## Bài tập ctdlgt-bt04

### Bài 1 (Solved)

Lấy mã nguồn chương trình bài toán ThreeSum (ThreeSum.java, ThreeSumFast.java, và dữ liệu 1Kints.txt, 2Kints.txt, 4Kints.txt, 8Kints.txt) tại [link mã nguồn môn học](http://algs4.cs.princeton.edu/code/). Tự chạy lại chương trình với dữ liệu test. Sau đó đọc mã nguồn.

**Chú ý**: Việc lấy thư viện [https://algs4.cs.princeton.edu/code/algs4.jar](http://algs4.cs.princeton.edu/code/algs4.jar) sử dụng trong chương trình của tài liệu môn và cách cài đặt:

* Mã nguồn cần import thư viện, bằng cách thêm vào đầu:  
   import edu.princeton.cs.algs4.In;  
   import edu.princeton.cs.algs4.StdOut;  
   import edu.princeton.cs.algs4.Stopwatch;  
   và bỏ đi dòng:  
   package edu.princeton.cs.algs4;
* Trong Intellij, thêm thư viện algs4.jar:  
   Click File from File menu.  
   Project Structure ( CTRL + SHIFT + ALT + S on Windows/Linux, ⌘ + ; on Mac OS X)  
   Select Modules at the left panel.  
   Dependencies tab.  
   Add... → Project Library → Attach Jar.
* Trong Intellij, thêm tham số dòng lệnh (là file dữ liệu 1Kints.txt) khi chạy chương trình:  
   Click Run menu. Chọn Edit Configurations ...  
   Tại Program arguments:, gõ vào 1Kints.txt  
   (Hoặc có thể sửa trực tiếp mã nguồn để đưa vào đường dẫn và tên file dữ liệu mỗi lần chạy thử.)
* Cách biên dịch và chạy sử dụng dòng lệnh, với chương ThreeSum.java và dữ liệu 1Kints:  
   Windows:  
   javac -cp .;algs4.jar ThreeSum.java  
   java -cp .;algs4.jar ThreeSum 1Kints.txt  
   Linux:  
   javac -cp .:algs4.jar ThreeSum.java  
   java -cp .:algs4.jar ThreeSum 1Kints.txt
* Cách đầy đủ với CLASSPATH: xem [tại đây](http://algs4.cs.princeton.edu/code/) - phần "Installing the textbook libraries".

### Bài 2 (Solved)

[Arrays - DS](https://www.hackerrank.com/challenges/arrays-ds)

Tóm tắt: Cho số n là kích cỡ của mảng, và sau đó là n phần tử của mảng. Hãy in ra các phần từ của mảng đã cho, nhưng theo thứ tự ngược lại.

class Result {

/\*

\* Complete the 'reverseArray' function below.

\*

\* The function is expected to return an INTEGER\_ARRAY.

\* The function accepts INTEGER\_ARRAY a as parameter.

\*/

public static List<Integer> reverseArray(List<Integer> a) {

// Write your code here

List <Integer> reverse = new ArrayList<Integer>();

for (int i = a.size() - 1; i >= 0; i--) {

reverse.add(a.get(i));

}

return reverse;

}

}

### Bài 3 (Solved)

[2D Array - DS](https://www.hackerrank.com/challenges/2d-array)

Tóm tắt: Cho ma trận 2 chiều 6x6. Trong đó có 16 hình đồng hồ cát (hourglass), mỗi đồng hồ tạo từ 7 con số. Mỗi đồng hồ như vậy có tổng của 7 con số tạo thành. Hãy tìm và in ra tổng lớn nhất.

class Result {

/\*

\* Complete the 'hourglassSum' function below.

\*

\* The function is expected to return an INTEGER.

\* The function accepts 2D\_INTEGER\_ARRAY arr as parameter.

\*/

public static int hourglassSum(List<List<Integer>> arr) {

// Write your code here

int max = -1000;

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

int tmp = arr.get(i).get(j) + arr.get(i).get(j+1) + arr.get(i).get(j+2) + arr.get(i+1).get(j+1) + arr.get(i+2).get(j) + arr.get(i+2).get(j+1) + arr.get(i+2).get(j+2);

if (tmp > max) max = tmp;

}

}

return max;

}

}

### 

### Bài 4 (Solved)

[Sherlock and Array](https://www.hackerrank.com/challenges/sherlock-and-array)

Tóm tắt: Cho mảng có n phần tử (a\_0, …, a\_n-1). Có hay không chỉ số i để sao cho tổng trái bằng tổng phải (không tính số a\_i), tức (a\_0 + … + a\_i-1) = (a\_i+1 + … + a\_n-1). Nếu có thì in ra “YES", nếu không thì in ra “NO".

class Result {

/\*

\* Complete the 'balancedSums' function below.

\*

\* The function is expected to return a STRING.

\* The function accepts INTEGER\_ARRAY arr as parameter.

\*/

public static String balancedSums(List<Integer> arr) {

// Write your code here

int sum = 0;

for (int i = 0; i < arr.size(); i++) sum += arr.get(i);

int left = 0;

for (int i = 0; i < arr.size(); i++)

{

if (sum - left - arr.get(i) == left) return "YES";

else left += arr.get(i);

}

return "NO";

}

}

### 

### Bài 5 (Solved)

[Closest Numbers](https://www.hackerrank.com/challenges/closest-numbers)

Tóm tắt: Cho một dãy các số chưa được sắp. Giữa hai số bất kỳ trong dãy, chúng ta có giá trị tuyệt đối của hiệu giữa chúng. Hãy tìm và in ra những cặp số có giá trị tuyệt đối chênh lệch trên là nhỏ nhất. Nếu có nhiều cặp như vậy thì in hết ra.

class Result {

/\*

\* Complete the 'closestNumbers' function below.

\*

\* The function is expected to return an INTEGER\_ARRAY.

\* The function accepts INTEGER\_ARRAY arr as parameter.

\*/

public static List<Integer> closestNumbers(List<Integer> arr) {

// Write your code here

int min\_diff = 32767; //INT MAX

Collections.sort(arr);

List <Integer> result = new ArrayList<Integer>();

for (int i = 1; i < arr.size(); i++)

{

if (arr.get(i) - arr.get(i-1) < min\_diff) {

min\_diff = arr.get(i) - arr.get(i-1);

}

}

for (int i = 1; i < arr.size(); i++) {

if (arr.get(i) - arr.get(i-1) == min\_diff)

{

result.add(arr.get(i-1));

result.add(arr.get(i));

}

}

return result;

}

}

### Bài 6 (Solved)

[Pairs](https://www.hackerrank.com/challenges/pairs)

Tóm tắt: Cho N số nguyên. Hãy đếm và in ra số các cặp số có chênh lệch giữa chúng là K.

class Result {

/\*

\* Complete the 'pairs' function below.

\*

\* The function is expected to return an INTEGER.

\* The function accepts following parameters:

\* 1. INTEGER k

\* 2. INTEGER\_ARRAY arr

\*/

public static int pairs(int k, List<Integer> arr) {

// Write your code here

int count = 0;

Collections.sort (arr);

for (int i = 0; i < arr.size() - 1; i++)

{

for (int j = i + 1; j < arr.size(); j++)

{

if (arr.get(j) - arr.get(i) == k) count ++;

else if (arr.get(j) - arr.get(i) > k) break;

}

}

return count;

}

}

### Bài 7 (Solved)

Viết lại chương trình ThreeSum (tìm bộ 3 số có tổng bằng 0 - xem Bài 1 ở trên), nhưng với thời gian ~N^2 (N bình phương).

**Chú ý**: Trong Java có sẵn phương thức **sort** giúp sắp xếp các phần tử của mảng. [Xem ví dụ tại đây.](http://www.tutorialspoint.com/java/util/arrays_sort_int.htm)

import java.util.\*;

public class ThreeSumFast {

public static int ThreeSum(int[] arr)

{

int count = 0;

Arrays.sort(arr);

for (int i = 0; i < arr.length - 2; i++)

{

int l = i+1;

int r = arr.length - 1;

while (l < r)

{

if (arr[i] + arr[l] + arr[r] == 0) {

count++;

l++;

r--;

}

else if (arr[i] + arr[l] + arr[r] < 0) l++;

else r--;

}

}

return count;

}

public static void main (String[] args) {

int[] arr = {-1, -2, -3, 0, 1, 2 ,3};

System.out.print(ThreeSum(arr)) ;

}

}

// -3 -2 -2- 1- 0 1 2 3