package AmazonQuestions;

/\*

Amazon | OA 2019 | Optimal Utilization

Given 2 lists a and b. Each element is a pair of integers where the first integer represents the unique id and the second integer represents a value. Your task is to find an element from a and an element form b such that the sum of their values is less or equal to target and as close to target as possible. Return a list of ids of selected elements. If no pair is possible, return an empty list.

Example 1:

Input:

a = [[1, 2], [2, 4], [3, 6]]

b = [[1, 2]]

target = 7

Output: [[2, 1]]

Explanation:

There are only three combinations [1, 1], [2, 1], and [3, 1], which have a total sum of 4, 6 and 8, respectively.

Since 6 is the largest sum that does not exceed 7, [2, 1] is the optimal pair.

Example 2:

Input:

a = [[1, 3], [2, 5], [3, 7], [4, 10]]

b = [[1, 2], [2, 3], [3, 4], [4, 5]]

target = 10

Output: [[2, 4], [3, 2]]

Explanation:

There are two pairs possible. Element with id = 2 from the list `a` has a value 5, and element with id = 4 from the list `b` also has a value 5.

Combined, they add up to 10. Similarily, element with id = 3 from `a` has a value 7, and element with id = 2 from `b` has a value 3.

These also add up to 10. Therefore, the optimal pairs are [2, 4] and [3, 2].

Example 3:

Input:

a = [[1, 8], [2, 7], [3, 14]]

b = [[1, 5], [2, 10], [3, 14]]

target = 20

Output: [[3, 1]]

Example 4:

Input:

a = [[1, 8], [2, 15], [3, 9]]

b = [[1, 8], [2, 11], [3, 12]]

target = 20

Output: [[1, 3], [3, 2]]

\*/

import java.util.\*;

public class OptimalUtilization {

public List<Integer[]> optimalUtilization\_2(List<Integer[]> a, List<Integer[]> b, int target) {

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

if (a.size() == 0 || b.size() == 0)

return result;

Collections.sort(a, new Comparator<Integer[]>() {

@Override

public int compare(Integer[] o1, Integer[] o2) {

return o1[1] - o2[1];

}

});

Collections.sort(b, new Comparator<Integer[]>() {

@Override

public int compare(Integer[] o1, Integer[] o2) {

return o1[1] - o2[1];

}

});

int i = 0;

int j = b.size() - 1;

int min = Integer.MIN\_VALUE;

while (i < a.size() && j >= 0) {

int sum = a.get(i)[1] + b.get(j)[1];

if (sum > target) {

j--;

} else {

if (min <= sum) {

if (min < sum) {

min = sum;

result.clear();

}

Integer[] tmp = new Integer[]{a.get(i)[0], b.get(j)[0]};

result.add(tmp);

int index = j - 1;

while (index >= 0 && b.get(index)[1] == b.get(index + 1)[1]) {

Integer[] newTmp = new Integer[]{a.get(i)[0], b.get(index)[0]};

result.add(newTmp);

index--;

}

}

i++;

}

}

return result;

}

}

/\*

if (diff == 0) {

if (newDiff == 0) {

Integer[] tmp = new Integer[]{a.get(i)[0], b.get(j)[0]};

result.add(tmp);

// diff = Math.abs(newDiff);

i++;

j--;

}

else{

if (newDiff < 0) {

i++;

} else {

j--;

}

}

}

else {

if (Math.abs(newDiff) < diff) {

result.clear();

diff = Math.abs(newDiff);

Integer[] tmp = new Integer[]{a.get(i)[0], b.get(j)[0]};

result.add(tmp);

}

if (newDiff < 0) {

i++;

} else {

j--;

}

}

}