

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, queremos 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++

```
1  #include <cmath>
2  #include <cstdio>
3  #include <vector>
4  #include <iostream>
5  #include <algorithm>
6  using namespace std;
7
8  int coger(vector<int> v, int k) {
9      return v[min(k, (int) v.size() - 1)];
10 }
11
12 int main() {
13     int n, m;
14     vector<int> choc, gomi, sumaC, sumaG;
15     cin >> n;
16     choc = vector<int>(n);
17     for (int i = 0; i < n; i++)
18         cin >> choc[i];
```

```

19     sort(choc.begin(), choc.end(), greater<int>());
20     cin >> m;
21     gomi = vector<int>(m);
22     for (int i = 0; i < m; i++)
23         cin >> gomi[i];
24     sort(gomi.begin(), gomi.end(), greater<int>());
25     sumaC = vector<int>(n+1);
26     sumaC[0] = 0;
27     for (int i = 1; i <= n; i++)
28         sumaC[i] = sumaC[i-1] + choc[i-1];
29     sumaG = vector<int>(m+1);
30     sumaG[0] = 0;
31     for (int i = 1; i <= m; i++)
32         sumaG[i] = sumaG[i-1] + gomi[i-1];
33     int ans = 0;
34     for (int i = 0; i <= 5; i++)
35         ans = max(ans, coger(sumaC, i) + coger(sumaG, 10 - 2*i));
36     cout << ans << '\n';
37     return 0;
38 }

```

Java

```

1  import java.io.*;
2  import java.util.*;
3
4  public class Solution {
5      static int[] choc, gomi, sumaC, sumaG;
6      static int cogerChoc(int k, int n) {
7          return sumaC[Math.min(k, n)];
8      }
9      static int cogerGomi(int k, int m) {
10         return sumaG[Math.min(k, m)];
11     }
12     public static void main(String[] args) {
13         Scanner s = new Scanner(System.in);
14         int n, m;
15         n = s.nextInt();
16         choc = new int[n];
17         sumaC = new int[n+1];
18         for (int i = 0; i < n; i++)
19             choc[i] = s.nextInt();
20         Arrays.sort(choc);
21         for (int i = 0; i < n/2; i++) {
22             int temp = choc[i];
23             choc[i] = choc[n-i-1];
24             choc[n-i-1] = temp;
25         }
26         sumaC[0] = 0;
27         for (int i = 1; i <= n; i++)
28             sumaC[i] = sumaC[i-1] + choc[i-1];
29         m = s.nextInt();
30         gomi = new int[m];
31         sumaG = new int[m+1];
32         for (int i = 0; i < m; i++)

```

```

33     gomi[i] = s.nextInt();
34     Arrays.sort(gomi);
35     for (int i = 0; i < m/2; i++) {
36         int temp = gomi[i];
37         gomi[i] = gomi[m-i-1];
38         gomi[m-i-1] = temp;
39     }
40     sumaG[0] = 0;
41     for (int i = 1; i <= m; i++)
42         sumaG[i] = sumaG[i-1] + gomi[i-1];
43     int ans = 0;
44     for (int i = 0; i <= 5; i++)
45         ans = Math.max(ans, cogerChoc(i, n) + cogerGomi(10 - 2*i, m));
46     System.out.println(ans);
47     s.close();
48 }
49 }

```

Python

```

1  def cogerdos(k):
2      if k >= len(totaldos):
3          return totaldos[-1]
4      else:
5          return totaldos[k]
6
7  def cogerruno(k):
8      if k >= len(totaluno):
9          return totaluno[-1]
10     else:
11         return totaluno[k]
12
13
14  n = int(input())
15  dos = list(map(int, input().split()))
16  m = int(input())
17  uno = list(map(int, input().split()))
18
19  dos.sort(reverse=True)
20  uno.sort(reverse=True)
21
22  totaldos = [0]
23  totaluno = [0]
24
25  for i in range(n):
26      totaldos.append(totaldos[-1] + dos[i])
27
28  for i in range(m):
29      totaluno.append(totaluno[-1] + uno[i])
30
31  record = 0
32  for i in range(6):
33      record = max(record, cogerdos(i) + cogerruno(10 - 2*i))
34
35  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++

```
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  int main()
7  {
8      vector<int> vi(4);
9      int n;
10
11     for (int i = 0; i < vi.size(); i++) cin >> vi[i];
12     cin >> n;
13
14     int t = -1; // Último tema practicado. Nos sirve para saber, en caso de empate
15     ↪ de puntuaciones, qué tema tenemos que elegir. La posición -1 es inválida
16     ↪ en el vector pero sabemos que en la primera iteración no hay empates por
17     ↪ lo que sabemos que se modificará su valor antes de usarse
18     for (int i = 0; i < n; i++) {
19         // Buscamos si por la mañana hay empate de puntuación entre dos temas
20         int v = -1;
21         for (int j = 0; j < vi.size(); j++) {
22             if (j == t || t == -1) continue;
23             if (vi[j] == vi[t]) {
24                 // Empate a puntuaciones, cogemos el tema que no sea el último
25                 ↪ practicado
26                 v = vi[j];
27                 t = j;
28                 break; // Como inicialmente no puede haber empate y siempre se
29                 ↪ desempata inmediatamente solo puede haber dos temas empatados.
30                 ↪ Eso nos evita además que t sea el valor inicial (por ejemplo
31                 ↪ si t == 1 y vi[0] == vi[1] y no pusiéramos este break en la
32                 ↪ primera iteración t pasaría a ser 0 pero en la siguiente
33                 ↪ iteración del mismo for() t volvería a ser 1
34             }
35         }
36     }
37     if (v == -1) { // Si había un empate no buscamos el de menor puntuación,
38         ↪ porque desempatar tiene prioridad
```

```

28         // Buscamos el tema con menor puntuación
29         t = 0;
30         v = vi[0];
31         for (int j = 1; j < vi.size(); j++) {
32             if (vi[j] < vi[t]) {
33                 t = j;
34                 v = vi[t];
35             }
36         }
37     }
38     // Sumamos al tema elegido las nuevas 5 puntuaciones del día
39     for (int j = 0; j < 5; j++) {
40         int p;
41         cin >> p;
42         vi[t] += p;
43     }
44 }
45 for (int i = 0; i < vi.size(); i++) {
46     if (i > 0) cout << " ";
47     cout << vi[i];
48 }
49 }

```

Java

```

1  import java.io.*;
2  import java.util.*;
3  import java.text.*;
4  import java.math.*;
5  import java.util.regex.*;
6
7  public class Solution {
8      public static void main(String[] args) {
9          int[] nums = new int[4];
10         int n;
11         Scanner s = new Scanner(System.in);
12
13         for (int i = 0; i < 4; i++)
14             nums[i] = s.nextInt();
15
16         n = s.nextInt();
17
18         int t = -1; // Último tema practicado. Nos sirve para saber, en caso de
19         ↪ empate de puntuaciones, qué tema tenemos que elegir. La posición -1 es
20         ↪ inválida en el vector pero sabemos que en la primera iteración no hay
21         ↪ empates por lo que sabemos que se modificará su valor antes de usarse
22
23         for (int i = 0; i < n; i++) {
24             // Buscamos si por la mañana hay empate de puntuación entre dos temas
25             int v = -1;
26             for (int j = 0; j < 4; j++) {
27                 if (j == t || t == -1) continue;
28                 if (nums[j] == nums[t]) {
29                     // Empate a puntuaciones, cogemos el tema que no sea el último
30                     ↪ practicado

```

```

27         v = nums[j];
28         t = j;
29         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
                ↳ pudiéramos este break en la primera iteracion t pasaría a
                ↳ ser 0 pero en la siguiente iteración del mismo for() t
                ↳ volvería a ser 1
30     }
31 }
32 if (v == -1) { // Si habia un empate no buscamos el de menor
                ↳ puntuación, porque desempatar tiene prioridad
                ↳ // Buscamos el tema con menor puntuación
33     t = 0;
34     v = nums[0];
35     for (int j = 1; j < 4; j++) {
36         if (nums[j] < nums[t]) {
37             t = j;
38             v = nums[t];
39         }
40     }
41 }
42 }
43 // Sumamos al tema elegido las nuevas 5 puntuaciones del día
44 for (int j = 0; j < 5; j++) {
45     int p;
46     p = s.nextInt();
47     nums[t] += p;
48 }
49 }
50 System.out.println(nums[0] + " " + nums[1] + " " + nums[2] + " " +
                ↳ nums[3]);
51 s.close();
52 }
53 }

```

Python

```

1  vi = list(map(int, input().split()))
2  n = int(input())
3  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
4  for i in range(n):
5      # Buscamos si por la mañana hay empate de puntuación entre dos temas
6      v = -1
7      for j in range(len(vi)):
8          if (j == t or t == -1):
9              continue
10         if (vi[j] == vi[t]):
11             # Empate a puntuaciones, cogemos el tema que no sea el último
                ↳ practicado
12             v = vi[j]
13             t = j

```

```

14         break # Como inicialmente no puede haber empate y siempre se desempata
           ↳ inmediatamente solo puede haber dos temas empatados. Eso nos evita
           ↳ además 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
15     if (v == -1):
16         # Si había un empate no buscamos el de menor puntuación, porque desempatar
           ↳ tiene prioridad
17         # Buscamos el tema con menor puntuación
18         t = 0;
19         v = vi[0];
20         for j in range(1, len(vi)):
21             if (vi[j] < vi[t]):
22                 t = j
23                 v = vi[t]
24         # Sumamos al tema elegido las nuevas 5 puntuaciones del día
25         p = list(map(int, input().split()))
26         vi[t] += sum(p)
27
28     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++

```
1  #include <cmath>
2  #include <cstdio>
3  #include <vector>
4  #include <iostream>
5  #include <algorithm>
6  using namespace std;
7
8
9  int main() {
10     ios::sync_with_stdio(false);
11     cin.tie(NULL);
12     int t, n, q, c, lo, hi, mid, k;
13     cin >> t;
14     vector<int> canicas;
15     vector<long long int> prod;
16     long long int mod = 1e9+7;
17     while(t--) {
18         cin >> n;
19         canicas = vector<int>(n);
20         prod = vector<long long int>(n+1);
21         for (int i = 0; i < n; i++)
22             cin >> canicas[i];
23         sort(canicas.begin(), canicas.end());
24         prod[0] = 1;
25         for (int i = 0; i < n; i++) {
26             prod[i+1] = (prod[i]*canicas[i])%mod;
27         }
28         cin >> q;
29         while(q--) {
30             cin >> c;
31             lo = 0;
32             hi = n-1;
```



```

33     k = 0;
34     while(lo <= hi) {
35         mid = lo + (hi-lo)/2;
36         if (canicas[mid] <= c)
37             k = lo = mid+1;
38         else hi = mid-1;
39     }
40     cout << prod[k] << '\n';
41 }
42 }
43 return 0;
44 }

```

Java

```

1  import java.io.*;
2  import java.util.*;
3
4  public class Solution {
5      public static void main(String[] args) throws IOException {
6          BufferedReader bufferedRead = new BufferedReader(new
7              ↳ InputStreamReader(System.in));
8          StringTokenizer stringTok = new
9              ↳ StringTokenizer(bufferedRead.readLine());
10         int t, n, q, c, lo, hi, mid, k;
11         t = Integer.parseInt(stringTok.nextToken());
12         int[] canicas;
13         long[] prod;
14         long mod = 1000000007;
15         for (int caso = 0; caso < t; caso++) {
16             stringTok = new StringTokenizer(bufferedRead.readLine());
17             n = Integer.parseInt(stringTok.nextToken());
18             canicas = new int[n];
19             prod = new long[n+1];
20             stringTok = new StringTokenizer(bufferedRead.readLine());
21             for (int i = 0; i < n; i++)
22                 canicas[i] = Integer.parseInt(stringTok.nextToken());
23             Arrays.sort(canicas);
24             prod[0] = 1;
25             for (int i = 0; i < n; i++) {
26                 prod[i+1] = (prod[i]*canicas[i])%mod;
27             }
28             stringTok = new StringTokenizer(bufferedRead.readLine());
29             q = Integer.parseInt(stringTok.nextToken());
30             stringTok = new StringTokenizer(bufferedRead.readLine());
31             for (int preg = 0; preg < q; preg++) {
32                 c = Integer.parseInt(stringTok.nextToken());
33                 lo = 0;
34                 hi = n-1;
35                 k = 0;
36                 while(lo <= hi) {
37                     mid = lo + (hi-lo)/2;
38                     if (canicas[mid] <= c)
39                         k = lo = mid+1;
40                     else hi = mid-1;

```

```

39     }
40     System.out.println(prod[k]);
41     }
42     }
43     }
44 }

```

Python

```

1  t = int(input())
2  mod = int(1e9+7)
3  for kkk in range(t):
4      n = int(input())
5      canicas = list(map(int, input().split()))
6      canicas.sort()
7      prod = [1]
8      for i in range(n):
9          prod.append((prod[-1]*canicas[i])%mod)
10     q = int(input())
11     queries = list(map(int, input().split()))
12     for ll in range(q):
13         c = queries[ll]
14         lo = 0
15         hi = n-1
16         k = 0
17         while(lo <= hi):
18             mid = lo + (hi-lo)//2
19             if (canicas[mid] <= c):
20                 k = lo = mid+1
21             else:
22                 hi = mid-1
23     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 $N! + 1$. $P \leq N + 1$, 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++

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  bool primo(int n) {
5      if (n <= 1 || (n % 2 == 0 && n != 2))
6          return false;
7      int i = 3;
8      while(i*i <= n) {
9          if (n % i == 0)
10             return false;
11             i += 2;
12         }
13         return true;
14     }
15
16     int main() {
17         int t;
18         cin >> t;
19         while (t-- > 0) {
20             int N;
21             cin >> N;
22             if (primo(N+1))
23                 cout << "SI\n";
24             else
25                 cout << "NO\n";
26         }
27     }
```

Java

```
1  import java.io.*;
2  import java.util.*;
3
4  public class Solution {
```

```

5     static boolean esPrimo(int n) {
6         if (n <= 1 || (n % 2 == 0 && n > 2))
7             return false;
8         int i = 3;
9         while(i*i <= n) {
10            if (n % i == 0)
11                return false;
12            i += 2;
13        }
14        return true;
15    }
16
17    public static void main(String[] args) {
18        int T, n;
19        Scanner s = new Scanner (System.in);
20        T = s.nextInt();
21        for (int caso = 0; caso < T; caso++) {
22            n = s.nextInt();
23            if (esPrimo(n+1))
24                System.out.println("SI");
25            else
26                System.out.println("NO");
27        }
28    }
29 }

```

Python

```

1  def esPrimo(x):
2      if x % 2 == 0:
3          return x == 2
4      i = 3
5      while i*i <= x:
6          if x % i == 0:
7              return False
8          i += 2
9      return True
10
11 n = int(input())
12 for caso in range(n):
13     k = int(input())
14     if esPrimo(k+1):
15         print("SI")
16     else:
17         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++

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  using namespace std;
5
6  typedef long long ll;
7
8  int n;
9  vector<vector<int>> g;
10 vector<int> litros;
11 vector<ll> dpCoger, dpNoCoger; // dpCoger no significa que cojamos, solo que
   ↳ podemos hacerlo
12
13 ll rec(int u, int par, bool puede) {
14     if (puede) {
15         if (dpCoger[u] != -1)
16             return dpCoger[u];
17         ll ans1 = rec(u, par, false), ans2 = litros[u];
18         for (int v: g[u]) {
19             if (v == par)
20                 continue;
21             ans2 += rec(v, u, false);
22         }
23         return dpCoger[u] = max(ans1, ans2);
24     } else {
25         if (dpNoCoger[u] != -1)
26             return dpNoCoger[u];
27         ll ans = 0;
28         for (int v: g[u]) {
29             if (v == par)
30                 continue;
31             ans += max(rec(v, u, false), rec(v, u, true));
32         }
33         return dpNoCoger[u] = ans;
34     }
}
```

```

35 }
36
37 int main() {
38     ios::sync_with_stdio(false);
39     cin.tie(NULL);
40     int t, u, v;
41     cin >> t;
42     while(t--) {
43         cin >> n;
44         g = vector<vector<int>>(n);
45         litros = vector<int>(n);
46         for (int i = 0; i < n; i++)
47             cin >> litros[i];
48         for (int i = 0; i < n-1; i++) {
49             cin >> u >> v;
50             g[u].push_back(v);
51             g[v].push_back(u);
52         }
53         dpCoger.assign(n, -1);
54         dpNoCoger.assign(n, -1);
55         cout << rec(0, -1, true) << "\n";
56     }
57     return 0;
58 }

```

Java

```

1  import java.io.*;
2  import java.util.*;
3
4  public class Solution {
5      static int n;
6      static int[] val = new int[10000];
7      static long[][] dp = new long[10000][2];
8      static ArrayList<ArrayList<Integer>> g;
9
10     static long rec() {
11         Stack<Integer> nodo = new Stack<Integer>(), padre = new Stack<Integer>(),
12         ↳ puedo = new Stack<Integer>();
13         nodo.push(0);
14         padre.push(-1);
15         puedo.push(1);
16         while(!nodo.isEmpty()) {
17             int u = nodo.peek();
18             int par = padre.peek();
19             int pue = puedo.peek();
20             boolean wait = false;
21             for (int v: g.get(u)) {
22                 if (v != par && dp[v][1] == -1) {
23                     nodo.push(v);
24                     padre.push(u);
25                     puedo.push(1);
26                     wait = true;
27                 }
28             }
29         }
30     }
31 }

```

```

28     if (pue == 1) {
29         for (int v: g.get(u)) {
30             if (v != par && dp[v][0] == -1) {
31                 nodo.push(v);
32                 padre.push(u);
33                 puedo.push(0);
34                 wait = true;
35             }
36         }
37     }
38     if (!wait) {
39         long ans1 = 0;
40         for (int v: g.get(u)) {
41             if (v != par)
42                 ans1 += dp[v][1];
43         }
44         if (pue == 0)
45             dp[u][0] = ans1;
46         else {
47             long ans2 = val[u];
48             for (int v: g.get(u)) {
49                 if (v != par)
50                     ans2 += dp[v][0];
51             }
52             dp[u][1] = Math.max(ans1, ans2);
53         }
54         nodo.pop();
55         padre.pop();
56         puedo.pop();
57     }
58 }
59 return dp[0][1];
60 }
61
62 public static void main(String[] args) {
63     int T;
64     Scanner s = new Scanner(System.in);
65     T = s.nextInt();
66     for (int caso = 0; caso < T; caso++) {
67         n = s.nextInt();
68         g = new ArrayList<ArrayList<Integer>>();
69         for (int i = 0; i < n; i++) {
70             val[i] = s.nextInt();
71             g.add(new ArrayList<Integer>());
72             dp[i][0] = -1;
73             dp[i][1] = -1;
74         }
75         for (int i = 1; i < n; i++) {
76             int u = s.nextInt();
77             int v = s.nextInt();
78             g.get(u).add(v);
79             g.get(v).add(u);
80         }
81         rec();
82         System.out.println(dp[0][1]);

```

```

83     }
84     s.close();
85 }
86 }

```

Python

```

1  import sys
2  sys.setrecursionlimit(200000)
3
4  def dfs_cnt(x, p):
5      res = 1
6      for i in t[x]:
7          if v != p:
8              res = res + dfs_cnt(v, x)
9      return res
10
11 def solve(x, p, can):
12     global dp
13     if dp[x][can] >= 0:
14         return dp[x][can]
15     res = 0
16     for v in t[x]:
17         if v != p:
18             res += solve(v, x, True)
19     if can:
20         tmp = l[x]
21         for v in t[x]:
22             if v != p:
23                 tmp = tmp + solve(v, x, False)
24     res = max(res, tmp)
25
26     dp[x][can] = res
27     return dp[x][can]
28
29 T = int(input())
30 for i in range(T):
31     n = int(input())
32     l = input().split(" ")
33     l = [int(x) for x in l]
34
35     t = [[] for i in range(n)]
36
37     for i in range(n-1):
38         [u, v] = [int(x) for x in input().split(" ")]
39         t[u].append(v)
40         t[v].append(u)
41
42     dp = [[-1,-1] for i in range(n)]
43
44     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++

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  using ld = long double;
5  using vs = vector<string>;
6  using vd = vector<ld>;
7  using vi = vector<int>;
8
9  ld zipf_dist(vi& times) {
10     int m = times.size();
11     vd f(m);
12     int total = 0;
13     for (int x : times) {
14         total += x;
15     }
16     for (int j = 0; j < m; ++j) {
17         f[j] = ld(times[j])/total;
18     }
19     sort(f.begin(), f.end());
20     reverse(f.begin(), f.end());
21     ld dist = 0;
22     for (int i = 1; i < m; ++i) {
23         ld diff = f[i] - f[0]/(i+1);
24         dist += diff*diff;
25     }
26     return dist;
27 }
28
29 int main() {
30     int n;
31     cin >> n;
32     map<string,int> times;
33     set<string> s;
34     for (int i = 0; i < n; ++i) {
35         string word;
36         cin >> word;
```

```

37     times[word]++;
38     s.insert(word);
39 }
40 vs words;
41 for (string word : s) {
42     words.push_back(word);
43 }
44 int k = words.size();
45 ld sol_dist = 99999999;
46 vi sol = vi(k, 99999999);
47 for (int mask = 0; mask < (1<<k); ++mask) {
48     vi chosen;
49     for (int i = 0; i < k; ++i) {
50         if (mask & (1<<i)) chosen.push_back(i);
51     }
52     if (chosen.size() >= 5) {
53         int m = chosen.size();
54         vi numbers(m);
55         for (int i = 0; i < m; ++i) {
56             numbers[i] = times[words[chosen[i]]];
57         }
58         ld dist = zipf_dist(numbers);
59         if (dist < sol_dist or (dist == sol_dist and chosen < sol)) {
60             sol_dist = dist;
61             sol = chosen;
62         }
63     }
64 }
65 cout << sol.size();
66 for (int i : sol) {
67     cout << " " << words[i];
68 }
69 cout << endl;
70 }

```

Java

```

1  import java.io.*;
2  import java.util.*;
3
4  public class Solution {
5      static ArrayList<String> unique = new ArrayList<String>();
6      static int n;
7      static ArrayList<Integer> f = new ArrayList<Integer>();
8      static ArrayList<String> pal = new ArrayList<String>();
9      static HashMap<String, Integer> freq = new HashMap<String, Integer>();
10     static double minimo = 1e10;
11     static ArrayList<String> sol = new ArrayList<String>();
12
13     static void calcula() {
14         if (f.size() < 5)
15             return;
16         double res = 0.0;
17         double total = 0.0;
18         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] /
31             ↪ total - ff[0] / (double) (total * (i + 1)));
32     if (res < minimo) {
33         sol = new ArrayList<String>(pal);
34         minimo = res;
35     }
36 }
37
38 static void rec(int i) {
39     if (i == n)
40         calcula();
41     else {
42         rec(i+1);
43         pal.add(unique.get(i));
44         f.add(freq.get(unique.get(i)));
45         rec(i+1);
46         pal.remove(pal.size()-1);
47         f.remove(f.size()-1);
48     }
49 }
50
51 public static void main(String[] args) {
52     Scanner s = new Scanner(System.in);
53     n = s.nextInt();
54     String word;
55     for (int i = 0; i < n; i++) {
56         word = s.next();
57         if (!freq.containsKey(word)) {
58             freq.put(word, 0);
59             unique.add(word);
60         }
61         freq.put(word, freq.get(word)+1);
62     }
63     n = unique.size();
64     rec(0);
65     System.out.print(sol.size() + " " + sol.get(0));
66     for (int i =1; i < sol.size(); i++)
67         System.out.print(" " + sol.get(i));
68     s.close();
69 }

```

Python

```
1 m = int(input())
2 n = 0
3 palabras = input().split()
4 ocurrencias = dict()
5 unicas = []
6 minimo = 1e10
7 sol = []
8 pal = []
9 f = []
10
11 def calcula():
12     global sol, minimo, pal, f
13     if len(f) < 5:
14         return
15     res = 0.0
16     total = float(sum(f))
17     ff = sorted(f, reverse=True)
18     for i in range(1, len(ff)):
19         res += (ff[i] / total - ff[0] / float(total * (i + 1))) * (ff[i] / total -
20             ↪ ff[0] / float(total * (i + 1)))
21     if (res < minimo - 1e-9):
22         sol = pal.copy()
23         minimo = res
24
25 def rec(i):
26     global n, pal, f
27     if (i == n):
28         calcula()
29     else:
30         rec(i+1)
31         pal.append(unicas[i])
32         f.append(ocurrencias[unicas[i]])
33         rec(i+1)
34         pal.pop()
35         f.pop()
36
37 for i in range(m):
38     if not (palabras[i] in ocurrencias):
39         ocurrencias[palabras[i]] = 0
40         unicas.append(palabras[i])
41     ocurrencias[palabras[i]] += 1
42
43 n = len(unicas)
44 rec(0)
45
46 print(str(len(sol)) + " ", end = '')
47 print(" ".join(sol))
```