys.setrecursionlimit(200000)

def dfs_cnt(x, p):
    res = 1
    for i in t[x]:
        if v != p:
            res = res + dfs_cnt(v, x)
    return res

def solve(x, p, can):
    global dp
    if dp[x][can] >= 0:
        return dp[x][can]
    res = 0
    for v in t[x]:
        if v != p:
            res += solve(v, x, True)
    if can:
        tmp = l[x]
        for v in t[x]:
            if v != p:
                tmp = tmp + solve(v, x, False)
        res = max(res, tmp)

    dp[x][can] = res
    return dp[x][can]

T = int(input())
for i in range(T):
    n = int(input())
    l = input().split(" ")
    l = [int(x) for x in l]

    t = [[] for i in range(n)]

    for i in range(n-1):
        [u, v] = [int(x) for x in input().split(" ")]
        t[u].append(v)
        t[v].append(u)

    dp = [[-1,-1] for i in range(n)]

    print(solve(0, 0, True))
```

Descifrando el manuscrito Voynich

Autor: Izan Beltrán

Teoría

- Backtracking

Solución

Primero, obtenemos las palabras únicas y sus frecuencias. Después, hacemos backtracking por todas las posibilidades de coger/no coger una palabra e imprimimos la mejor solución.

Código

C++

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

using ld = long double;
using vs = vector<string>;
using vd = vector<ld>;
using vi = vector<int>;

ld zipf_dist(vi& times) {
    int m = times.size();
    vd f(m);
    int total = 0;
    for (int x : times) {
        total += x;
    }
    for (int j = 0; j < m; ++j) {
        f[j] = ld(times[j])/total;
    }
    sort(f.begin(), f.end());
    reverse(f.begin(), f.end());
    ld dist = 0;
    for (int i = 1; i < m; ++i) {
        ld diff = f[i] - f[0]/(i+1);
        dist += diff*diff;
    }
    return dist;
}

int main() {
        int n;
        cin >> n;
        map<string,int> times;
        set<string> s;
        for (int i = 0; i < n; ++i) {
            string word;
            cin >> word;
```

```
            times[word]++;
            s.insert(word);
        }
        vs words;
        for (string word : s) {
            words.push_back(word);
        }
        int k = words.size();
        ld sol_dist = 99999999;
        vi sol = vi(k, 99999999);
        for (int mask = 0; mask < (1<<k); ++mask) {
            vi chosen;
            for (int i = 0; i < k; ++i) {
                if (mask & (1<<i)) chosen.push_back(i);
            }
            if (chosen.size() >= 5) {
                int m = chosen.size();
                vi numbers(m);
                for (int i = 0; i < m; ++i) {
                    numbers[i] = times[words[chosen[i]]];
                }
                ld dist = zipf_dist(numbers);
                if (dist < sol_dist or (dist == sol_dist and chosen < sol)) {
                    sol_dist = dist;
                    sol = chosen;
                }
            }
        }
        cout << sol.size();
        for (int i : sol) {
            cout << " " << words[i];
        }
        cout << endl;
}
```

Java

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

public class Solution {
    static ArrayList<String> unique = new ArrayList<String>();
    static int n;
    static ArrayList<Integer> f = new ArrayList<Integer>();
    static ArrayList<String> pal = new ArrayList<String>();
    static HashMap<String, Integer> freq = new HashMap<String, Integer>();
    static double minimo = 1e10;
    static ArrayList<String> sol = new ArrayList<String>();

    static void calcula() {
        if (f.size() < 5)
            return;
        double res = 0.0;
        double total = 0.0;
        for (int i = 0; i < f.size(); i++)
```

```
19              total += f.get(i);
20          int[] ff = new int[f.size()];
21          for (int i = 0; i < f.size(); i++)
22              ff[i] = f.get(i);
23          Arrays.sort(ff);
24          for (int i = 0; i < f.size()/2; i++) {
25              int temp = ff[i];
26              ff[i] = ff[ff.length - i - 1];
27              ff[ff.length - i - 1] = temp;
28          }
29          for (int i = 1; i < ff.length; i++)
30              res += (ff[i] / total - ff[0] / (double) (total * (i + 1))) * (ff[i] /
                ↪  total - ff[0] / (double) (total * (i + 1)));
31          if (res < minimo) {
32              sol = new ArrayList<String>(pal);
33              minimo = res;
34          }
35      }
36
37      static void rec(int i) {
38          if (i == n)
39              calcula();
40          else {
41              rec(i+1);
42              pal.add(unique.get(i));
43              f.add(freq.get(unique.get(i)));
44              rec(i+1);
45              pal.remove(pal.size()-1);
46              f.remove(f.size()-1);
47          }
48      }
49
50      public static void main(String[] args) {
51          Scanner s = new Scanner(System.in);
52          n = s.nextInt();
53          String word;
54          for (int i = 0; i < n; i++) {
55              word = s.next();
56              if (!freq.containsKey(word)) {
57                  freq.put(word, 0);
58                  unique.add(word);
59              }
60              freq.put(word, freq.get(word)+1);
61          }
62          n = unique.size();
63          rec(0);
64          System.out.print(sol.size() + " " + sol.get(0));
65          for (int i =1; i < sol.size(); i++)
66              System.out.print(" " + sol.get(i));
67          s.close();
68      }
69  }
```

Python

```python
m = int(input())
n = 0
palabras = input().split()
ocurrencias = dict()
unicas = []
minimo = 1e10
sol = []
pal = []
f = []

def calcula():
    global sol, minimo, pal, f
    if len(f) < 5:
        return
    res = 0.0
    total = float(sum(f))
    ff = sorted(f, reverse=True)
    for i in range(1, len(ff)):
        res += (ff[i] / total - ff[0] / float(total * (i + 1))) * (ff[i] / total -
        ↪  ff[0] / float(total * (i + 1)))
    if (res < minimo - 1e-9):
        sol = pal.copy()
        minimo = res

def rec(i):
    global n, pal, f
    if (i == n):
        calcula()
    else:
        rec(i+1)
        pal.append(unicas[i])
        f.append(ocurrencias[unicas[i]])
        rec(i+1)
        pal.pop()
        f.pop()


for i in range(m):
    if not (palabras[i] in ocurrencias):
        ocurrencias[palabras[i]] = 0
        unicas.append(palabras[i])
    ocurrencias[palabras[i]] += 1

n = len(unicas)
rec(0)

print(str(len(sol)) + " ", end = '')
print(" ".join(sol))
```