CS709a: Algorithms and Complexity

Focus: Spatial Data Structures and Algorithms

Instructor: Dan Coming
dan.coming@dri.edu

Thursdays 4:00-6:45pm
Office hours after class
(or by appointment)

Today

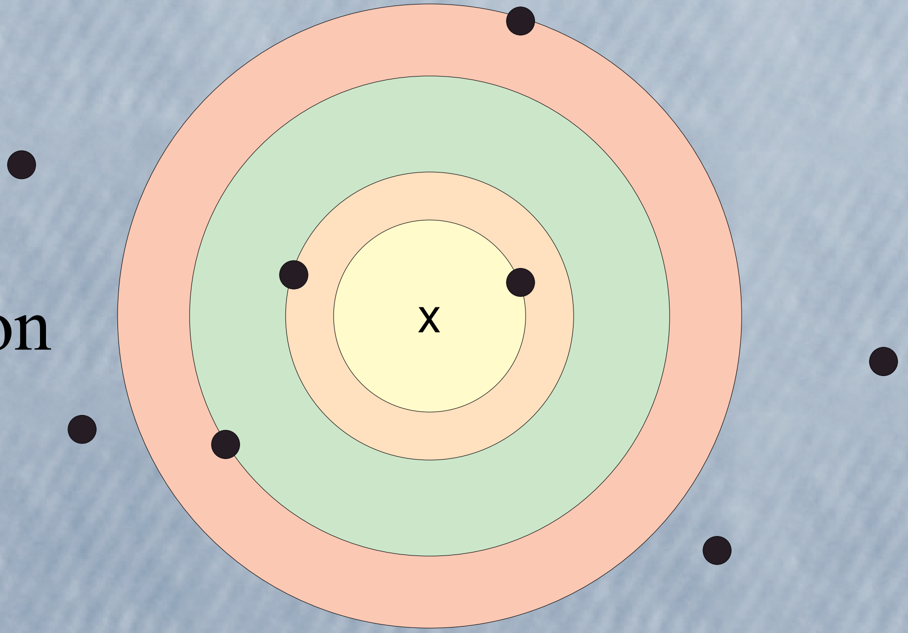
- Nearest neighbor searches
- Midterm Review
- After class: Steve's practice talk for VR conference

Tentative Calendar

- 2/12 – Paper selection due
- 2/19 – Paper Presenter: Joe
- 2/26 – Paper Presenter: Matt
  - Present Project 1 in class (Project 1 Due 2/25)
- 3/5 – Bounding volumes
- 3/12 – Nearest neighbor
- 3/14-22 Spring Break
- 3/26 – Midterm review
  - Present Project 2 in class (Project 2 Due 3/25)
- 4/2 – Paper Presenters: Mark, Scott, Cody, and Steve
- 4/9 – Midterm
- 4/16 – Present Project 3 in class (Project 3 Due 4/15)
- 4/23 – Paper Presenter: Roger
- 4/30
- 5/7-13 Finals Week
  - Final Projects and Presentations Due

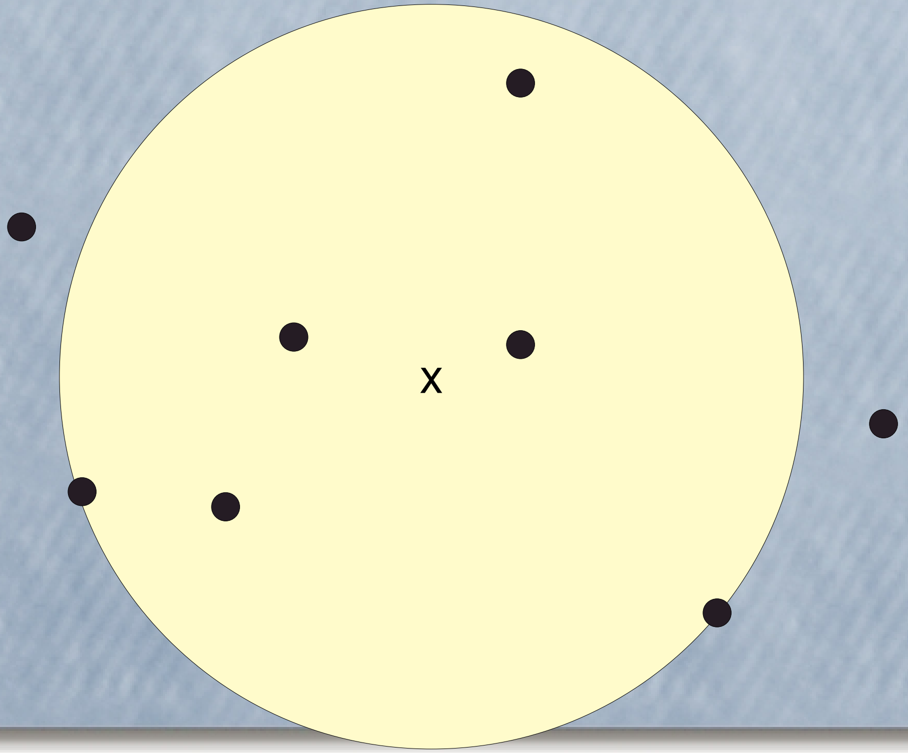
k Nearest Neighbor

- vector<iterator> find_nearest(const point_t & x, size_type k)
- Find the *k* closest points to x
- Example, 4 nearest neighbors:
- Useful for
  - Surface reconstruction
  - Scattered data interpolation
  - Finding similar points



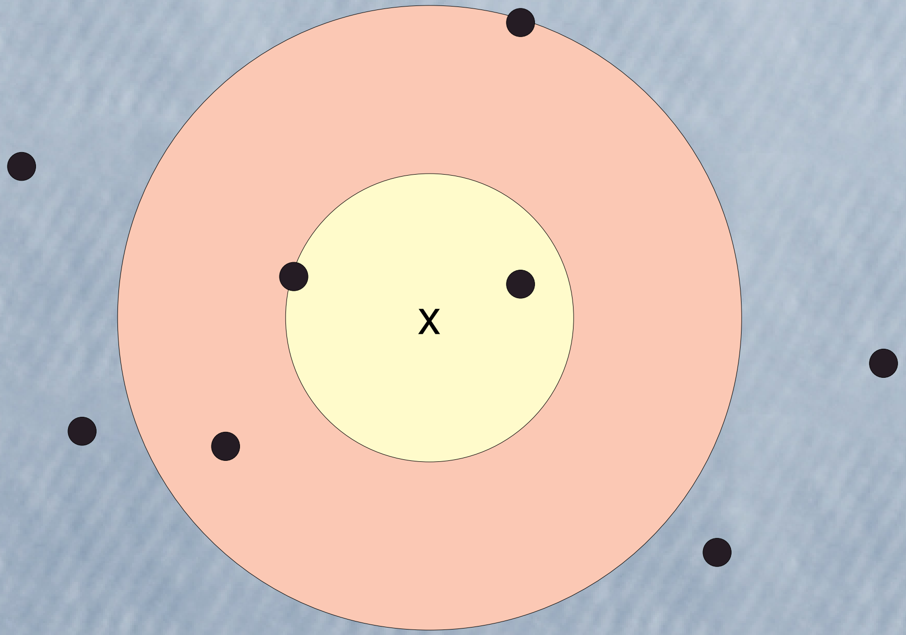
k Nearest Neighbor

- Brute force approach... compute distance to all points, sort by increasing distance, take first *k*
- Alternative approach growing radius search
- Start with some radius, see how many points are within that radius
- If enough, sort by distance and take first *k*



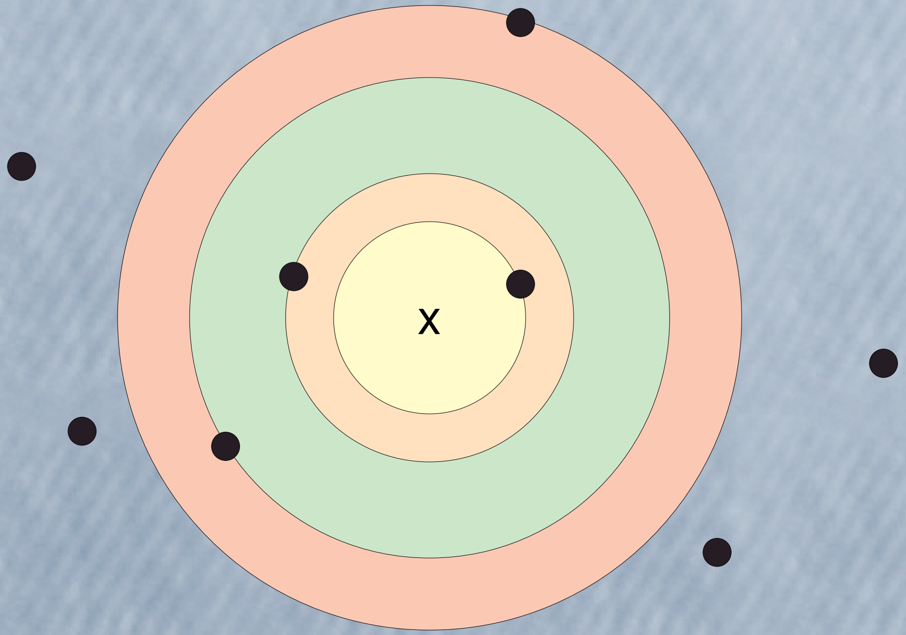
k Nearest Neighbor

- If not enough points inside, grow radius until enough
- How do we find which points are inside?
- But how much do we grow the radius by?



Which points are inside?

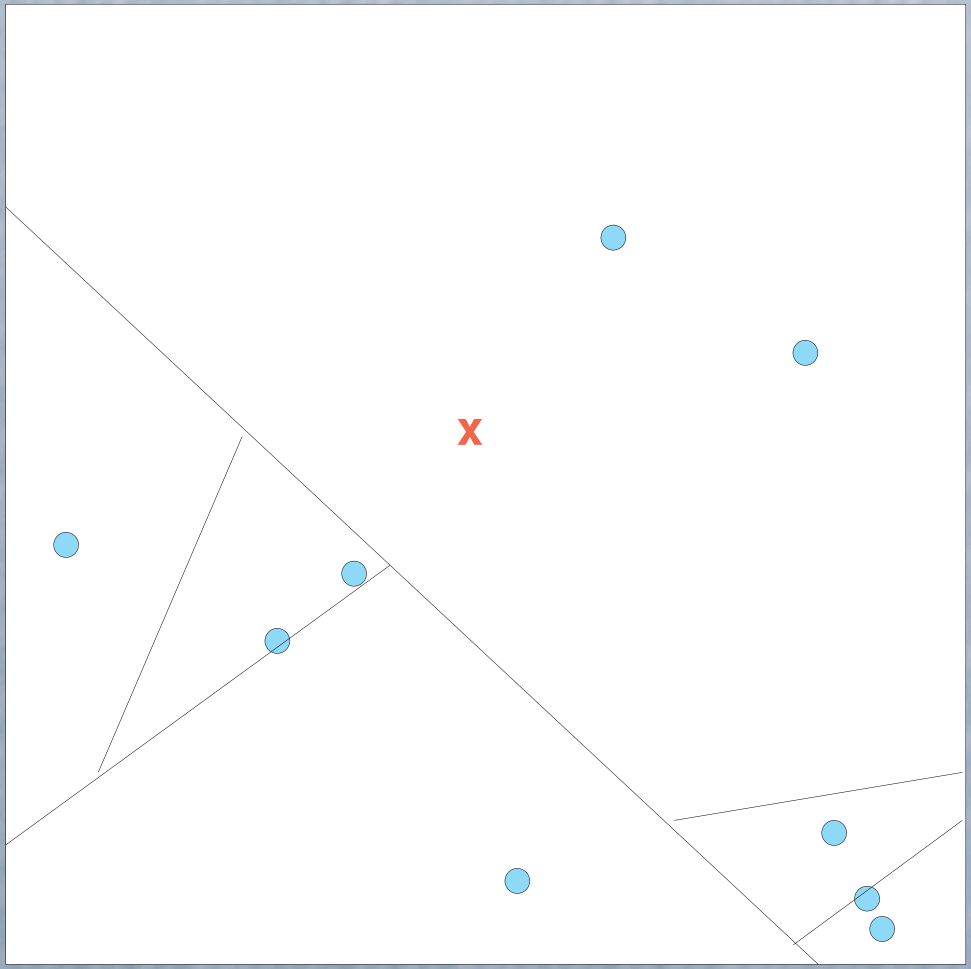
- Radius search (or incremental version)
- What's the radius?



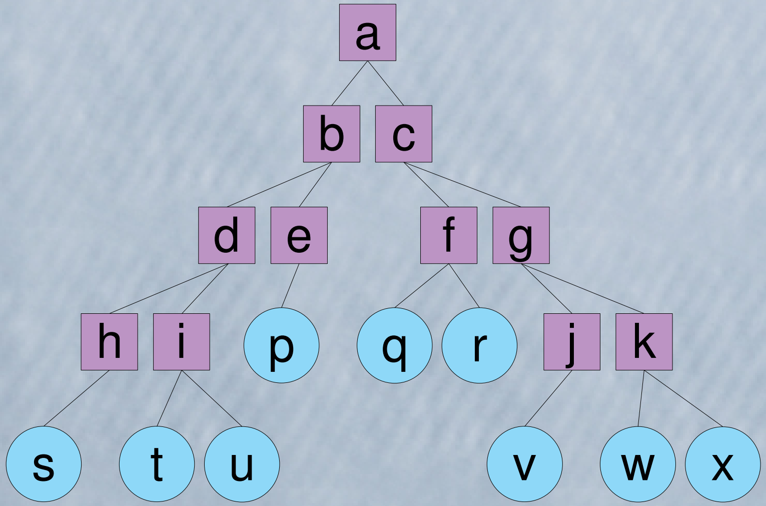
Best-first / Incremental Nearest Neighbor

```
push root of a hierarchical spatial data structure on
  priority queue P
until $k$  found
   element = P.dequeue
   if element is an object
      add element to found
   else for each child of element
      P.enqueue(child,distance(x, child))
   endif
return found
```

Example



Tree Data Structure



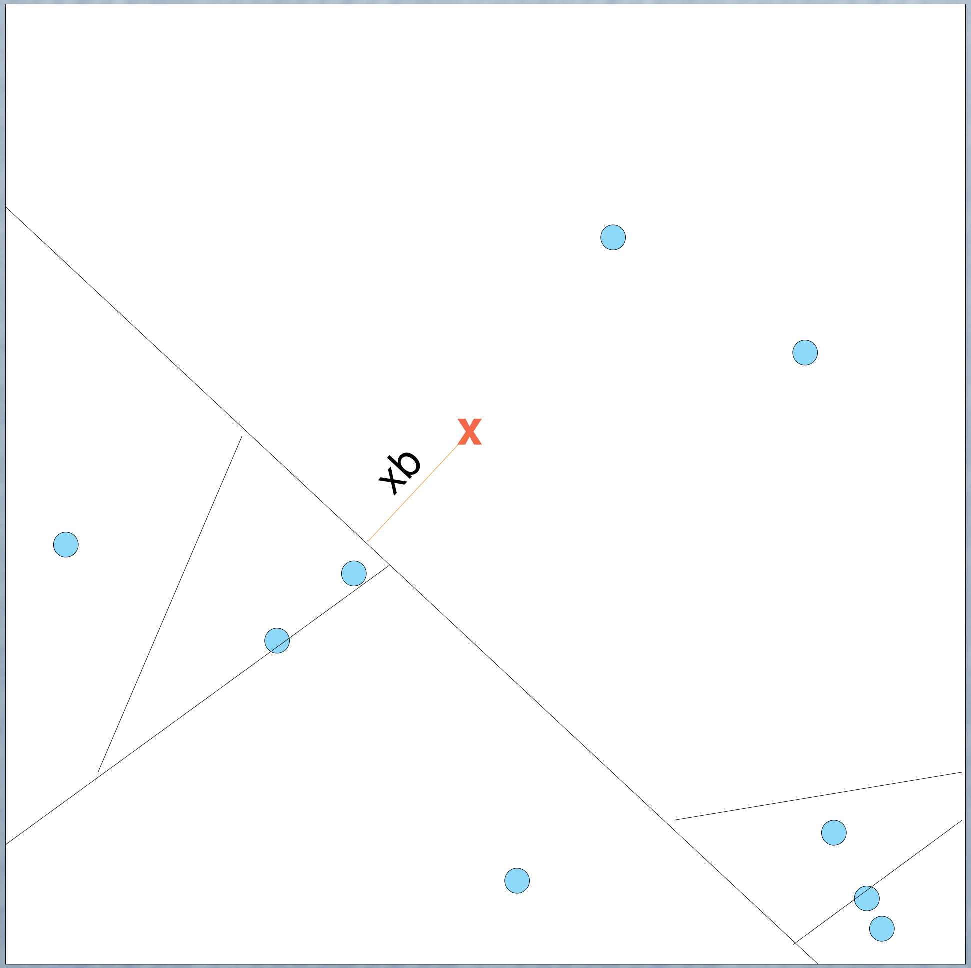
xa = 0

Priority Queue

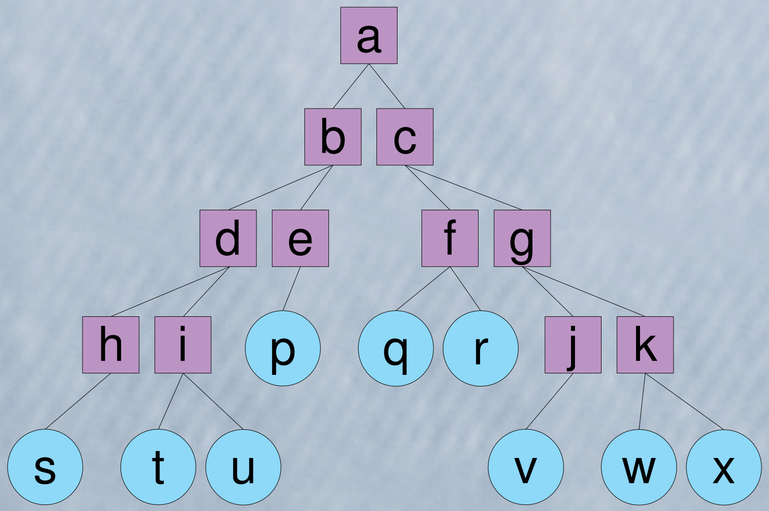
a

Head

Example



Tree Data Structure

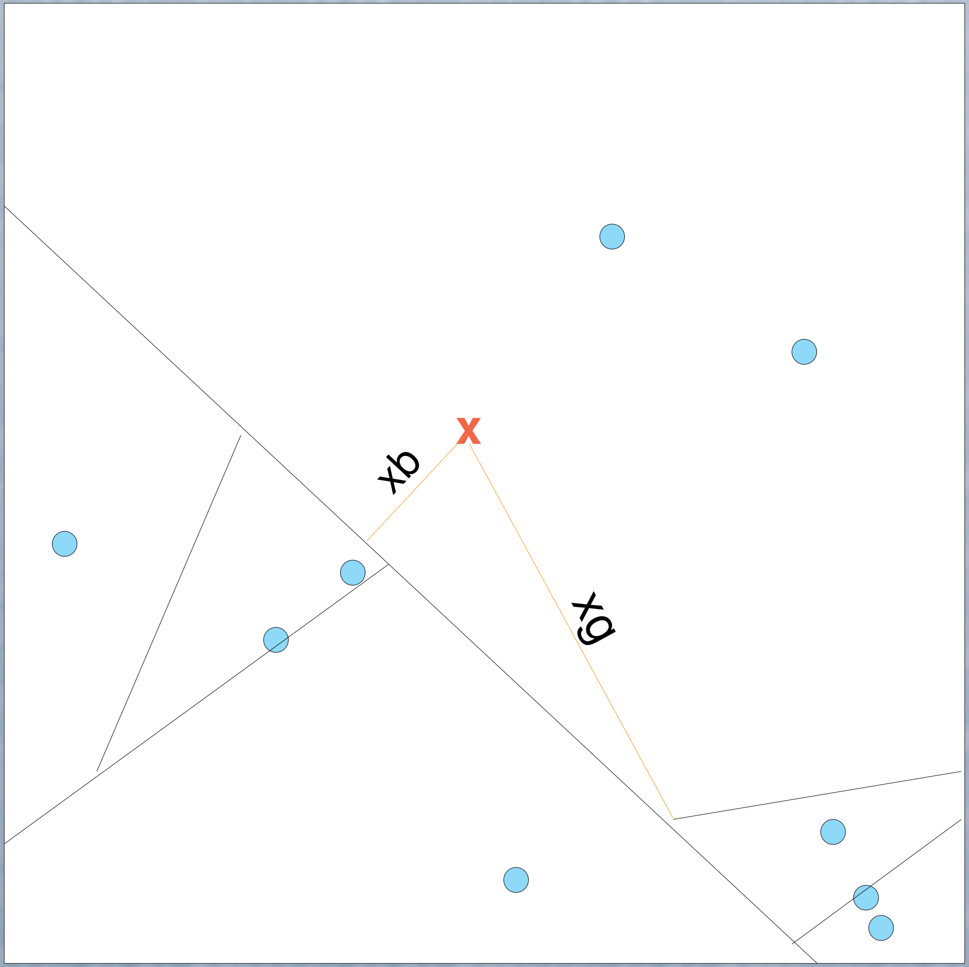


Priority Queue

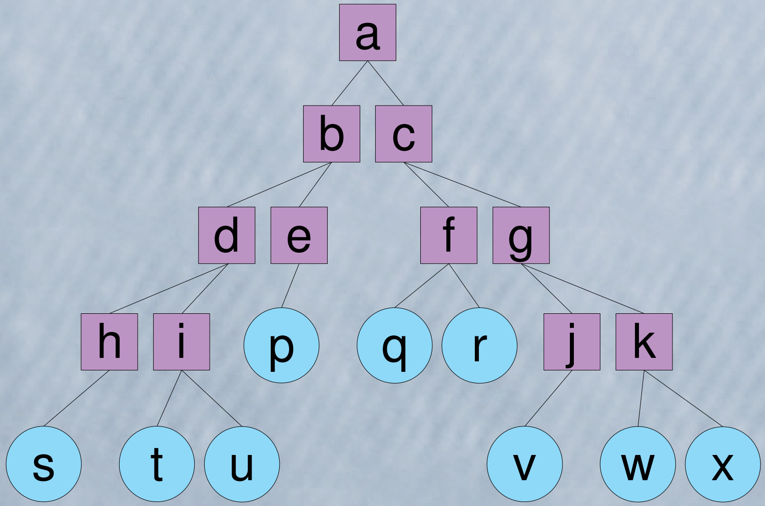
~~a~~ c b

Head

Example



Tree Data Structure

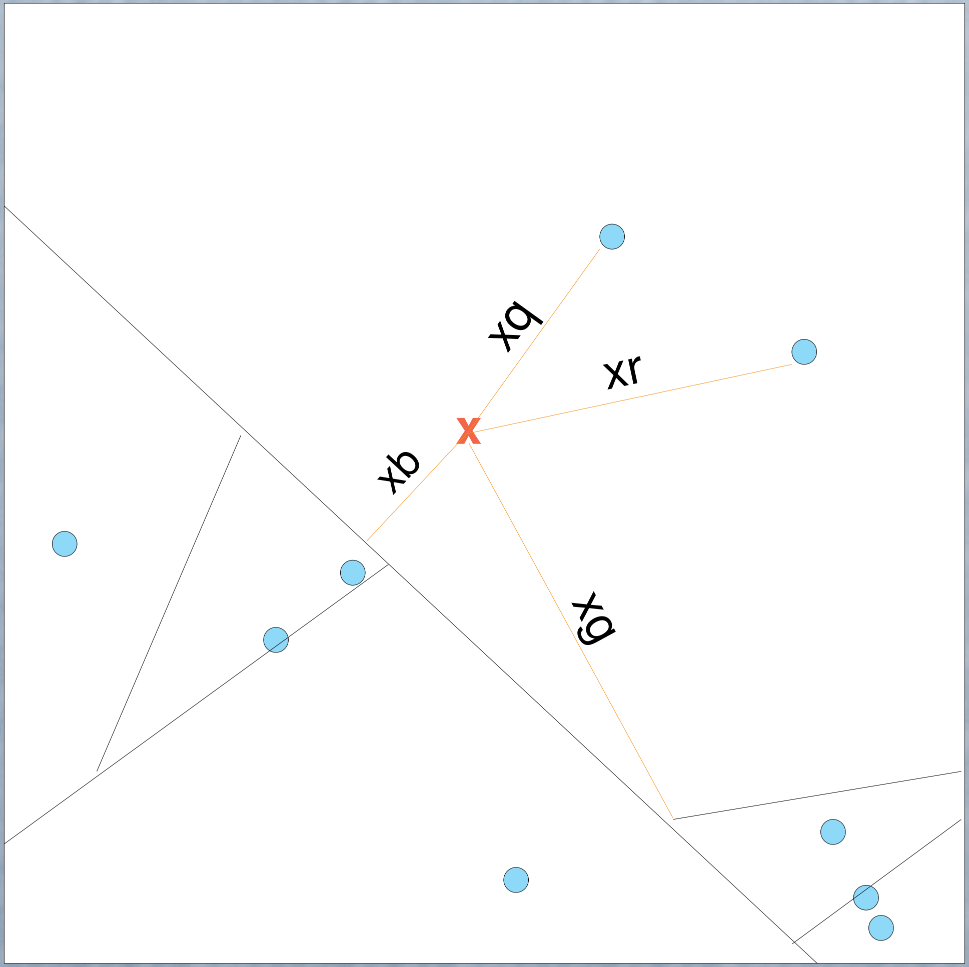


Priority Queue

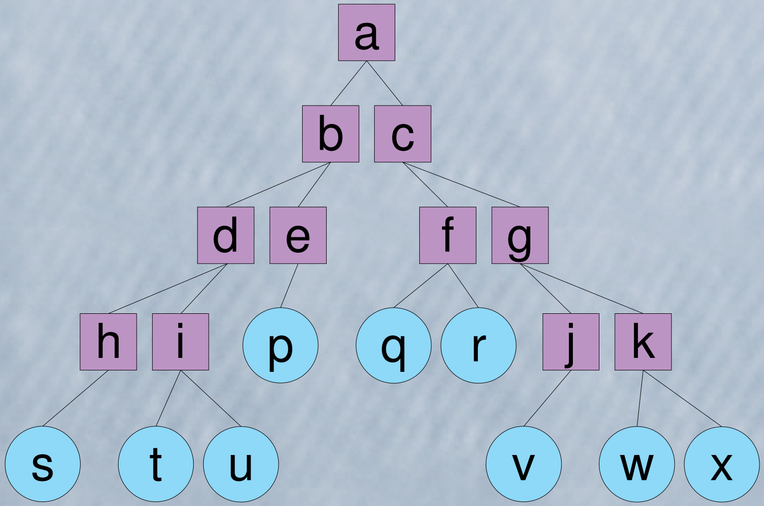
~~a c~~ f b g

Head

Example



Tree Data Structure

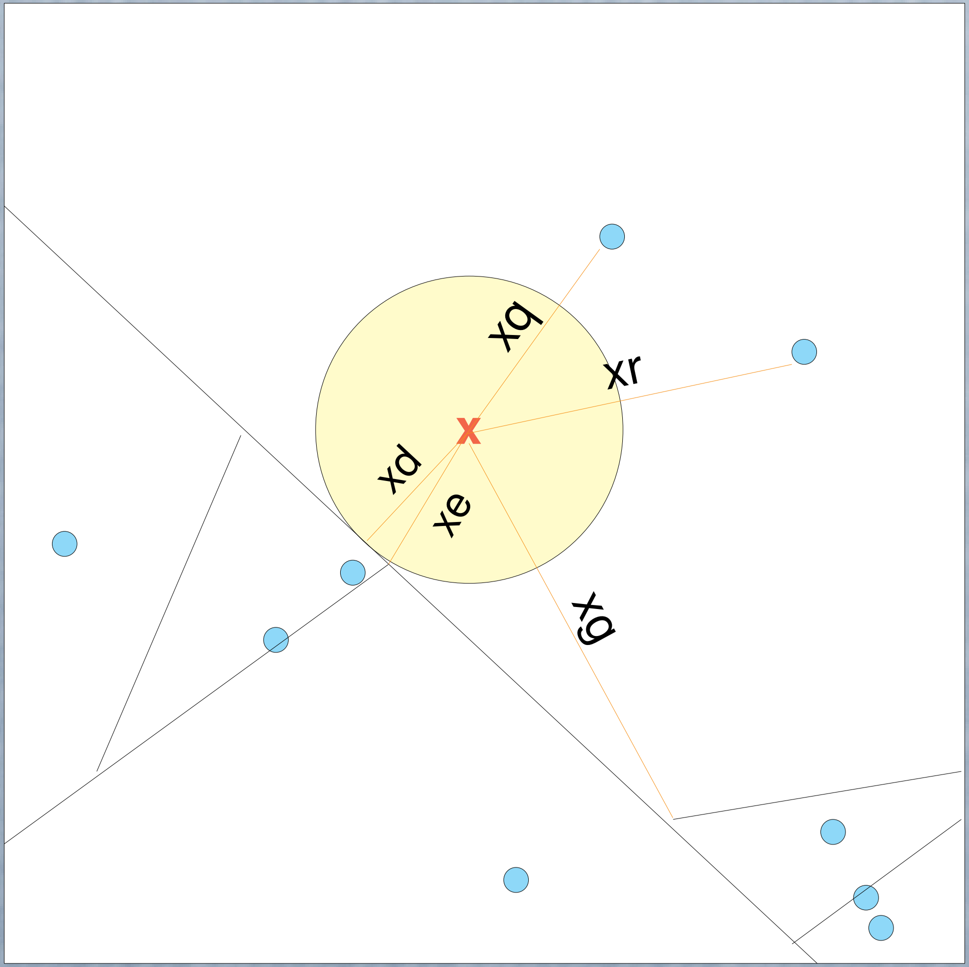


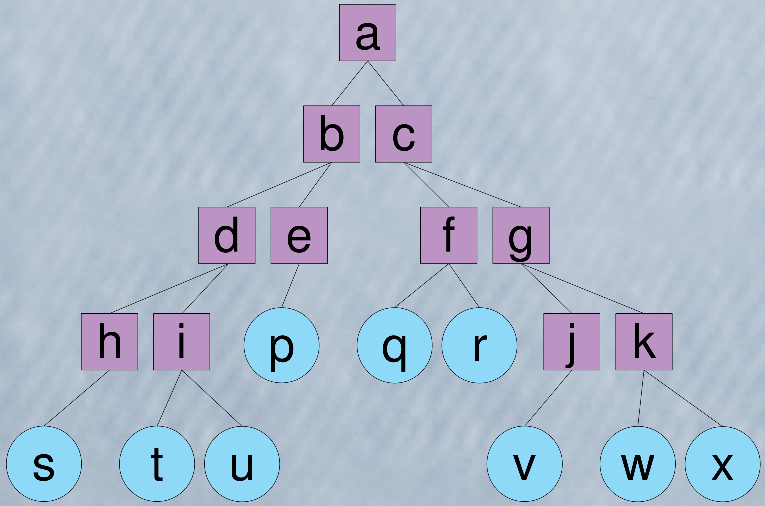
Priority Queue

~~a c f~~ b q r g

Head

Example



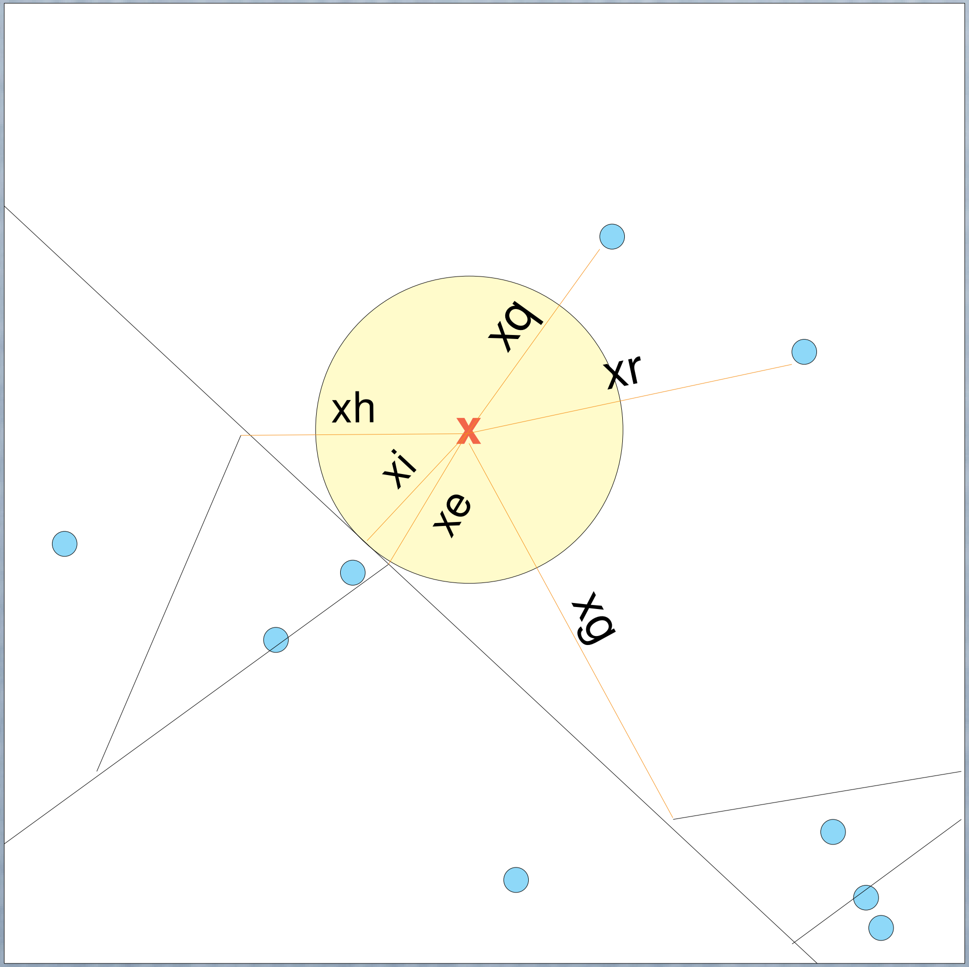


Priority Queue

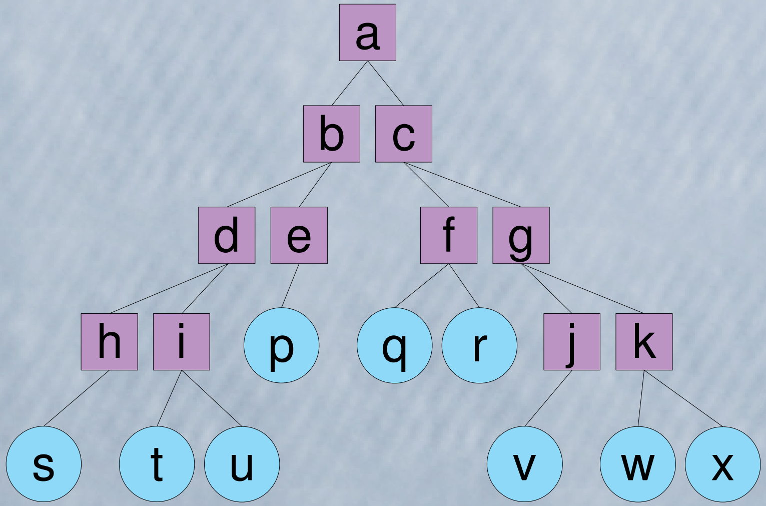
~~a c f b~~ d e q r g

Head

Example



Tree Data Structure

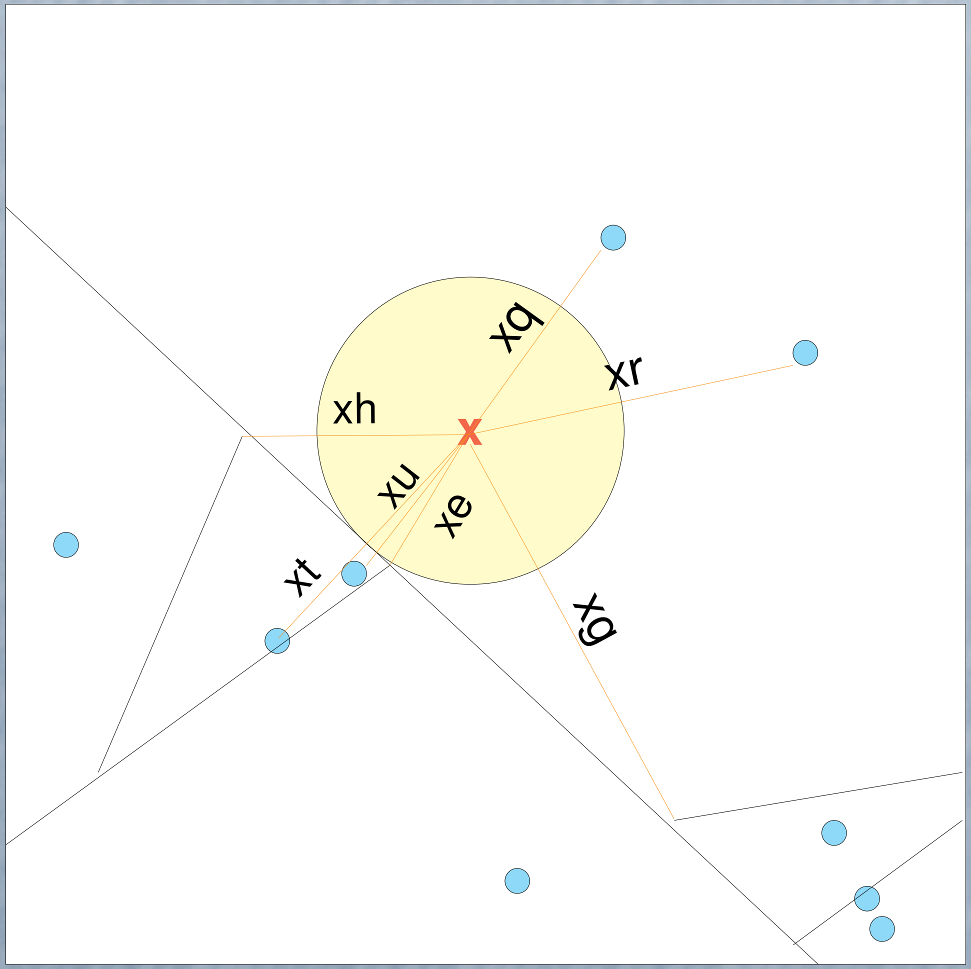


Priority Queue

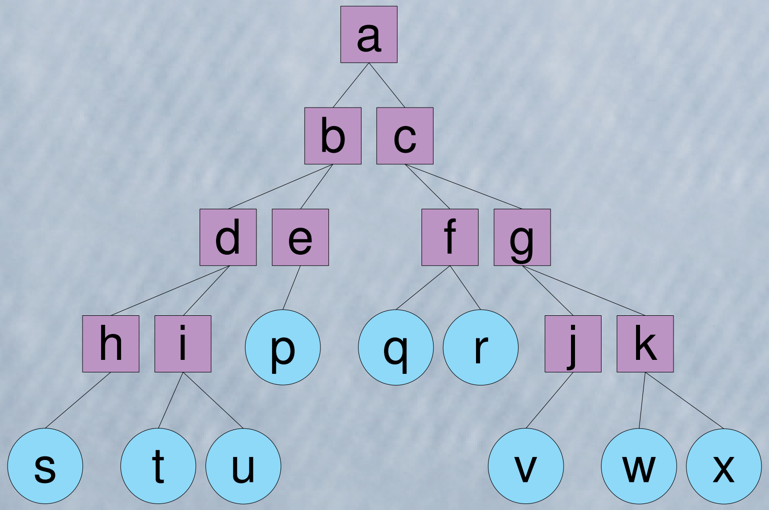
~~a c f b d~~ i e h q r g

Head

Example



Tree Data Structure

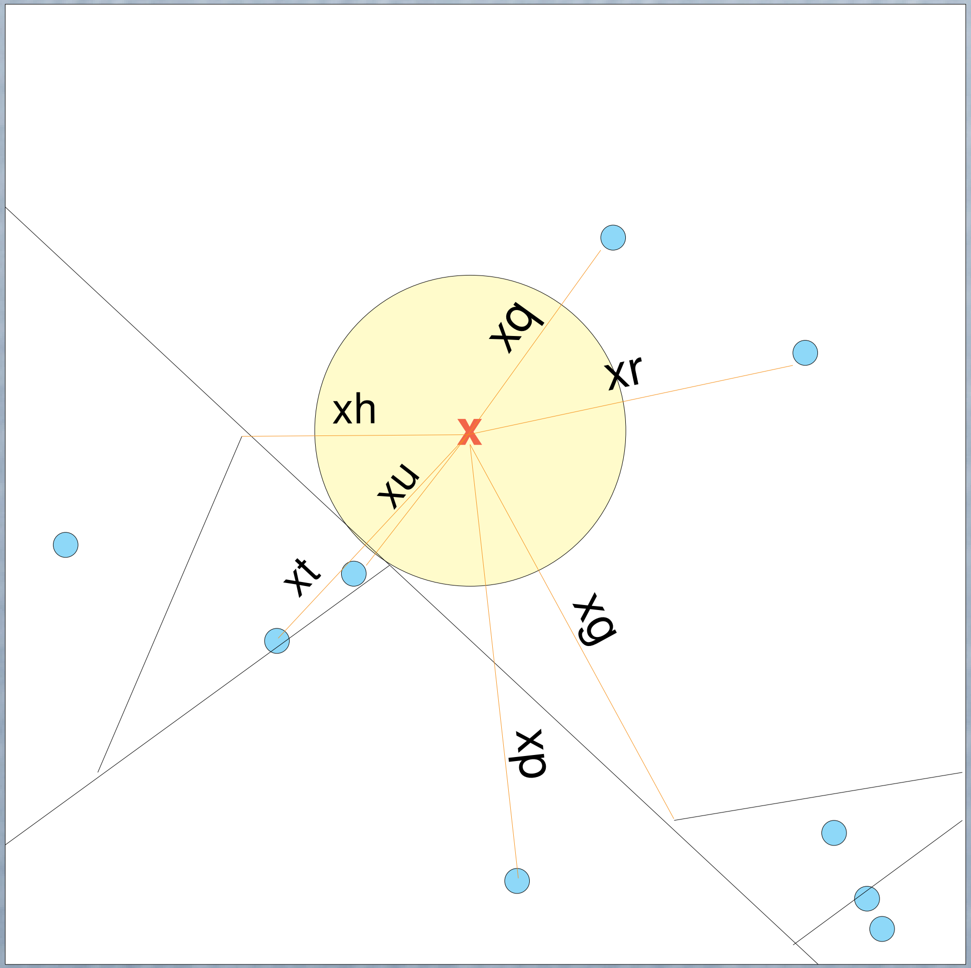


Priority Queue

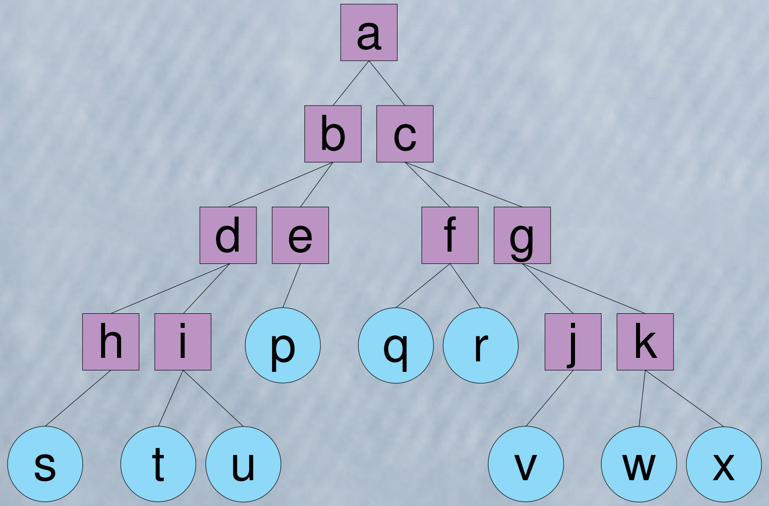
~~a c f b d i~~ e u h q t r g

Head

Example



Tree Data Structure

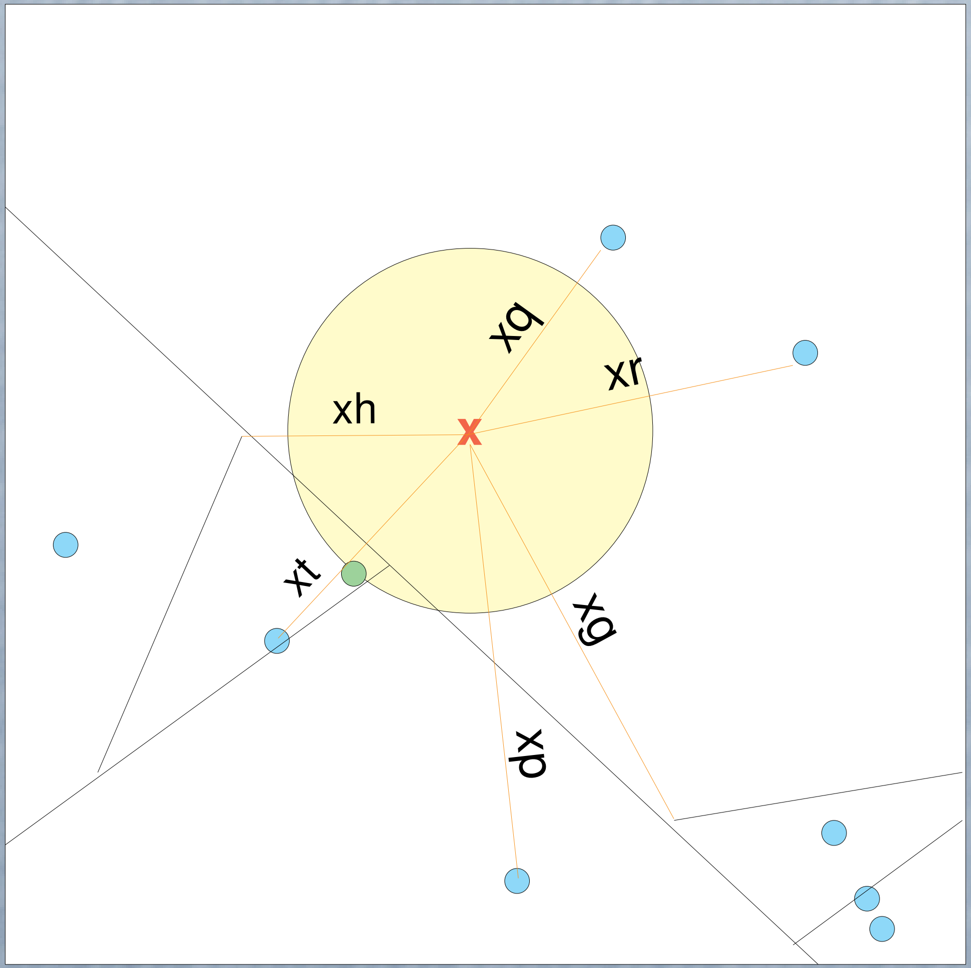


Priority Queue

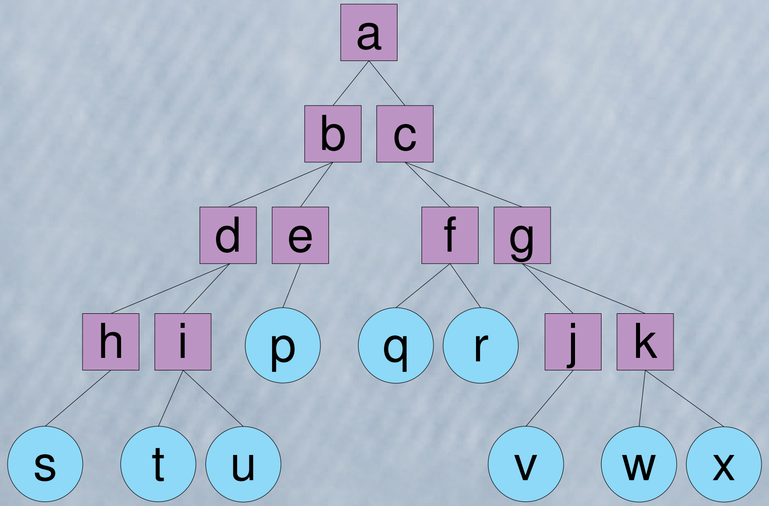
~~a c f b d l e~~ u h q t r g p

Head

Example



Tree Data Structure

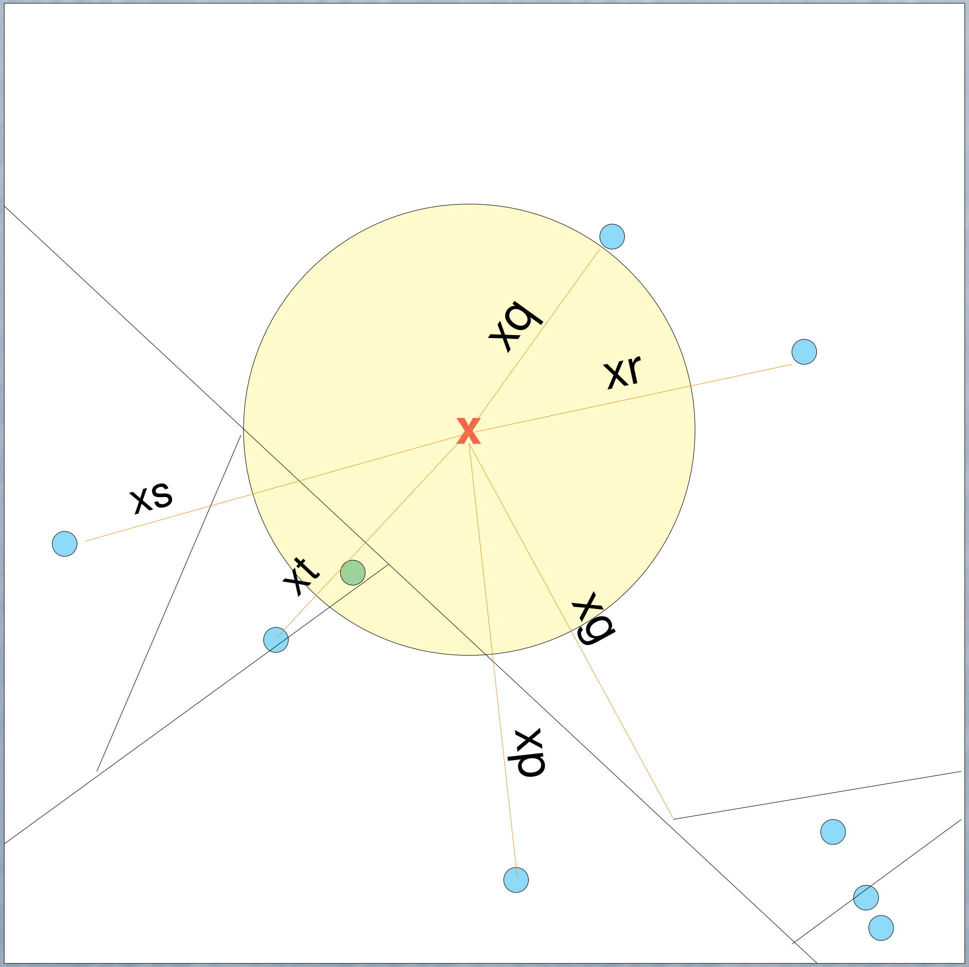


Priority Queue

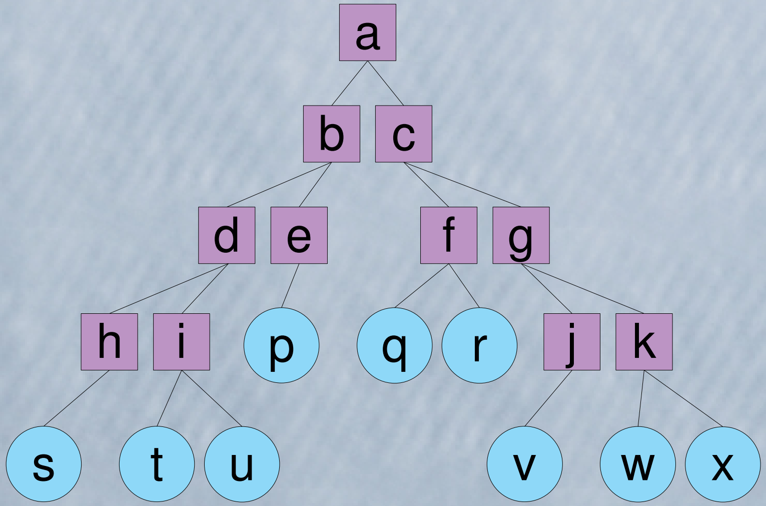
~~a c f b d l e u~~ h q t r g p

Head

Example



Tree Data Structure

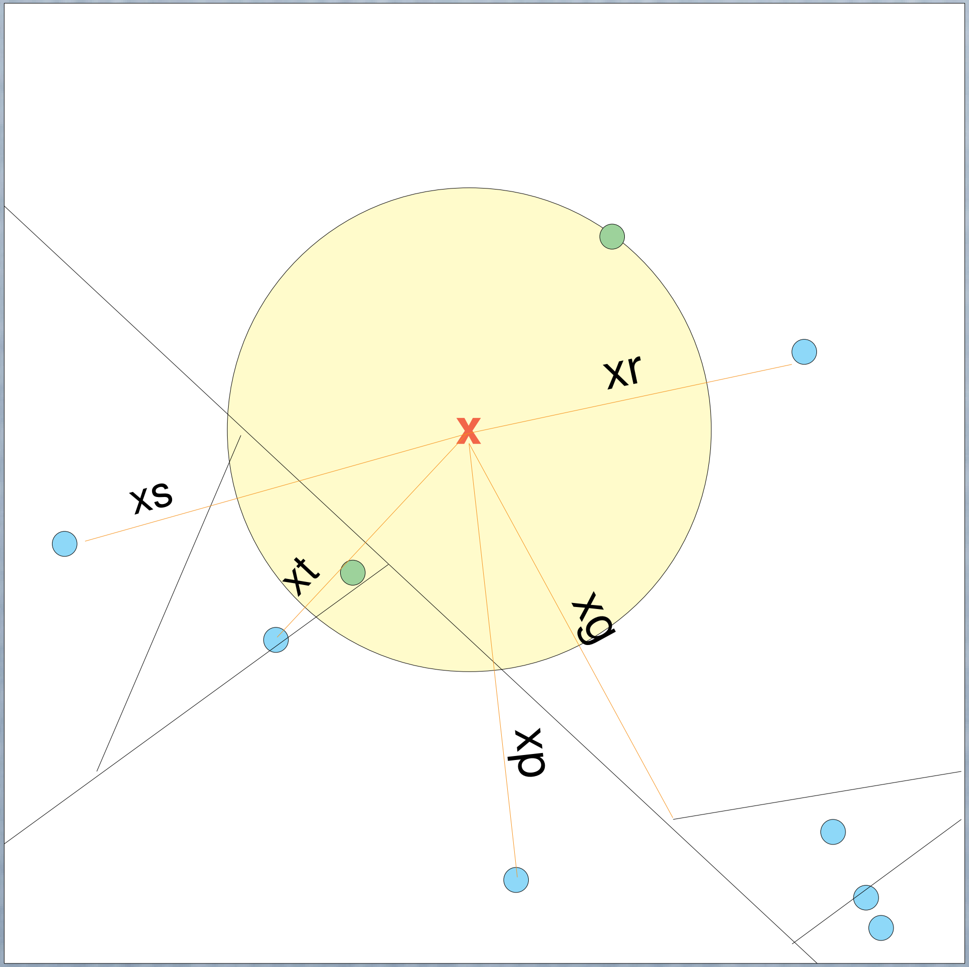


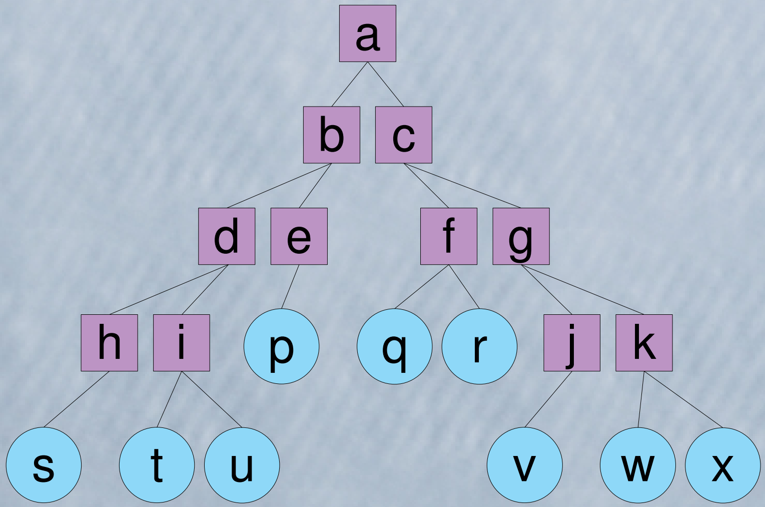
Priority Queue

~~a c f b d l e u h~~ q t r s g p

Head

Example



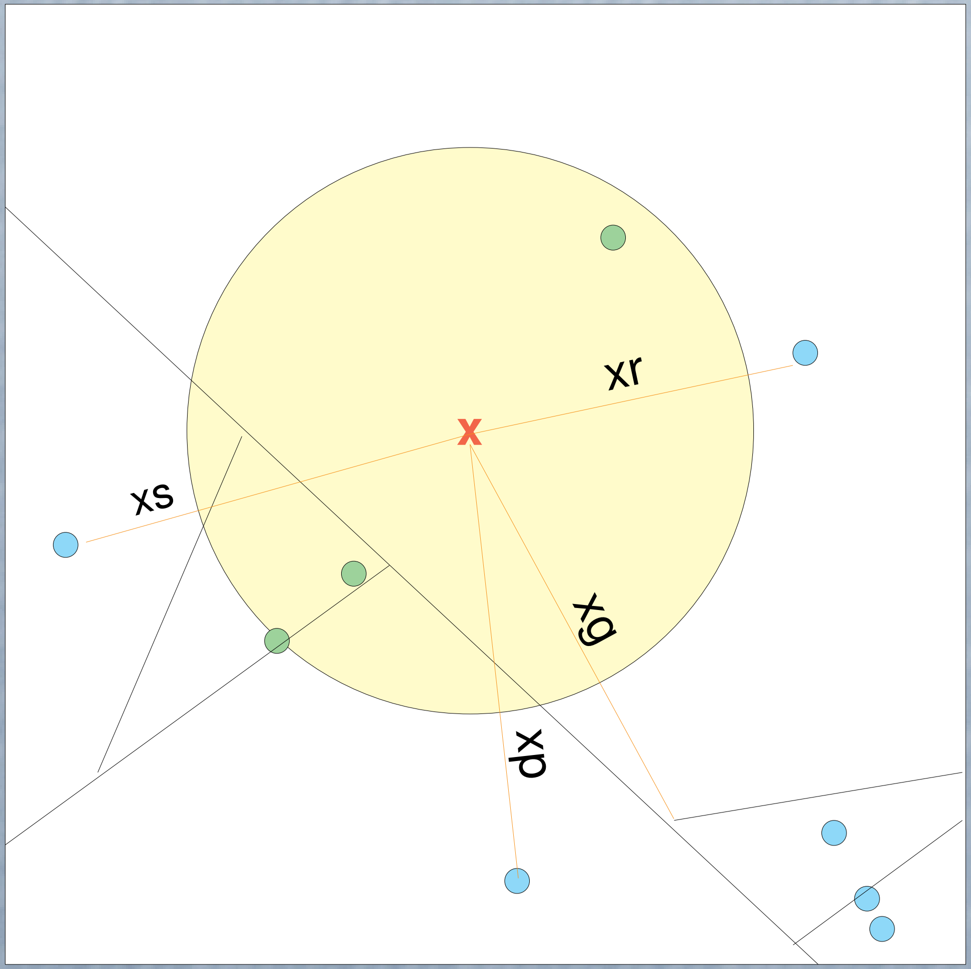


Priority Queue

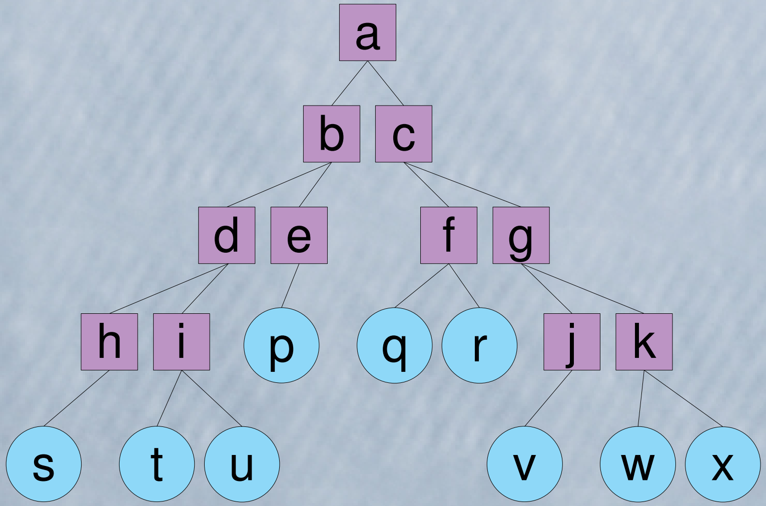
~~a c f b d l e u h q~~ t r s g p

Head

Example



Tree Data Structure

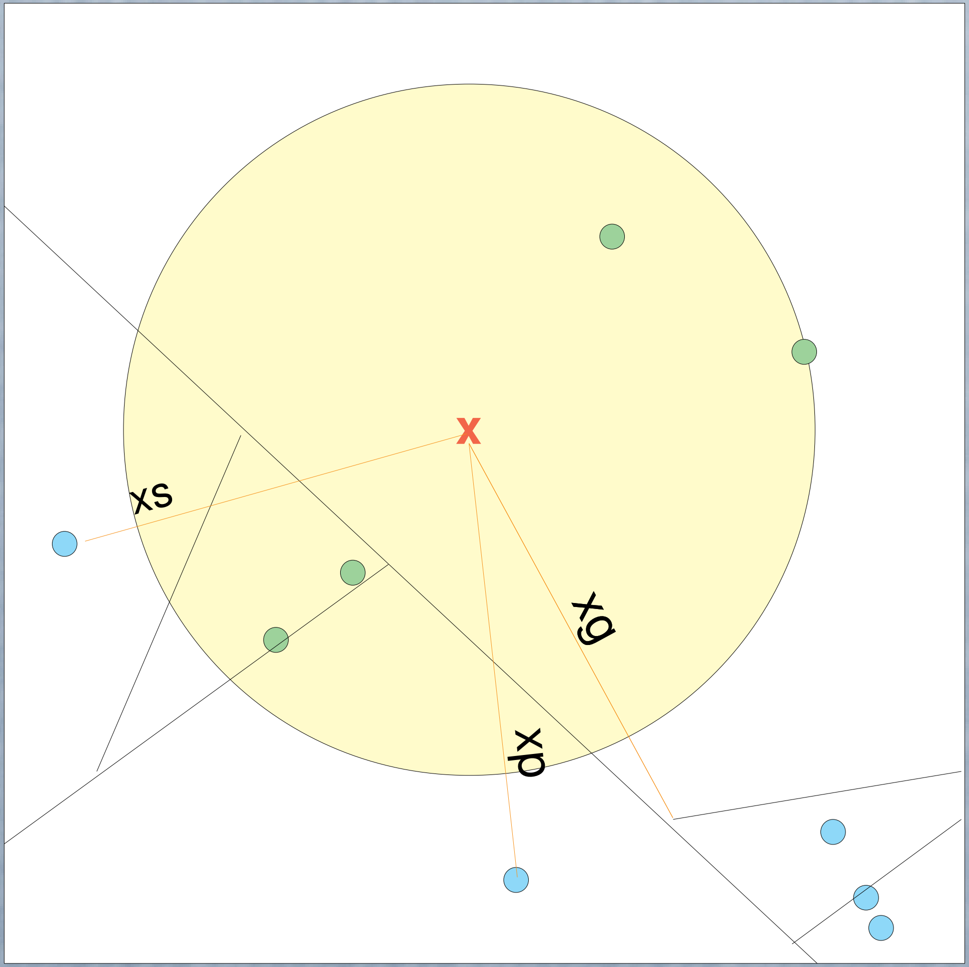


Priority Queue

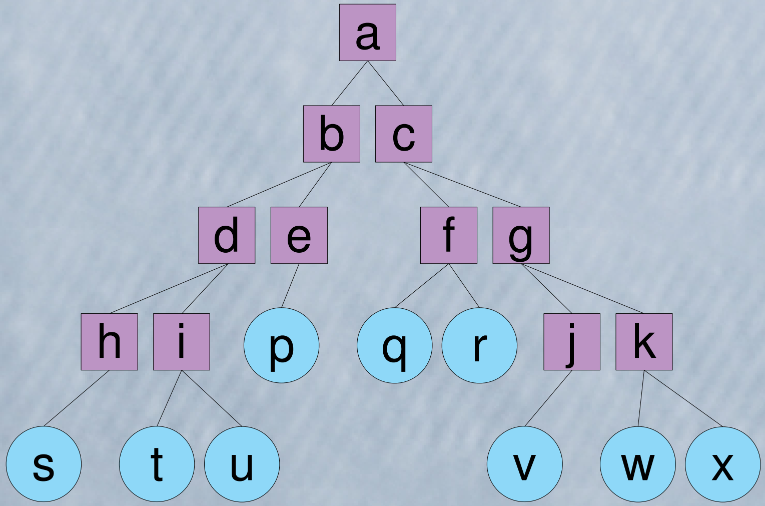
~~a c f b d l e u h q t~~ r s g p

Head

Example



Tree Data Structure

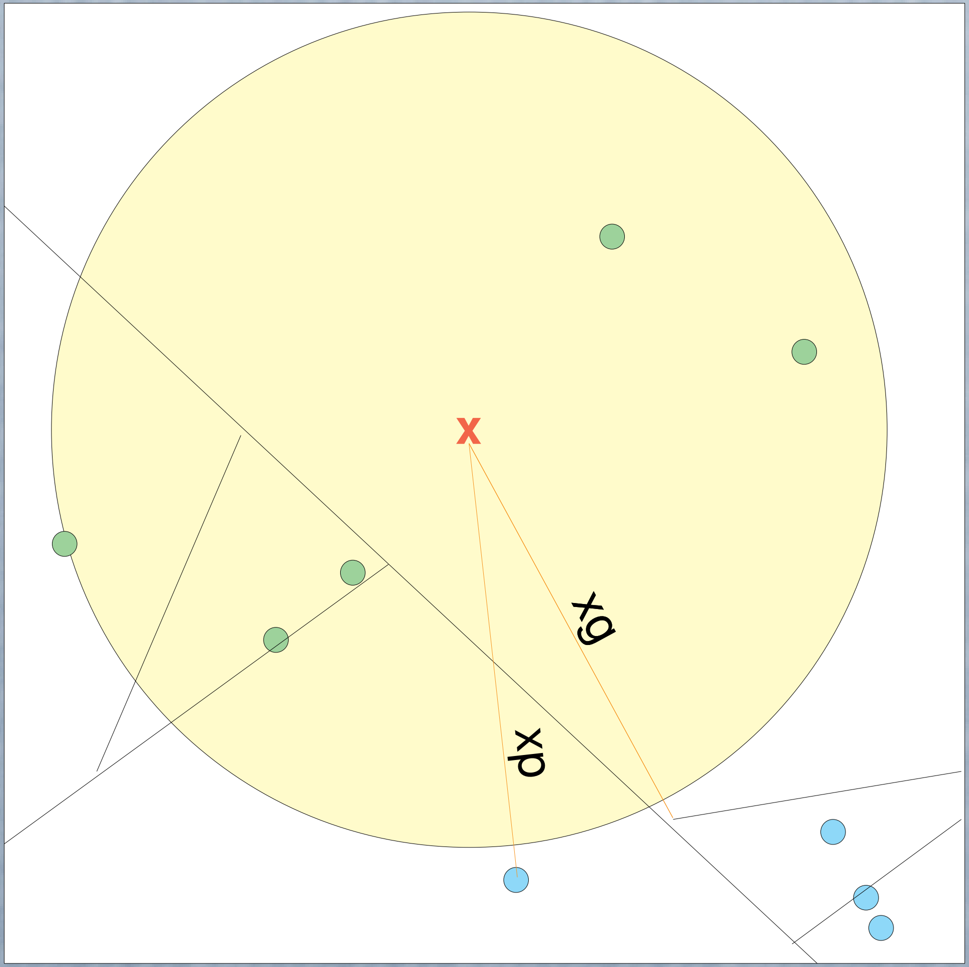


Priority Queue

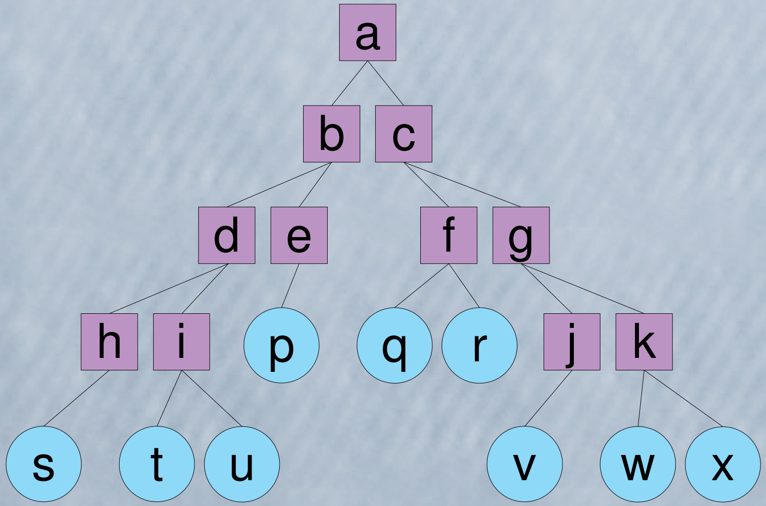
~~a c f b d l e u h q t r~~ s g p

Head

Example



Tree Data Structure

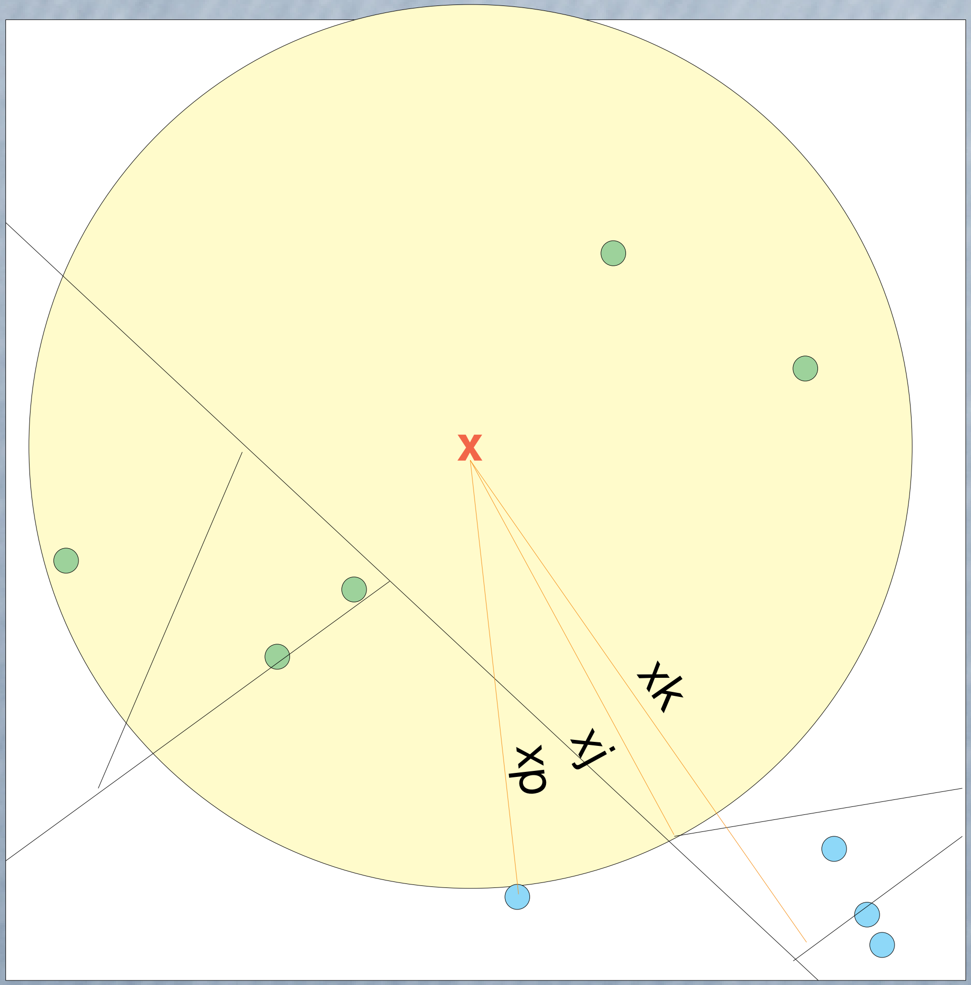


Priority Queue

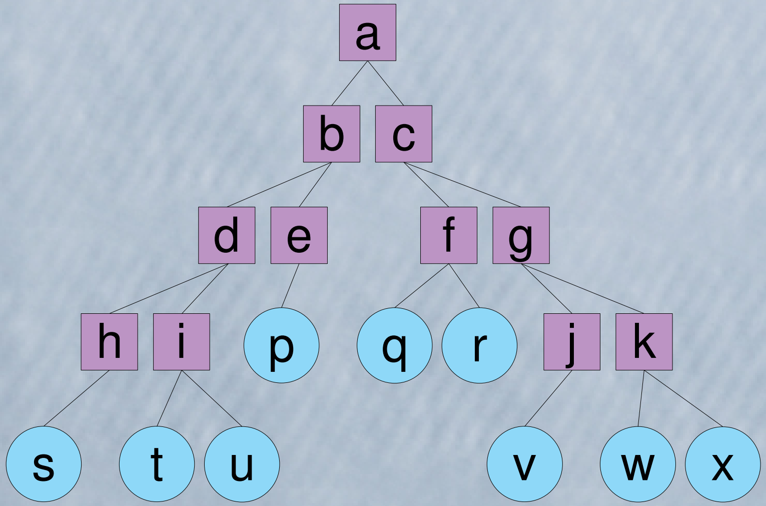
~~a c f b d l e u h q t r s~~ g p

Head

Example



Tree Data Structure

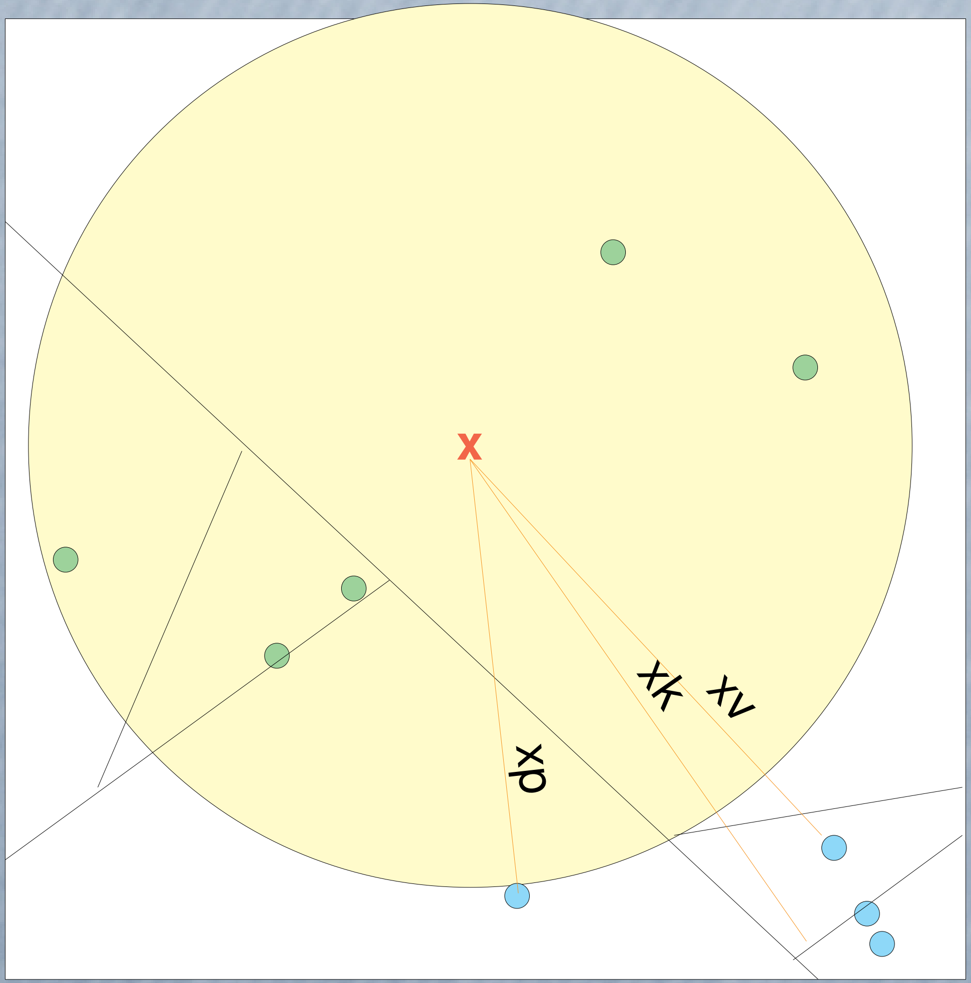


Priority Queue

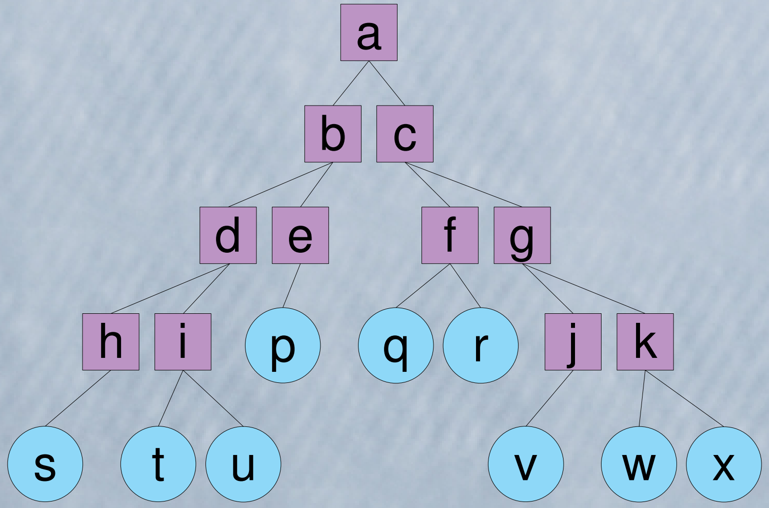
~~a c f b d l e u h q t r s g~~ j p k

Head

Example



Tree Data Structure

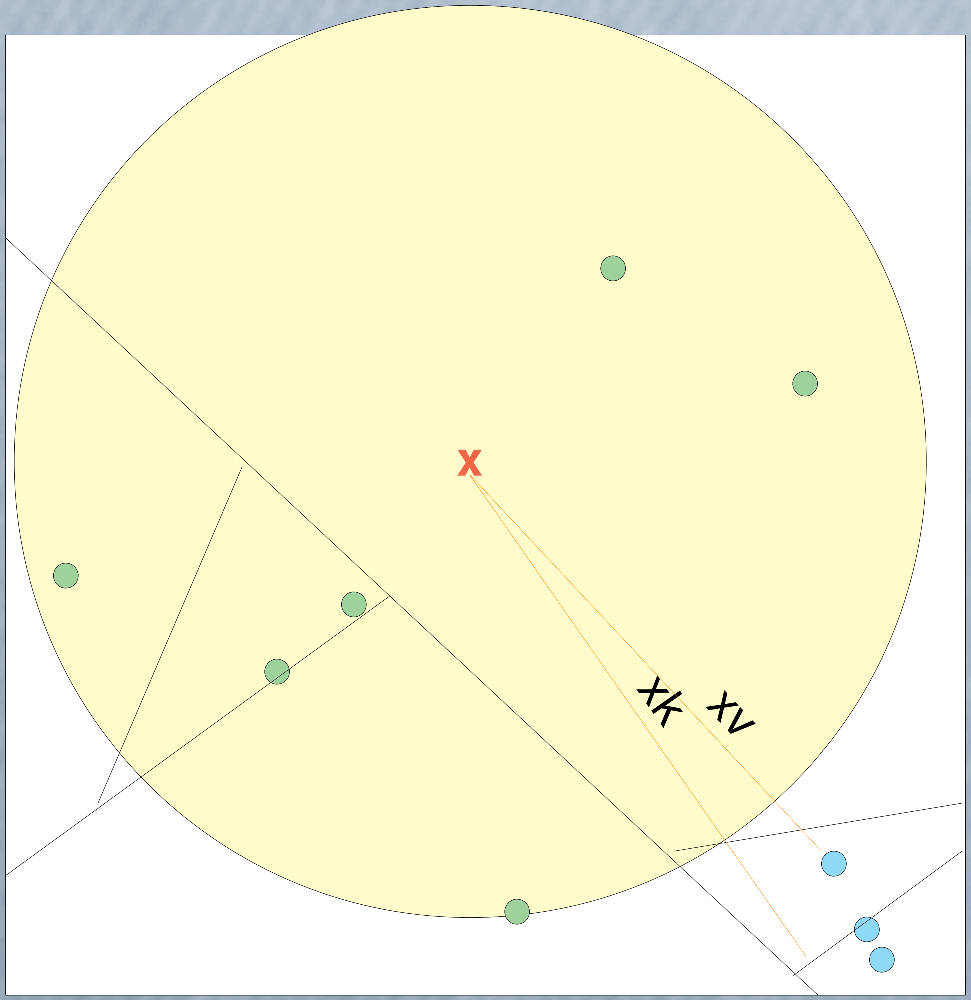


Priority Queue

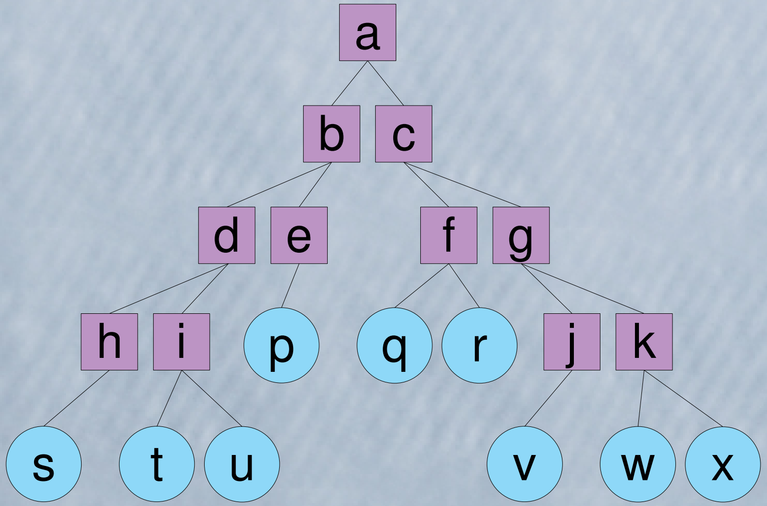
~~a c f b d l e u h q t r s g j~~ p v k

Head

Example



Tree Data Structure



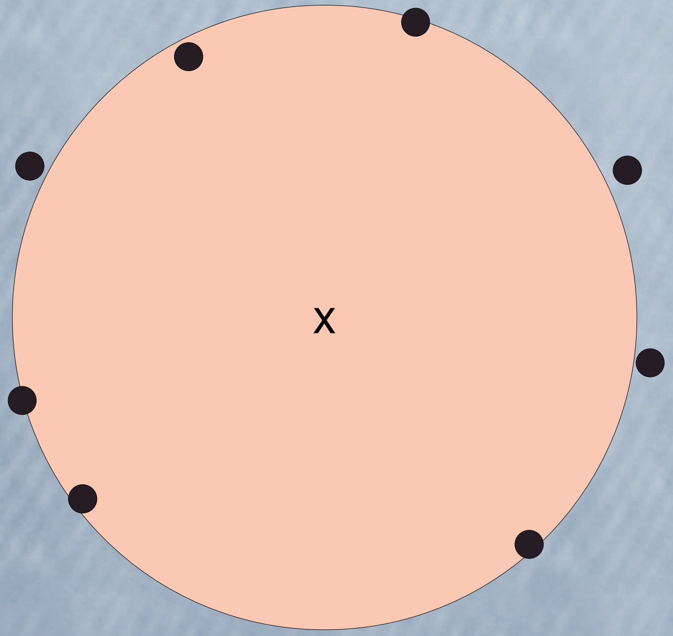
Priority Queue

~~a c f b d i e u h q t r s g j p~~ v k

Head

Worst case

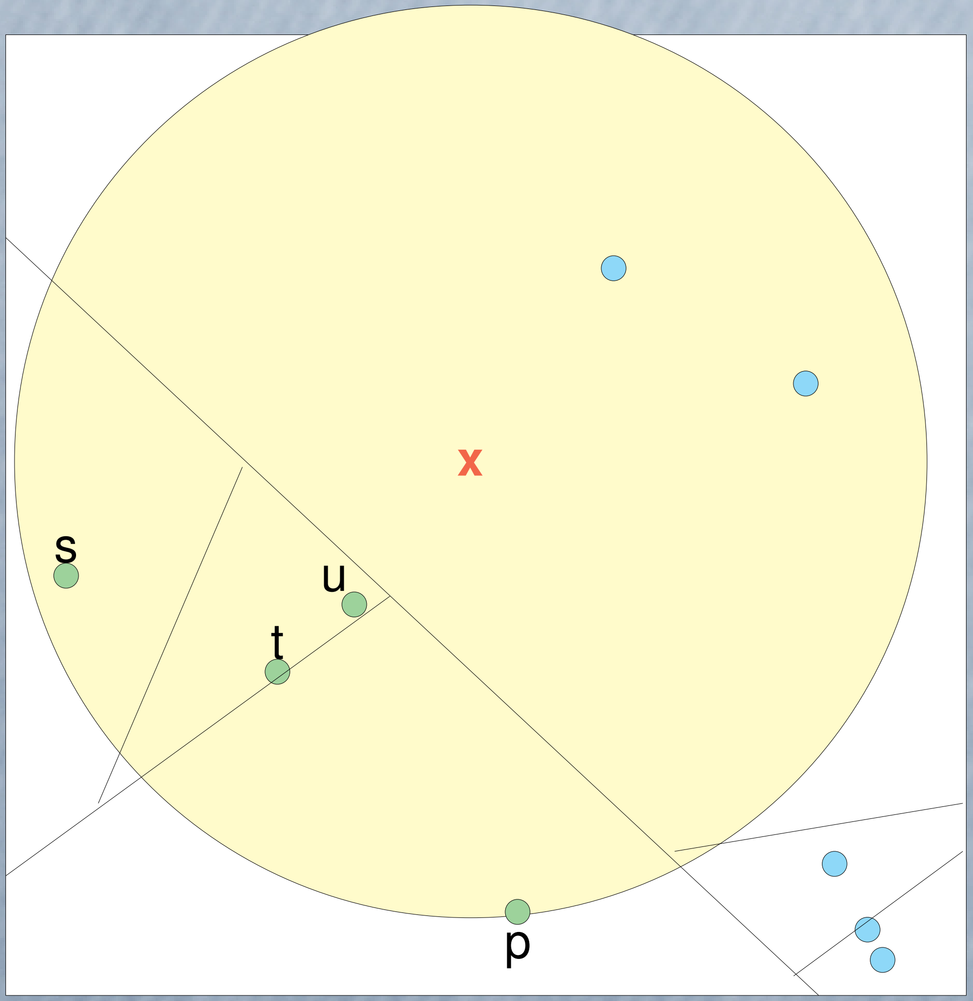
- All points about the same distance from x
- Why bad?



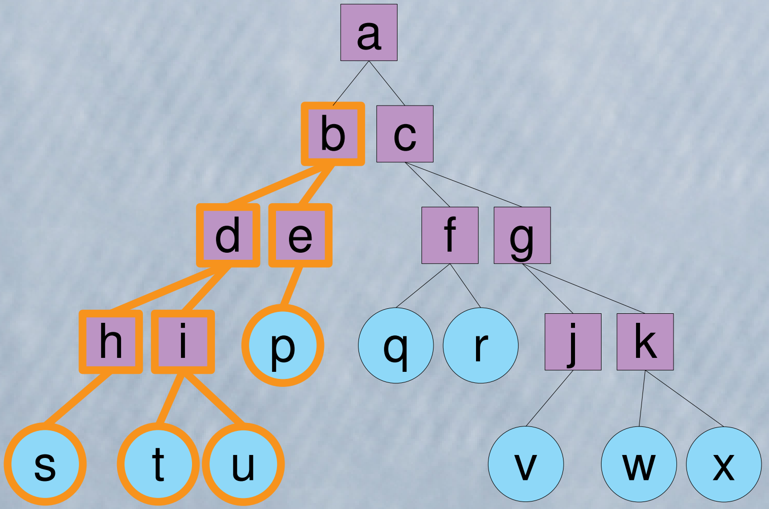
Depth first KNN

- Maintain a candidate set of *k* points
- Only searches within the range of the farthest of the candidates
- Priority queue of farthest candidates away
- Depth-first to reach points quickly

Example



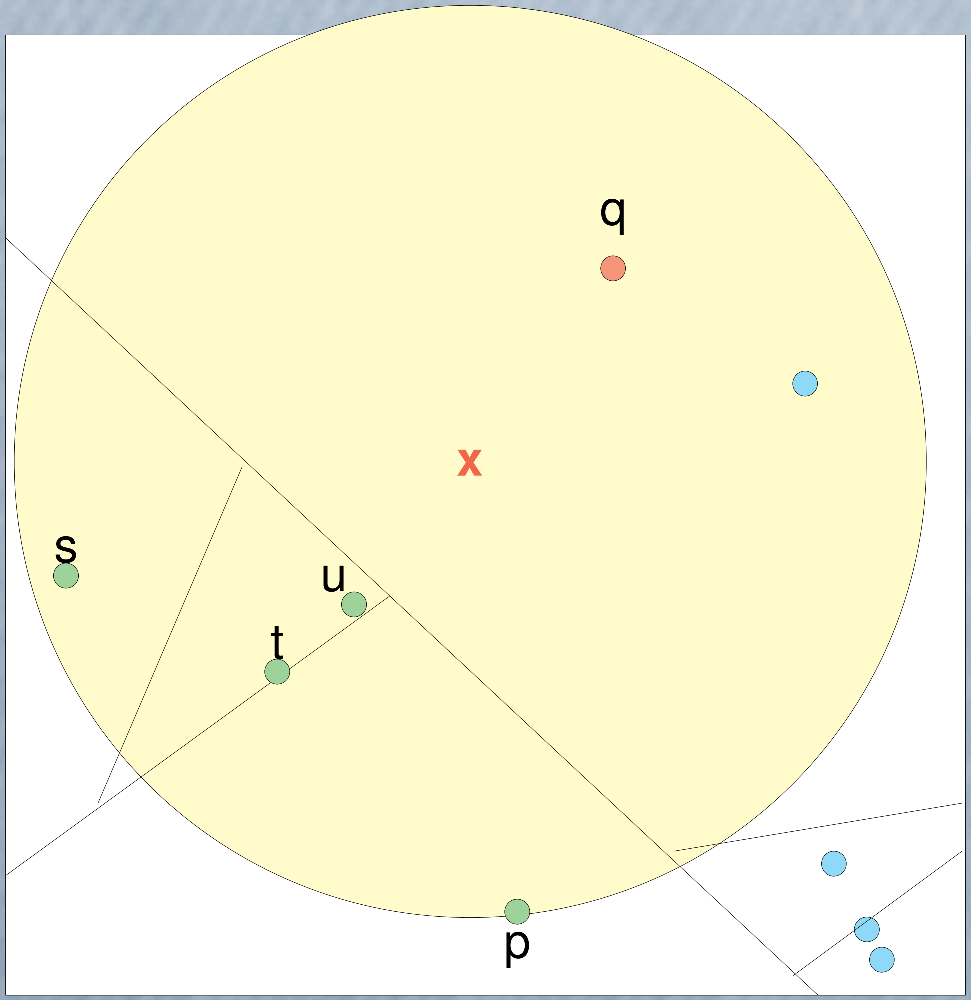
Tree Data Structure



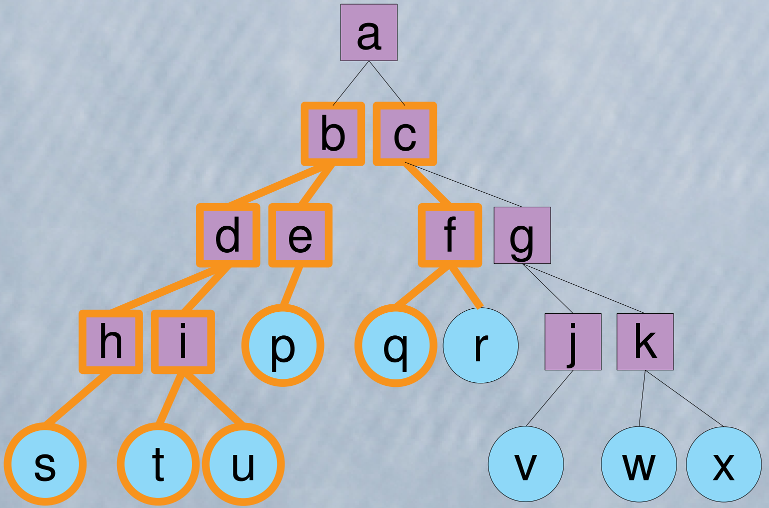
Candidates priority queue

p s t u

Example



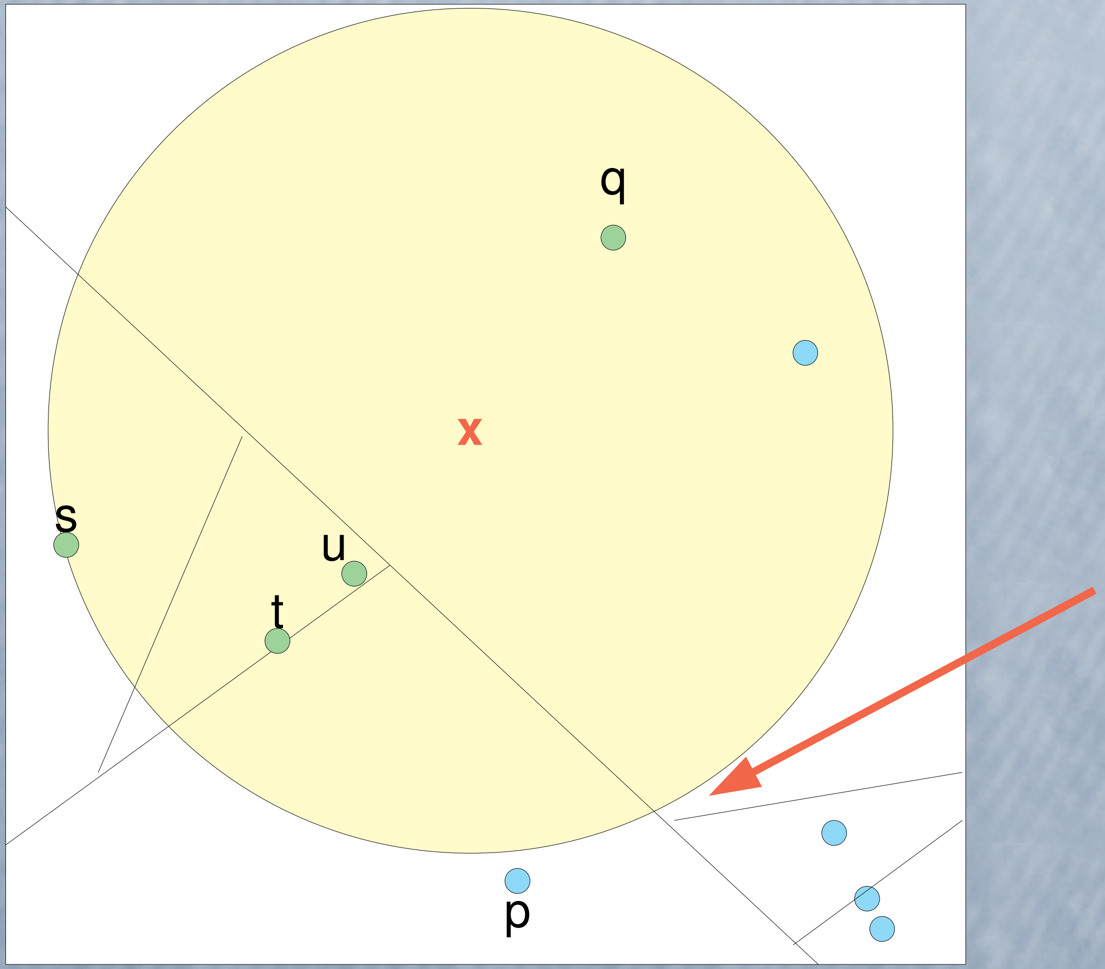
Tree Data Structure



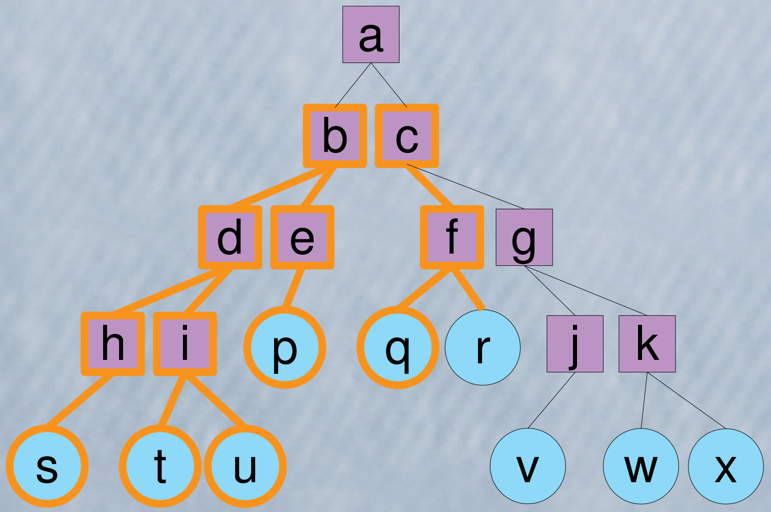
Candidates priority queue

p s t u

Example



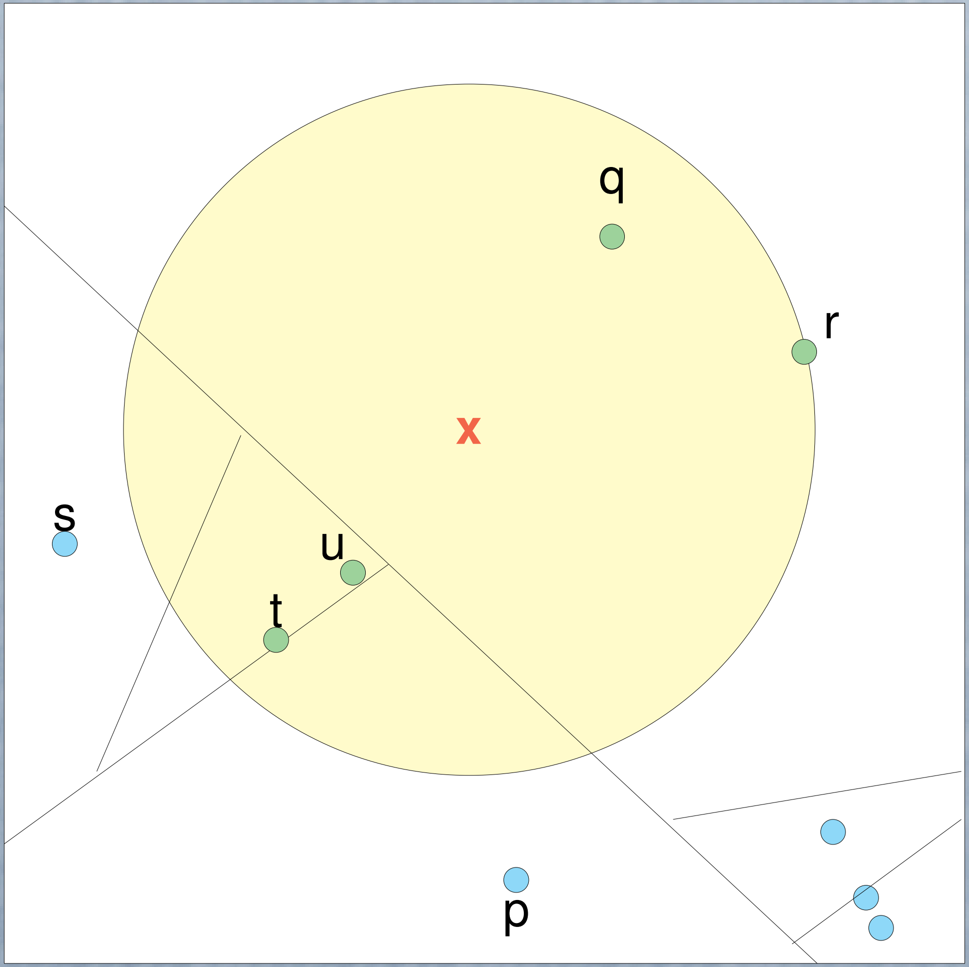
Tree Data Structure



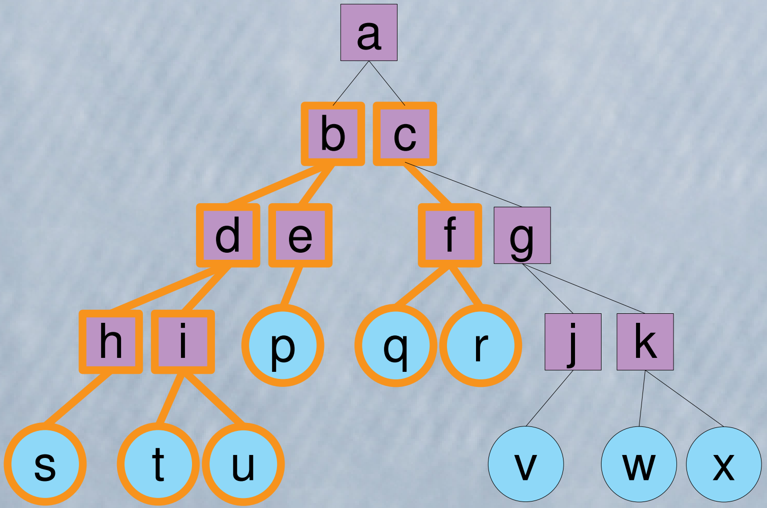
Candidates priority queue

s q t u

Example



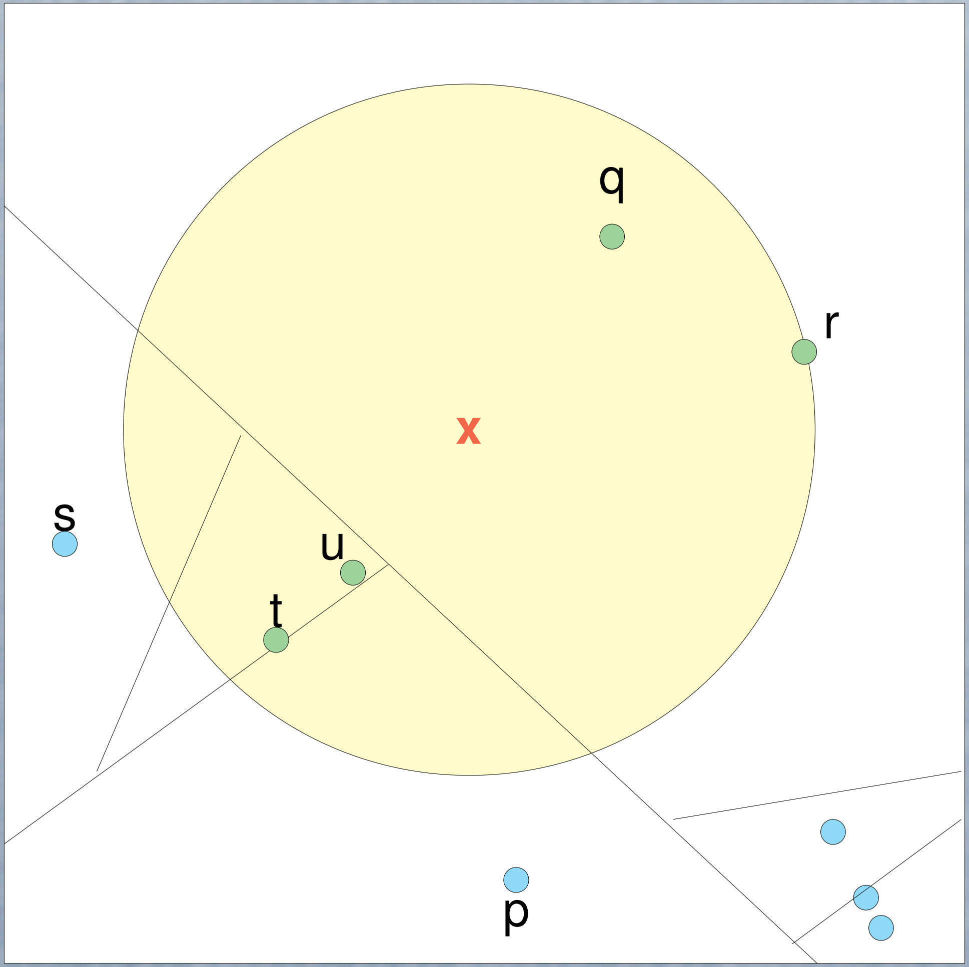
Tree Data Structure



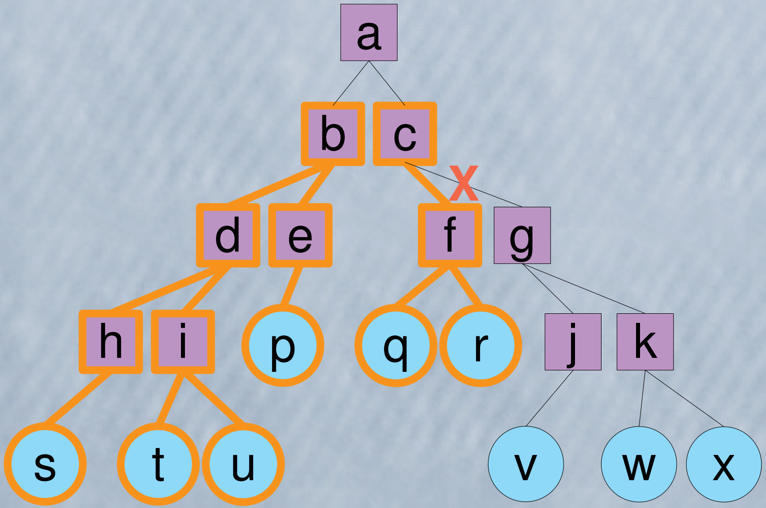
Candidates priority queue

r q t u

Example



Tree Data Structure



Candidates priority queue

r q t u

Midterm – April 9

- 90 Minutes
- Points breakdown (out of 100 possible):
  - 30pts Short answer questions
  - 30pts Coding (You must write any functions you call that aren't already provided)
  - 40pts Long answer questions (essay style)
- Jury still out on open-book vs closed-book

Midterm topics

- Spatial data structures and operations on them
- Space-filling curves
- Bounding volumes
- Geometric operations like intersection, topo-skeleton, etc.
- Anything else covered in lecture / projects
- Reading assignments

Data structures

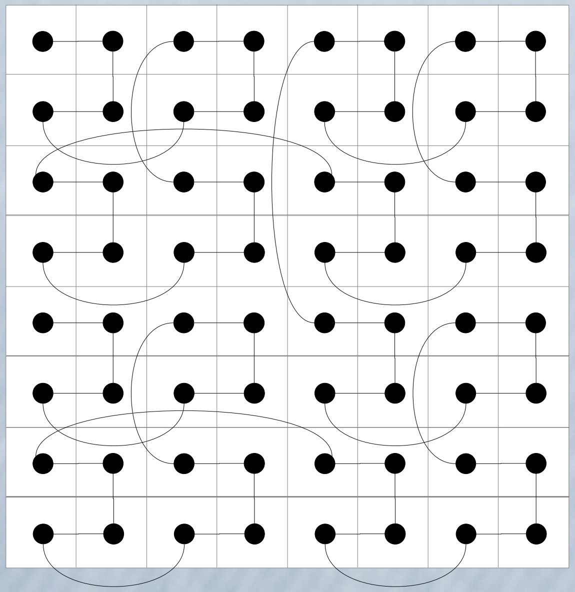
- Multi-D Range trees
- Bounding volume hierarchies
- KD-trees / BSP-trees
- Regular / irregular grids
- Quadtrees / Octrees

Operations

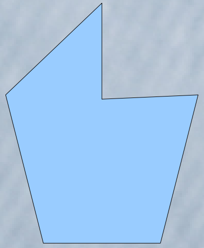
- Build / Insert / Erase
- Splitting heuristics
- Range search / radius search
- Ray cast
- (k-) Nearest neighbor

- Know how to perform them and the caveats

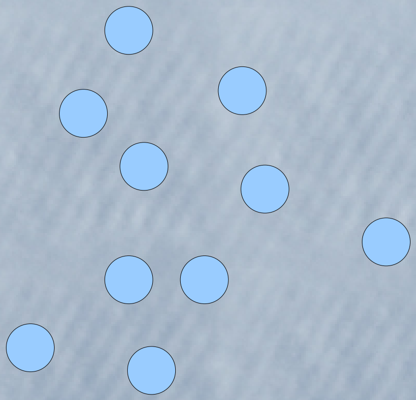
Short Question Examples



- Draw the double gray order space filling curve for an 8x8 cell grid. Is it admissible? Why?



- Draw the topological skeleton for this polygon



- Build a KD-tree from these data points using median split and alternating split axes starting with x. Represent it both spatially and as memory would be allocated for the data structure. What is the average-case complexity of this build?

Coding Question Examples

- Provide pseudo-code to perform a range search on an irregular grid.
- Fill in the C++ function to grow a d-dimensional axis-aligned bounding box to include point X:

```
template <class real, int d>
class aabb{
   real m_min, m_max;
   ...
   void expand_to_include(real * X) {                }
};
```

Long Question Examples

- Given an application using airborne sensor data: you will get one data point at a time, not far from the previous data point, and each time you get a point, you need to perform a k-nearest neighbor search from that point. Which data structure do you think is most appropriate for this application and why?
- Compare and contrast the benefits and drawbacks of grids (applying to both regular/irregular) versus kd-trees for ray casting on static data.

Next Time

- Project 2 Presentations
- Midterm review open session (Bring questions!)
- Project 3 Specs
- Required Reading: Chapter 4 (p. 485-499, 502-507, 517-521, 557-563)
- Recommended: Ch4, Sec. 1-3 (p. 485 – 563)