deep se
dependable evolvable pervasive software engineering group

Exercise Session – Interpolation

Federica Filippini

Politecnico di Milano

federica.filippini@polimi.it

POLITECNICO DI MILANO

Goal

- Implement a library that, given a vector of points, computes their **interpolation** in a **two-dimensional Euclidean space** with **three different methods.**

- The method **`double interpolate(double x) const;`** receives as input the `x` coordinate of a point and returns the corresponding `y` value, according to the chosen method. **If `x` does not belong to the domain**, the method should return `NaN`.

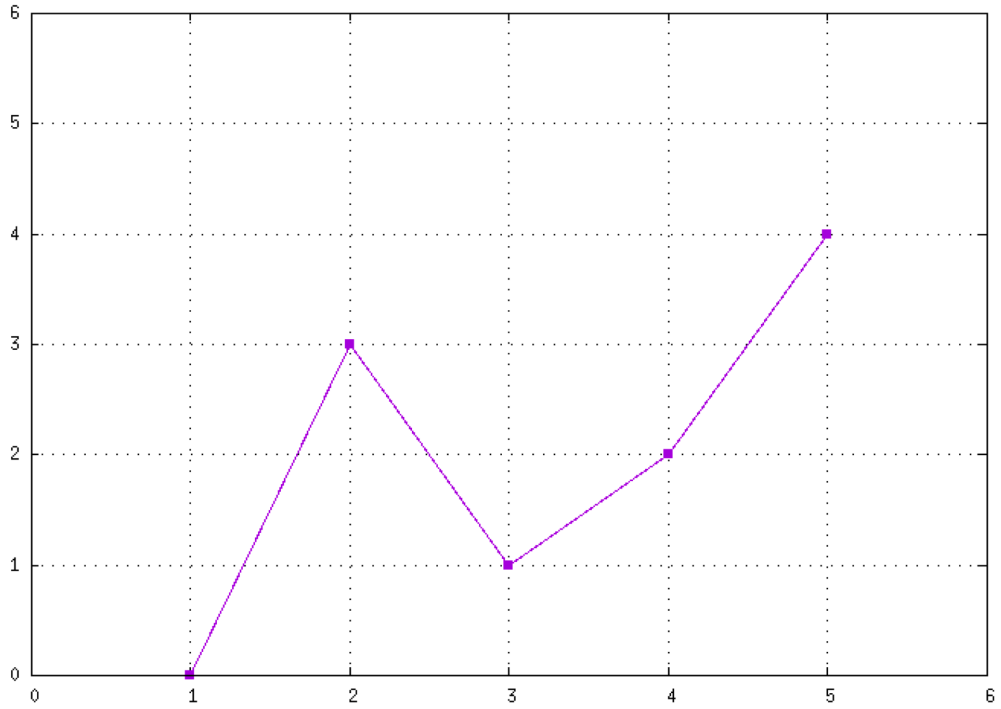
- **It is not allowed** to have an instance of class `Interpolation` without knowing which scheme it implements.

Linear Interpolation

- Given a vector of points, e.g.,

```
{(1.0, 0.0), (2.0, 3.0), (3.0, 1.0), (4.0, 2.0), (5.0, 4.0)}
```

connects the points with a straight line.

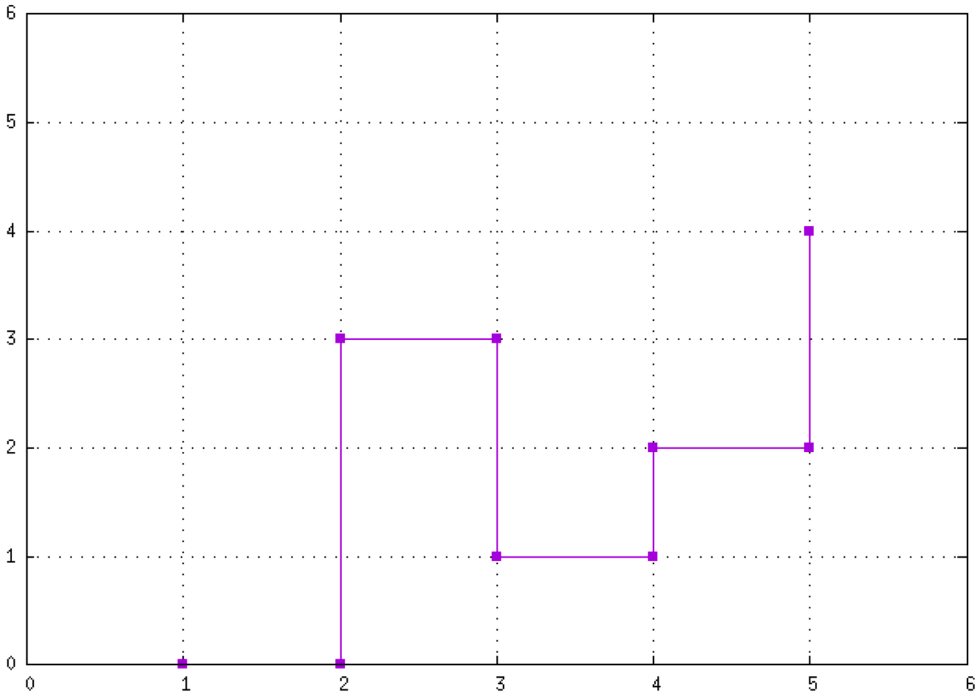


Stepwise Interpolation

- Given a vector of points, e.g.,

  `{(1.0, 0.0), (2.0, 3.0), (3.0, 1.0), (4.0, 2.0), (5.0, 4.0)}`

  provides, within an interval bounded by subsequent points, the y value of the left extreme.

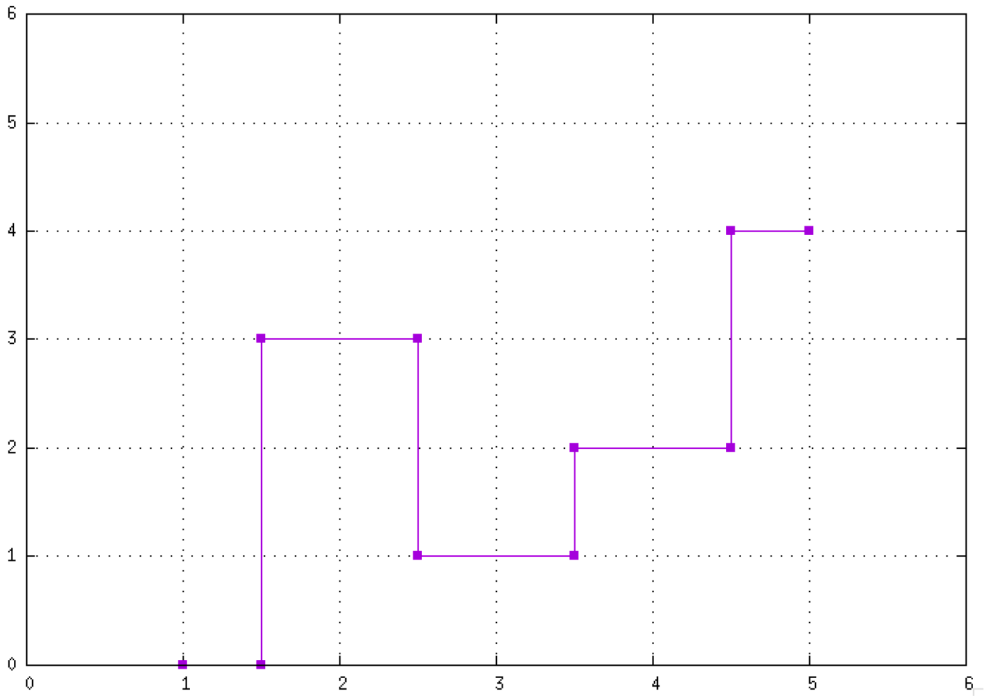


Nearest-neighbor Interpolation

- Given a vector of points, e.g.,

```
{(1.0, 0.0), (2.0, 3.0), (3.0, 1.0), (4.0, 2.0), (5.0, 4.0)}
```

locates in the vector the nearest point and assigns the same value.



Assumptions

- The class `Point` is defined as follows:

```
class Point {
  private:
     double x;
     double y;
     // …
};
```

- The vector of `Point`s in `Interpolation` is sorted according to the value of the x-axis coordinate.