Recurring Section 8

Aniruth – 5 PM

Announcements

- Midterm this Monday!
  - Exam tips
- Quick (conceptual) review on tries and k-d trees
  - Introduced from today in lab, so conceptual big picture ideas and then questions on the worksheet today.
  - From today’s lab, there were some worksheets linked – make sure you do those this weekend. These are in scope for the midterm.

Content Review

Midterm Tips

- Exams (in this class) are generally designed to go from easy to hard.
    - Past exams are the source for this tidbit.
- Write everything you would have memorized on your cheat sheet. Everything.
    - All the runtimes, one or two examples. Consider the process of making the cheat sheet to be your review.
    - Go through all the lectures and labs and keep adding to them.
- Take a practice exam first.
    - This way, you know what you know and what you don’t know.
    - Spend time reviewing what you don’t and be extra careful to put more effort in your cheat sheet on those areas.

Midterm Tips

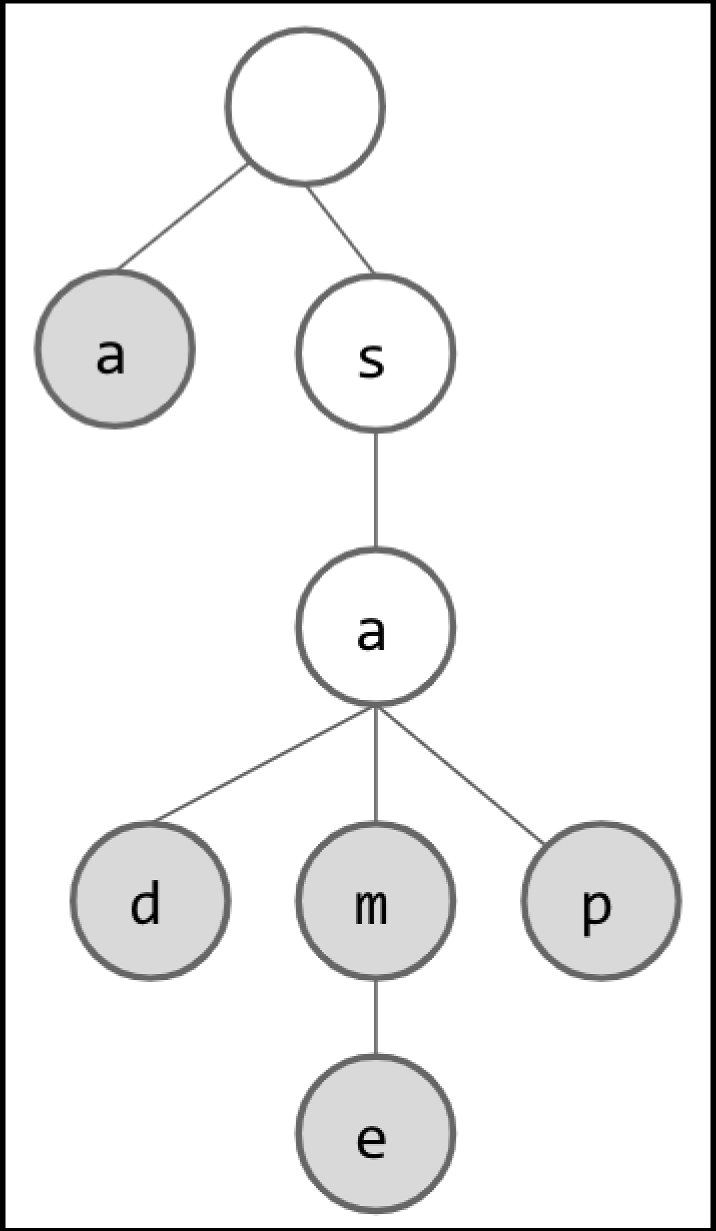
- If you see something you don’t know/might take a while, feel free to skip.
  - Personal Strategy: I will try/think about it for a few minutes. I find it helps me as I have it in the back of my mind as I go through the rest of the exam.
- Look for common areas that trip you up. Practice using old exams. For example:
  - Using .equals vs comparing pointers for objects
  - Selecting multiple options for bounds that could be true
  - Order of tiebreaks for disjoint sets
- Write down questions you miss from practice exams on your cheat sheet.

Midterm Tips

- Breathe and pace yourself! You got this.
- Exams at Berkeley, especially in CS, are quite difficult. All that matters is that you know your material and apply it the best you can.

Tries

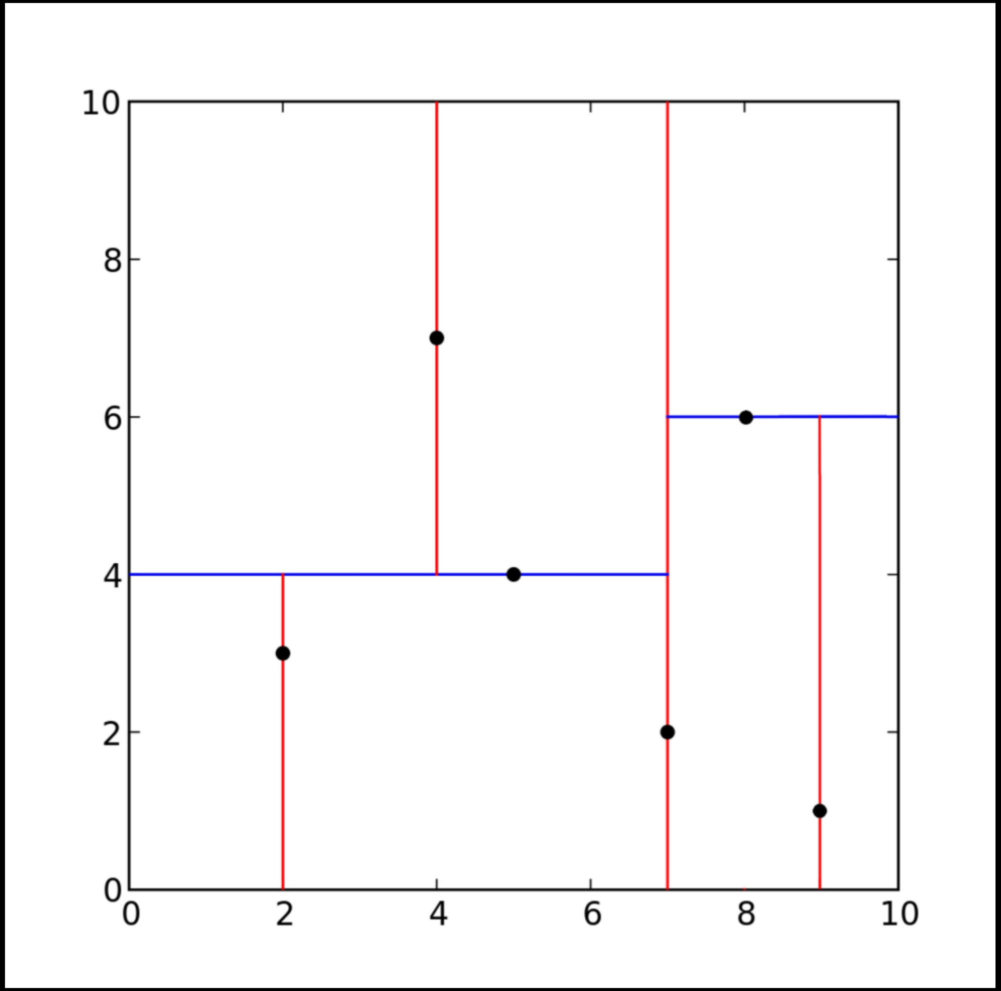
- Think of it as a tree of nodes except you want to take advantage of looking for prefixes/store things with common prefixes together. (image from [cs61bl](cs61bl))
- For example, words (i.e. autocomplete) can have each node be a character and mark each node whenever it's the end of a stored word.
  - This way, storing "win", "wind", "window", and "windows" takes only 7 nodes.



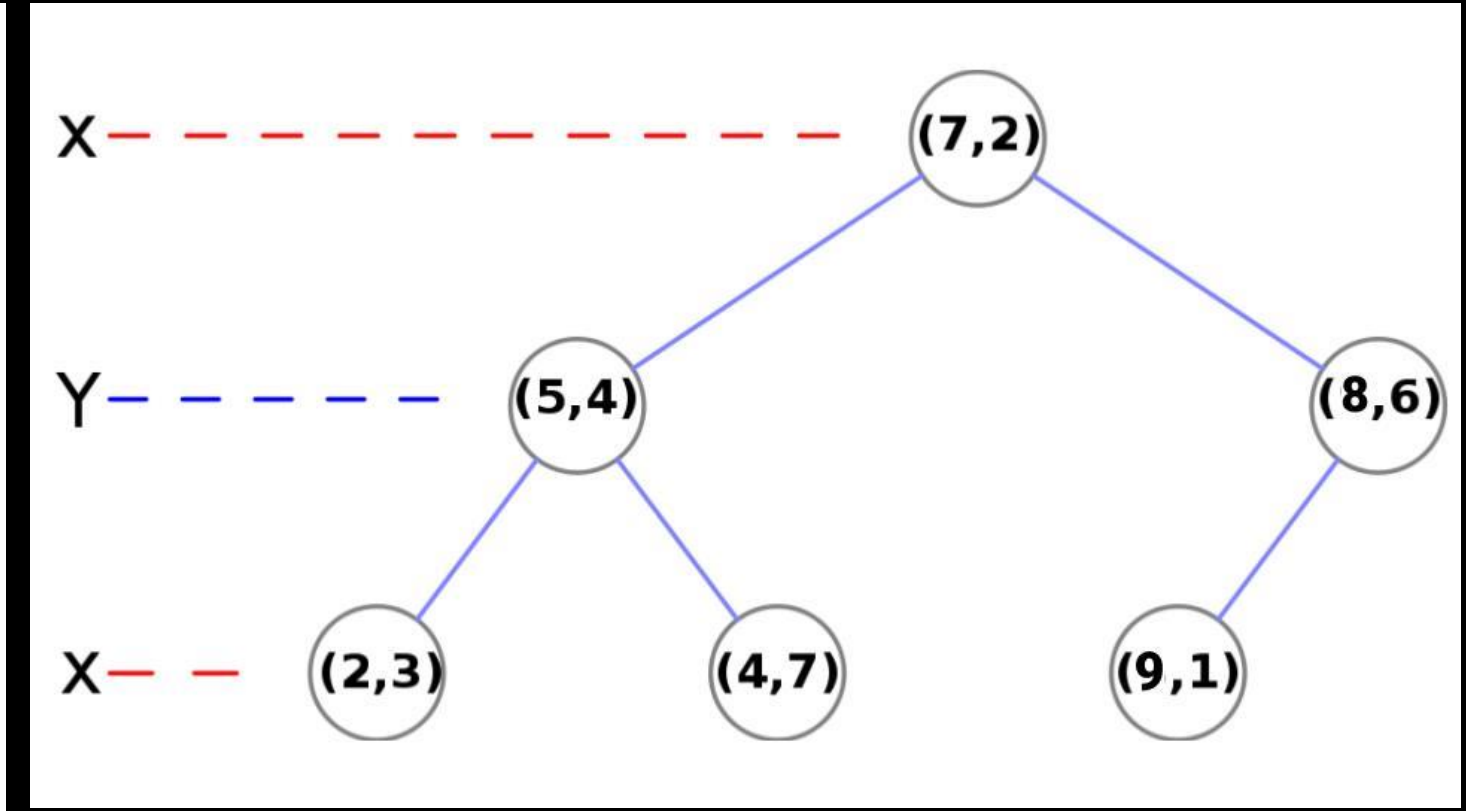
k-d Trees

- What made binary trees so great?
- Binary trees are one dimensional (we compare on one thing). How can we compare for more dimensions (on more things)?
- Used for multidimensional data, each level is switching the dimension we compare on.
- This way, we can traverse the entirety of the data more easily, since we know that certain spaces (branches) of the data (tree) do not need to be searched, cutting down our time needed.

Example of k-d trees (from cs61bl)



0
2
4
6
8
10
0
2
4
6
8
10



X
Y
X
(7,2)
(5,4)
(8,6)
(2,3)
(4,7)
(9,1)