Daily Coding Problem #100

Problem

This problem was asked by Google.

You are in an infinite 2D grid where you can move in any of the 8 directions:

```
 (x,y) to
    (x+1, y),
    (x - 1, y),
    (x, y+1),
    (x, y-1),
    (x-1, y-1),
    (x+1,y+1),
    (x-1,y+1),
    (x+1,y-1)
```

You are given a sequence of points and the order in which you need to cover the points. Give the minimum number of steps in which you can achieve it. You start from the first point.

Example:

```
Input: [(0, 0), (1, 1), (1, 2)]
Output: 2
```

It takes 1 step to move from `(0, 0)` to `(1, 1)`. It takes one more step to move from `(1, 1)` to `(1, 2)`.

Solution

We can see that the minimum number of steps would be to walk as many diagonal steps as possible, and then walk directly to the second point. If we were to walk directly vertically and then horizontally, it would be a greater number of steps. We can break down the diagonal and vertical/horizontal components by taking the adding the minimum of the vertical or horizontal differences with the remaining distance.

```
// X and Y co-ordinates of the points in order.
// Each point is represented by (X.get(i), Y.get(i))
public int coverPoints(ArrayList<Integer> X, ArrayList<Integer> Y) {
    int totalDistance = 0;
    for (int i = 1; i < X.size(); i++) {
        totalDistance += getDistance(X.get(i - 1), Y.get(i - 1), X.get(i),
Y.get(i));
    }
    return totalDistance;
}

private int getDistance(int x1, int y1, int x2, int y2) {
    /* Get diagonal distance component */
    int dist1 = (int)Math.min(Math.abs(x2 - x1), Math.abs(y2 - y1));
    /* Get horizontal/vertical distance component */
    int dist2 = (int)Math.max(Math.abs(x2 - x1), Math.abs(y2 - y1)) - dist1;
    return dist1 + dist2;
}
```

Or, we can simply take the maximum of the vertical and horizontal distances -- this is the total distance.

```
// X and Y co-ordinates of the points in order.
// Each point is represented by (X.get(i), Y.get(i))
public int coverPoints(ArrayList<Integer> X, ArrayList<Integer> Y) {
    int totalDistance = 0;
    for (int i = 1; i < X.size(); i++) {
        totalDistance += getDistance(X.get(i - 1), Y.get(i - 1), X.get(i),
Y.get(i));
    }
    return totalDistance;
}
```

```
private int getDistance(int x1, int y1, int x2, int y2) {
    return (int)Math.max(Math.abs(x2 - x1), Math.abs(y2 - y1));
}
```