Java với lập trình cạnh tranh

Trần Việt Khoa
tvkhoa.husc@gmail.com,
Khoa Công nghệ thông tin, Đại học Khoa học Huế

Ngày 14 tháng 3 năm 2017

1 Giới thiệu

Sử dụng ngôn ngữ lập trình nào cho một bài toán thi là một vấn đề đặt ra, hầu hết các thí sinh đề mong muốn chọn cho mình ngôn ngữ quen thuộc, phù hợp với tư duy giải quyết bài toán (ví dụ sử ngôn ngữ lập trình Pascal ở học sinh phổ thông). Tuy nhiên các cuộc thi lập trình, các trang web thi trực tuyến đều đề xuất các ngôn ngữ lập trình tiên tiến có tính cộng đồng cao như: C, C++, Java, Python. Do đó, xu thế sử dụng các ngôn ngữ trên là tất yếu. Một thuận lợi là với các ngôn ngữ kể trên là một số thư viện được phép sử dụng trong lập trình giúp cho việc triển khai bài giải tốt hơn.

Ví dụ nếu bạn chọn C++ với thư viện STL, một số thuật toán quan trọng cần phải nhớ khi thi như quick sort, binary search v..v, không cần phải cài đặt nữa mà đã có sẵn các hàm sort(), bsearch() được cài đặt sẵn trong thư viện algorithm. Nếu bạn gặp phải một bài toán ví dụ như tính 25! (kết quả là: 15,511,210,043,330,985,984,000,000) con số này là quá lớn các kiểu dữ liệu cơ bản mà các ngôn ngữ lập trình cung cấp, như với C++ kiểu unsigned long long (64 bit) cũng chỉ tính được với 20!. Nhưng nếu bạn dùng Java thì với lớp BigInteger bạn có thể dễ dàng giải quyết vấn đề trên thậm chí với 500!.

```
import java.math.*;
class fact500{
   public static void main(String[] args){
      BigInteger fact= BigInteger.valueOf(1);
      for (int i=2; i<=500; i++)
         fact=fact.multiply(BigInteger.valueOf(i));
      System.out.println(fact);
   }
}
```

Như vậy, việc sử dụng các ngôn ngữ lập trình với các thư viện có sẵn sẽ giúp rất nhiều cho thí sinh khi giải bài toán thi. Ví dụ sau cho thấy các chương trình chỉ giải quyết trong vòng 20 câu lệnh.

Bài toán. Đếm số hoán vị

Phát biểu

Có bao nhiêu xâu khác nhau có thể lập được từ các chữ cái trong từ "SUCCESS".

Dữ liệu vào

- Xâu ký tự, kích thước từ nhỏ hơn hoặc bằng 20 ký tự.

Kết quả

- In số nguyên dương n là số xâu được thành lập.

Ví dụ

Dữ liệu vào	Kết quả
SUCCESS	420

Lời giải Bài toán thuộc lớp bài toán ad hoc, không hạn chế về thời gian hoặc ràng buộc với dữ liệu nhỏ, tức là độ dài của xâu vừa đủ. Bởi vì đây là một bài toán có không gian tìm kiếm lớn O(n!). Sử dụng hàm next_permutation() của thư viện algorithm của C++ (Java không có hàm cài sẵn này) để liệt kê tất cả các hoán vị của một xâu và đếm nó.

```
#include <algorithm>
#include <string>
#include <iostream>
using namespace std;
int main()
{
    int count=0;
    string s = "SUSSCESS";
    sort(s.begin(), s.end());
    do {
        count++;
    } while(std::next_permutation(s.begin(), s.end()));
    cout<<count<<endl;
return 0;
}
```

Các bài toán vừa trình bày ở trên về mặt thuật toán là không khó, tuy nhiên cách giải vận dụng các hàm có sẵn (API) của ngôn ngữ lập trình giúp giải quyết được bài toán một cách nhanh chóng.

2 Java - Ngôn ngữ lập trình cạnh tranh

Java là một ngôn ngữ lập trình mạnh được dùng để lập trình trong nhiều lĩnh vực, trên phương diện giải một bài toán trong kỳ thi lập trình cạnh tranh thì Java không những đáp ứng được các yêu cầu mà còn thể hiện làm một ngôn ngữ mạnh, trong sáng và đơn giản, phù hợp với việc triển khai một bài toán. Một vài so sánh giữa Java với ngôn ngữ lập trình C, C++:

- Các kiểu dữ liệu nguyên thủy được định nghĩa rõ ràng, bạn không bị nhập nhằng giữa int, long, long long như ngôn ngữ C, C++.

- Các kiểu dữ liệu tham chiếu (reference) cực kỳ chặt chẽ, đơn giản. Java không có kiểu dữ liệu như union, struct như trong C, C++. Các kiểu tham chiếu của Java là lớp (class), giao diện (inteface) và mảng. Trong đó dữ liệu biến kiểu mảng luôn được cấp phát động trong Heap khác hẳn với C, C++ khi sử dụng mảng với nhiều cách khác nhau.

- Kiểu dữ liệu BigInteger vô cùng mạnh mẽ trong việc giải những bài toán số vốn rất

phức tạp khi phải cài đặt các phép cộng, trừ, nhân chia trên số nguyên lớn ở C, C++.

- Các kiểu dữ liệu nâng cao như: Danh sách (list, ArrayList, Vector), ngăn xếp (Stack), hàng đợi (Queue), tập hợp (Set), từ điển (Map) với đầy đủ các phương thức giúp tổ chức và giải bài toán thi một cách hiệu quả.

- Loại bỏ đi nhiều kỹ thuật lập trình phức tạp, rắc rối như dùng con trỏ, dùng template, dùng biến global như trong C, C++.

- Java với khả năng bắt lỗi chặt chẽ, giúp lập trình viên dễ dàng tìm ra lỗi của chương trình.

Mặc dù tốc độ thực thi (một yếu tố quan trọng trong giải các bài toán thi) của Java kém hơn so với C, C++ nhưng với các thuận lợi kể trên Java cũng được xem là một ngôn ngữ lập trình được nhiều lập trình viên chọn làm ngôn ngữ giải bài toán thi cạnh tranh.

Để thấy được sức mạnh, tính giản tiện của Java, chúng ta sẽ khảo sát qua các bài toán sau, mỗi bài toán là một số kỹ thuật mà ta sử dụng trong Java.

3 Một số bài toán tiêu biểu

Bài toán 1. Dãy tỉ lệ - Câu 3 đề chuyên tin 2014

Xét dãy số Fibonacci Fn theo định nghĩa $F_1 = F_2 = 1; F_n = F_{n-1} + F_{n-2}, n > 2$. Dãy số nguyên $a_n = a_1, a_2, \ldots, a_n$ được gọi là dãy tỉ lệ của F_n nếu ta có: $\frac{a_1}{F_1} = \frac{a_2}{F_2} = \cdots = \frac{a_n}{F_n}$

Cho a_1 và n, hãy tính tổng $S = a_1 + a_2 + \cdots + a_n$. Do S lớn nên kết quả lấy phần dư khi chia cho $(10^9 + 7)$.

Dữ liệu: vào từ file văn bản RATEEQUA.INP gồm một dòng ghi hai số nguyên a_1 và n ($0 < a_1, n \leq 10^{15}$).

Kết quả: ra file văn bản RATEEQUA.OUT một số nguyên – số dư tìm được.

Ví dụ:

RATEEQUA.INP

3 5

RATEEQUA.OUT

36

Phân tích bài toán:

Sau khi thực hiện một số biến đổi, bài toán quy về tính tổng $S = a_1 \times (f_1 + f_2 + \cdots + f_n)$, trong đó f_i là các số Fibonacci. Đây là một bài toán quen thuộc đối với chương trình dành cho sinh viên CNTT. Tuy nhiên bài toán này có ba vấn đề cần giải quyết:

- Không thể tính tổng S trên bằng thuật toán Naïve mà cần phải tìm công thức cho S.

- Phải tìm một thuật toán cỡ O(log) để tính cho được giá trị của f_n với $n \leq 10^{15}$.

- Vấn đề xử lý phép chia modulo cho $10^9 + 7$.

Lời giải

i) Mấu chốt của bài toán được giải quyết thông qua công thức: $S = f_1 + f_2 + \cdots + f_n = f_{n+2} - 1$. Dễ dàng chứng minh công thức trên bằng quy nạp.

ii) Thuật toán tính phần tử fibonacci thứ n với độ phức tạp O(log)

```
Function Fib(count)
a = 1; b = 0; p = 0; q = 1
While count > 0 Do
If Even(count) Then
p = p.p + q.q; q = 2pq + q.q; count = count / 2;
Else
a = bq + aq + ap; b = bp + aq; count = count -1;
End If
End While
Return b
End Function
```

iii) Vấn đề chia modulo được giải quyết bằng các quy tắc sau:

1. (a + b) % c = ((a % c) + (b % c)) % c

2. (a * b) % c = ((a % c) * (b % c)) % c

3. (a – b) % c = ((a % c) – (b % c)) % c

Mã chương trình:

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

public class RateEqua {
   static long mod = 1000000007L;
   static long fib(long n) {
      long a=1, b=0, p=0, q=1, prev_a, prev_p = 0;
      while(n>0) {
         if (n%2 == 0) {
            prev_p = p; p = (p*p%mod + q*q%mod)%mod;
            q = (2*prev_p*q%mod + q*q%mod)%mod; n /= 2;
         }
         else {
            prev_a = a; a = (b*q%mod + a*q%mod + a*p%mod)%mod;
            b = (b*p%mod + prev_a*q%mod)%mod; n -= 1;
         }
      }
      return b;
   }

   public static void main(String[] args) throws IOException{
      Reader reader = new FileReader("RATEEQUA.INP");
      Writer writer = new FileWriter("RATEEQUA.OUT");
      BufferedReader in = new BufferedReader(reader);
      PrintWriter out = new PrintWriter(writer);

      String line = in.readLine();
      StringTokenizer token = new StringTokenizer(line);

      long a = Long.parseLong(token.nextToken());
      long n = Long.parseLong(token.nextToken());
```

```
        long S=0L;
        long D= System.currentTimeMillis();
        S=fib(n+2)-1;
        S = (a * S)%mod;
        out.println(S);
        long E= System.currentTimeMillis();
        System.out.println(E-D +"ms");
        out.close();
        in.close();
    }
}
```

Bài toán 2. Dấu ngoặc cân bằng.

Cho một xâu ký bao gồm 2 ký tự: dấu ngoặc [] và []. Một xâu kiểu này được gọi là đúng cú pháp nếu:

- Nếu nó là một xâu rỗng.

- Nếu A và B là hai xâu đúng cú pháp thì xâu AB cũng đúng cú pháp.

- Nếu A là xâu đáng cú pháp thì (A) và [A] cũng đúng cú pháp.

Viết chương trình kiểm tra các xâu cho trước có đúng cú pháp hay không, giả sử chiều dài tối đa của xâu là 128.

Dữ liệu: vào từ file văn bản DAUNGOAC.INP với dòng thứ nhất chứa số nguyên dương n, n dòng kế tiếp mỗi dòng chứa một xâu các dấu () và [].

Kết quả: đưa ra file văn bản DAUNGOAC.OUT - In ra "Yes" hoặc "No" tương ứng với mỗi bộ dữ liệu vào.

Ví dụ:

DAUNGOAC.INP

3

([])

(([()]))

([()[]()])()

DAUNGOAC.OUT

Yes

No

Yes

Phân tích bài toán:

Với đề bài toán được phát biểu dưới dạng đệ quy như trên, phương án giải đầu tiên nghĩ đến chính là đệ quy, trong trường hợp này chúng ta sử dụng một cấu trúc lưu trữ là Stack để xử lý, thay vì dùng stack của hệ thống khi triển khai bằng đệ quy. Đây là một bài toán có ý tưởng đẹp.

Mã chương trình:

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

public class DauNgoac {
public static void main(String[] args) throws IOException{
   Reader reader = new FileReader("DAUNGOAC.INP");
   Writer writer = new FileWriter("DAUNGOAC.OUT");
   BufferedReader in = new BufferedReader(reader);
   PrintWriter out = new PrintWriter(writer);
   String input;
   StringBuffer sb = new StringBuffer("");
   String line = in.readLine();
   int T = Integer.parseInt(line);
   for (int i = 0; i < T; i++) {
      boolean iscorrect = true;
      input = in.readLine();
      Stack<Character> st = new Stack<Character>();
      for (int j = 0; j < input.length(); j++){
         if (iscorrect)
            switch (input.charAt(j)){
            case '(': st.push('(');break;
            case '[': st.push('[');break;
            case ']':{
               if(st.isEmpty()){
                  iscorrect=false;break;
               }
               char c = st.pop();
               if (c != '[')iscorrect = false;
               break;
            }
            case ')':{
               if(st.isEmpty()){
                  iscorrect=false; break;
               }
               char c = st.pop();
               if (c != '(')
                  iscorrect = false; break;
            }
            }
      }
      if(iscorrect && st.isEmpty()) sb.append("Yes");
      else sb.append("No");
      sb.append("\n");
   }
   out.print(sb);
   in.close();
   out.close();
}
}
```

Bài toán 3. Bạn bè (Friends) Chu tiên trấn có N công dân, được biết một số người trong thị trấn là bạn bè theo quy tắc sau: "Những người bạn của bạn tôi là bạn bè của tôi". Ví dụ A và B là bạn bè, B và C là bạn bè thì A và C cũng là bạn bè. Hãy đếm xem có bao nhiêu người nằm trong nhóm bạn bè có số lượng người lớn nhất của thị trấn.

Dữ liệu: vào từ file văn bản FRIENDS.INP gồm:

- Dòng đầu tiên là 1 số T là số bộ test case.

- Dòng tiếp theo là số nguyên $N(1 \leq N \leq 30000)$ là số lượng dân và $M(0 \leq M \leq 500000)$ là số lượng cặp người có mối quan hệ bạn bè. M dòng tiếp theo bao gồm 2 số nguyên A và B $(1 \leq A, B \leq N, A \neq B)$ trong đó A và B là bạn bè (Có thể lặp lại).

Kết quả: In ra FRIENDS.OUT, mỗi dòng là một nghiệm S của mỗi bộ dữ liệu chứa số người nằm trong nhóm bạn bè lớn nhất.

Ví dụ:

FRIENDS.INP

2

3 2

1 2

2 3

10 12

1 2

3 1

3 4

5 4

3 5

4 6

5 2

2 1

7 1

1 2

9 10

8 9

FRIENDS.OUT

3

7

Phân tích bài toán:

Đây chính là một bài toán duyệt đồ thị, có thể duyệt theo chiều sâu bằng Stack, duyệt theo chiều rộng bằng Queue, thậm chí có thể dùng cấu trúc các tập rời nhau (Union Disjoint Set). Thuật toán sau dùng Queue để duyệt.

Mã chương trình:

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

```
import java.util.*;
public class Friends {
static class Graph{
   private int V,Count[];

   private boolean Visit[];
   private LinkedList<Integer> adj[];//Mang chua dinh lien ke

   Graph(int v){
      V = v;
      Visit=new boolean[v];
      Count=new int[v];
      adj = new LinkedList[v];
      for (int i=0; i<v; ++i) adj[i] = new LinkedList();
   }

   void addEdge(int v,int w){
      if(!adj[v].contains(w)) adj[v].add(w);
      //Dung contains loc cac phan tu trung
      if(!adj[w].contains(v)) adj[w].add(v);
   }
   //dem so canh lien thong
   int BFS(int s){
      int count=0;
      //Tao queue cho BFS
      LinkedList<Integer> queue = new LinkedList<Integer>();
      //Danh dau dinh hien hanh duoc tham dua vao queue
      Visit[s]=true;
      queue.add(s);
      while (queue.size() != 0){
         //Lay dinh ra khoi queue va in no
         s = queue.poll();
         count++;
      //day cac dinh lan can voi dinh vua tham vao queue
      //chi lay dinh chua danh dau
         Iterator<Integer> i = adj[s].listIterator();
         while (i.hasNext()) {
            int n = i.next();
            if (!Visit[n]){
               Visit[n] = true;
               queue.add(n);
            }
         }
      }
      return count;
   }
   int Max(){ //tim max
      int max=0;
      for(int i=0;i<V;i++) max=Math.max(max,BFS(i));
      if(max==1) max=0;
      return max;
   }
}
public static void main(String[] args) throws IOException {
      Reader reader=new FileReader("FRIENDS.INP");
      Writer writer= new FileWriter("FRIENDS.OUT");
      BufferedReader in= new BufferedReader(reader);
      PrintWriter out=new PrintWriter(writer);
```

```
        int t=Integer.parseInt(in.readLine());
        for(int i=0;i<t;i++){
           StringTokenizer str=new StringTokenizer(in.readLine());
           int n=Integer.parseInt(str.nextToken());
           int m=Integer.parseInt(str.nextToken());
           Graph G=new Graph(n+1);
           for(int j=0;j<m;j++){
              str=new StringTokenizer(in.readLine());
              int v=Integer.parseInt(str.nextToken());
              int w=Integer.parseInt(str.nextToken());
              G.addEdge(v, w);
           }
           out.println(G.Max());
        }
        out.close();
        in.close();
     }
}
```

Bài toán 14. Chơi đẹp (Olympic 2014-bài 3)

Để tạo không khí vui vẻ náo nhiệt, trong buổi giao lưu giữa sinh viên các trường tham dự OLP –ACM, trường đăng cai OLP năm tới đề xuất tổ chức một cuộc thi đấu game online tay đôi giữa sinh viên trường mình với sinh viên trường sở tại. Mỗi trường cử ra một đội n người, tạo thành n cặp đấu, sinh viên cùng trường không đấu với nhau.

Trò chơi được chọn là một trò chơi rất phổ biến, được các bạn trẻ yêu thích, ai cũng biết và đã từng chơi nhiều trước đó. Mọi người đều biết chỉ số năng lực của mình trong trò chơi này và biết rằng nếu đấu tay đôi, ai có năng lực cao hơn sẽ thắng. Trong các trận đấu tay đôi, người thắng sẽ được 1 điểm, người thua – 0 điểm. Thời gian chơi được quy định đủ để phân biệt thắng thua. Các trận hòa sẽ kéo dài vô hạn và sẽ bị hủy kết quả khi hết thời gian. Với tinh thần fair play các bạn trường đề xuất ngồi vào vị trí thi đấu, truy nhập vào hệ thống và gửi về máy chủ chỉ số năng lực của mình. Trưởng đoàn của trường sở tại có 0.5 giây để xử lý thông tin, phân công ai đấu với ai để tổng số điểm thu được là lớn nhất.

Hãy xác định, với cách bố trí tối ưu các cặp đấu, đội của trường sở tại sẽ có bao nhiêu điểm.

Dữ liệu: vào từ file văn bản FAIRPLAY.INP có dạng dòng đầu tiên chứa số nguyên n ($1 \le n \le 105$). Dòng thứ 2 chứa n số nguyên a1, a2, . . ., an, trong đó ai – chỉ số năng lực của người thứ i thuộc đội của trường đề xuất, $1 \le ai \le 10^9$. Dòng thứ 3 chứa n số nguyên b1, b2, . . ., bn, trong đó bi – chỉ số năng lực của người thứ i thuộc đội của trường sở tại, $1 \le bi \le 10^9$.

Kết quả: đưa ra file văn bản FAIRPLAY.OUT. In một số nguyên – số điểm đội trường sở tại có thể đạt được với cách bố trí cặp chơi tối ưu.

Ví dụ:

FAIRPLAY.INP

5

10 15 30 20 25

28 24 20 16 14

FAIRPLAY.OUT

4

Phân tích bài toán:

Bài toán này đơn thuần chỉ là một bài toán sắp xếp, Sắp xếp tăng dần chỉ số năng lực của đội A, sau đó đội B cũng sắp xếp tăng dần về chỉ số năng lực theo một hàng đợi, sau đó tuần tự từng người một ra khỏi hàng đợi và chọn đối thủ tương xứng.

Mã chương trình:

```
import java.io.*;
import java.util.StringTokenizer;
import java.util.Arrays;
import java.util.PriorityQueue;

public class FairPlay {
public static int[] A;
public static int[] B;
public static void main(String[] args) throws IOException{
   //Dinh huong vao ra
   Reader reader = new FileReader("FAIRPLAY.INP");
   Writer writer = new FileWriter("FAIRPLAY.OUT");
   BufferedReader in = new BufferedReader(reader);
   PrintWriter out = new PrintWriter(writer);
   //doc n
   String line = in.readLine();
   int n = Integer.parseInt(line);
   A = new int[n];
   B = new int[n];
   //doc lan luot cac phan tu dua vao mang A
   StringTokenizer stk1=new StringTokenizer(in.readLine()," ");
   for(int i=0;i<n;i++)
      A[i]=Integer.parseInt(stk1.nextToken());
   Arrays.sort(A);
   //doc lan luot cac phan tu dong hai dua vao mang B
   PriorityQueue < Integer > q = new PriorityQueue < Integer > ();
   StringTokenizer stk2=new StringTokenizer(in.readLine()," ");
   for(int i=0;i<n;i++)
      q.add(Integer.parseInt(stk2.nextToken()));
   int i=0, count=0;
   while (!q.isEmpty()){
      int cur = q.remove();
      if (cur > A[i]){
         count++; i++;
      }
   }
   out.println(count);
   in.close();
   out.close();
}
}
```

Bài toán 5. Đếm phần tử

Cho 2 dãy số A và B là các số nguyên dương. A có n phần tử, B có m phần tử. Tìm số lượng các phần tử khác nhau trong dãy B xuất hiện trong dãy A.

Dữ liệu: Vào từ file văn bản COUNT.INP gồm ba dòng:dòng thứ nhất chứa 2 số nguyên dương n, m với 0<m, n<106; 0<x<105. Dòng thứ 2 chứa n số là các số của dãy A. Dòng thứ 3 chứa m số là các số của dãy B.

Kết quả: Ghi ra file văn bản COUNT.OUT gồm một dòng ghi số lượng tìm được.

Ví dụ:

COUNT.INP

3 4 2

2 5 6

7 5 5 6

COUNT.OUT

2

Phân tích bài toán:

Bài toán này sử dụng cấu trúc dữ liệu tập hợp với phép kiểm tra phần tử x có thuộc trong một tập hợp cho trước hay không. Bài toán dễ dàng thực hiện bằng cấu trúc Set.

Mã chương trình:

```
import java.util.*;
import java.io.*;
class Count {
   public static void main(String[] args) throws IOException {
      Reader r=new FileReader("Count.inp");
      Writer w=new FileWriter("Count.out");

      BufferedReader in=new BufferedReader(r);
      PrintWriter out=new PrintWriter(w);
      StringTokenizer s=new StringTokenizer(in.readLine());

      int n=Integer.parseInt(s.nextToken());
      int m=Integer.parseInt(s.nextToken());
      Set<Integer> s1=new HashSet<Integer>(), s2=new HashSet<Integer>();

      s=new StringTokenizer(in.readLine());
      for(int i=0;i<n;i++)
         s1.add(Integer.parseInt(s.nextToken()));

      s=new StringTokenizer(in.readLine());
      for(int i=0;i<m;i++){
         int x=Integer.parseInt(s.nextToken());
         if(s1.contains(x)) s2.add(x);
      }
      System.out.println(s2.size());
      in.close();
      out.close();
   }
}
```

Bài toán 6. Thống kê tần suất.

Viết chương trình thống kê tần suất xuất hiện của các số nguyên trong một dãy số cho trước, chương trình in ra danh sách các số nguyên theo sau là tần suất xuất hiện của nó.

Dữ liệu: vào từ file văn bản TANSUAT.INP chứa một dãy số nguyên (dương, âm hoặc không) độ dài của dãy là tùy ý.

Kết quả: đưa ra file văn bản TANSUAT.OUT bằng cách in ra một dãy các bộ gồm hai phần tử: phần tử thứ nhất là số nguyên xuất hiện trong dãy, số thứ hai là tần suất xuất hiện của nó trong dãy, các bộ in trên từng dòng.

Ví dụ:

TANSUAT.INP

3 1 2 2 1 3 5 3 3 2

TANSUAT.OUT

3 4

1 2

2 3

5 1

Phân tích bài toán:

Bài toán này sử dụng cấu trúc dữ liệu cây tìm kiếm.

Mã Chương trình:

```
import java.io.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

public class TanSuat {
public static void main(String[] args) throws Exception {
   Reader reader = new FileReader("TANSUAT.INP");
   Writer writer = new FileWriter("TANSUAT.OUT");
   BufferedReader br = new BufferedReader(reader);
   PrintWriter pw = new PrintWriter(writer);
   String s;
   HashMap<Integer, Integer> map = new HashMap<Integer, Integer>();
   List<Integer> list = new ArrayList<Integer>();
   while ((s = br.readLine()) != null) {
      String[] line = s.split(" ");
      for (int i = 0; i < line.length; i++) {
         int x = Integer.parseInt(line[i]);
         if (map.containsKey(x)) {
               map.put(x, map.get(x) + 1);
         } else {
               list.add(x);
               map.put(x, 1);
         }
      }
   }
   for (int i : list) {
```

```
            int x = map.get(i);
            pw.println(i + " " + x);
        }
        pw.flush();
        pw.close();
        br.close();
    }
}
```

4 Thực hành

Bài tập 1. Giải bài toán ở địa chỉ http://oj.hueuni.edu.vn/practice/problem/160/details.

Bằng cách sử dụng số nguyên lớn (BigInteger). Bài toán sau khi đã biến đổi công thức với S cần tìm là: $S = (1 + BigInt(m) * BigInt(m + 1) * BigInt(2 * m + 1) * 8/3 + 2 * BigInt(m).pow(2) + 6 * m)\%moduloP$

Bài tập 2. Giải bài toán http://oj.hueuni.edu.vn/practice/problem/9/details. Sử dụng Queue với phép duyệt BFS.