Point Location in Voronoi Diagrams

Anders Høst Kjærgaard and Hildur Uffe Flemberg
{ahkj, hufl}@itu.dk

IT University of Copenhagen, Rued Langgaards Vej 7, 2300 Copenhagen S, Denmark

Abstract. Voronoi diagrams are used in a variety of contexts in mathematics, biology and computing to name a few. We are interested in analysing the composition of the voronoi diagram, as to say something interesting about the underlying data. In this paper we descibe have a data set containing the geographical location of a handful of fastfood restaurants can be analysed when representing them in a voronoi diagram. In particular, we provide background and solution for performing point location in a voronoi diagram by altering an existing open-source solution for generating the voronoi diagram, with our own implementation of trapezoidal map algorithm, to be able to perform point location in O(log n) time. The motivation for this is to be able to answer what fast-food restaurants a given set of geographical points, or maybe customers, are closest to. Secondly, we provide an analysis of the expected number of custmers that will end in the bigget voronoi cell when distributed randomly across the fast-food restaurant diagram. We find that such information querying on the target data set, could be of potential interest when dealing with location planning of new restaurants, or analysing customer behavior.

Keywords: algorithms, computational geometry, point location, software development

1 Introduction

Voronoi diagrams have been know for... In YYYY Fortune invented the algorithm for ... They are interesting because this and that. Get examples from `http://voronoi.com/wiki/index.php?title=Voronoi_Applications` We use restaurants, defined as sites.

Various interesting questions could be asked about the diagram, that would also pose of interest in algorithm design. In which cell does a given point reside? Which cell is the largest? The questions are not jsut interest for the algorithmic challange, but also due to how we can interpret the answer to such questions in relation to the sites that composit the diagram. E.g. we could assume that if a given

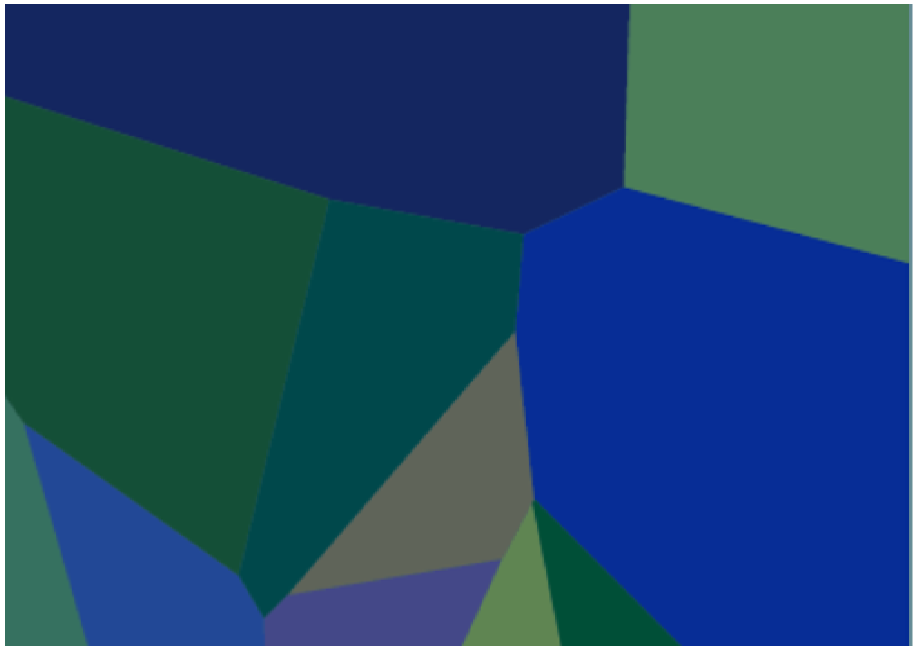

Fig. 1. A Voronoi Diagram with 12 sites

2 Background

3 Problem Analysis and Requirements

3.1 Problem Analysis

4 Solution

The solution is two fold and described in the following.

4.1 Point location in Voronoi

While there are existing algorithms for the two problems with good and optimal running times, an analysis of the two algoritms makes it clear that, for the scope of this project, it would not make much sense to look for an improvement or more elegant way of acomplishing what the two algorithm do. By realizing that the trapezoidal map simply reuires a random edge at the time until all edges have been traversed, makes it somewhat trivial to see that the running output of the voronoi algorithm can be used as input for the trapezoidal map algorithm. Seen in another way, while merging the two algorithm might be a learningful experience, it would in the general case make sense to just keep the two algorithms seperated and rund them in sequense, as the complexity and result is exactly the same. On the down side one could argue that merging them is just a minus as it simply makes the implementation harder to understand. In fact, by merging them we do save some nlognn computations, but unless we come up with some extreme case, we see this as a premature optimization. However, we find the task of combining the two of sufficient academic interest to have carried it through and provided an implementation. describe voronoi and how one needs to modify to the creation of segments upon circle events, maybe discuss why this is actually a final segment, refer to DCEL data structure and half edges not knowing their other ends. Describe the open source solution that we use, problems with it, and bridge it to our trapezoidal map implementation. Roud off with nice features of our trapezoidal map implementation. And show the simplicity in getting the tree DS from it for querying.

4.2 Expected Customers

5 Evaluation

Test with a few screen shots showing that algo actually found the right cell.

6 Threats to Validity

Buggy software til generering af Voronoi.

7 Plan For Future Work

7.1 Future Work

Create a clean and generic voronoi implementation in .NET 4.x Find best point for planar location. Not trivial, would probably be Hawaii, but that would be winning a lot of oceanic area.

7.2 Something else?

8 Conclusion

We have shown the design of two algorithms in the field of computational geometry. We found that altering the the Fortune algorithm to return a fast search data structure of a trapezoidal map for point location was indeed feasible, and showed that by querying random points at the datastructure we could reason about which cell is the biggest. The algorithms presented are implemented in a generic fashion and it should be easy to see the solution used on top of another any other data set.

Acknowledgements We would like to thank Rasmus Pagh for his lectures and his on the spot guidance that has been essential for moving this project in the right direction.

References

[1] Sławek Staworko, Iovka Boneva, and Benoît Groz. The view update problem for xml. In *Updates in XML (EDBT Workshop Proceedings)*, 2010.

[2] Jakob Beetz and al. Bimserver.org – an open source ifc model server. *In: Proc. of. 27th International Conference on Applications of IT in the AEC Industry CIB-W78*, 2010.

[3] buildingSMART. Model view definition description. `http://buildingsmart.com/standards/mvd`, 2012.

[4] Krzysztof Czarnecki and Simon Helsen. Feature-based survey of model transformation approaches. *IBM Systems Journal - Model-driven software development*, 2006.

[5] Kaj A. Jørgensen, Jørn Skauge, Per Christiansen, Kjeld Svidt, Kristian Birch Sørensen, and John Mitchell. Use of ifc model servers. 2010.

[6] Kaj Asbjørn Jørgensen. Collaboration regarding modelling of construction and building services. 2012.

[7] Gabor Karsai, Holger Krahn, Claas Pinkernell, Bernhard Rumpe, Martin Schindler, and Steven Völkel. Design guidelines for domain specific languages. *9th OOPSLA Workshop on Domain-Specific Modeling*, 2009.

[8] Wiet Mazairac and Jakob Beetz. Towards a framework for a domain specific open query language for building information models. *Proceedings DDSS conference Eindhoven*, 2010.

[9] Dr. Mohamed Nour. A graphical user interface for handling ifc partial model exchange. *http://www.inpro-project.eu/publications.asp*, 2008.

[10] James Steel, Robin Drogemuller, and Bianca Toth. Model interoperability in building information modelling. *Software and Systems Modeling*, 11(1):99–109, 2010.

[11] Federico Tomasetti and Marco Torchiano. Prede: a projectional editor for the eclipse modeling framework. *http://2011.eclipse-it.org/ProcEclipse-IT11/*, 2011.

[12] Markus Voelter. Embedded software development with projectional language workbenches. In Dorina C. Petriu, Nicolas Rouquette, and Øystein Haugen, editors, *MoDELS (2)*, volume 6395 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 2010.