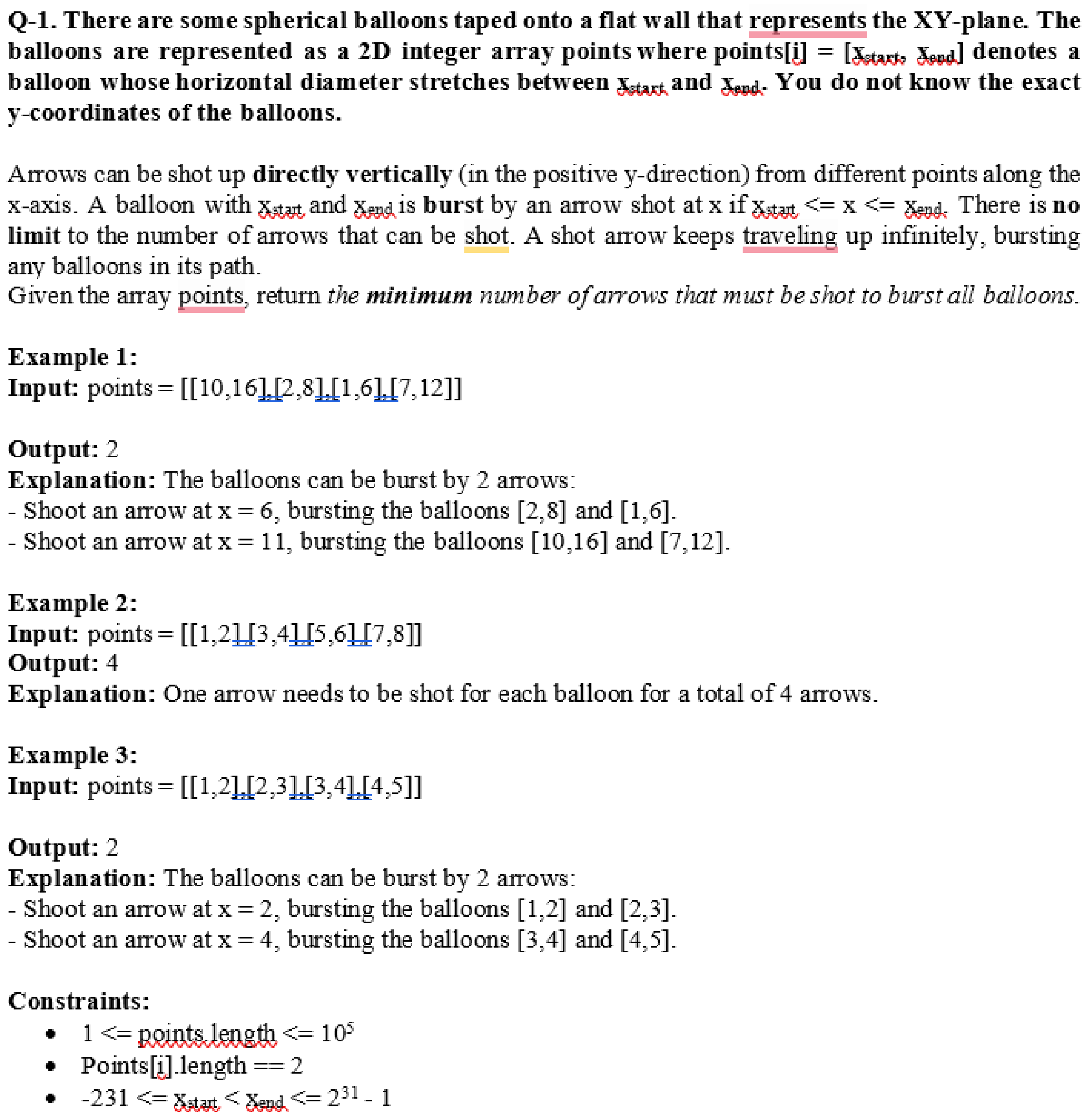
# ODD: Q1



**Q-1. There are some spherical balloons taped onto a flat wall that represents the XY-plane. The balloons are represented as a 2D integer array points where points[i] = [$x_{start}$, $x_{end}$] denotes a balloon whose horizontal diameter stretches between $x_{start}$ and $x_{end}$. You do not know the exact y-coordinates of the balloons.**

Arrows can be shot up **directly vertically** (in the positive y-direction) from different points along the x-axis. A balloon with $x_{start}$ and $x_{end}$ is **burst** by an arrow shot at x if $x_{start}$ <= x <= $x_{end}$. There is **no limit** to the number of arrows that can be shot. A shot arrow keeps traveling up infinitely, bursting any balloons in its path.
Given the array points, return *the **minimum** number of arrows that must be shot to burst all balloons.*

**Example 1:**
**Input:** points = [[10,16],[2,8],[1,6],[7,12]]

**Output:** 2
**Explanation:** The balloons can be burst by 2 arrows:
- Shoot an arrow at x = 6, bursting the balloons [2,8] and [1,6].
- Shoot an arrow at x = 11, bursting the balloons [10,16] and [7,12].

**Example 2:**
**Input:** points = [[1,2],[3,4],[5,6],[7,8]]
**Output:** 4
**Explanation:** One arrow needs to be shot for each balloon for a total of 4 arrows.

**Example 3:**
**Input:** points = [[1,2],[2,3],[3,4],[4,5]]

**Output:** 2
**Explanation:** The balloons can be burst by 2 arrows:
- Shoot an arrow at x = 2, bursting the balloons [1,2] and [2,3].
- Shoot an arrow at x = 4, bursting the balloons [3,4] and [4,5].

**Constraints:**

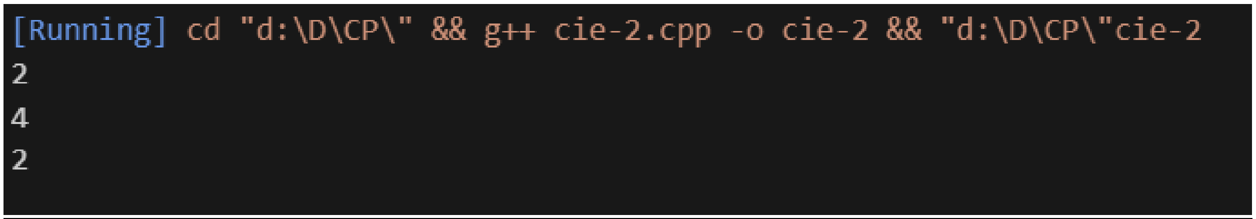
- 1 <= points.length <= $10^5$
- Points[i].length == 2
- -231 <= $x_{start}$ < $x_{end}$ <= $2^{31}$ - 1

**CODE:**

```
#include <iostream>
#include <vector>

using namespace std;
class balloons{
  int xst;
  int xend;

public:
  balloons(int xstart,int xende){
    this->xend=xende;
    this->xst=xstart;
  }

  int burst(vector<balloons> v){
    int count=1;
    int end=v[0].xend;
    for(int i=1;i<v.size();i++){
      if(v[i].xst>end){
        count++;
        end=v[i].xend;
      }
      else{
        end=min(end,v[i].xend);
      }
    }
    return count;
  }
};
int main() {
  vector<balloons> v1,v2,v3;
```

```
    balloons b1(1,6);
    v1.push_back(balloons(1,6));
    v1.push_back(balloons(2,8));
    v1.push_back(balloons(7,12));
    v1.push_back(balloons(10,16));
    cout<<b1.burst(v1)<<endl;

    balloons b2(1,2);
    v2.push_back(balloons(1,2));
    v2.push_back(balloons(3,4));
    v2.push_back(balloons(5,6));
    v2.push_back(balloons(7,8));
    cout<<b2.burst(v2)<<endl;

    balloons b3(1,2);
    v3.push_back(balloons(1,2));
    v3.push_back(balloons(2,3));
    v3.push_back(balloons(3,4));
    v3.push_back(balloons(4,5));
    cout<<b3.burst(v3)<<endl;

    return 0;

}
```

**OUTPUT:**



```
[Running] cd "d:\D\CP\" && g++ cie-2.cpp -o cie-2 && "d:\D\CP\"cie-2
2
4
2
```