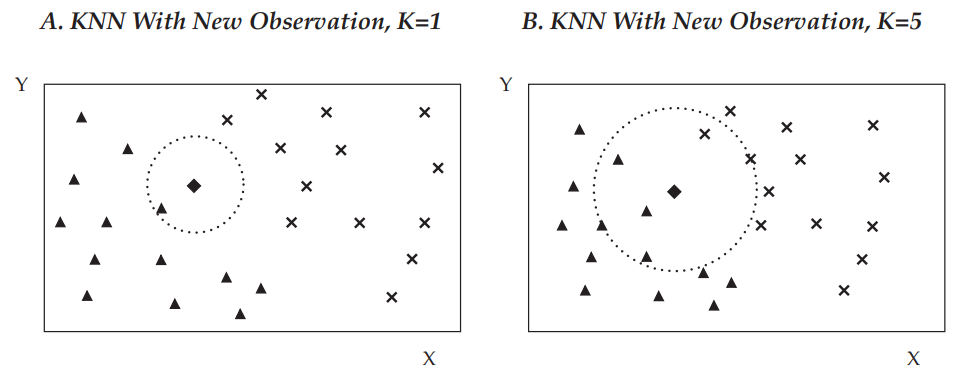
1. **Thuật toán gốc – KNN**

**Thuật toán K láng giềng gần nhất** trong tiếng Anh là **K-Nearest Neighbor**, viết tắt là **KNN**.

**Thuật toán K láng giềng gần nhất** là một kĩ thuật học có giám sát (supervised learning) dùng để phân loại quan sát mới bằng cách tìm điểm tương đồng giữa quan sát mới này với dữ liệu sẵn có.

**Ví dụ**

[](https://cdn.vietnambiz.vn/2020/2/29/hinh-1-1582949835129453959061.png)

Hình thoi trong Hình 1 đang cần được phân loại thuộc hình chữ thập hoặc hình tam giác.

- Nếu k = 1, hình thoi sẽ được phân loại vào cùng loại với điểm dữ liệu gần nhất của nó (tức là hình tam giác trong bảng bên trái - bảng A).

 Bảng bên phải (bảng B) thể hiện trường hợp k = 5, thuật toán sẽ xem xét 5 điểm dữ liệu gần hình thoi nhất, đó là 3 hình tam giác và 2 hình chữ thập. Qui tắc quyết định là chọn phân loại có số lượng lớn nhất trong 5 điểm dữ liệu được xem xét. Vì vậy, trong trường hợp này, hình thoi cũng được xếp vào phân loại tam giác.

1. **Ý tưởng MapReduce hóa**

Trong thuật toán KNN, tại bước tính khoảng cách từ điểm cần phân lớp đến các điểm trong tập dữ liệu mẫu, ta có thể thực hiện tính toán song song với nhau. Nên có thể áp dụng MapReduce tại đây

1. **Giải pháp MapReduce**

Map(keyIn, valueIn) => list(keyInt, valueInt)

Reduce(keyInt, list(valueInt)) => list(keyOut, valueOut)

Tương ứng với bài toán KNN:

**Map**:

keyIn: offset\_row

valueIn: row\_content

keyInt: điểm cần gán nhãn(point)

valueInt: khoảng cách đến điểm đang xét trong dữ liệu huấn luyện(distance) + nhãn(label)

* **Map(offset\_row, row\_content) => list(point, (distance,label))**

**Reduce**:

keyInt: điểm cần gán nhãn

valueInt: : khoảng cách đến điểm đang xét trong dữ liệu huấn luyện + nhãn

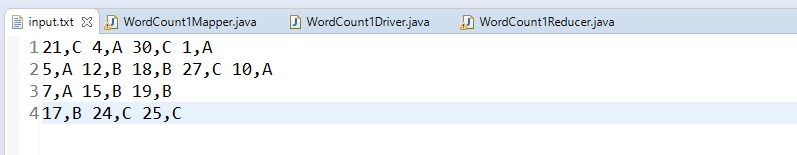
keyOut: điểm cần gán nhãn

valueOut: nhãn gán

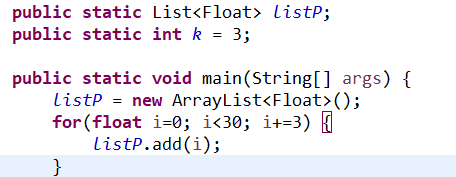
* **Reduce(point, list(distance,label)) => list(point, label)**

1. **Thuật toán MapReduce hóa mới**

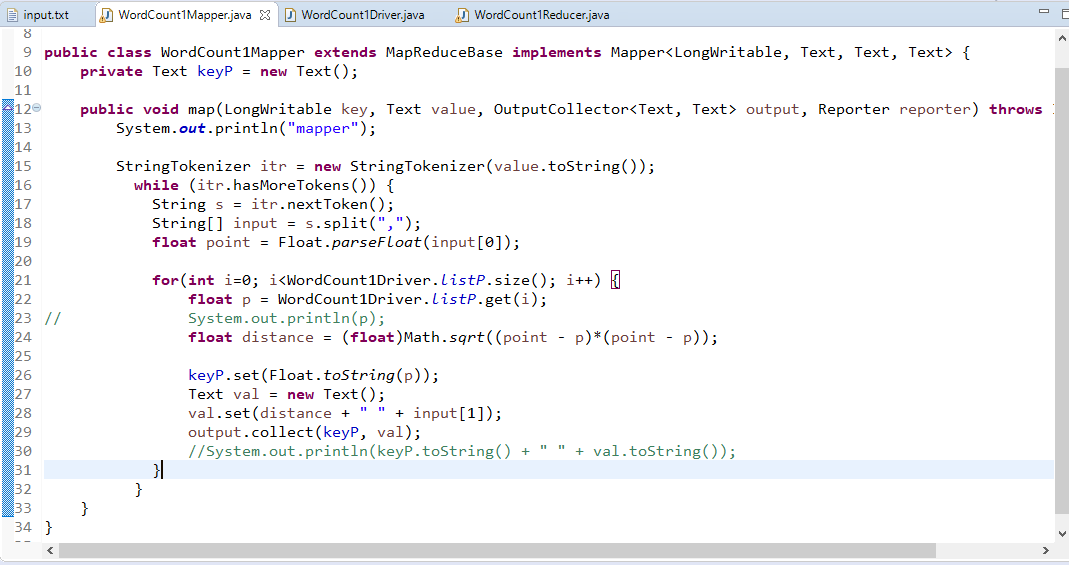
File input gồm nhiều dòng, mỗi dòng bao gồm các cặp tọa độ điểm và nhãn cho điểm đó



Ví dụ trong hình, điểm có tọa độ 21 được gán nhãn C, điểm 4 có nhãn A, điểm 30 có nhãn C, .....



Biến k để lưu giá trị của k, listP lưu danh sách các điểm mình cần phân lớp

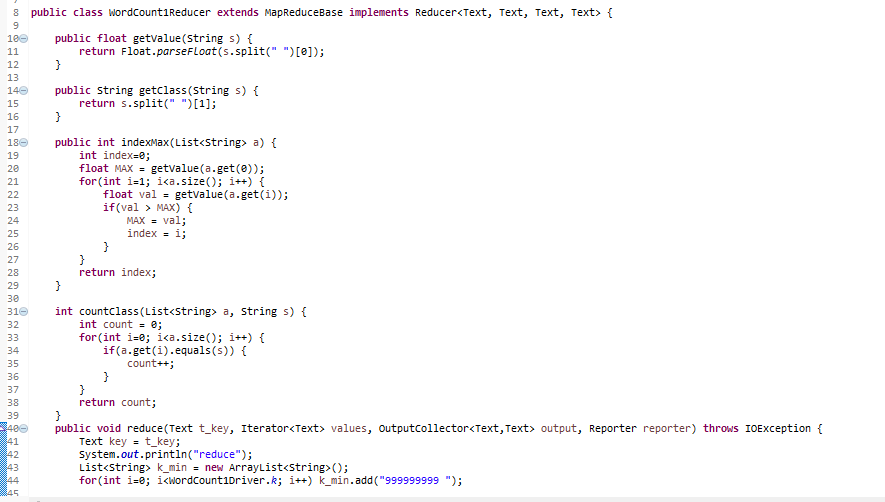


Tại mapper, ta sẽ đọc file input các điểm trong tập dữ liệu mẫu, thực hiện tính khoảng cách euclid từ các điểm cần phân lớp trong listP đến các điểm trong file input, theo công thức:

Distance(x,y) =

Ta sẽ thu được các cặp (keyP, val) với keyP là điểm cần gán nhãn trong listP, val là khoảng cách từ keyP đến điểm trong dữ liệu mẫu

Ví dụ, đang cần gán nhãn cho điểm 7, và dữ liệu mẫu là điểm 3 có nhãn A, ta có output.collect(7, 4 A)

Tại reducer, ta có t\_key là điểm cần gán nhãn, và values danh sách các cặp (distance, label) được gửi lên từ mapper

Ví dụ, t\_key 7, values: [ “25 C”, “10 B”, “15 C”, “7 B”, “16 B”, “20 C”, “3 A”, “6 A”]

Với k=3

Giải thuật như sau:

Tạo 1 list gồm 3 điểm với giá trị +, duyệt lần lượt các điểm trong values, thay thế điểm đó với điểm lớn nhất trong list, sau khi duyệt hết values sẽ thu được k điểm có khoảng cách gần với t\_key nhất

B1. List: [**+**]

B2. Xét 25C, List: [25C]

B3. Xét 10B, List: [25C]

B4. Xét 15C, List: [**25C**]

B5. Xét 7B, List: [7B]

B6. Xét 16B, List: [7B]

B7. Xét 20C, List: [7B]

B8. Xét 3A, List: [7B]

B9. Xét 6A, List: [**7B**]

Sau khi duyệt hết giá trị values, ta thu được list gồm 3 điểm có khoảng cách nhỏ nhất đến điểm t\_key = 7

Ta đếm trong list có 2 nhãn A, 1 nhãn B, nên điểm 7 sẽ được gán cho nhãn A

1. Chạy demo gán nhãn cho các điểm 3 6 9 12 15 18 21 24 27

