Introduction to K-Dimensional Tree

Pingchuan Ma*

College of Software, Nankai University

pika7ma@gmail.com

Abstract

This state-of-the-art report summarizes k-dimensional tree research and further applications within computer science.

Keywords: Data Structure, K-dimensional Tree, Combinatorics

1 Introduction

K-dimensional tree [1] (or k-d tree, where k is a random integer indicates the dimensionality of the search space) has been established as one of the major solutions to modelling multidimensional space. While k-d tree initially gained popularity as multidimensional data structure, it has emerged to play a role in other fields (e.g. particle simulation and clustering). K-d tree is also a special case of binary space partitioning tree.

*Student ID: 1511442

2 The Method

Given a low dimensional continuous space, the basic idea of k-d trees is attempting to cut this space into halves and limiting the searching into one of the quadrants to reduce the computational complexity.

To simplify the problem in order to provide insight into this artful but confusing data structure, the sample dimensionality has been set into 2D. Consider this training data set:

$$\{(1,9),(2,3),(4,1),(3,7),(5,4),$$
$$(6,8),(7,2),(8,8),(7,9),(9,6)\}$$

Now the axes have two attributes. The algorithm works like that we pick an attribute and find the median, then we split the data set along that median. For example, in this case, the median happens to be 6 at the first time split, so we have the points in the upper line on the left side and the lower ones on the right. Similarly, we do the same things to the second attributes, then we work out a tree-like data structure. The pseudocode is followed:

```
   Data: axis, median: integer
1  function kdtree(pointSet, depth)
2  begin
3      axis ← depth mod k;
4      median ← selected based on axis from pointSet;
5      newNode.location ← median;
6      newNode.left ← kdtree(points in pointSet before median, depth + 1);
7      newNode.right ← kdtree(points in pointSet after median, depth + 1);
8  end
```

As a result, we are going to fracture the space into some little cubes and hy-

percubes. When it comes to inserting, every time we get a new data point, we can just walk down the tree until we end up in one of these hypercubes.

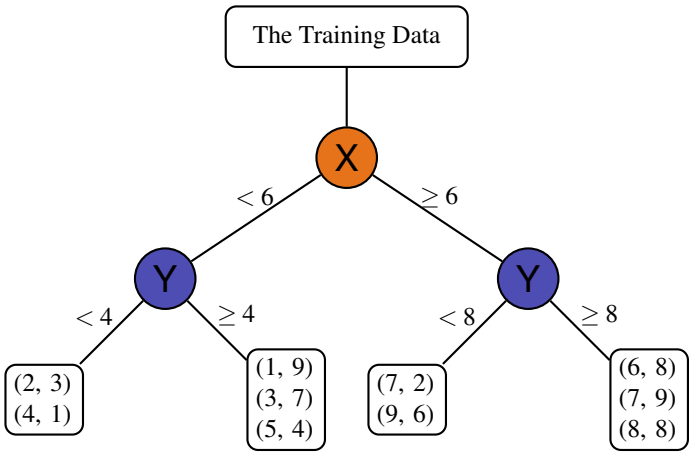

Figure 1: Building a *k*-d tree from training data.

Apparently, *k*-d tree is an approximate technique that you may make mistakes when implementing searching, but it allows you to quickly drill down to a bucket that has roughly correct position.

3 Applications

3.1 Clustering

Since our tree-like data structure is modeled based on a multidimensional space, intuitively, its main function will be implemented on space working. Clustering is a well-known phenomenon in geometric data, and has application in machine learning. To simplify the problem, we define the distance function in standardized euclidean metric:

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

In a low dimensionality, the quickly drilling feature of k-d tree enables it as a speeding up of k-means clustering [3] (Lloyd's algorithm is more efficient for higher dimension).

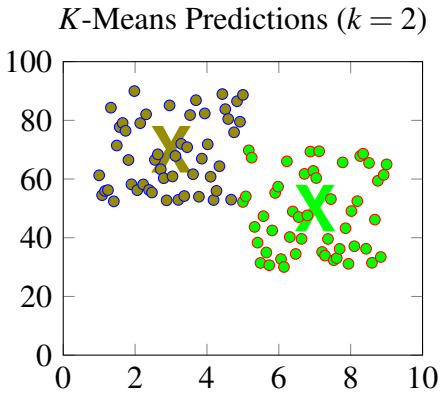

Figure 2: One case in clustering.

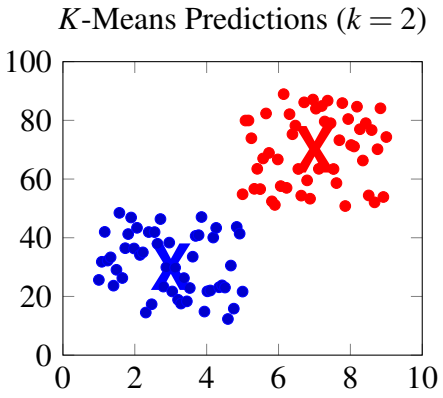

Figure 3: Another case in clustering.

3.2 Particle Simulation

In computer graphics, we compute the movement of fluid particles with the help of multiple numerical methods which trying to solve the famous *incompressible Navier-Stokes equation* [2]. It describes the motion of fluid substances as a result of forces.

$$\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \vec{\nabla})\vec{v} = \frac{-\vec{\nabla} p}{\rho_m} + \frac{\mu}{\rho_m}\nabla^2 \vec{v} + \frac{\vec{f}_{ext}}{\rho_m} \tag{1}$$

$$\nabla v = 0 \tag{2}$$

Though it seems extremely complex at first glance, we have two intuitive approach to make it: Lagrangian and Eulerian viewpoint, which is particle-based and grid-based respectively.

When it comes to the Lagrangian viewpoint, Smoothed Particle Hydrodynamics [4] (or SPH) has been considered as one of the major particle-based approach to simulate fluid. However, a space-fitting data structure is in need to search the neighbor particles in order to boost the kernel function used in SPH.

Since then, plentiful applications of particle-based fluid simulation, such as Gasoline [5], have been done with the hierarchy data structure k-d tree in terms of the quickness of searching, deleting and inserting.

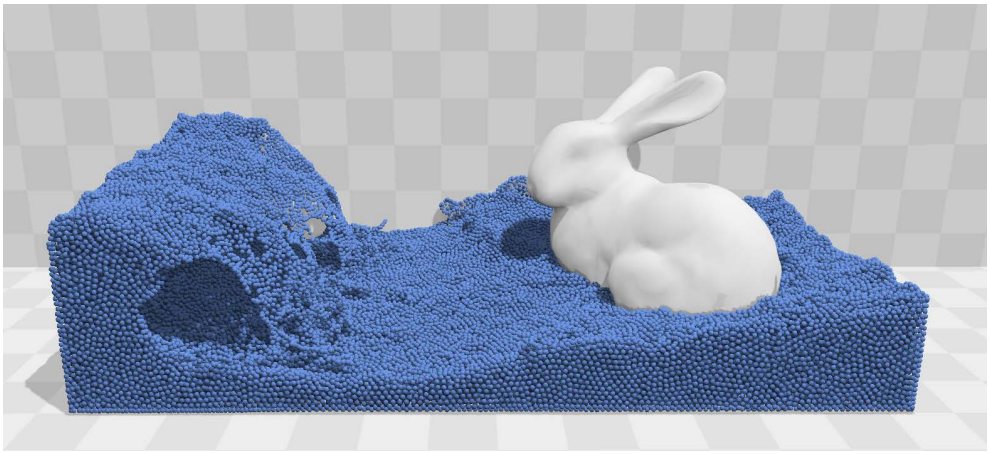

Figure 4: A sample SPH particle system.

4 Connection to Combinatorics

After the brief introduction, we can find that k-d tree is an optimization of binary search tree and a special case of binary space partitioning tree. In terms of com-

puter science, k-d tree has been built as a branch of the search tree structure which aims to reduce the complexity of searching, deleting, sorting and inserting. Similarly, when it comes to combinatorics, k-d tree is a part of graph theory which is one of the combinatorial concepts covered in our class. On one hand, k-d tree is an extended branch of tree in graph theory. On the other hand, the space partition is also a topic in combinatorics.

Furthermore, in our continuous space in real world, it is pretty intuitive to deal with and judge the different part of space. However, to computer, which has no sight and is able to process discrete data merely, modeling the presentative but complicated problem into a batch of code or data somehow plays a vital role in computer science. From my point of view, combinatorics is a specific science that breaks up information into discrete state and designs algorithm to deal with them in a mathematical view. As a result, k-d tree can be abstracted as an imaginary hierarchy structure in abstract mathematics or even a clever pattern of designing discrete data.

5 Conclusion

This report concludes the background, basic principles, thinking patterns, implementations, applications and mathematical connections of k-d tree. Based on the coverage of combinatorial concepts during summer course and my independent reflection, this report can be qualified for a k-d tree primer and a learning stimulator for data structure and combinatorics. In the course of working out this report, I gained experience of manipulating the LaTeX typesetting and programming language. Moreover, I gained an insight into combinatorics and data structure.

Finally, I would like to extend my sincere gratitude to my combinatorics teacher during summer semester, Hua Wang, for his patient lectures and professional instruction.

References

[1] BENTLEY, J. L. Multidimensional binary search trees used for associative searching. *Commun. ACM 18*, 9 (Sept. 1975), 509–517.

[2] BRIDSON, R. Fluid simulation for computer graphics, 2015.

[3] HAMERLY, G. Making k-means even faster. In *SDM* (2010), pp. 130–140.

[4] MÜLLER, M., CHARYPAR, D., AND GROSS, M. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 154–159.

[5] WADSLEY, J., STADEL, J., AND QUINN, T. Gasoline: a flexible, parallel implementation of treesph. *New Astronomy 9*, 2 (2004), 137 – 158.