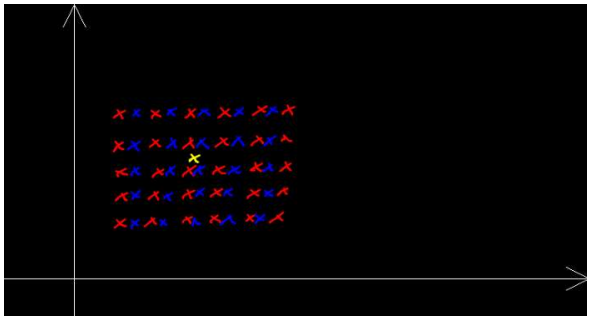
Question And Answer KNN

1. **Explain about K-Nearest Neighbors?**

Ans :- The k-nearest neigbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

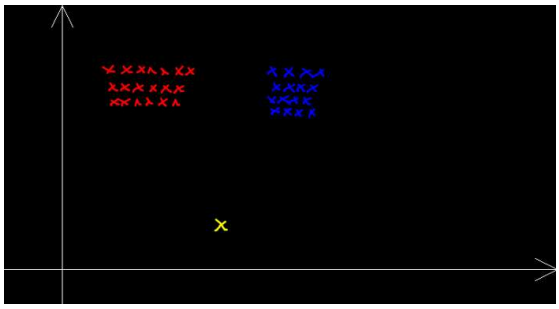
2. **Failure cases of KNN?**

Ans :- Case 1:



In this case, the data is randomly spread and hence no useful information can be obtained from it. Now in such a scenario when we are given a query point (yellow point), the KNN algorithm will try to find the k nearest neighbors but since the data points are jumbled, the accuracy is questionable.

Case 2:



In this case, the data is grouped in clusters but the query point seems far away from the actual grouping. In such a case, we can use K nearest neighbors to identify the class, however, it doesn't make much sense because the query point (yellow point) is really far from the data points and hence we can't be very sure about its classification.

3. **Define Distance measures: Euclidean(L2) , Manhattan(L1), Minkowski, Hamming?**

Ans :-

The formula for Minkowski Distance is given as:

$$D = \left(\sum_{i=1}^{n} |p_i - q_i|^p \right)^{1/p}$$

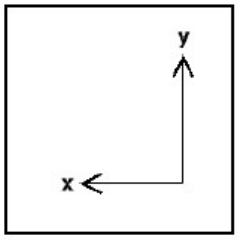
If p = 1 this is Manhattan distance

If p = 2 Euclidian distance

Manhattan Distance (L1 Norm):

We use Manhattan Distance if we need to calculate the distance between two data points in a grid-like path. As mentioned above, we use the Minkowski distance formula to find Manhattan distance by setting p's value as 1.

Let's say, we want to calculate the distance, d, between two data points- x and y.



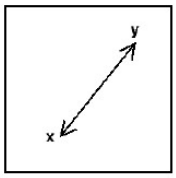
Manhattan

Distance d will be calculated using an absolute sum of the difference between its Cartesian coordinates as below :

$$d = \sum_{i=1}^{n} |x_i - y_i|$$

Euclidean Distance (L2 Norm):

Euclidean distance formula can be used to calculate the distance between two data points in a plane.



Euclidean

Euclidean distance is one of the most used distance metrics. It is calculated using the Minkowski Distance formula by setting p's value to 2. This will update the distance d' formula as below

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Hamming Distance:

A Hamming distance in information technology represents the number of points at which two corresponding pieces of data can be different. It is often used in various kinds of error correction or evaluation of contrasting strings or pieces of data.

Example:

Suppose there are two strings 11011001 and 10011101.

11011001 ⊕ 10011101 = 01000100. Since, this contains two 1s, the Hamming distance, d(11011001, 10011101) = 2.

4. What is Cosine Distance & Cosine Similarity?

Ans :- The relation between cosine similarity and cosine distance can be define as below.

a. Similarity decreases when distance between two vectors increases
b. Similarity increases when distance between two vectors decreases.

Cosine similarity says that to find the similarity between two points or vectors we need to find Angle between them.

Formula to find the Cosine Similarity and Distance is as below:

$$1 - \text{Cosine\_Similarity} = \text{Cosine\_Distance}$$

$$\text{Cosine\_Similarity} = \cos \Theta(\text{theta})$$

Cosine Similarity and Cosine Distance is heavily used in recommendation systems to recommend products to the users based on there likes and dislikes.

It has three case:

Case1: When angle between points P1 & P2 is 45 Degree then

cosine_similarity= Cos 45 = 0.525

P1 and P2 Are 52% Similar

Cosine_distance = 1-0.525 = 0.475

Case2: When two points P1 & P2 are far from each other and angle between points is 90 Degree then

cosine_similarity= Cos 90 = 0

P1 and P2 Are disimilar

Cosine_distance = 1-0 = 1

Case3: When two points P1 & P2 are very near and lies on same axis to each other and angle between points is 0 Degree then

cosine_similarity= Cos 0 = 1

P1 and P2 Are Similar

Cosine_distance = 1-1 = 0

Here we see when distance is less the similarity is more (points are near to each other) and distance is more, two points are dissimilar (far away from each other).

5. How to measure the effectiveness of k-NN?

Ans:- Knn(distance) + Majority Vote

1st break the actual data into two parts train and test.

Dtrain union Dtest = Actual data set

Dtrain intersection Dtest = Null

2nd then train Knn model by using train data and to measure the model use test dataset.

Now measure :

$$\text{Accuracy} = \text{count}/n2$$

Where:

Count = no of point which dtrain + Knn gave correct class value

n2 = no of point in dtest

Accuracy lies [0-1]

6. Limitations of KNN?

Ans:- Knn Time And Space Complexity is O(n)

- When we take KNN Algorithm in real-time it takes lot of time and space to evaluate as its tc and sc is O(n).
- For large scale of application (low latency system) knn fails.
- In higher dimensional space, the cost to calculate distance becomes expensive and hence impacts the performance.
- KNN is sensitive to outliers and missing values and hence we first need to impute the missing values and get rid of the outliers before applying the KNN algorithm.
-

7. How to handle Overfitting and under fitting in KNN?

Ans :- The model is overfitting when the train data is good but worse performance in cross validation (test data).

The model is under fitting when both train and cross validation (test) both badly performance.

To overcome this we use best value of K by using cross validation or elbow method.

And also we resampling and holding and validate the dataset

8. Need for Cross validation?

Ans:- suppose we have 1000 dataset if in train 70% data and in test in 30% data. When I do basic train test split our model use 70% for train the model and remain 30% test data for check then say the model accuracy here happen that 70% and 30% is randomly selected when this type selection happen then may be all data is not present in 70% train data set then accuracy goes down.

When we use train-test split we select randomly. Here also every new data or we refresh the model the value will change it is major disadvantage of this train-test split.it shuffled every time when we add new data so accuracy fluctuate.

To prevent this we use cross validation

https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/

Types:

1. Leave one out cross validation (LOOCV)
2. k-fold cross validation
3. Stratified k-fold cross validation
4. Adversarial validation
5. Cross validation for time series
6. Custom cross validation techniques

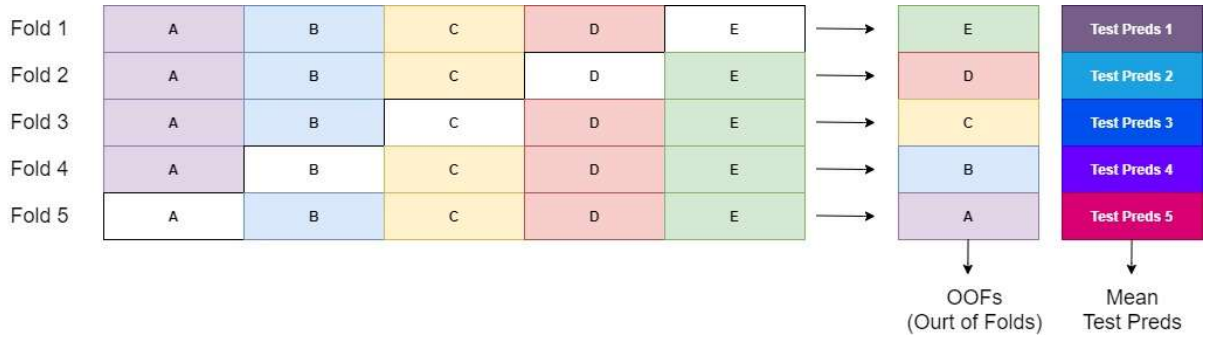
9. What is K-fold cross validation?

k-fold cross validation

It's easy to follow and implement. Below are the steps for it:

1. Randomly split your entire dataset into k"folds"
2. For each k-fold in your dataset, build your model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for *kth* fold
3. Record the error you see on each of the predictions
4. Repeat this until each of the k-folds has served as the test set
5. The average of your k recorded errors is called the cross-validation error and will serve as your performance metric for the model

Below is the visualization of a k-fold validation when k=10.



10. What is Time based splitting?

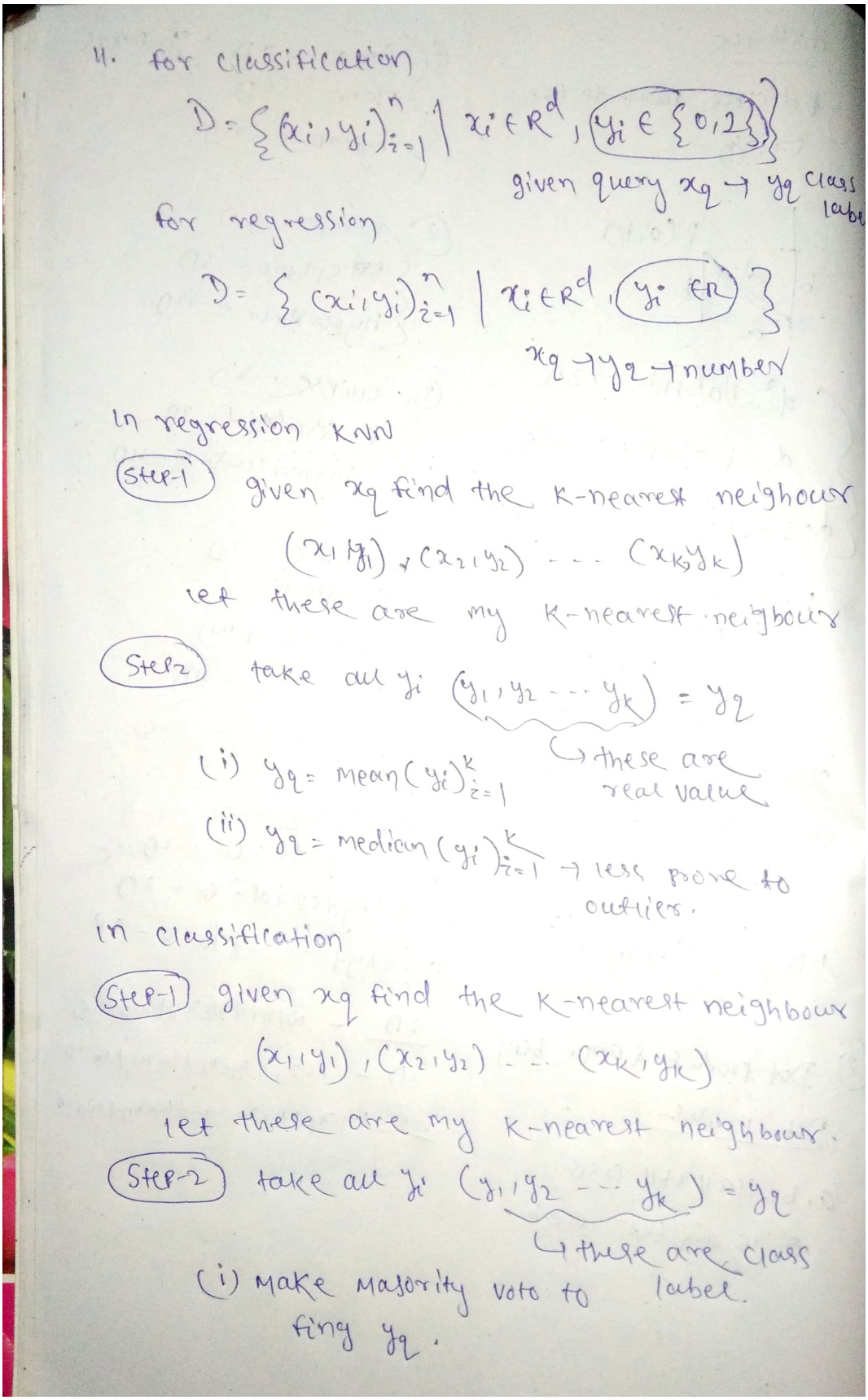
Ans:- In a Machine Learning algorithm we can split the given dataset into training and test data.

We can either split randomly or use time based splitting. For time based splitting we need a timestamp as one of the attributes / features. Like in case of e-commerce website we can have reviews for various products. These reviews can have timestamps also. In such scenarios it's better to use time-based strategy. To do this we can first sort the reviews using timestamp and then do the split.

Sort data by time.

Split – Training (80%) and Testing (20%)

11.Explain k-NN for regression ?



11. for classification

$$D = \{(x_i, y_i)_{i=1}^n \mid x_i \in R^d, y_i \in \{0,2\}\}$$

given query $x_q \rightarrow y_q$ class label

for regression

$$D = \{(x_i, y_i)_{i=1}^n \mid x_i \in R^d, y_i \in R\}$$

$x_q \rightarrow y_q \rightarrow$ number

In regression KNN

Step-1 given x_q find the K-nearest neighbour

$(x_1, y_1), (x_2, y_2) \ldots (x_k, y_k)$

let these are my K-nearest neighbour

Step-2 take all y_i $(y_1, y_2 \ldots y_k) = y_q$

↳ these are real value

(i) $y_q = \text{mean}(y_i)_{i=1}^k$

(ii) $y_q = \text{median}(y_i)_{i=1}^k$ → less prone to outliers.

In classification

Step-1 given x_q find the K-nearest neighbour

$(x_1, y_1), (x_2, y_2) \ldots (x_k, y_k)$

let these are my K-nearest neighbour.

Step-2 take all y_i $(y_1, y_2 \ldots y_k) = y_q$

↳ these are class label

(i) make majority vote to find y_q.

12.Weighted k-NN ?

Ans:- Weighted kNN is a modified version of k nearest neighbors.

a. In this technique we give the weight to the every k nearest neighbour

W = 1/d for give the weight to the knn.

As weight increase distance decrease and vice versa.

b. After give the proper weight to the every knn then calculate total weight in –ve point and total weight in +ve point. Then choose the higher value of weight of the knn.

13.How to build a kd-tree.?

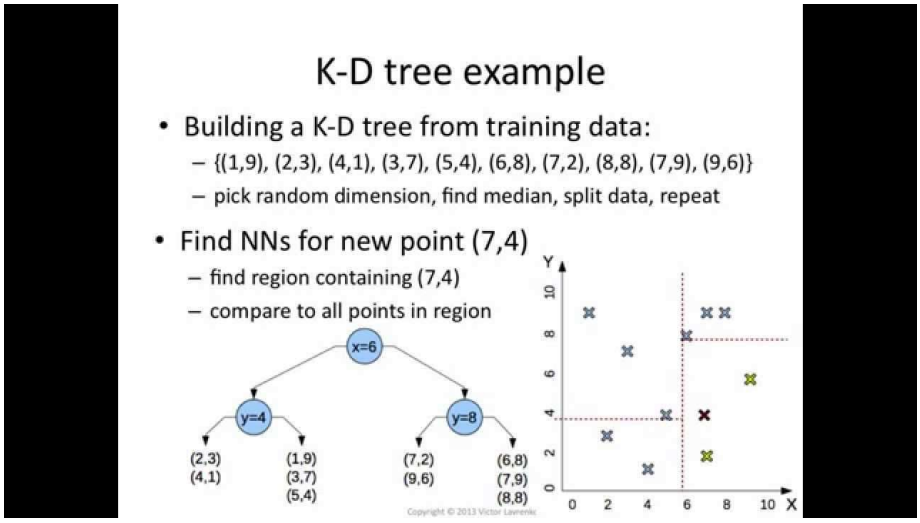
Ans :- A K-D Tree(also called as K-Dimensional Tree) is a binary search tree where data in each node is a K-Dimensional point in space. In short, it is a space partitioning (details below) data structure for organizing points in a K-Dimensional space.

A non-leaf node in K-D tree divides the space into two parts, called as half-spaces.

Points to the left of this space are represented by the left subtree of that node and points to the right of the space are represented by the right subtree. We will soon be explaining the concept on how the space is divided and tree is formed.

For the sake of simplicity, let us understand a 2-D Tree with an example.

The root would have an x-aligned plane, the root's children would both have y-aligned planes, the root's grandchildren would all have x-aligned planes, and the root's great-grandchildren would all have y-aligned planes and so on.

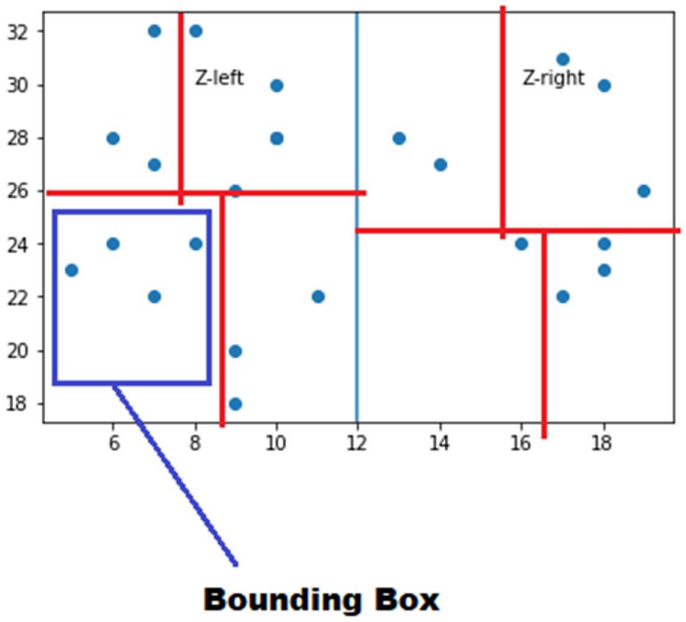


14. Find nearest neighbors using kd-tree?

Before we start we must understand what are the values that are stored at each node to build an intuitive data structure like K-D Tree

We will basically store following three information at each node:

1. Axis X or Y used for splitting?
2. Split value
3. Bounding Box - It is the smallest box that contains all the points within that node.



Steps

The path traversed to reach Query point *A(2,8)* satisfies all the splitting conditions down the tree at each node to this point and we will continue searching until a leaf node is reached.

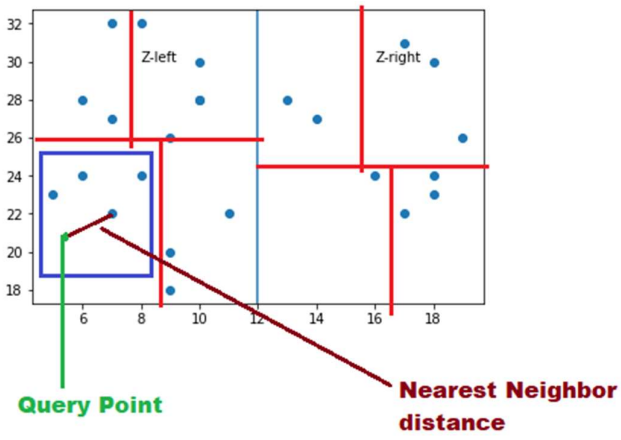
1. Start from the root node and moves recursively in the same way as it is build. The *x* value of Query point *A(2,8)* is compared with that of *Z (8,12)* to determine in which direction to branch. Because *A x*'s value is less than *P*'s *x* value, we branch to the left subtree

2. At root level we really don't care about the Query point A y's value and it's doesn't affect the decision on which way to branch at root level.

3. At the second level it is Query point A y's value is compared with the node (3,10) y's value i.e. 8 is greater than 6 so we will branch to the right subtree.

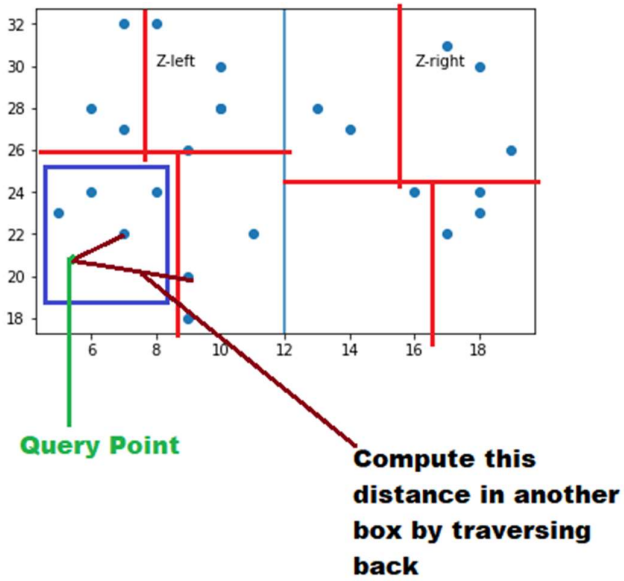
4. Compare the Node value *(2,8)* with *A(2,8)*, Is it same? Yes, then Stop.

5. Compute the distance of each point contained in the bounding box to the Query point A.



6. Can a nearest neighbor necessarily be inside the bounding box? Not really, it can live in another box also. There can be another points closer in adjacent boxes.

7. So what we will do find other points in adjacent boxes that can be nearest compared to the nearest one in the same box. We will traverse backward and start finding the distance with the other points in neighboring boxes.



8. Compute the distance between Query point and all the points in neighboring box. And we will say that a new nearest neighbor is found in the adjacent box which is smaller than the nearest neighbor point in the same box found in Step 5.

9. As a final step, We will traverse backward to the previous node and check if the distance to that bounding box is less than the new nearest neighbor found in step 8 or not.

10. Prune all the nodes where the distance of bounding box is greater than the nearest neighbor and that way you don't have to compute the distance with each point inside that box.

11. After pruning all such boxes whatever is left will give the nearest neighbor based on their distance from the query point. so if query is to find two nearest neighbor then first neighbor is point found in step 8 and second nearest neighbor is point found in step 5.

---------- END ----------