//Bessie the running cow that will run in John's Farm

//N checkpoints 3 <= N <= 100,000 --> 1 to N

//can skip at least one checkpoint at maximum

//(x1, y1) and (x2, y2)

//"Manhattan" Distance: |x1-x2| + |y1-y2|

//Inputs

/\*

 \* first line gives the value of N.

 \* next N lines each contain two space-separated integers, x and y,

 \*

 \* ex)

 \* 4

 \* 0 0

 \* 8 3

 \* 11 -1

 \* 10 0

 \*/

//Outputs:

//minimum distance that Bessie can run by skipping up to one checkpoint.

/\*

 \* Distance

 \* 11

 \* 7

 \* 3

 \*

\*/

//java

//Set Up

import java.io.\*;

import java.util.\*;

public class marathon {

public static void main(String[] args) throws IOException {

        BufferedReader r = new BufferedReader(new FileReader("marathon.in"));

        PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter("marathon.out")));

        StringTokenizer st = new StringTokenizer(r.readLine());

        int n = Integer.parseInt(st.nextToken());

        st = new StringTokenizer(r.readLine());

        //what do I want to store in this array?

        /\*

         \* [(x1,y1) , (x2, y2), (x3, y3), (x4, y4), (xn, yn)]

         \*

        \*/

        // int[] ar = new int[n];

        /\*

         \* [1,

            3,

            4]

            [(0,0)

            ,(8,3)

            ,(11,-1),etc.

            ]

            2D array:

        \*/

        //2d Array

        //int[][] 2DArray = new int[row][col];

        int [][] twoDArray = new int[n][2];

        //put points in an array

        for(int i = 0; i < n; i++){

          int x = Integer.parseInt(st.nextToken());

          int y = Integer.parseInt(st.nextToken());

          twoDArray[i][0] = x; // at row i the col of 0 = x

          twoDArray[i][1] = y;

          if(i == n-1){

            break;

          }

          st = new StringTokenizer(r.readLine());

        }

        int total = 0; // 0 , Integer.MIN\_VALUE, Integer.MAX\_VALUE;

        //"Manhattan" Distance: |x1-x2| + |y1-y2|

        //calculate distance between two points and add it to total

        for(int i = 0; i < n - 1; i++){

          //distance = Math.abs(x1 - x2) + Math.abs(y1 - y2);

          int distance = Math.abs(twoDArray[i][0] - twoDArray[i+1][0]) + Math.abs(twoDArray[i][1] - twoDArray[i+1][1]);

          total += distance;

        }

        //Minumum value to return

        int min = Integer.MAX\_VALUE; //--> inf

        //Find Routes

        for(int i = 0; i < n-2; i++){

          //int route = total - (point 1 to pt 2) - (pt 2 to pt 3) + (pt 1 to pt 3)

          int dist1to2 = Math.abs(twoDArray[i][0] - twoDArray[i+1][0]) + Math.abs(twoDArray[i][1] - twoDArray[i+1][1]);

          int dist2to3 = Math.abs(twoDArray[i+1][0] - twoDArray[i+2][0]) + Math.abs(twoDArray[i+1][1] - twoDArray[i+2][1]);

          int dist1to3 = Math.abs(twoDArray[i][0] - twoDArray[i+2][0]) + Math.abs(twoDArray[i][1] - twoDArray[i+2][1]);

          int route = total - dist1to2 - dist2to3 + dist1to3;

          min = Math.min(min, route);

        }

        //return value

        pw.println(min);

        r.close();

        pw.close();

    }

}