Nama : Meggy A Messah

NIM : 1306082038

Mata Kuliah : Riset Operasi

import java.util.Arrays;  
import static java.util.[Arrays](http://www.google.com/search?hl=en&q=allinurl%3Aarrays+java.sun.com&btnI=I%27m%20Feeling%20Lucky).stream;  
import java.util.concurrent.\*;  
   
public class VogelsApproximationMethod {  
   
 final static int[] demand = {30, 20, 70, 30, 60};  
 final static int[] supply = {50, 60, 50, 50};  
 final static int[][] costs = {{16, 16, 13, 22, 17}, {14, 14, 13, 19, 15},  
 {19, 19, 20, 23, 50}, {50, 12, 50, 15, 11}};  
   
 final static int nRows = supply.length;  
 final static int nCols = demand.length;  
   
 static boolean[] rowDone = new boolean[nRows];  
 static boolean[] colDone = new boolean[nCols];  
 static int[][] result = new int[nRows][nCols];  
   
 static ExecutorService es = Executors.newFixedThreadPool(2);  
   
 public static void main([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky)[] args) throws [Exception](http://www.google.com/search?hl=en&q=allinurl%3Aexception+java.sun.com&btnI=I%27m%20Feeling%20Lucky) {  
 int supplyLeft = stream(supply).sum();  
 int totalCost = 0;  
   
 while (supplyLeft > 0) {  
 int[] cell = nextCell();  
 int r = cell[0];  
 int c = cell[1];  
   
 int quantity = [Math](http://www.google.com/search?hl=en&q=allinurl%3Amath+java.sun.com&btnI=I%27m%20Feeling%20Lucky).min(demand[c], supply[r]);  
 demand[c] -= quantity;  
 if (demand[c] == 0)  
 colDone[c] = true;  
   
 supply[r] -= quantity;  
 if (supply[r] == 0)  
 rowDone[r] = true;  
   
 result[r][c] = quantity;  
 supplyLeft -= quantity;  
   
 totalCost += quantity \* costs[r][c];  
 }  
   
 stream(result).forEach(a -> [System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println([Arrays](http://www.google.com/search?hl=en&q=allinurl%3Aarrays+java.sun.com&btnI=I%27m%20Feeling%20Lucky).toString(a)));  
 [System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Total cost: " + totalCost);  
   
 es.shutdown();  
 }  
   
 static int[] nextCell() throws [Exception](http://www.google.com/search?hl=en&q=allinurl%3Aexception+java.sun.com&btnI=I%27m%20Feeling%20Lucky) {  
 Future<int[]> f1 = es.submit(() -> maxPenalty(nRows, nCols, true));  
 Future<int[]> f2 = es.submit(() -> maxPenalty(nCols, nRows, false));  
   
 int[] res1 = f1.get();  
 int[] res2 = f2.get();  
   
 if (res1[3] == res2[3])  
 return res1[2] < res2[2] ? res1 : res2;  
   
 return (res1[3] > res2[3]) ? res2 : res1;  
 }  
   
 static int[] diff(int j, int len, boolean isRow) {  
 int min1 = [Integer](http://www.google.com/search?hl=en&q=allinurl%3Ainteger+java.sun.com&btnI=I%27m%20Feeling%20Lucky).MAX\_VALUE, min2 = [Integer](http://www.google.com/search?hl=en&q=allinurl%3Ainteger+java.sun.com&btnI=I%27m%20Feeling%20Lucky).MAX\_VALUE;  
 int minP = -1;  
 for (int i = 0; i < len; i++) {  
 if (isRow ? colDone[i] : rowDone[i])  
 continue;  
 int c = isRow ? costs[j][i] : costs[i][j];  
 if (c < min1) {  
 min2 = min1;  
 min1 = c;  
 minP = i;  
 } else if (c < min2)  
 min2 = c;  
 }  
 return new int[]{min2 - min1, min1, minP};  
 }  
   
 static int[] maxPenalty(int len1, int len2, boolean isRow) {  
 int md = [Integer](http://www.google.com/search?hl=en&q=allinurl%3Ainteger+java.sun.com&btnI=I%27m%20Feeling%20Lucky).MIN\_VALUE;  
 int pc = -1, pm = -1, mc = -1;  
 for (int i = 0; i < len1; i++) {  
 if (isRow ? rowDone[i] : colDone[i])  
 continue;  
 int[] res = diff(i, len2, isRow);  
 if (res[0] > md) {  
 md = res[0]; // max diff  
 pm = i; // pos of max diff  
 mc = res[1]; // min cost  
 pc = res[2]; // pos of min cost  
 }  
 }  
 return isRow ? new int[]{pm, pc, mc, md} : new int[]{pc, pm, mc, md};  
 }  
}