CS226: Computational Geometry
Homework 2

This homework is for extra credit; certainly do the problems if you feel the need to raise your grade in the class. Otherwise I will grade you based on class participation, HW 1, your presentation, and your write-up for the project.
Feel free to discuss these questions with your classmates or friends, but please write up all of the solutions in your own words. Include a list of people you got ideas from, and also a list of Web sites, books, papers, etc. that you used in solving each problem.
Always explain your reasoning using complete sentences, punctuation, and paragraphs. Unless your handwriting is very, very readable, please type. Using LaTeX would be best, and also good practice. Each part of each problem should be answered in at most a page; longer answers are not better than short, clear answers.
If you cannot completely solve a problem but you can say something relevant, write it down.

1. (kd-trees suck, part 6)

   It is widely assumed that if a kd-tree is constructed on n points distributed uniformly in the unit square, using the construction in Chapter 5, then its cells will be close to cubical, and all of them will be roughly of the same size. Since there are $O(n)$ such cells, the average side-length should be roughly $1/\sqrt{n}$.

   Argue, on the contrary, that a kd-tree on uniform random data is almost certain to contain at least some "thin cells" with at least one side-length roughly $1/n$. Hint: look near the cutting line corresponding to the root of the tree. Is this the only place you would expect to find thin cells? Estimate (informally is fine) the total number of thin cells in the kd-tree.

2. (k-nearest-neighbors via sorting)

   Let's consider the sorting-based nearest-neighbor algorithm of Liao, Lopez, and Leutenegger, based on the Lemma of Chan (the paper is linked from the Feb 22 lecture). Rather than one approximate nearest neighbor, we often want to find k approximate nearest neighbors. Given an input point set P, a point $p \in P$ is a k-approximate nearest neighbor of a query point q if the distance $d(q,p) \leq (1+c)d(q,p^*)$, where p^* is the true kth nearest point to q, and c is the approximation factor.

   In Section 2.5, they explain how to get k approximate nearest neighbors instead of just one: rather than look at just one point adjacent to q in each sorted list, look at k on either side. They say, "An argument similar to the one presented at the end of the previous section guarantees that the distance from q to the k-th neighbor returned is within a factor of $O(d^{1+\frac{1}{t}})$ of the true k-th distance." Modify the proof of Theorem 1 to show formally that this claim is correct.

3. (molecular Voronoi diagrams)

   Consider a molecule represented as the union of a set of balls. The balls representing a real molecule have two special properties: 1) no two ball centers can get closer to

each other than a, and 2) the radius of any ball is at least b and no more than c, for fixed constants a, b, c.

Recall that the restricted weighted Voronoi diagram of a molecule (the dual of the α-shape) is the intersection of the union with the power diagram of the balls.
a) Argue that the restricted weighted Voronoi diagram has size $O(n)$. One way to do this is to consider the union superimposed on an appropriately sized voxel grid in $\mathbb{R}^3$ (explain how the size depends on a, b and/or c), and argue that only a constant number of atoms can intersect a grid cell.
b) Recall the spatial hash data structure: given an appropriate grid size, we represent voxels that contain parts of atoms in a hash table. Argue that by using the spatial hash structure we can compute the restricted weighted Voronoi diagram of a molecule in time $O(n)$.