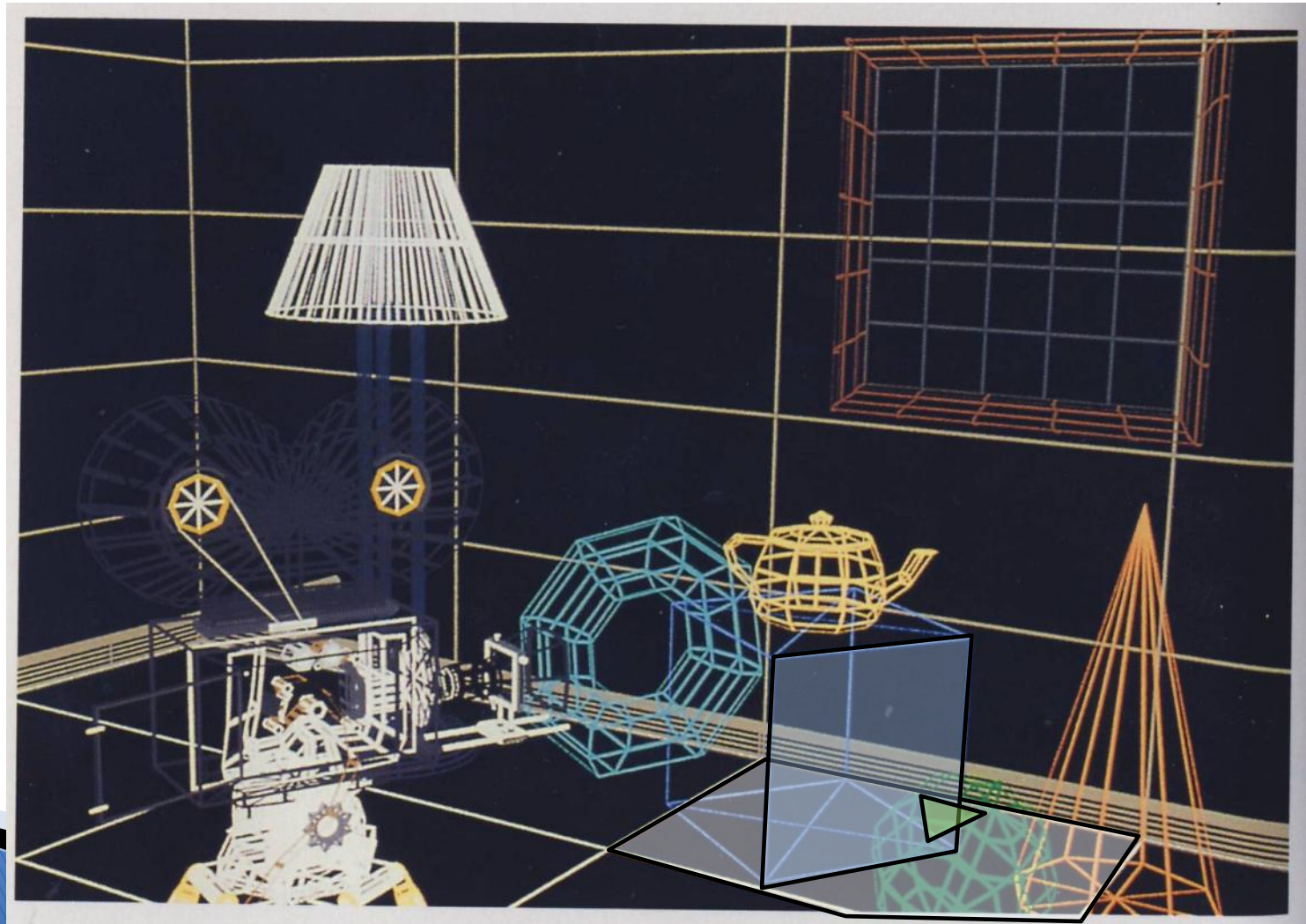
Computer Graphic

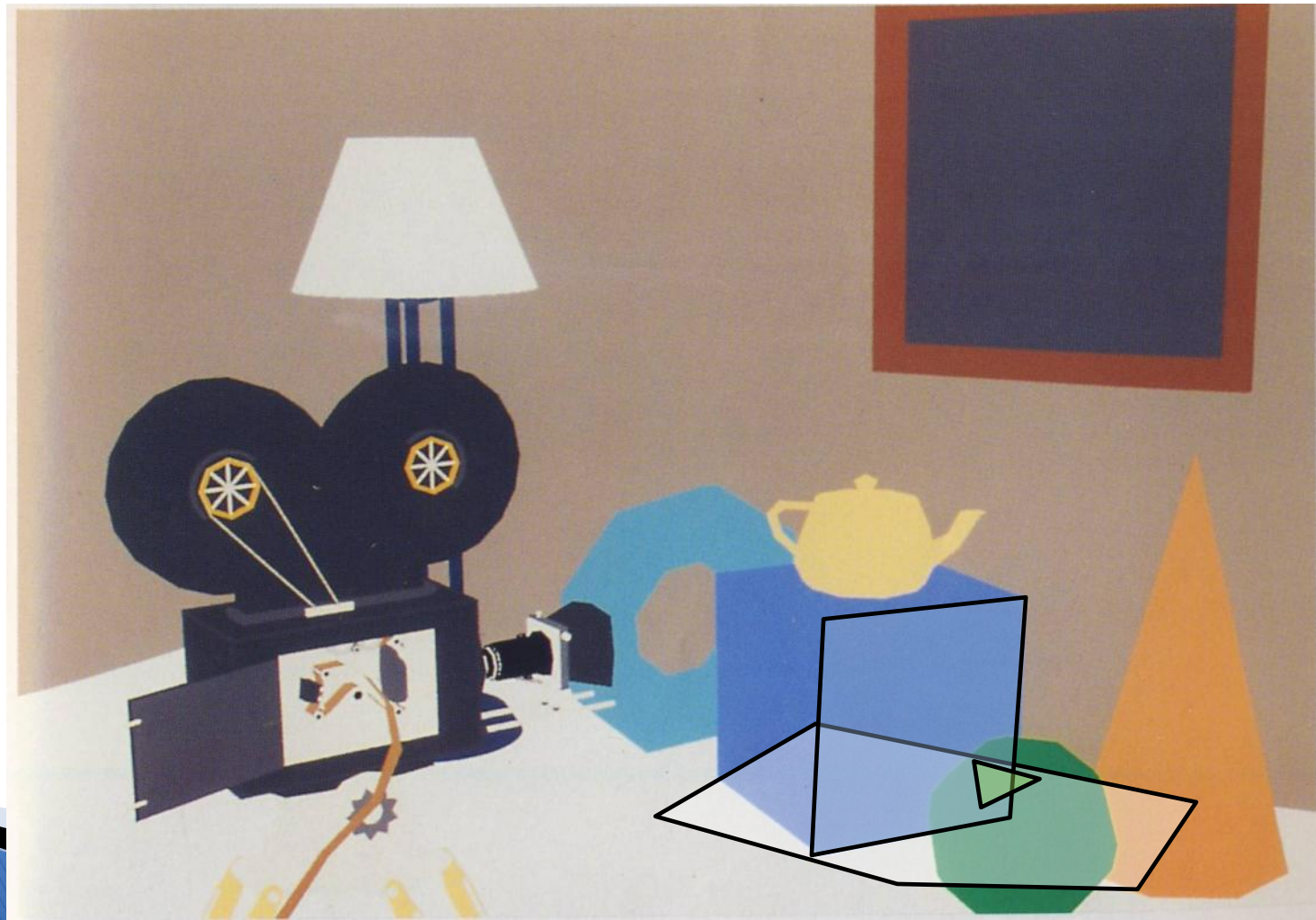
Hidden Surface Removal

Transformation/Viewing

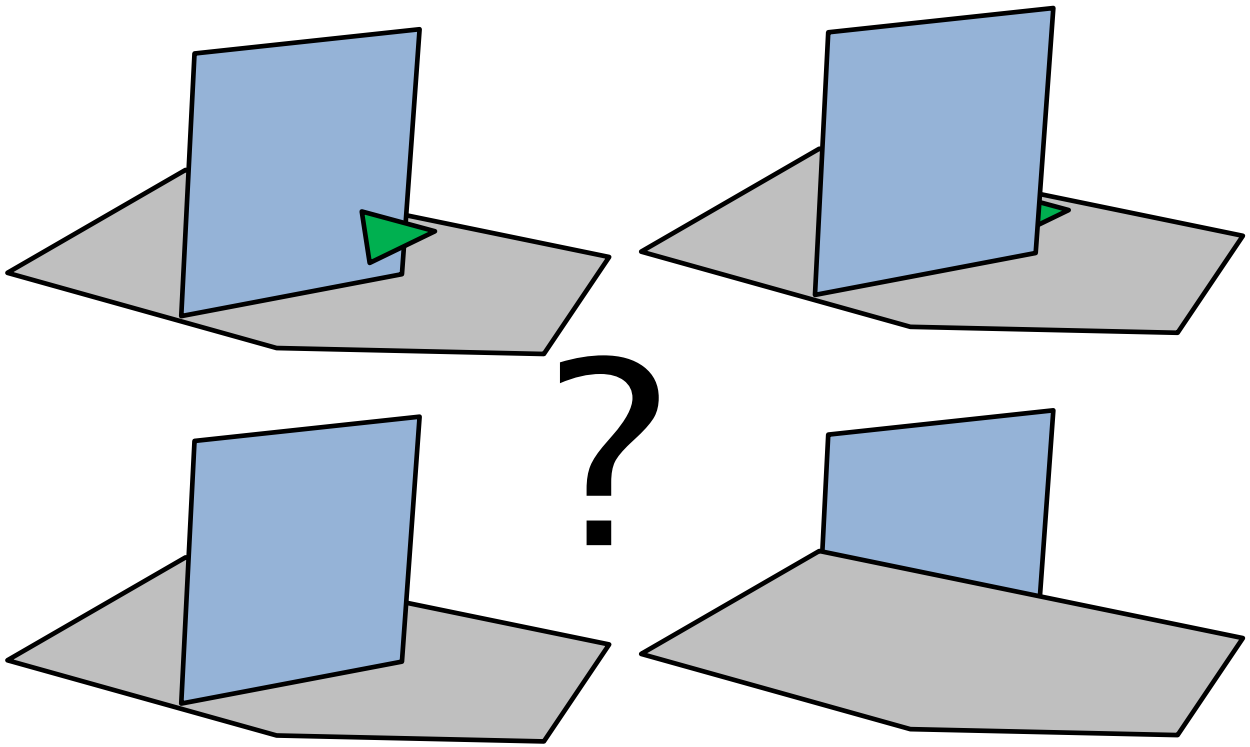


Scan Convert Algorithm

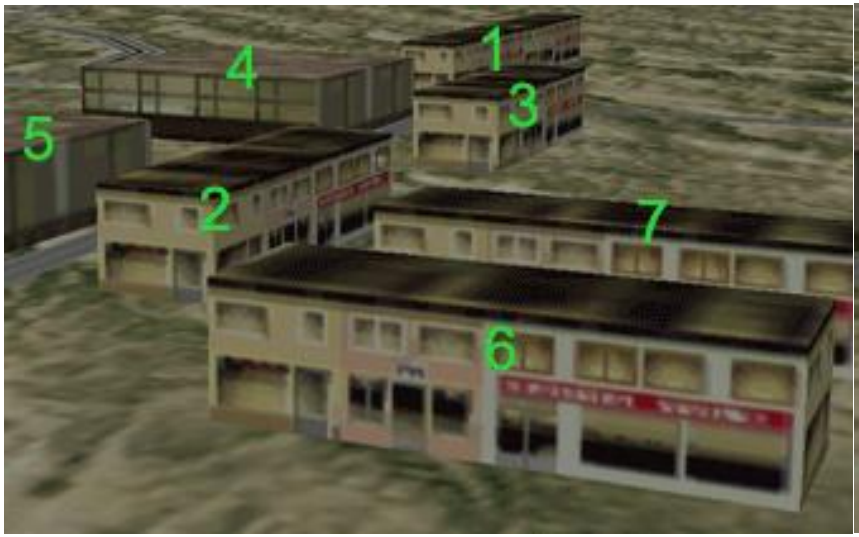
- But how do we know the order?

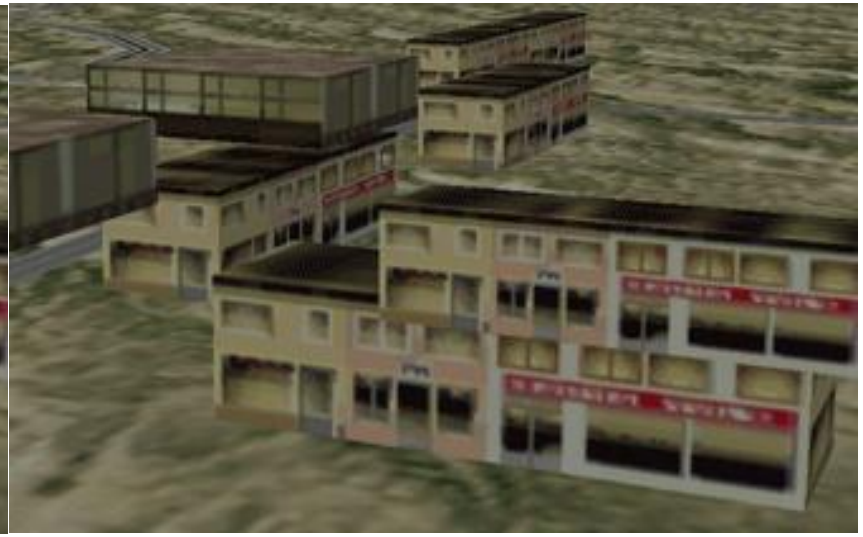


Drawing Order?



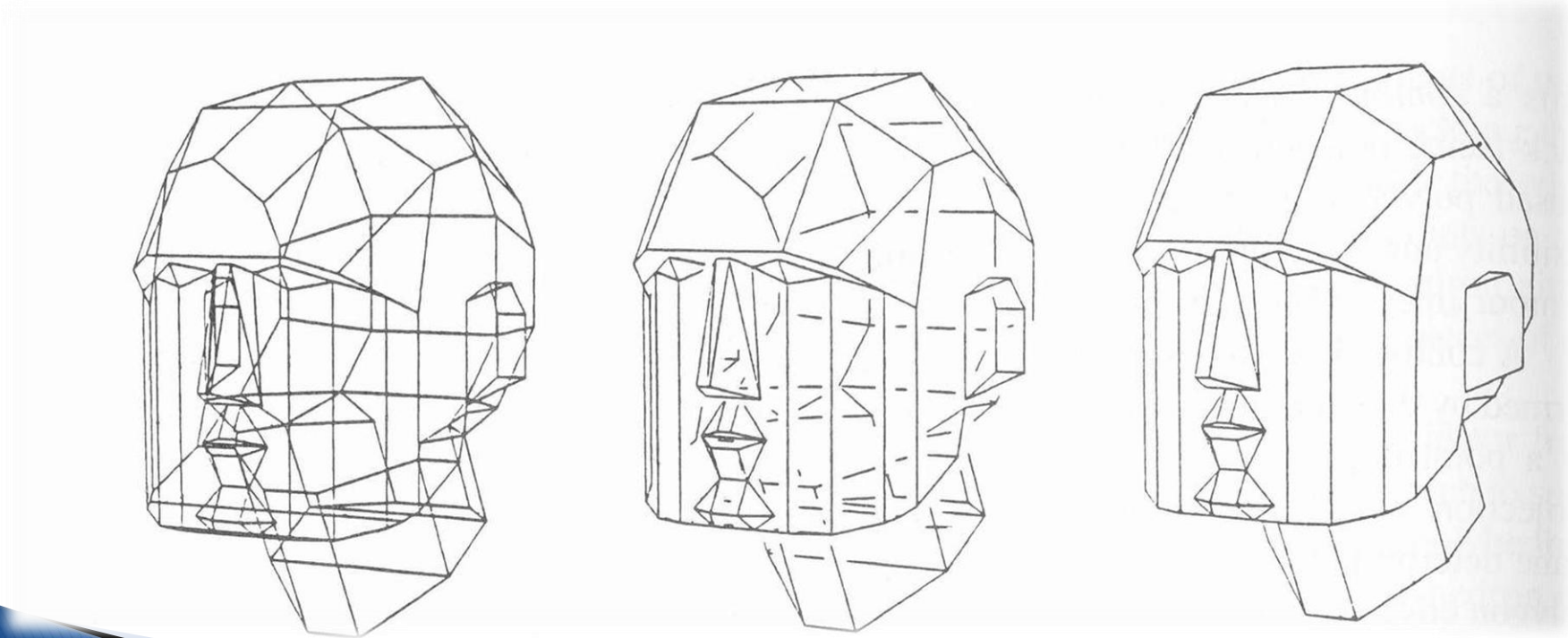
Wrong Drawing Order





Hidden Surface Removal (HSR)

- Managing the polygons so that they are drawn in the right order



Two Main Approaches

I. Control the drawing order of objects
   - ***Object Precision Algorithm***
   - Sort the order of polygons according to "depth"
   - Draw polygons from furthest to nearest
   - View dependent
     - The depth order changes when the viewpoint changes

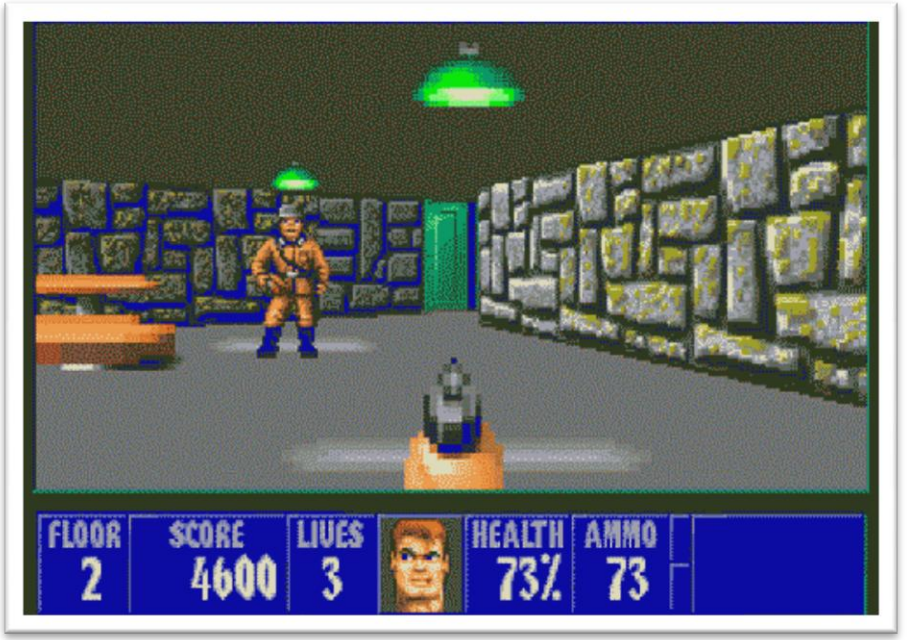
II. Draw objects in a random order
   - ***Image Precision Algorithm***
   - Then for each pixel drawn by a new object
     - If there is already some object drawn, decide if the new object should overwrite it or not

Binary Space Partitioning

Type Ib: Binary Space Partitioning

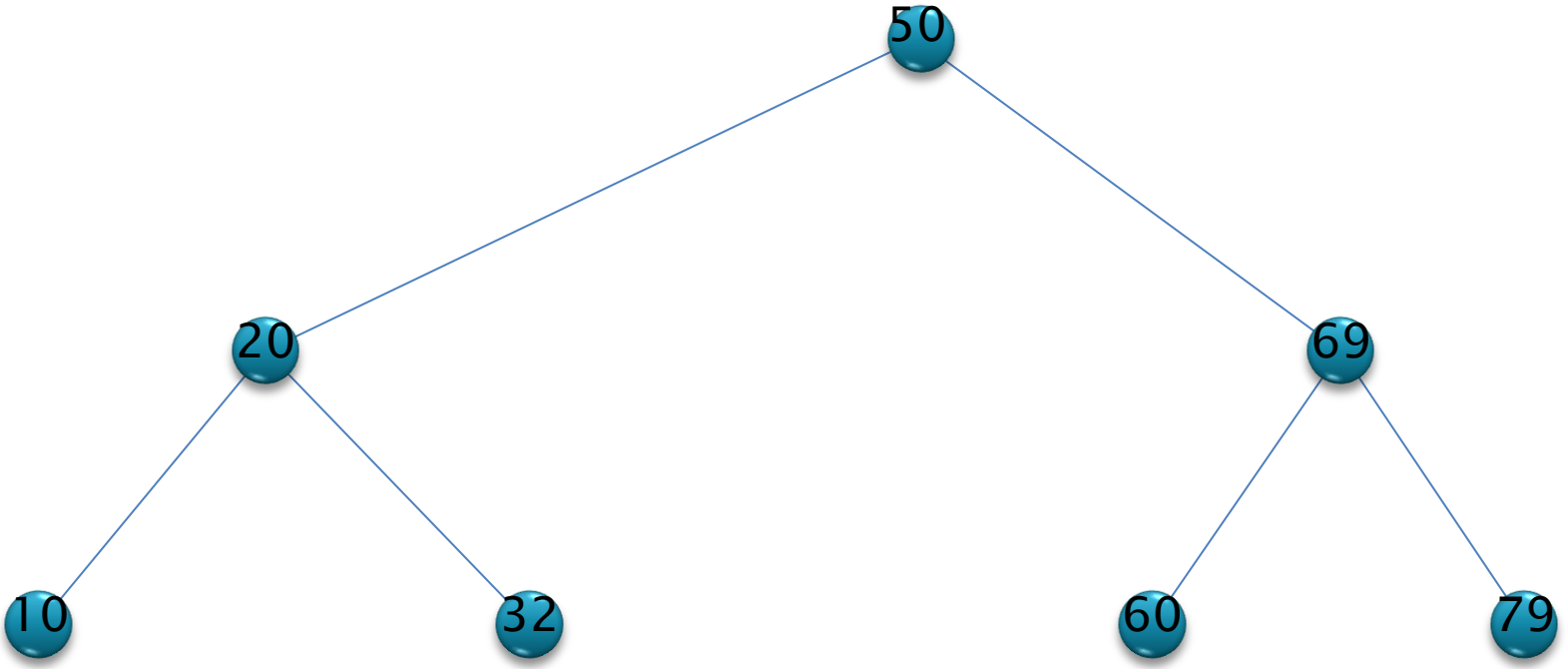
- An industrial standard for real time 3D games such as FPS
  - Doom, Counterstrike, Quake, etc…
- Input:
  - An environment modeled by polygon
- Preprocess the environment and BSP tree
- Output
  - Base on the BSP tree, output a drawing order from the furthest away to the nearest polygon

The First FPS



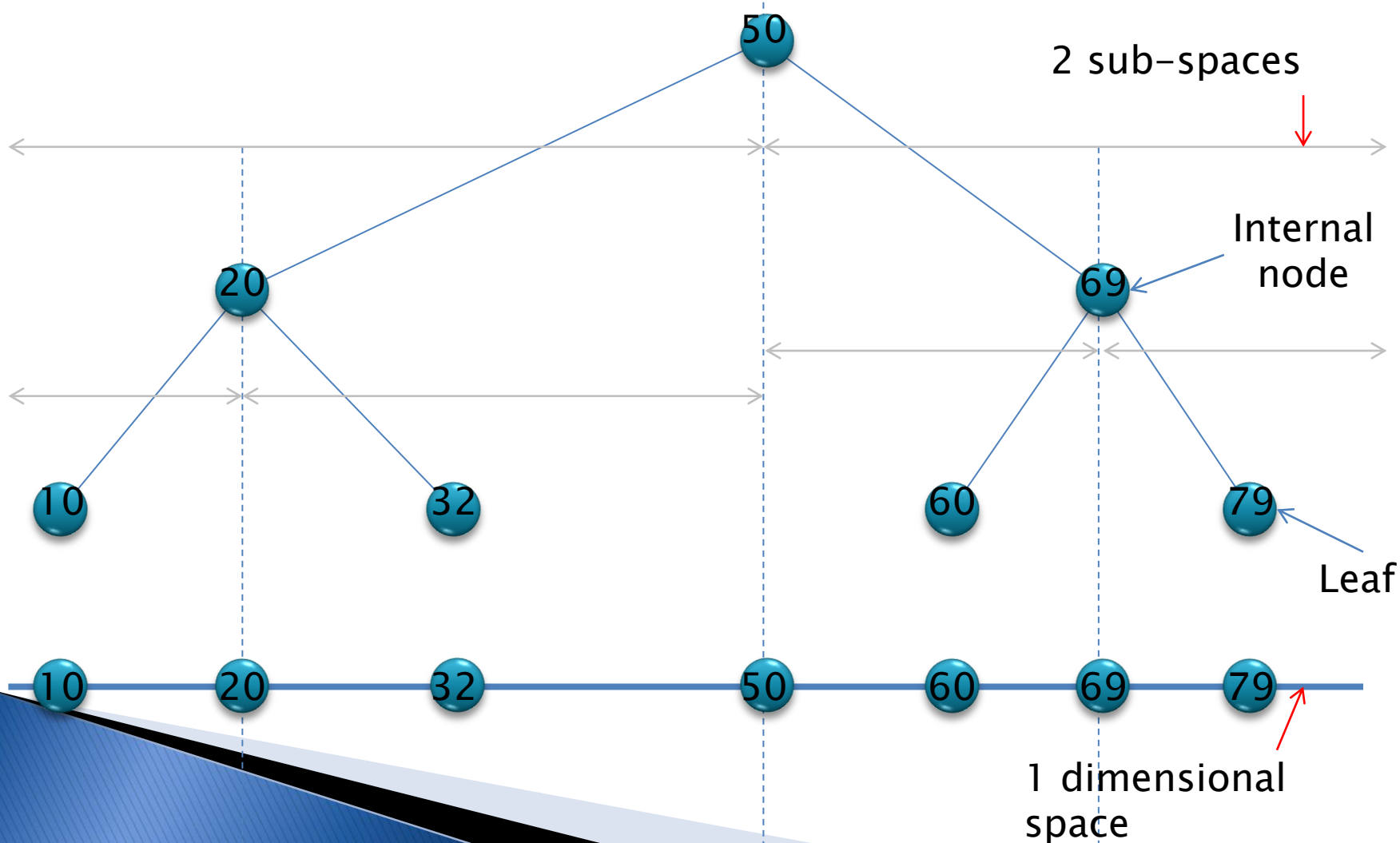
- There are free versions on web and iPad/iPhone
  - The iPad version is totally free on the Japanese iStore

Recap: Binary Search Tree



1 dimensional space

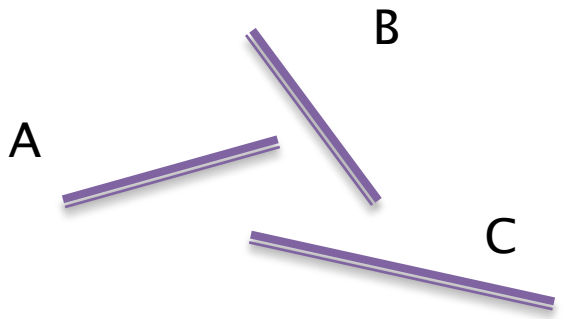
Recap: Binary Search Tree



Type 1b: Binary Space Partitioning

- Idea: Subdivide the entire space by a binary tree
  - Each internal node is a division of a partition/space
  - Each leaf is a part of the space with only **one** polygon

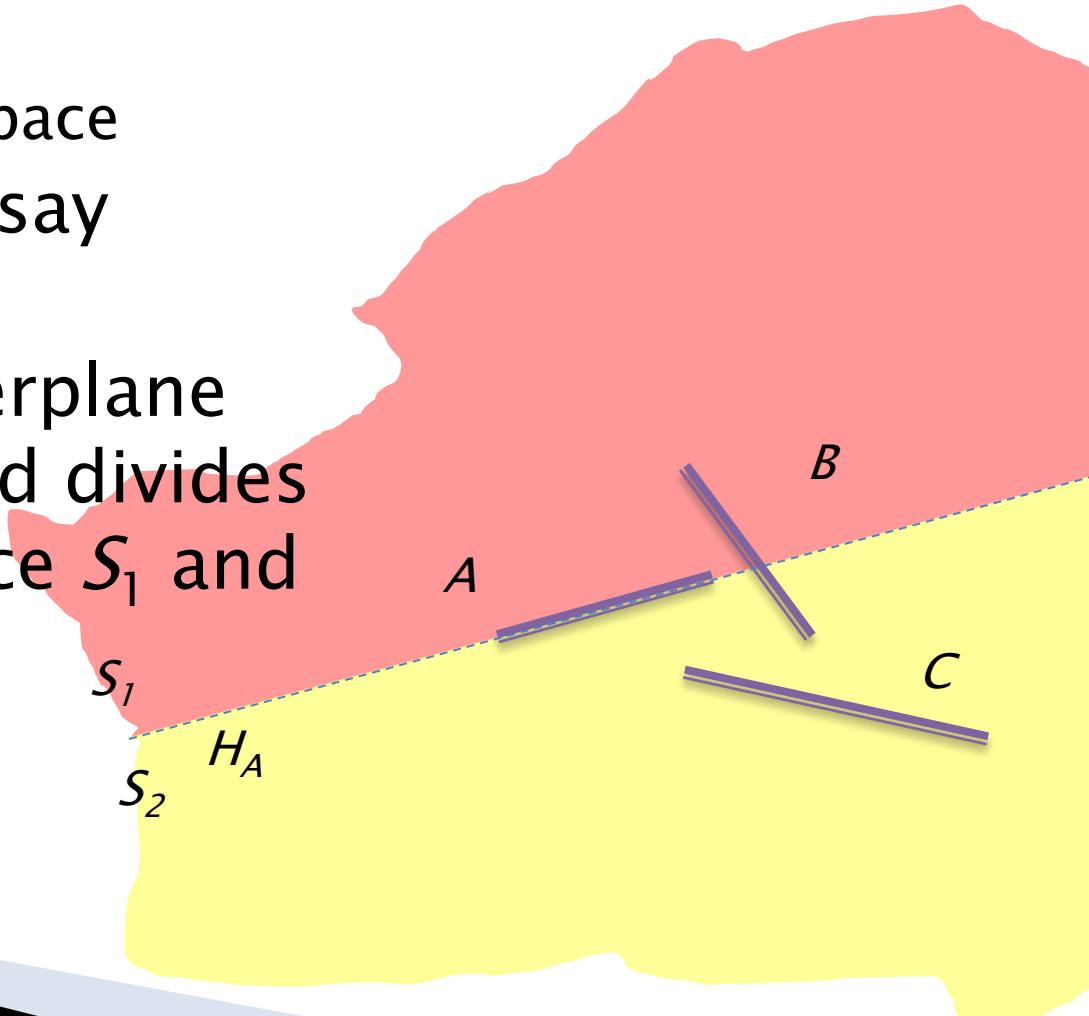
- Divide into two steps
  I. Preparation
  II. Rendering



(Use a bird's eye view for illustration)

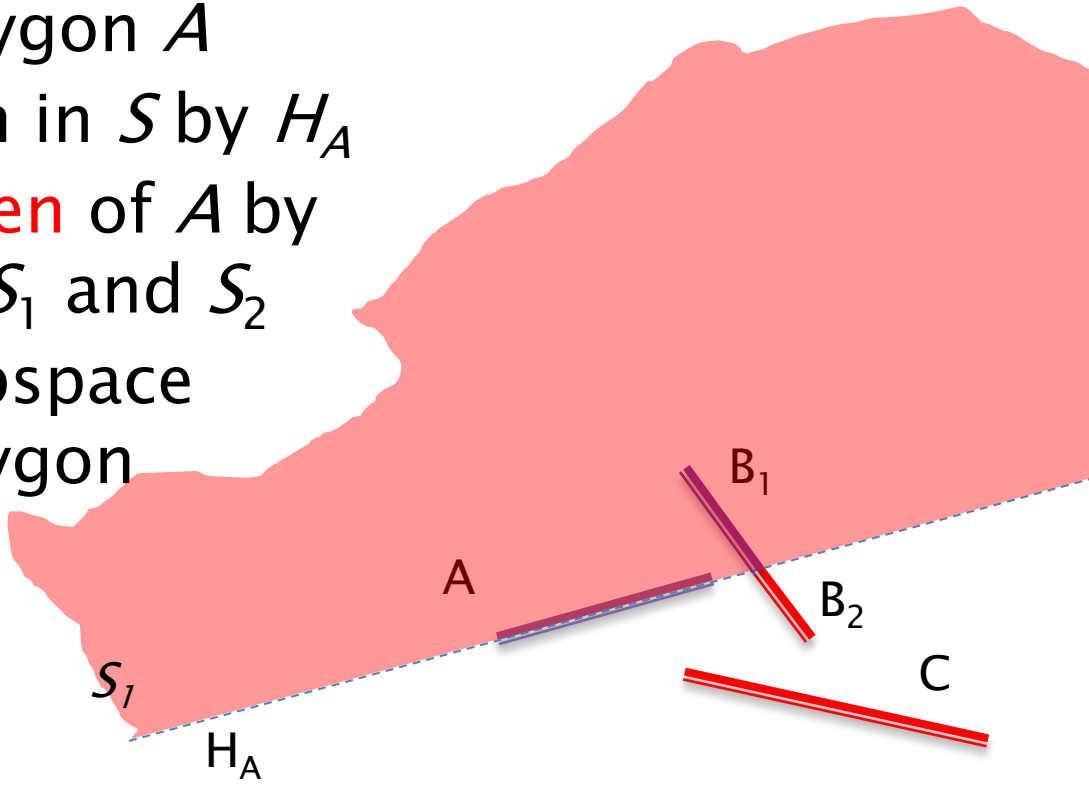
Type 1b: BSP-tree Preparation

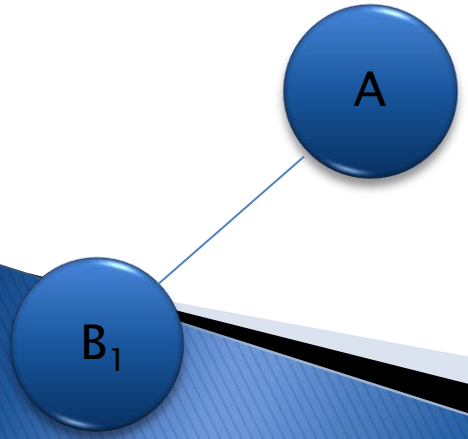
- Initialization
  - Let S = the entire space
- Pick any polygon, say Polygon A
- Let H_A be the hyperplane that contains A and divides S into two subspace S_1 and S_2



Type 1b: BSP-tree

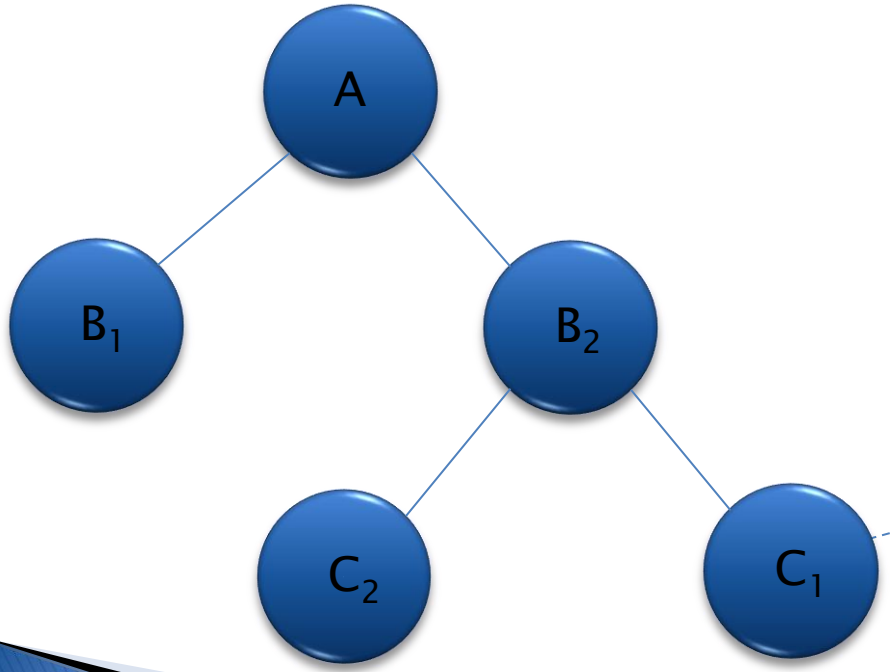
- Build a node by polygon A
- Break every polygon in S by H_A
- Build the two children of A by the two subspaces S_1 and S_2
- Repeat until the subspace contains only 1 polygon

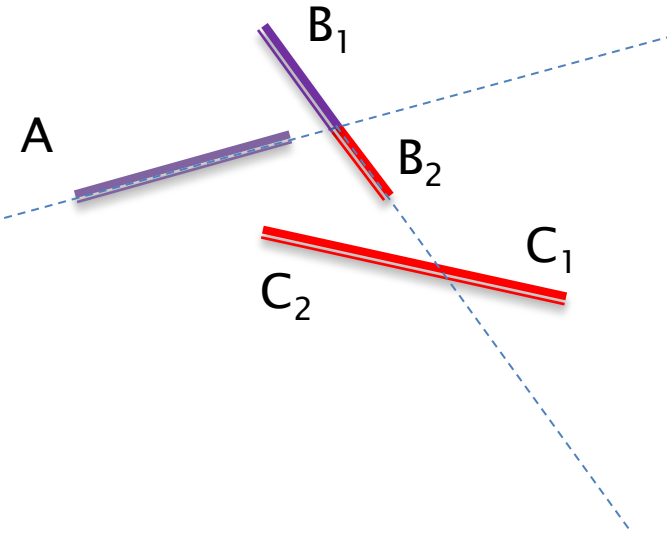




Type 1b: BSP-tree

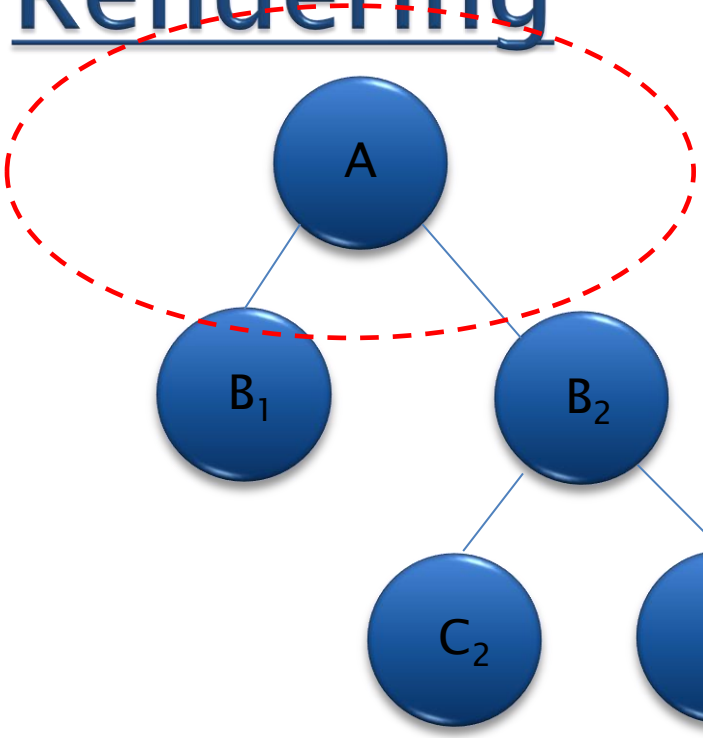
- Prepare the BSP-tree for rendering:

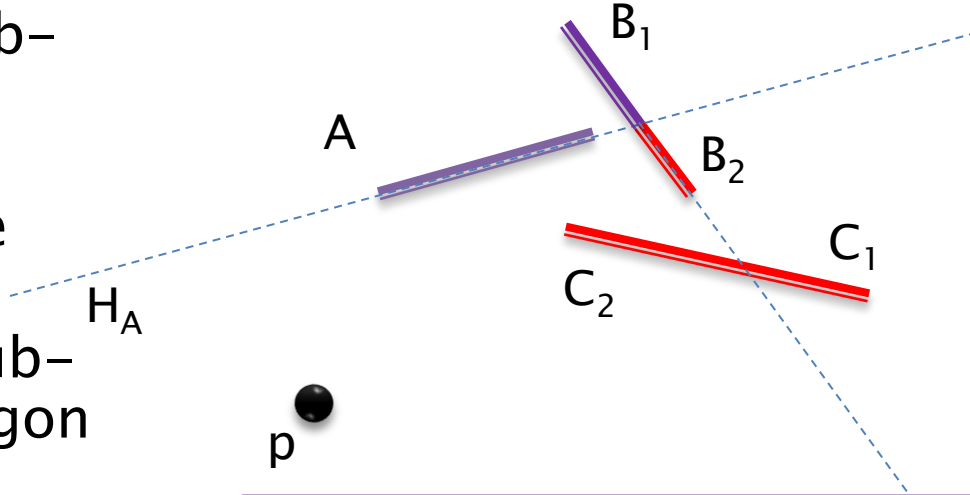




Type 1b: BSP-tree Rendering

- Depending on the viewpoint *p*
- Start from the root
  - For each node there is one polygon and two sub-spaces in the two children
    1. Recursively draw the sub-tree behind the polygon from the view point *p*
    2. Draw the polygon of the node
    3. Recursively, draw the sub-tree in front of the polygon from the view point *p*

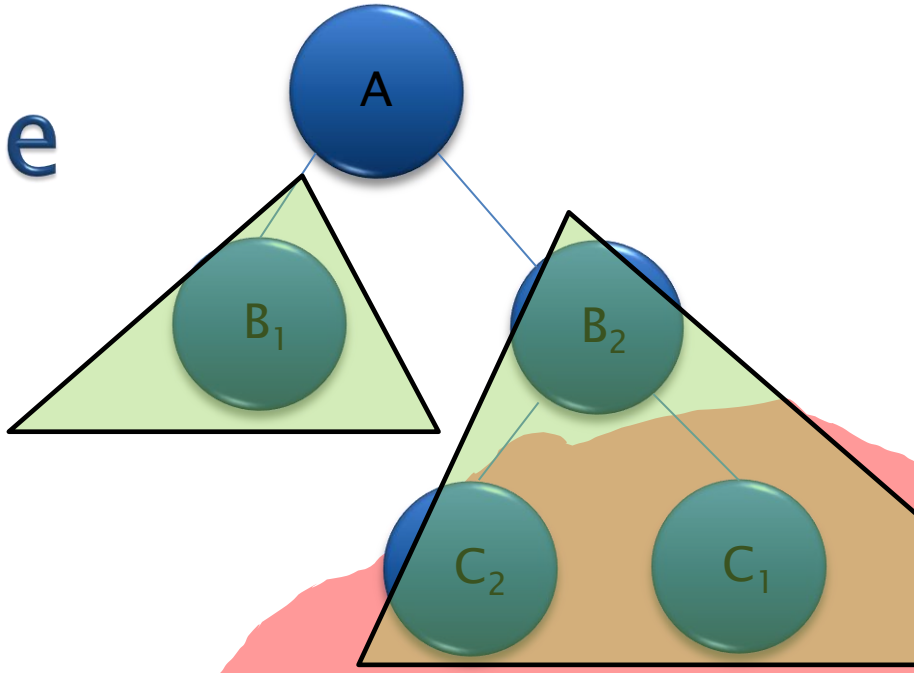


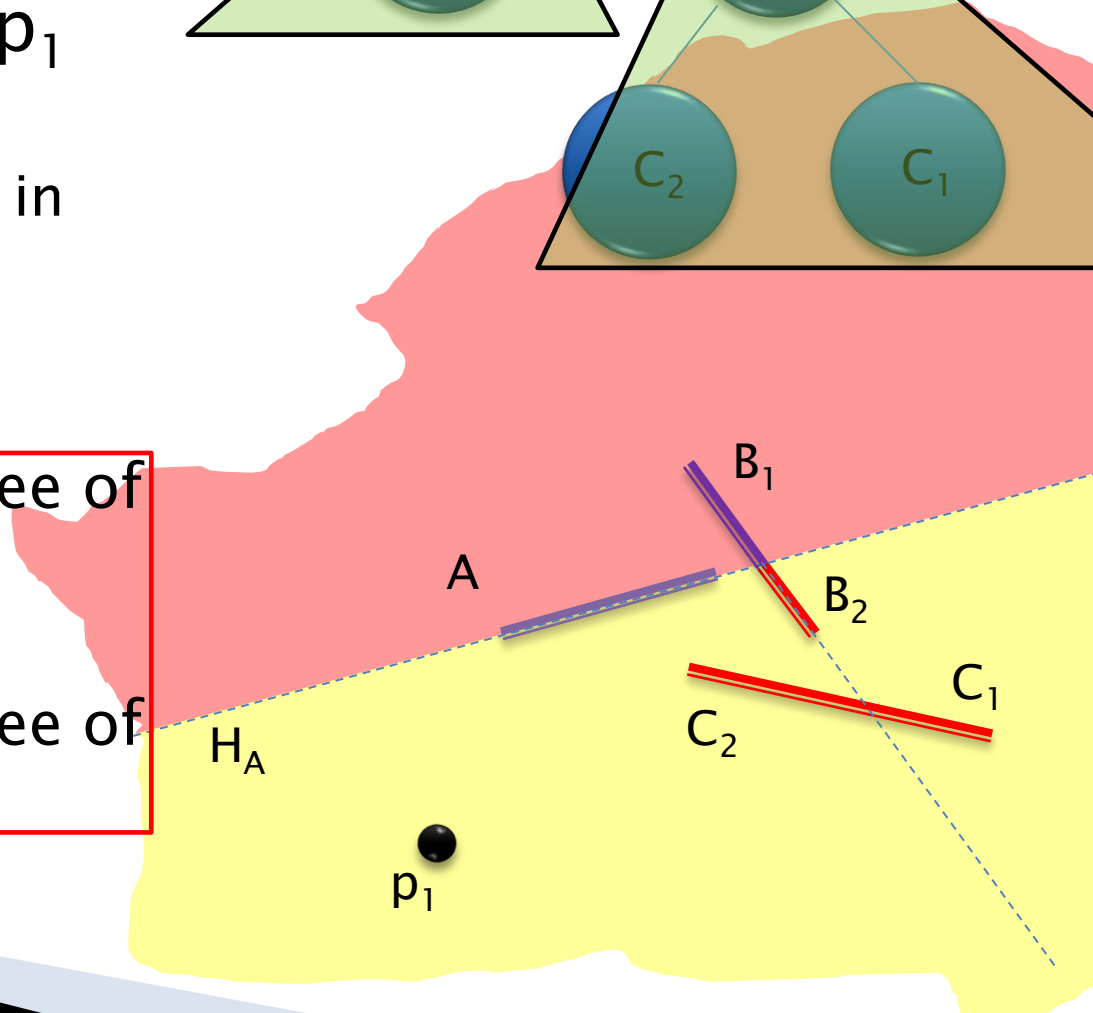


= in-order traversal of a BSP

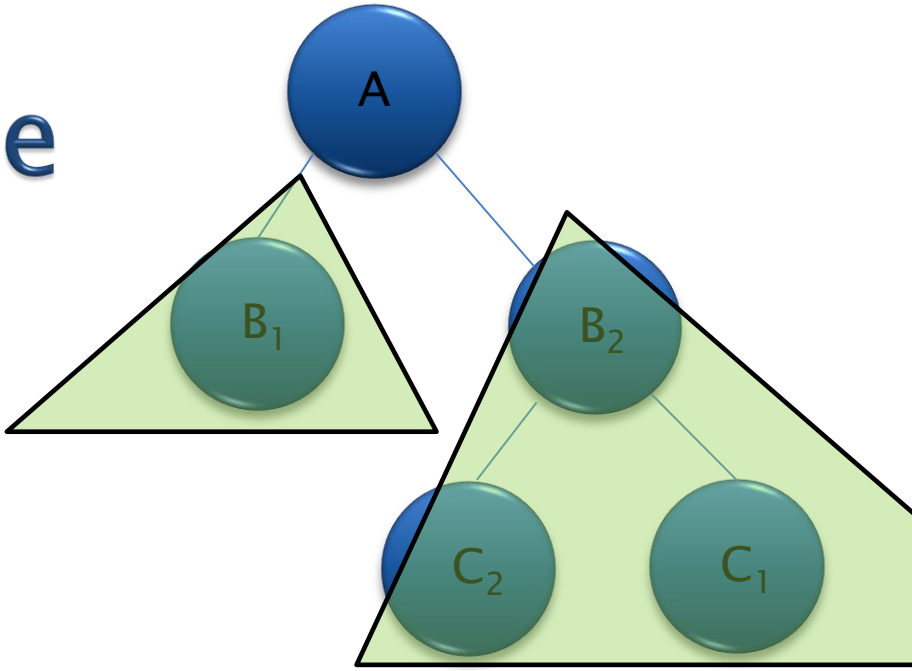
Type 1b: BSP-tree

- For the viewpoint p_1
  - Starting from A
    - The subtree of B_2 is in front of A
    - The subtree of B_1 is behind of A
  - Travserse the subtree of B_1 first
  - Then A
  - Travserse the subtree of B_2

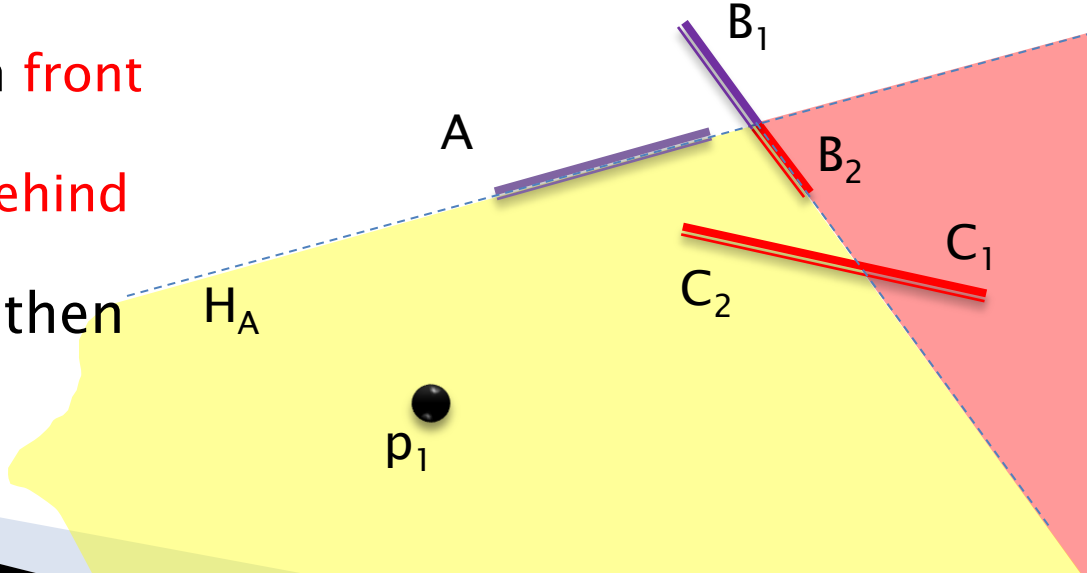




Type 1b: BSP-tree

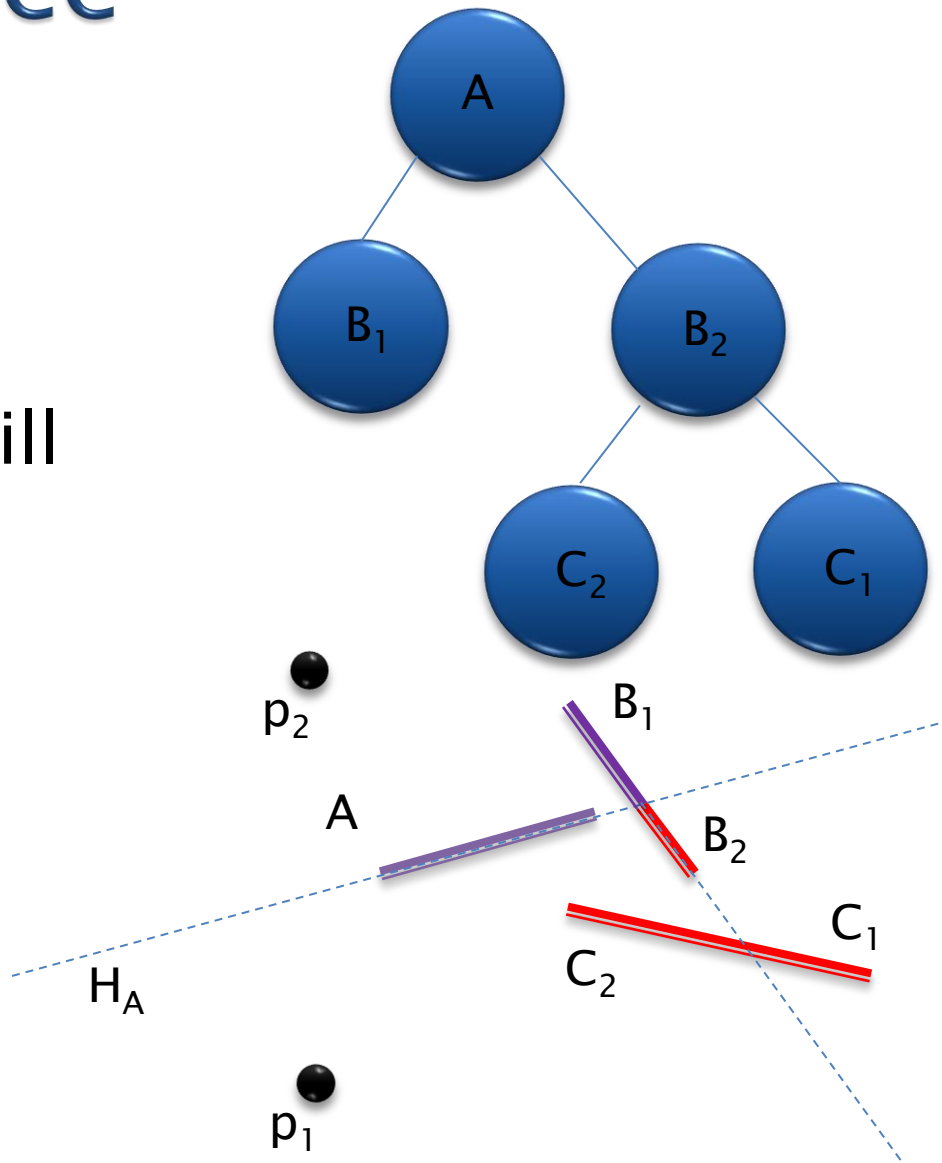


- Travsers the subtree of B_1 first
  - Output B_1
- Then output A
- Then Travserse the subtree of B_2
  - Starting from B_2
    - The subtree of C_2 is in front of B_2
    - The subtree of C_1 is behind of B_2
  - Therefore output C_1 , then B_2, and finally C_2



Type 1b: BSP-tree

- For example, for the following viewpoints, their drawing order will be
  - p_1 : B_1, A, C_1, B_2 , C_2
  - p_2: C_1, B_2, C_2, A, B_1



Detailed exercises in

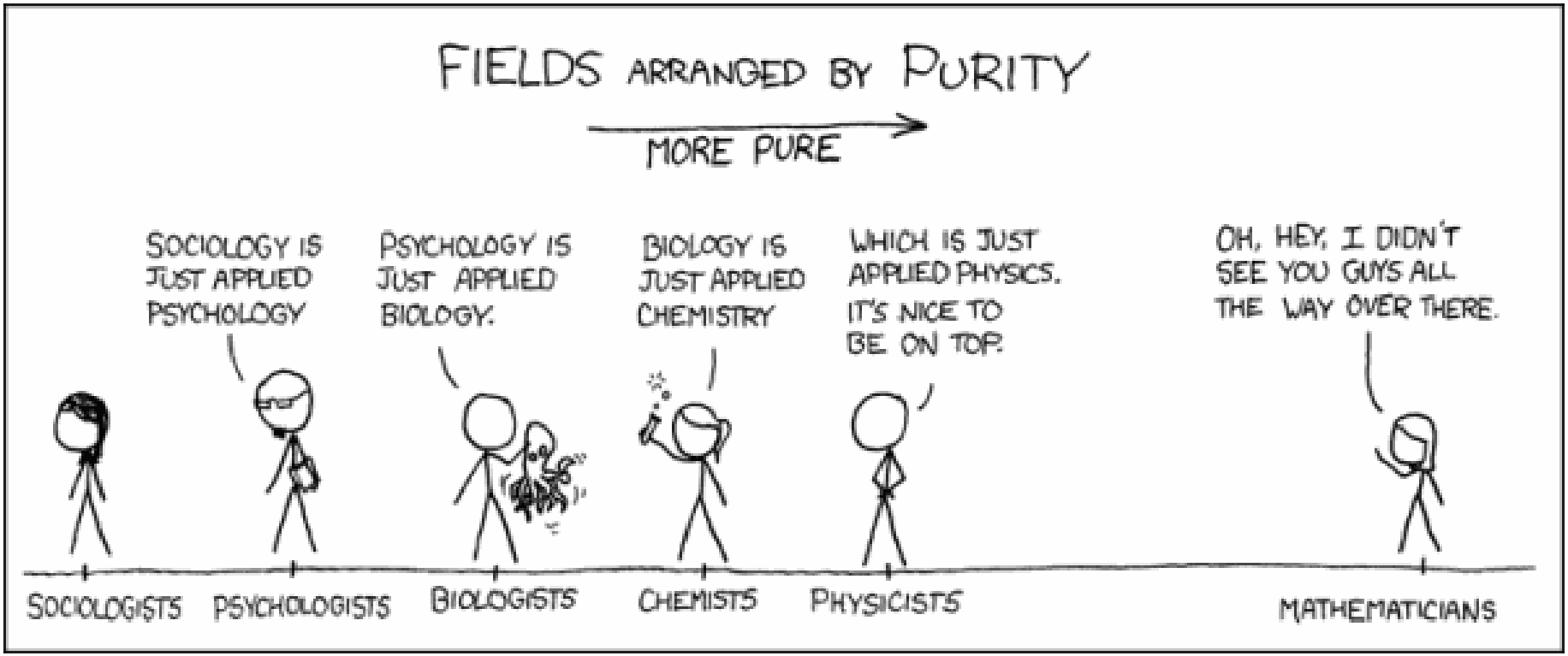
Type 1b: BSP-tree

- **Advantage**:
  - Once the tree is computed, the tree can handle all viewpoints without reconstructing the tree, i.e. efficient
  - Handle transparency
  - Indeed, it's a standard format (**`.BSP`** files) to store the environment for many games
    - E.g. Quake, Half-life, Call of Duty, etc
- **Disadvantages**
  - Cannot handle moving/changing environments
  - Preprocessing time for tree construction is long

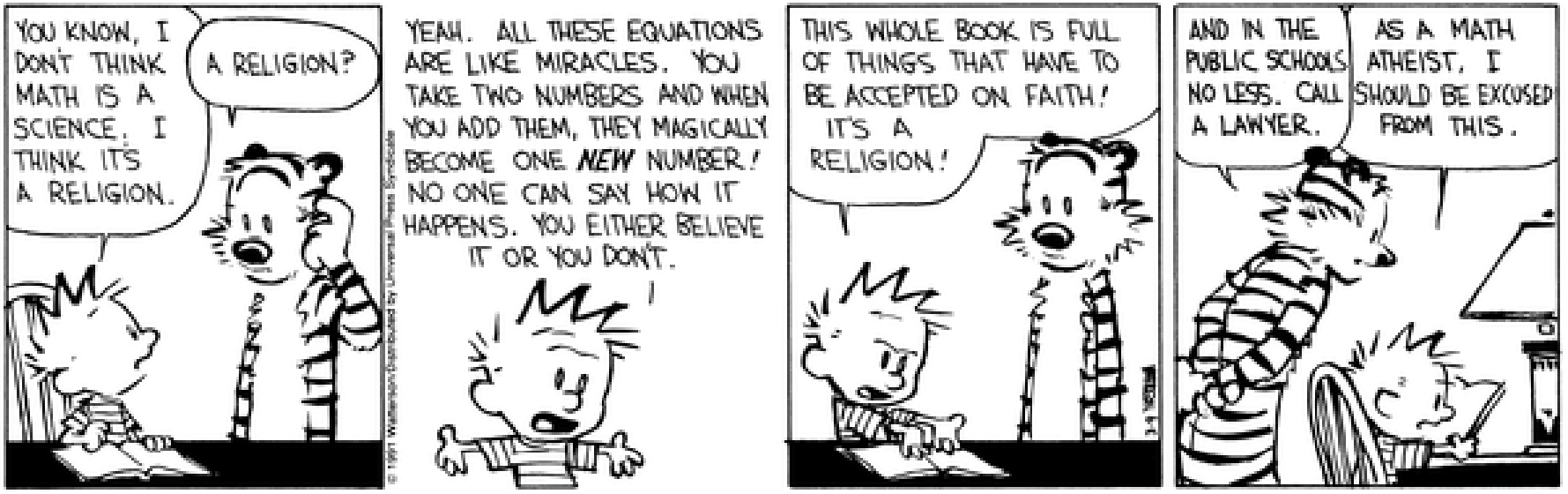
Computational Geometry

A Hybrid of Mathematics and Computer Science

Different Fields

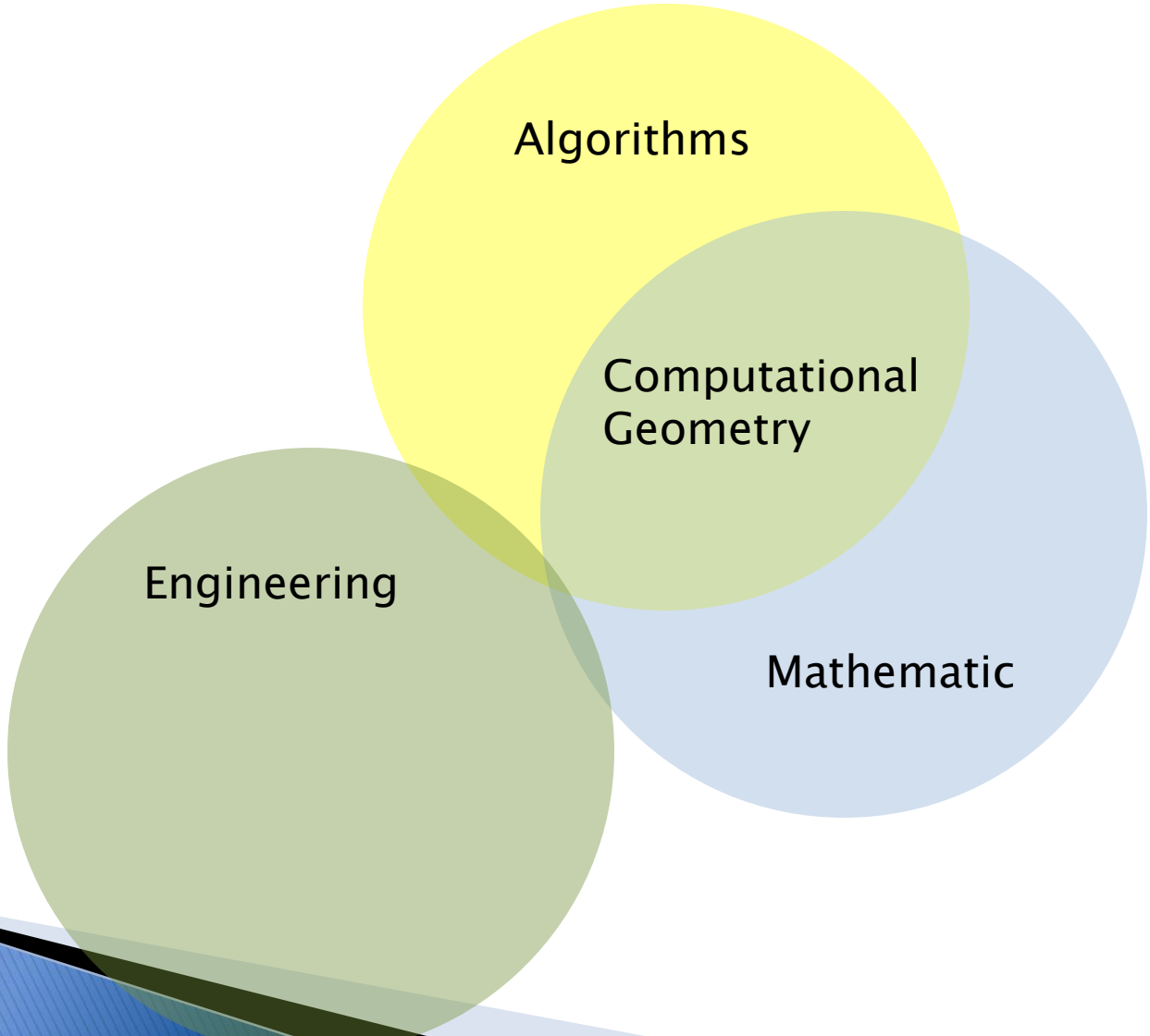


Mathematic as a Religion



- At least mathematic is an art subject

Computational Geometry



Computational Geometry

- Solving computational problems by geometric methods
  - E.g. Simplex algorithm in linear programming

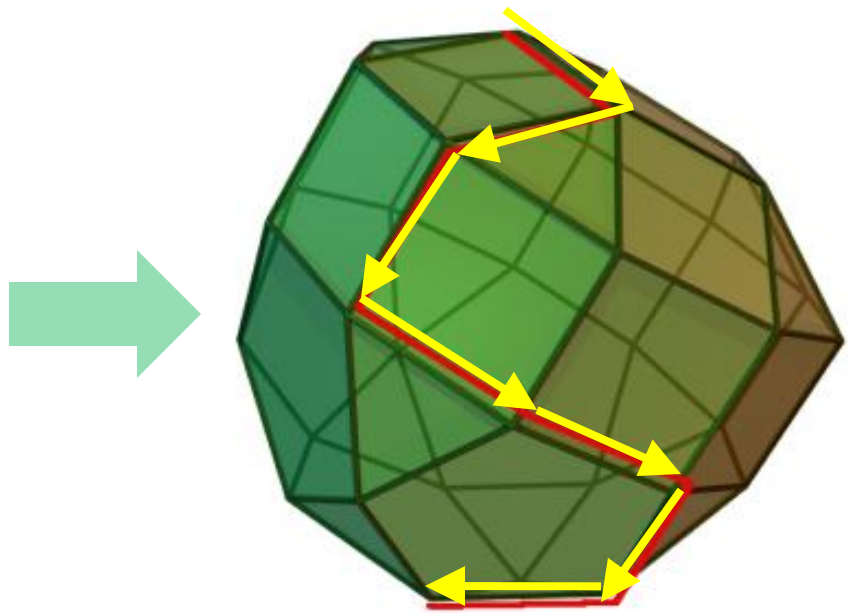
Minimize $z=x_1+x_2+x_3$

Subject to constraints: $x_1-3x_2+4x_3=5$

$x_1-2x_2 \leq 3$

$2x_2+x_3 \geq 4$

$x_1 \geq 0$, $x_2 \geq 0$ and x_3 is unrestricted.



Applications: Solving What?

- Geometric Problems
  - Graphics
    - Radiosity, ray tracing
  - Geometric design
    - Shape manipulation, shape reconstruction
  - GIS/GPS
    - Data structure tuning, polygon overlay and update
  - Molecular science
    - Protein Structures, docking, drug design
  - Medical imaging
    - Reconstruction, feature detection/matching
  - Engineering analysis
    - Mesh generation
  - Robotics
    - Configuration space

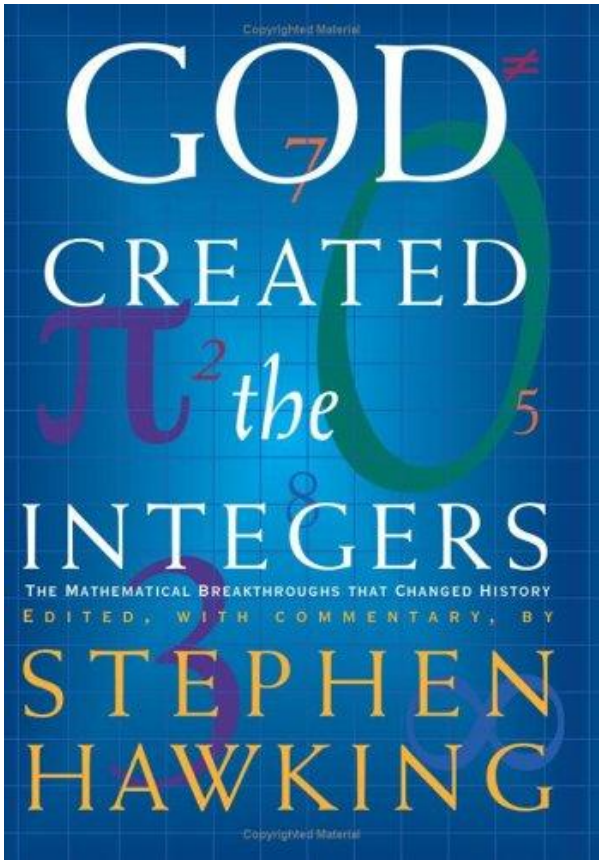
Applications: Solving What?

- General Problems
  - Data mining, searches, optimization
  - Computer vision
    - Model-based recognition
  - Game, genetics, etc.

Different Types of Geometry

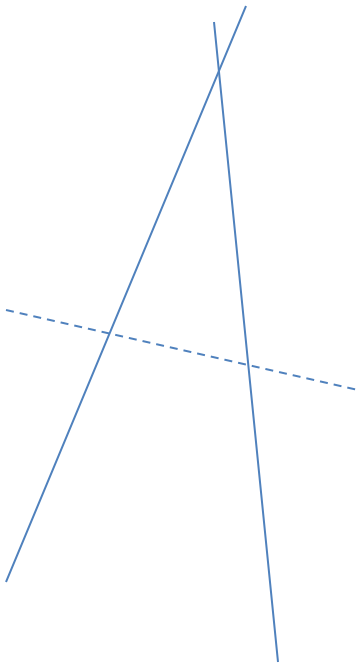
- Euclidean geometry
  - Elements, Euclid, 300 B.C.
- Analytic geometry
  - Further, algebraic geometry, 1637, René Descartes
- Differential geometry
  - Calculus, Gauss late 18^{th} and early 19^{th} century
- Projective geometry
  - J. V. Poncelet (1822)
- Non-Euclidean geometry
  - N. I. Lobachevsky (1826) and János Bolyai (1832).

Nothing Religious



- Euclid
- Archimedes
- Isaac Newton
- Jean Baptiste Joseph Fourier
- Carl Friedrich Gauss
- Georg Friedrich Bernhard Riemann
- Alan Mathison Turing

Euclidean Geometry

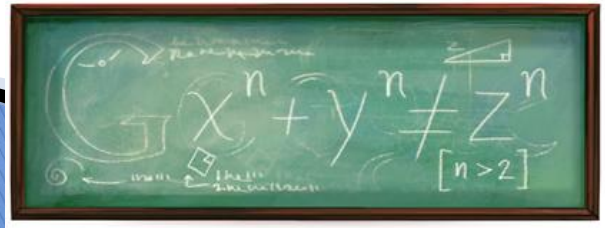


- Based on the FIVE Postulates
  I. A straight line segment can be drawn joining any two points.
  II. Any straight line segment can be extended indefinitely in a straight line.
  III. Given any straight line segment, a circle can be drawn having the segment as radius and one endpoint as center.
  IV. All right angles are congruent.
  V. If two lines are drawn which intersect a third in such a way that the sum of the inner angles on one side is less than two right angles, then the two lines inevitably must intersect each other on that side if extended far enough. This postulate is equivalent to what is known as the parallel postulate.

(V: Not Proven)

Analytic (Algebraic) geometry

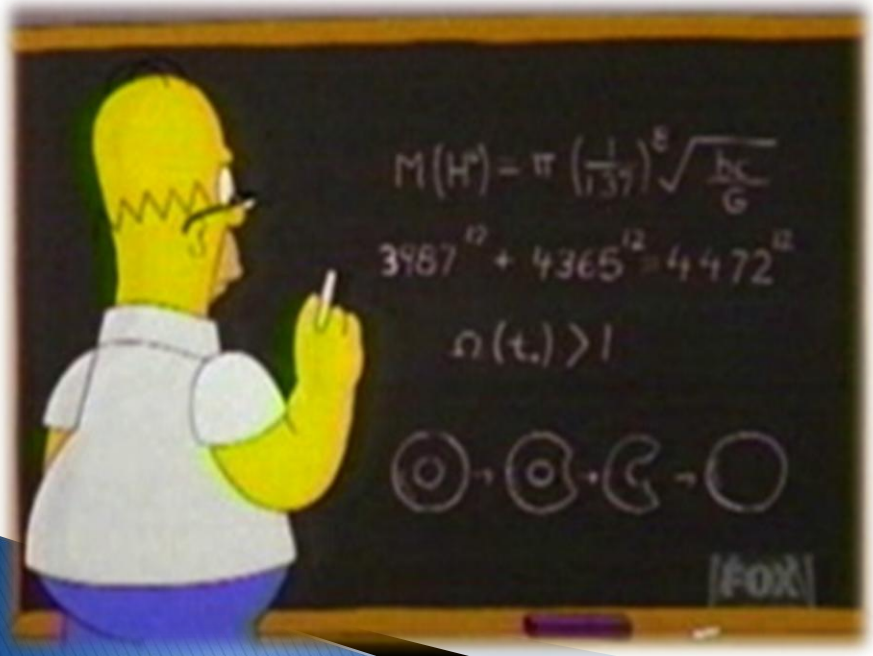
- Algebraic representation
  - Coordinate system
  - Location as an n-tuple
  - Thus, $x^2 + y^2 = 1$
  - Help in proving
    - Fermat's Last Theorem



Fermat's Last Theorem

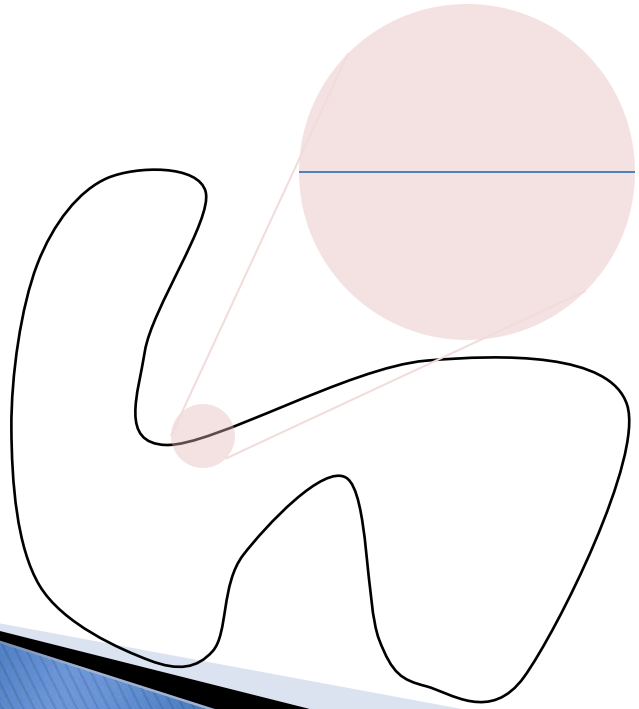
- For n > 2, there is **NO** integer solution for the equation

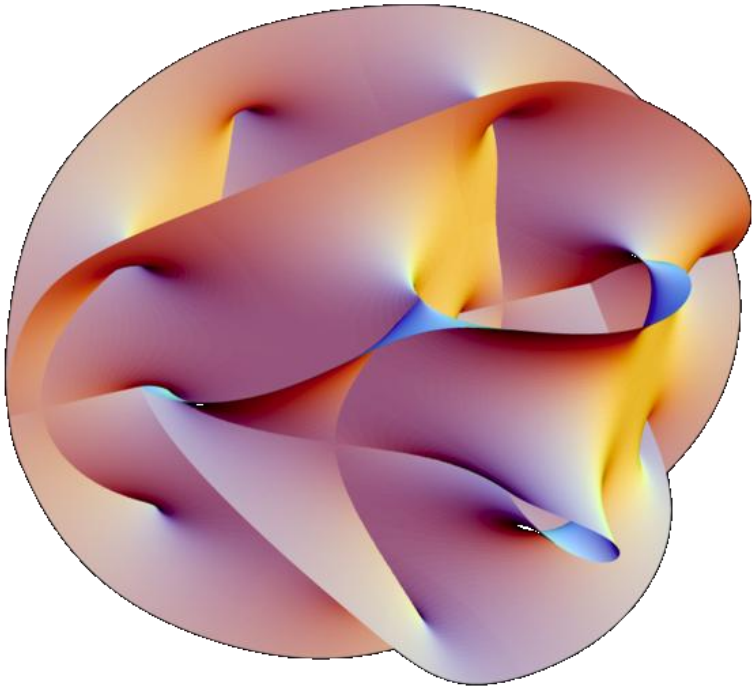
$$x^n + y^n = z^n$$



Differential Geometry

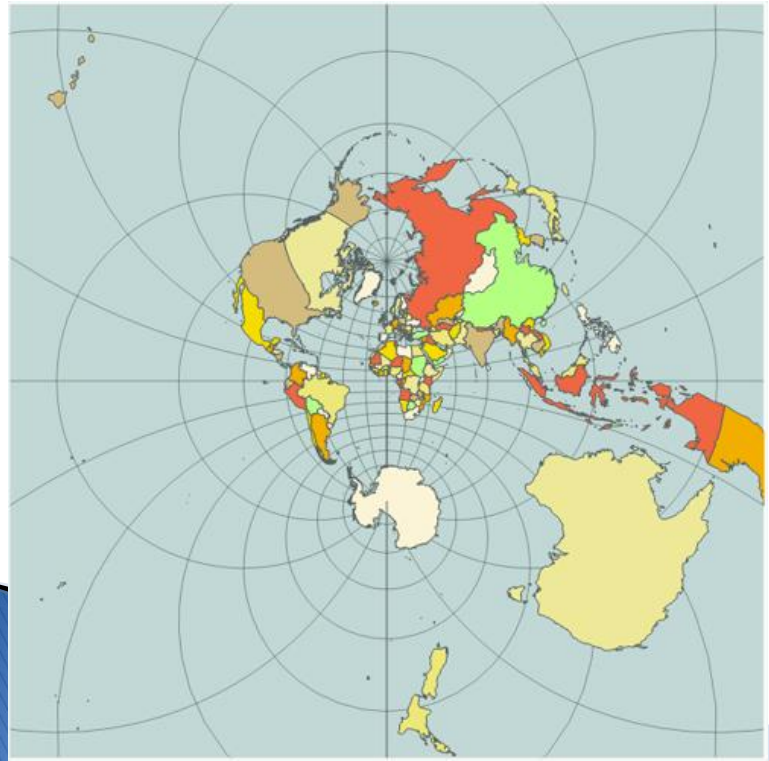
- Manifolds
  - A manifold is a (topological) space that is locally Euclidean
- Curvatures, tensors, etc…

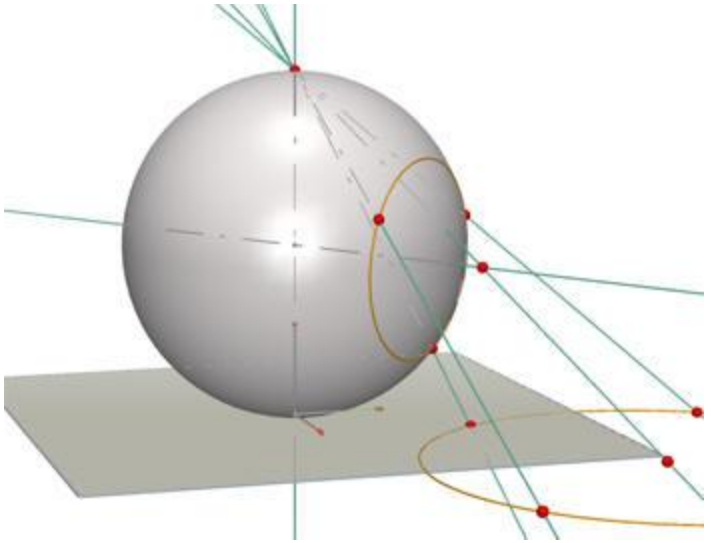




Projective Geometry

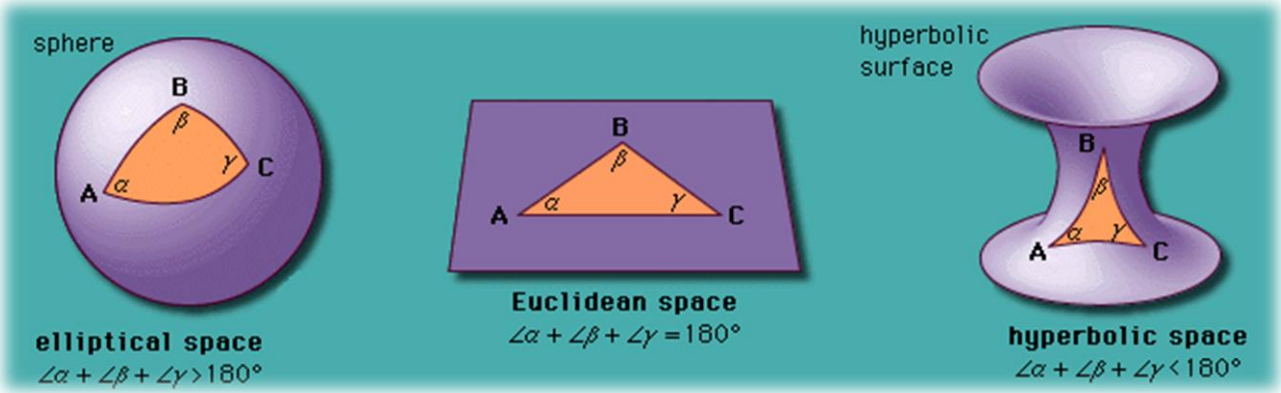
- The properties and invariants of geometric figures under projection
  - For example, stereographic projection





Non-Euclidean Geometry

- Euclidean minus the fifth postulates
  - Hyperbolic Geometry
  - Elliptic Geometry
  - Etc…
- 2D examples:



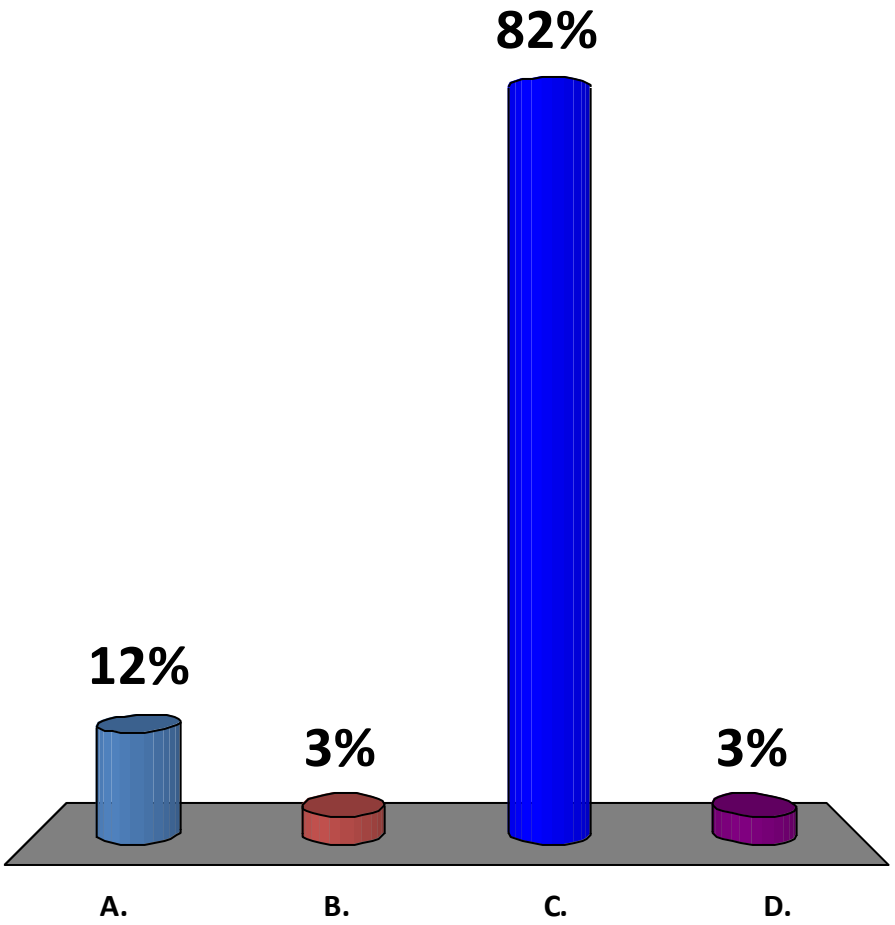
- Is our 3D space is Euclidean?

Mathematical Thinking

- Question: Is there a place on earth that you can
  - Travel south for 10 km
  - Travel west for 10 km
  - Travel north for 10 km
  - But you return to the same place where you start with?
  - return to the same place where you start with?

The place is…..

A. My home
B. Any place on the Equator like Singapore
C. North Pole
D. There is no such place, silly…



Mathematical Thinking

- Question: Is there a place on earth that you can
  - Travel south for 10 km
  - Travel west for 10 km
  - Travel north for 10 km
  - But you return to the same place where you start with?
  - return to the same place where you start with?
- Answer: North Pole
- Step 1: Does the solution exist?
  - Or, if we do not know any solution, can we prove that there is NO solution?

Maths vs CS vs Engineering

- Three good friends, an engineer, a mathematician and a computer scientist, are driving on a highway that is in the middle of no where. Suddenly one of the tires went flat and they have no spare tire.

Maths vs CS vs Engineering

- Engineer
  - “Let’s use bubble gum to patch the tire and use the strew to inflate it again”
- Mathematician
  - “I can prove that there is a good tire exists in somewhere this continent”
- Computer Scientist
  - “Let’s remove the tire, put it back, and see if it can fix itself again”

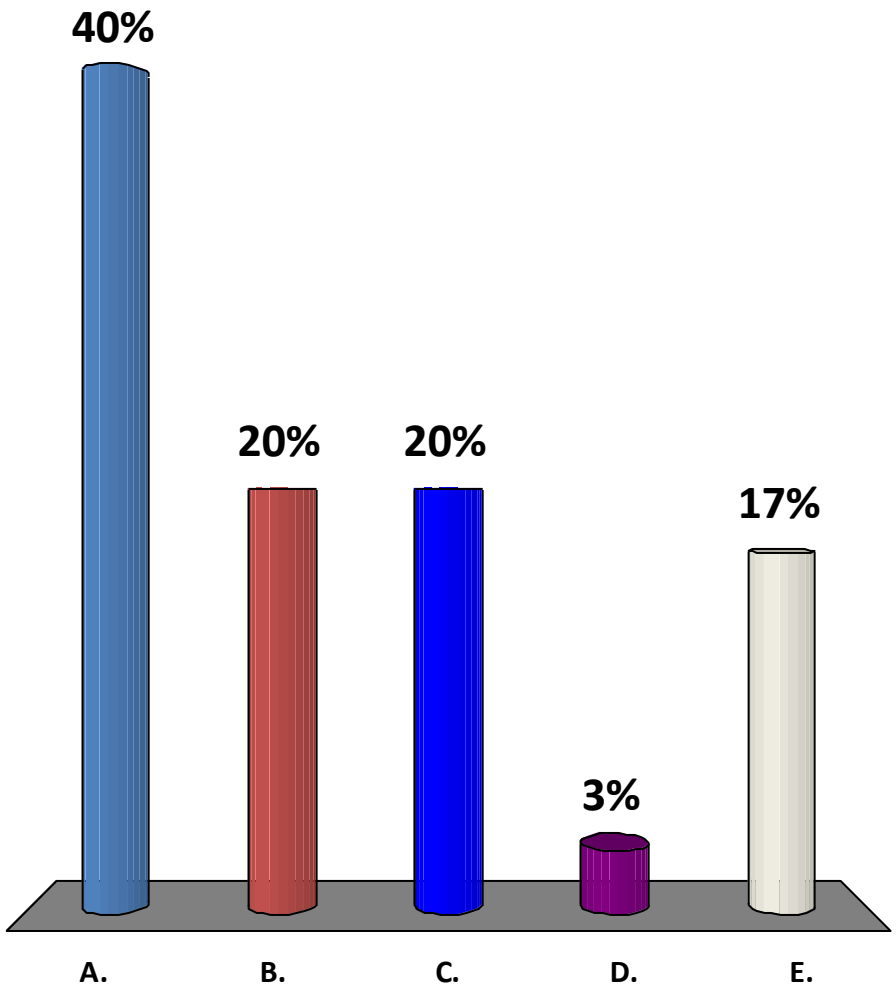
Step 2

- Question: Is there a place on earth that you can
  - Travel south for 10 km
  - Travel west for 10 km
  - Travel north for 10 km
  - But you return to the same place where you start with?

- Is there **another** place other than North Pole?

Is there another place?

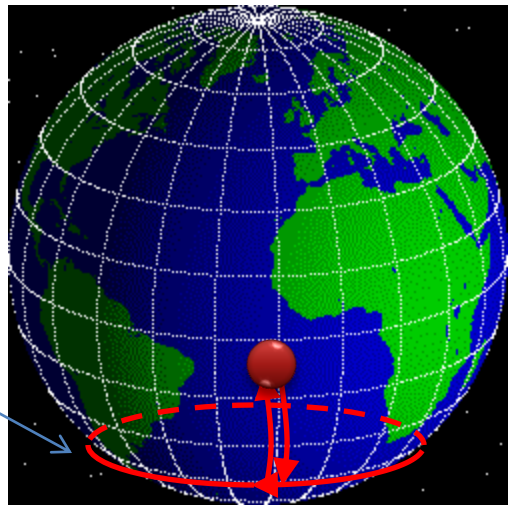
A. Another North Pole in another planet
B. Somewhere near the South Pole
C. South Pole
D. Singapore
E. No such place, NP is unique



Mathematical Thinking

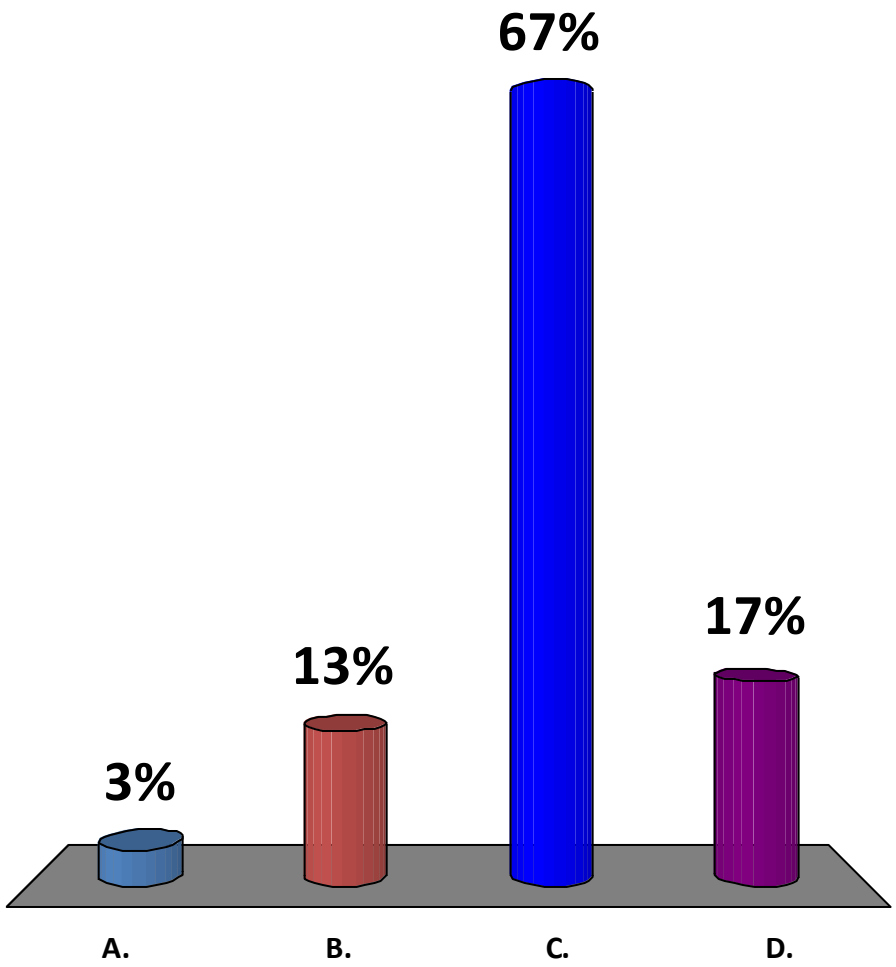
- Step Two: Is there more than one solution?
- Step Three: Altogether how many solutions are there?

A circle with 1 km perimeter



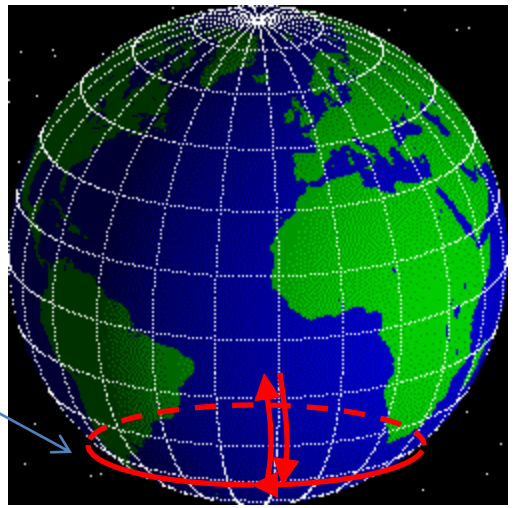
How many places are there?

A. Two
B. $(\text{Log } d)+1$ where d is the distance between the Equator and the South Pole
C. Infinite ✓
D. South Pole is too cold for me..



Mathematical Thinking

- Step Two: Is there more than one solution?
- Step Three: Altogether how many solutions are there?
  - Reaching the circle with
    - 1 km perimeter
    - 0.5 km perimeter
    - 1/3 km perimeter
    - …
    - 1/n km perimeter for n = 1 to ∞



A circle with 1 km perimeter

Mathematical Thinking

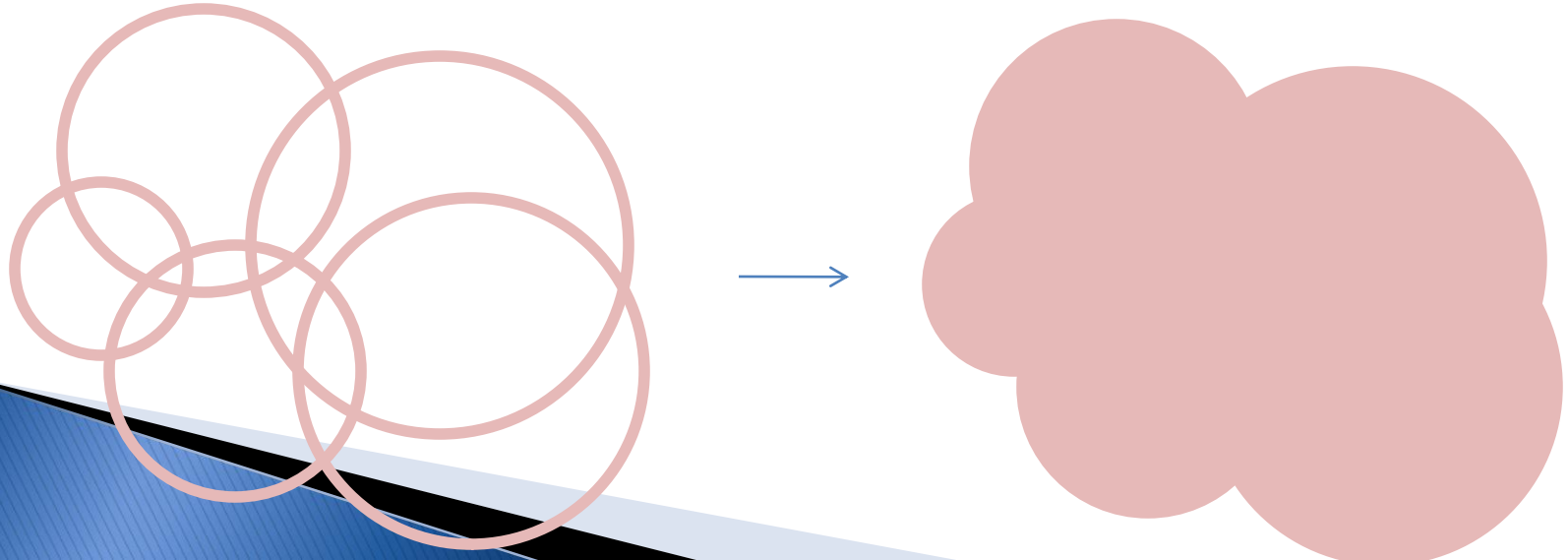
- Step 1: Does a solution exist?
- Step 2: Is there more than one solution?
- Step 3: How many solutions in total?
- However ……, all these can be achieved **WITHOUT** knowing any actual solution

Computational Thinking

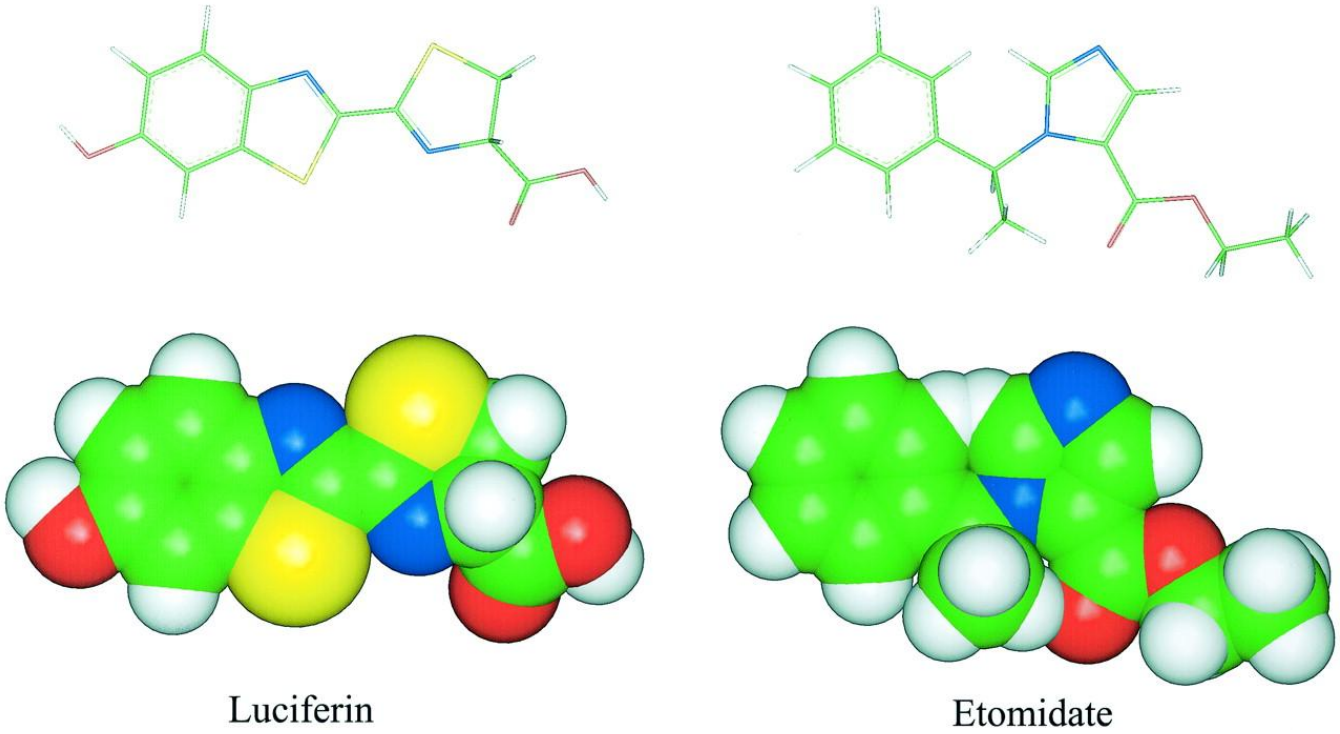
- Is it computable?
- How fast can we compute?
- Example:
  - Given a set of disks

$$B = \{b_i = \{x \mid |x - z_i| \le r_i, \text{for } i = 1..n\}$$

  - Computing the area of union of disks



Wait, why do we have to do this



- The molecular volume of luciferase is 271 $Å^3$ and that of etomidate is 286 $Å^3$.
- Franks and Lieb raises the question of how closely luciferase resembles the actual site of anesthetic action.

Computational Thinking

- Mathematically
  - Inclusion-exclusion formula

$$\text{Area}(\cup B) = \sum_{X \subseteq B} (-1)^{\text{card}(X)-1} \, \text{Area}(\cap X)$$

  - When $\text{card}(X) =$
    - 1: $\text{Area}(b_1) + \text{Area}(b_2) + \text{Area}(b_3) + ... \text{Area}(b_n)$
    - 2: $-\left(\text{Area}(b_1 \cap b_2) + \text{Area}(b_1 \cap b_3) + ... \text{Area}(b_{n-1} \cap b_n)\right)$
    - 3: $\text{Area}(b_1 \cap b_2 \cap b_3) + \text{Area}(b_1 \cap b_2 \cap b_4) + ... \text{Area}(b_{n-2} \cap b_{n-1} \cap b_n)$
    - 4: …
    - ……
    - n: $\text{Area}(b_1 \cap b_2 \cap b_3 .... \cap b_n)$

Computational Thinking

- Mathematically
  - Inclusion-exclusion formula

$$\text{Area}(\bigcup B) = \sum_{X \subseteq B} (-1)^{\text{card}(X)-1} \, \text{Area}(\bigcap X)$$

  - Complexity $= \text{O}({}_nC_1) + \text{O}({}_nC_2) + \text{O}({}_nC_3) + \ldots + \text{O}({}_nC_n)$

$$= \text{O}(2^n)$$

- Computationally, it's a sin!
- Edelsbrunner, $\text{O}(N)$, N is the number of simplices
  - In 2D, N is $\text{O}(n)$
  - In 3D, N is $\text{O}(n^2)$

Between the Two Thinkings

- What is the difference between an engineer, a physicist, and a mathematician?
  - An engineer believes equations approximate the world.
  - A physicist believes the world approximates equations.
  - A mathematician….
    - sees no connection between the two.

Computational Geoemtry

- k-d Tree
- Line Intersection
- Triangulating a polygon

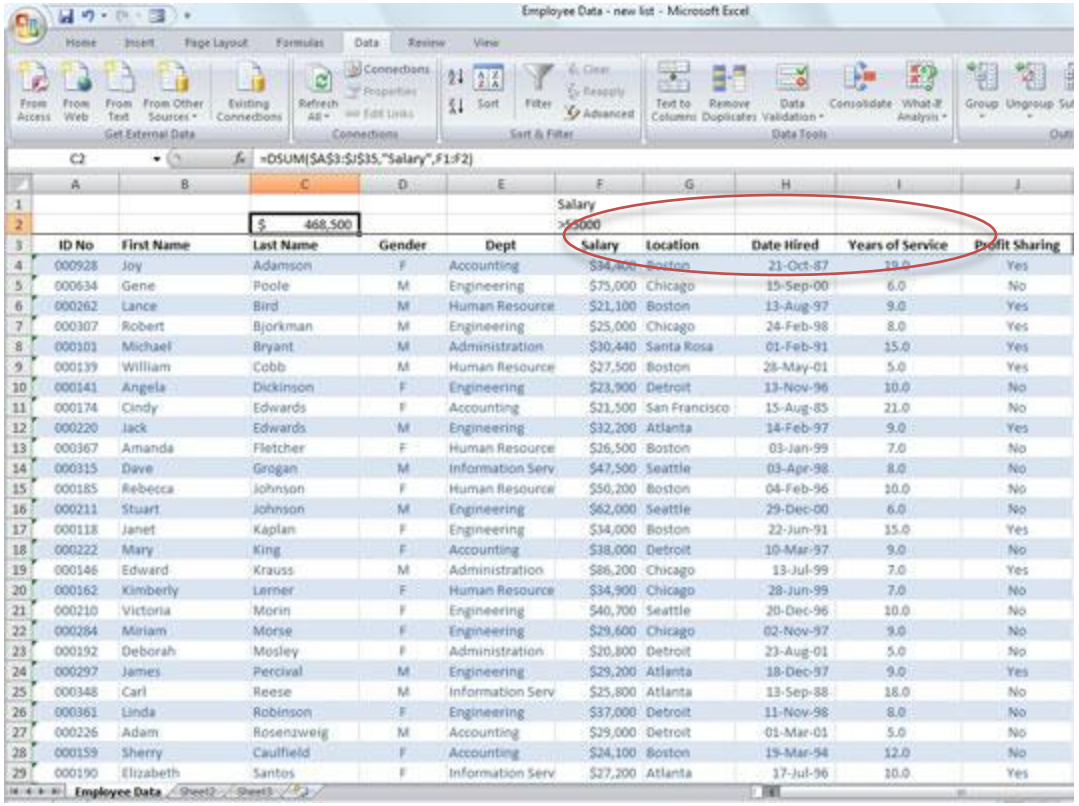
Orthogonal Range Query

Have you ever taste the real power of "recursive thinking"?

Motivation

Database

- In a database, fields are textual or numerical, for example
  - Name
  - ID code
  - Date of birth
  - Height
  - Salary
  - Etc.



ID No	First Name	Last Name	Gender	Dept	Salary	Location	Date Hired	Years of Service	Profit Sharing
000928	Joy	Adamson	F	Accounting	$34,800	Boston	21-Oct-87	19.0	Yes
000634	Gene	Poole	M	Engineering	$75,000	Chicago	15-Sep-00	6.0	No
000262	Lance	Bird	M	Human Resource	$21,100	Boston	13-Aug-97	9.0	Yes
000307	Robert	Bjorkman	M	Engineering	$25,000	Chicago	24-Feb-98	8.0	Yes
000101	Michael	Bryant	M	Administration	$30,440	Santa Rosa	01-Feb-91	15.0	Yes
000139	William	Cobb	M	Human Resource	$27,500	Boston	28-May-01	5.0	Yes
000141	Angela	Dickinson	F	Engineering	$23,900	Detroit	13-Nov-96	10.0	No
000174	Cindy	Edwards	F	Accounting	$21,500	San Francisco	15-Aug-85	21.0	No
000220	Jack	Edwards	M	Engineering	$32,200	Atlanta	14-Feb-97	9.0	Yes
000367	Amanda	Fletcher	F	Human Resource	$26,500	Boston	03-Jan-99	7.0	No
000315	Dave	Grogan	M	Information Serv	$47,500	Seattle	03-Apr-98	8.0	No
000185	Rebecca	Johnson	F	Human Resource	$50,200	Boston	04-Feb-96	10.0	No
000211	Stuart	Johnson	M	Engineering	$62,000	Seattle	29-Dec-00	6.0	No
000118	Janet	Kaplan	F	Engineering	$34,000	Boston	22-Jun-91	15.0	Yes
000222	Mary	King	F	Accounting	$38,000	Detroit	10-Mar-97	9.0	No
000146	Edward	Krauss	M	Administration	$86,200	Chicago	13-Jul-99	7.0	Yes
000162	Kimberly	Lerner	F	Human Resource	$34,900	Chicago	28-Jun-99	7.0	No
000210	Victoria	Morin	F	Engineering	$40,700	Seattle	20-Dec-96	10.0	No
000284	Miriam	Morse	F	Engineering	$29,600	Chicago	02-Nov-97	9.0	No
000192	Deborah	Mosley	F	Administration	$20,800	Detroit	23-Aug-01	5.0	No
000297	James	Percival	M	Engineering	$29,200	Atlanta	18-Dec-97	9.0	Yes
000348	Carl	Reese	M	Information Serv	$25,800	Atlanta	13-Sep-88	18.0	No
000361	Linda	Robinson	F	Engineering	$37,000	Detroit	11-Nov-98	8.0	No
000226	Adam	Rosenzweig	M	Accounting	$29,000	Detroit	01-Mar-01	5.0	No
000159	Sherry	Caulfield	F	Accounting	$24,100	Boston	19-Mar-94	12.0	No
000190	Elizabeth	Santos	F	Information Serv	$27,200	Atlanta	17-Jul-96	10.0	Yes

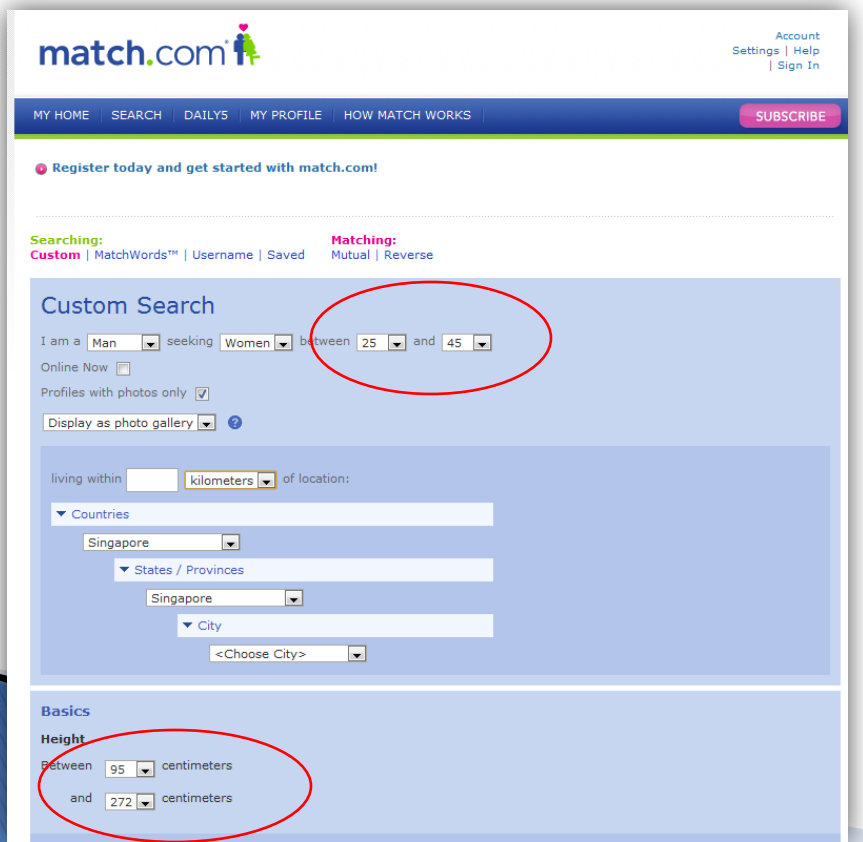
Queries

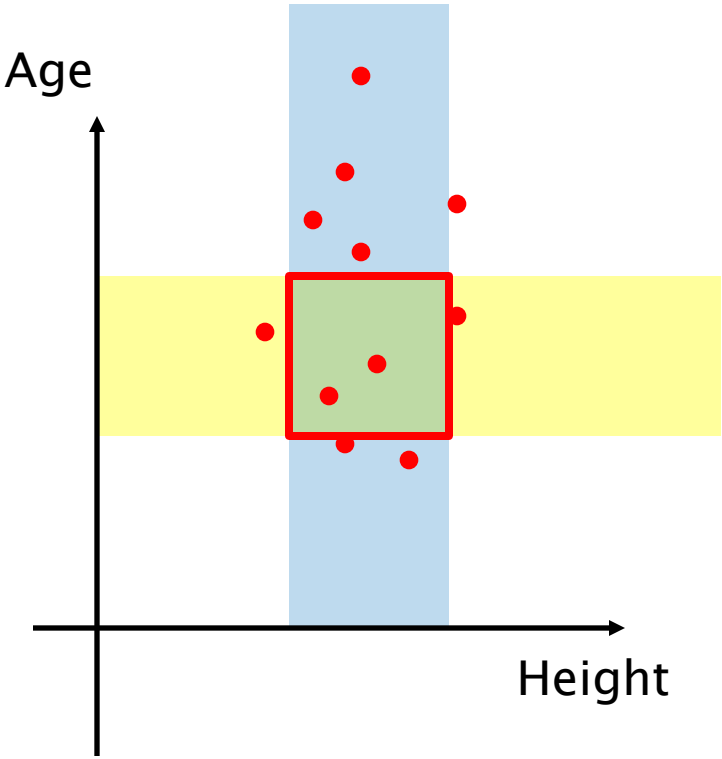
- Exact match query
  - Asks for the objects whose coordinates match query coordinates exactly

- **Range query**
  - Asks for the objects who lie in a specified query range (interval) , e.g. Age between 25 to 30
    - Search (Report)
      - List out all objects in the range
    - Counting
      - Just the number of objects

Multi-dimensional Range Query

- Search for
  - Age between 25~30 AND
  - Height between 160~180cm

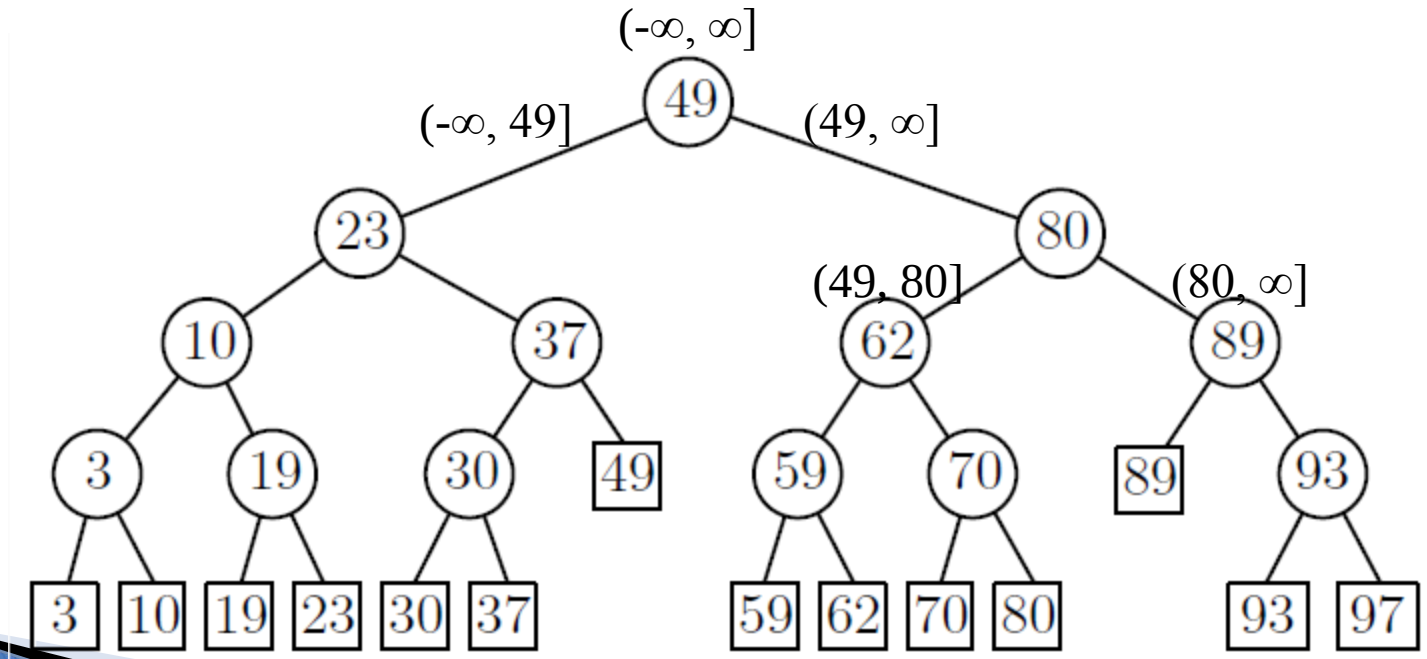




Recap: BST

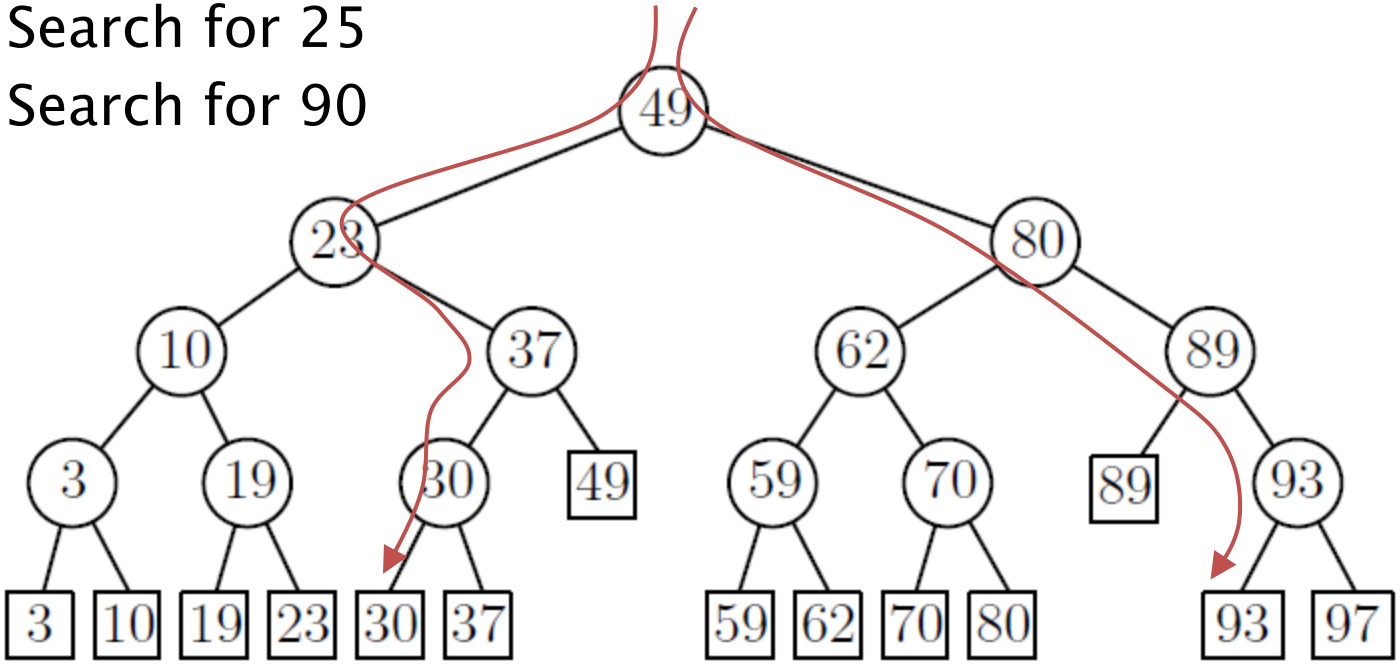
Recap: Balanced Binary Search Tree

- We assumed the BST is **balanced**
- Each internal node T divided its parent's interval $(a,b]$ into two $(a, \text{T.key}]$ and $(\text{T.key}, b]$



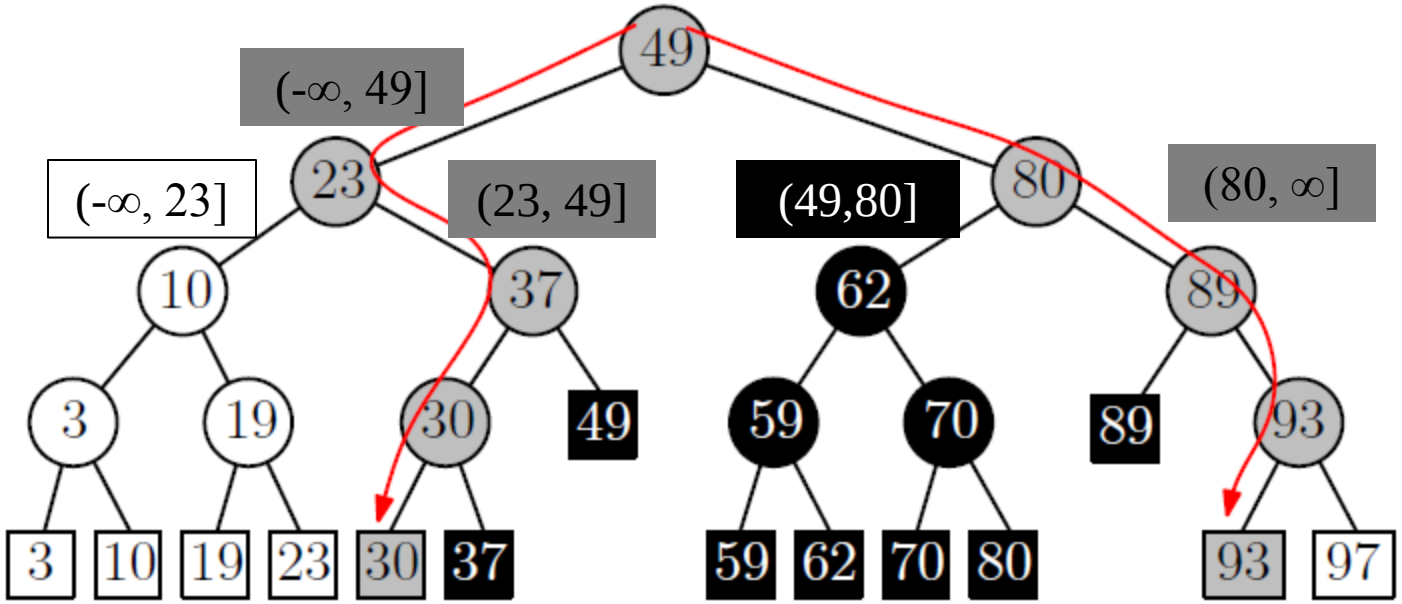
1D Range Search

- Balanced binary search tree
  - Search for all numbers between 25 and 90
  - Steps:
    - Search for 25
    - Search for 90



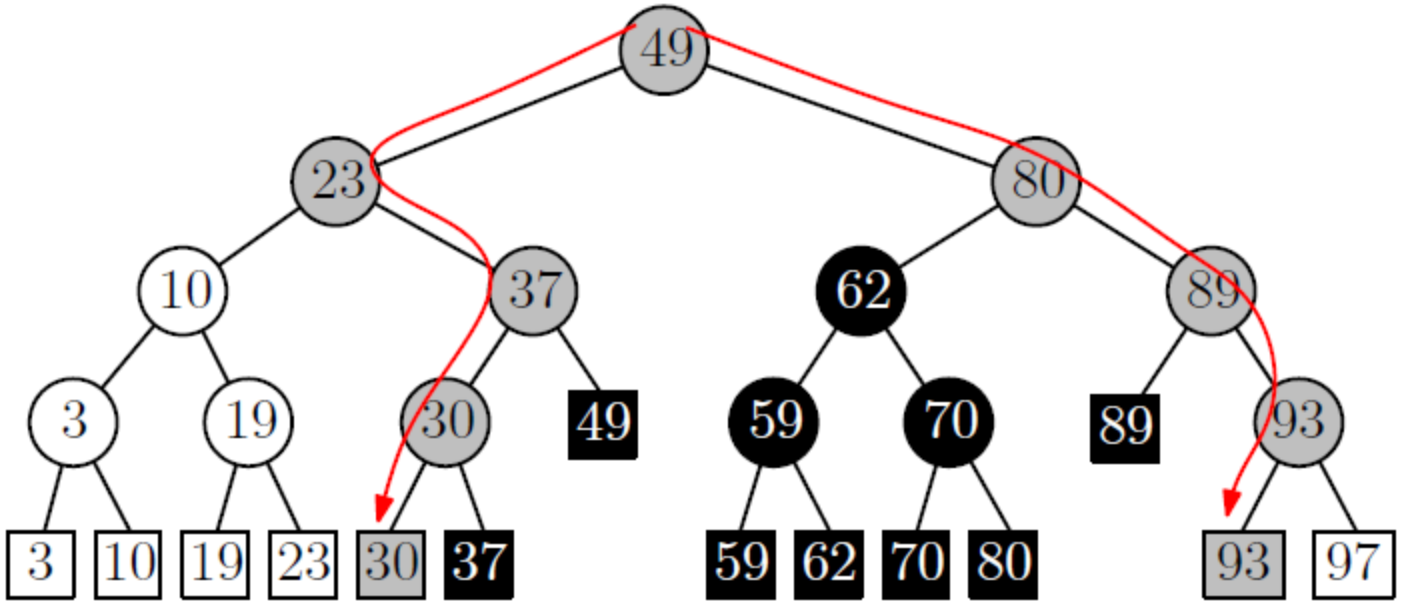
1D Range Search Complexity

- Interpretation
  - White: totally out of range
  - Black: totally within range
  - Grey: otherwise



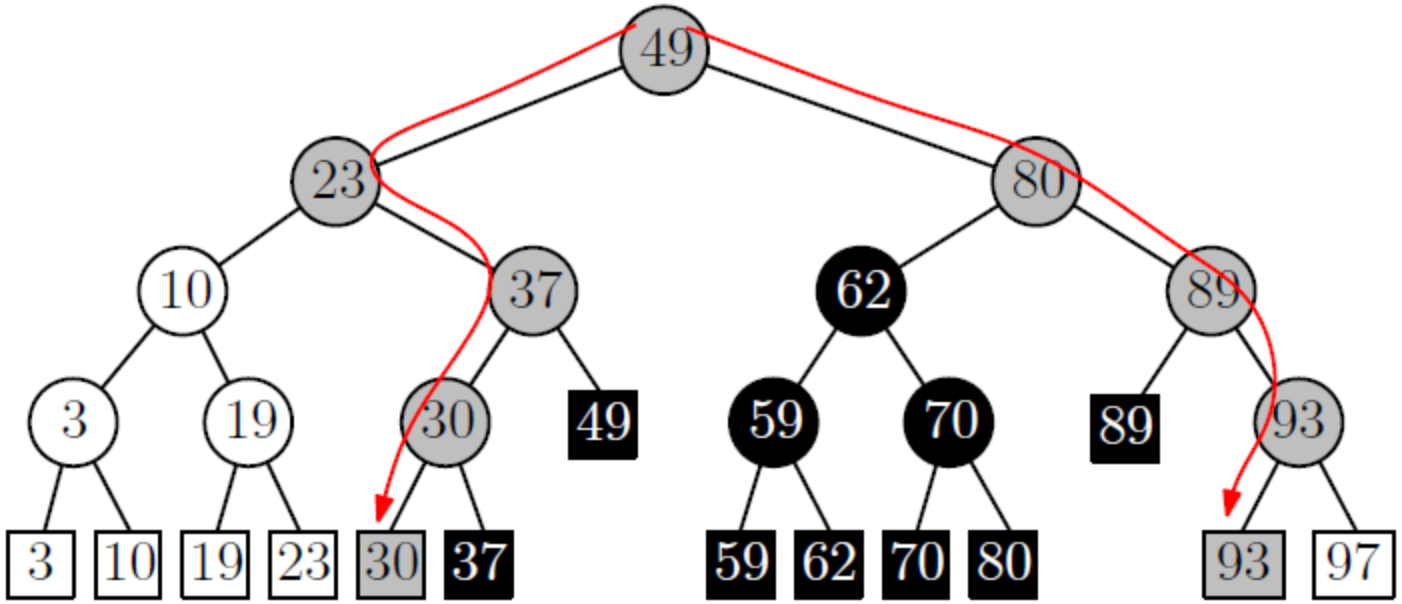
1D Range Search Complexity

- Note the THREE different types of nodes
  - White: never visited
  - Black: visited, and its whole subtree is the output
  - Grey: visited, unclear if they are the output



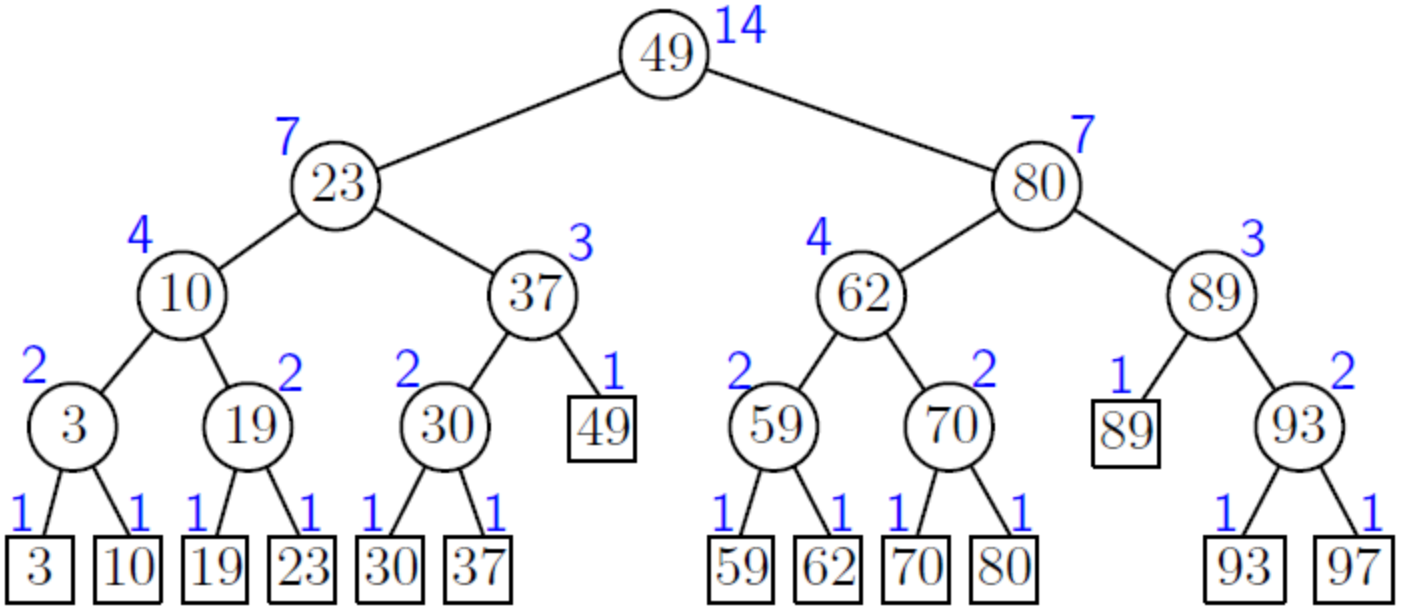
1D Range Search Complexity

- Complexity?
  - # black $<$ 2 x number of output $= O(k)$
  - # grey $<$ depth of the tree $= O(\log n)$
  - $O(\log n + k)$



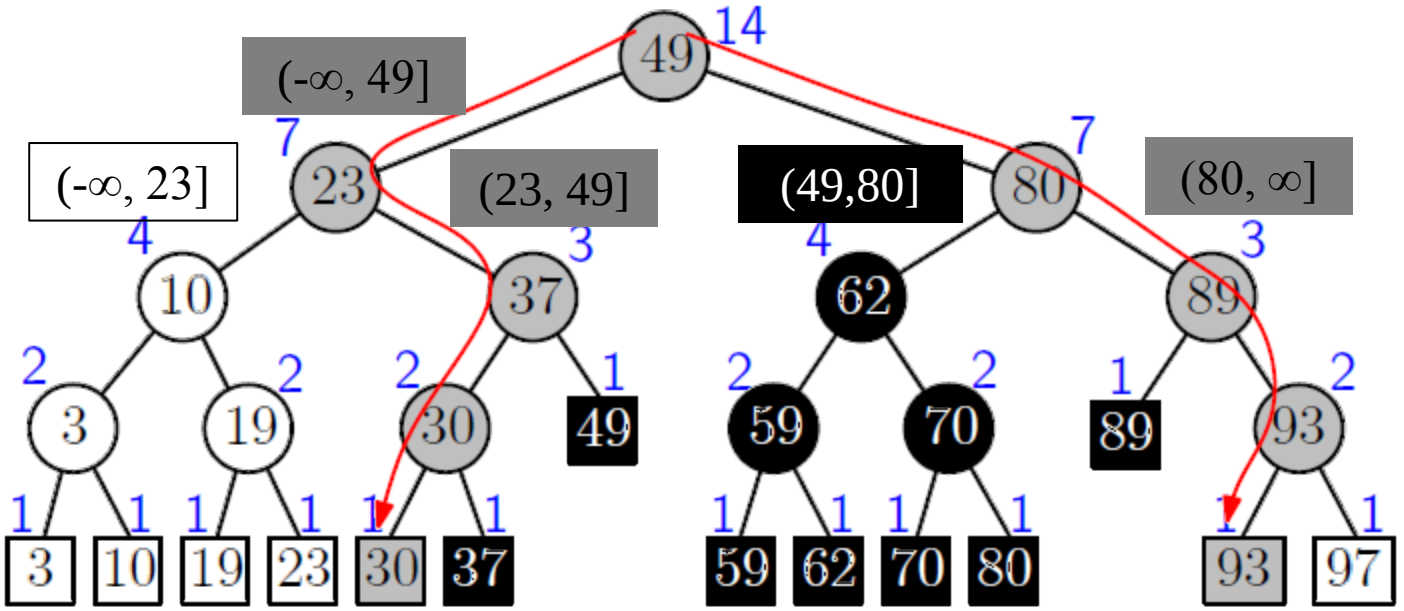
1D Range Count

- What if we just need to know the number of items within the range?
  - First, record the number of leaves in each node when the BST is constructed



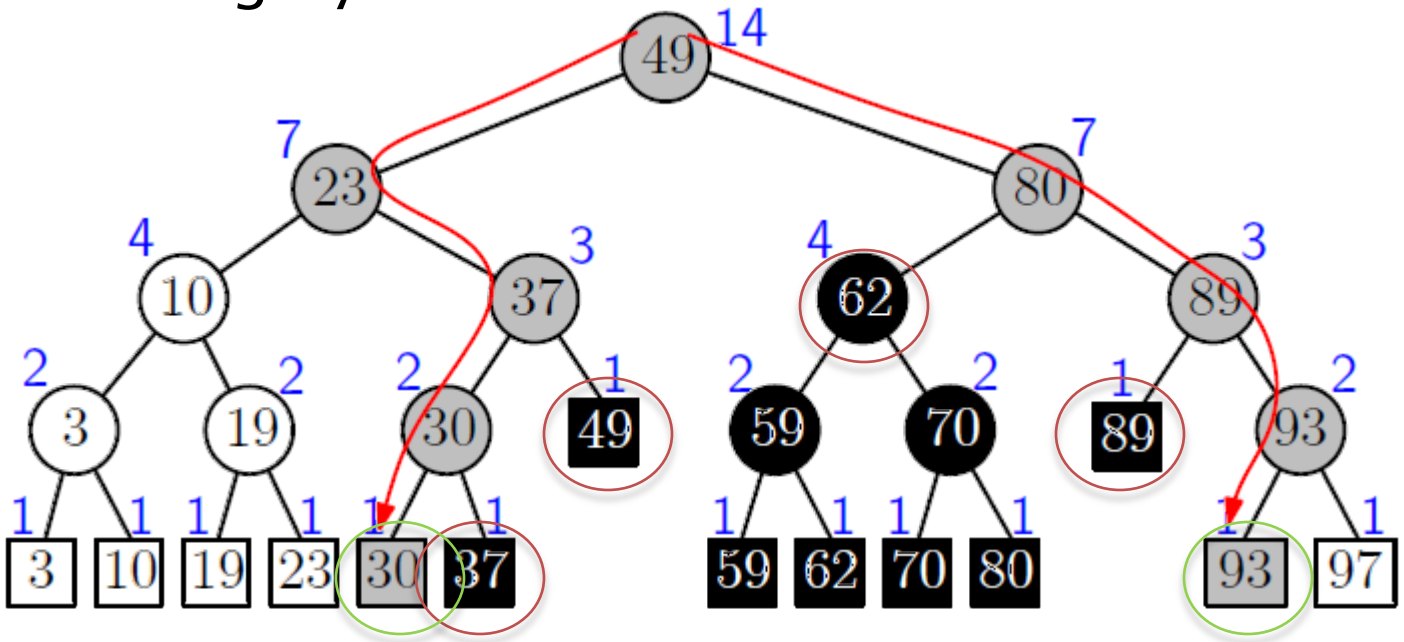
1D Range Search Complexity

- Interpretation
  - White: totally out of range
  - Black: totally within range
  - Grey: otherwise



1D Range Count

- Only need to consider
  - The ROOTS of the black node subtrees
    - At most equal to the number of grey nodes
  - The two grey leaves



Summary for 1D Range Search

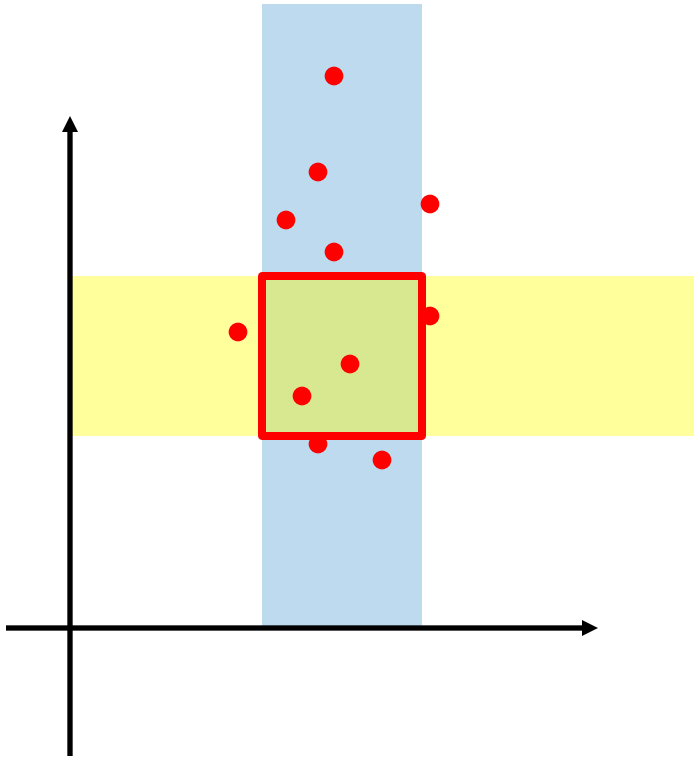
- Building BST
  - $O(n \log n)$
- Storage
  - $O(n)$
- Range query
  - Grey + Black nodes
  - $O(\log n + k)$
- Range count
  - Grey nodes + roots of Black subtree
  - $O(\log n)$

2D Range Search

kd-tree Construction

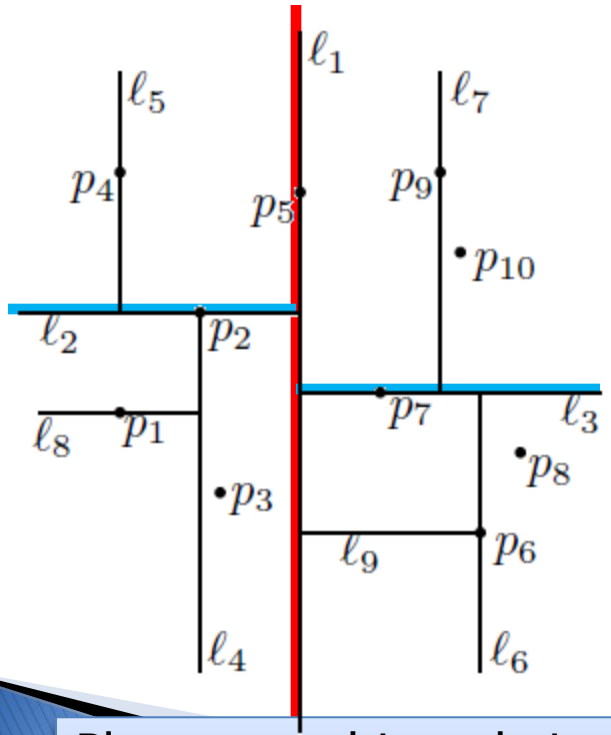
2D Range Search

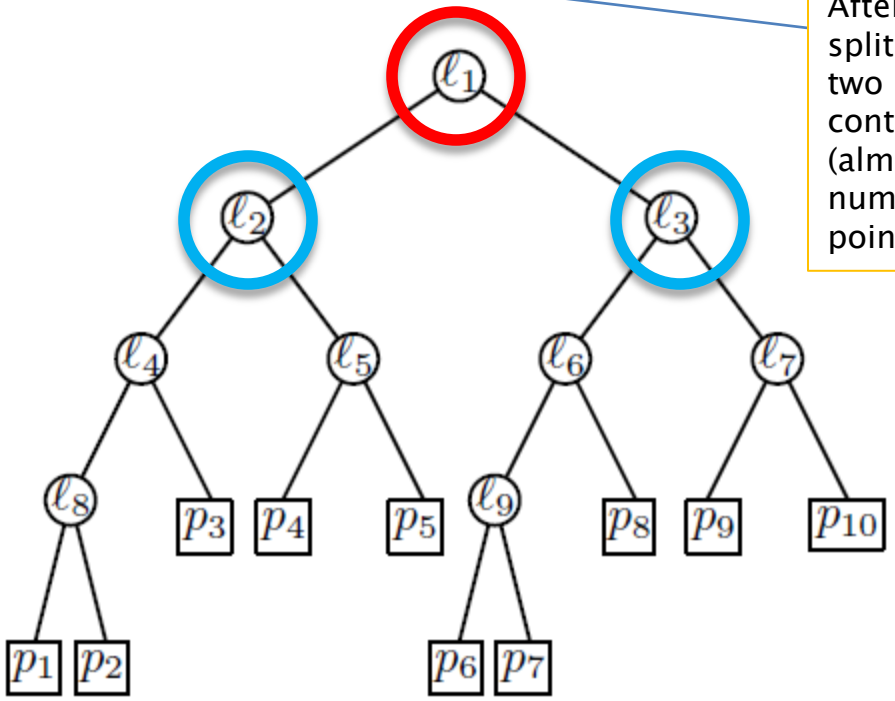
- Can we just use two BSTs, one for x, and one for y?
  - $O(n)$ if $k = n$ in the first dimension, but the final output is far less than that



kd-tree

- Idea
  - Split the points alternatively by x and y directions
  - Every split is on the median of the remaining points



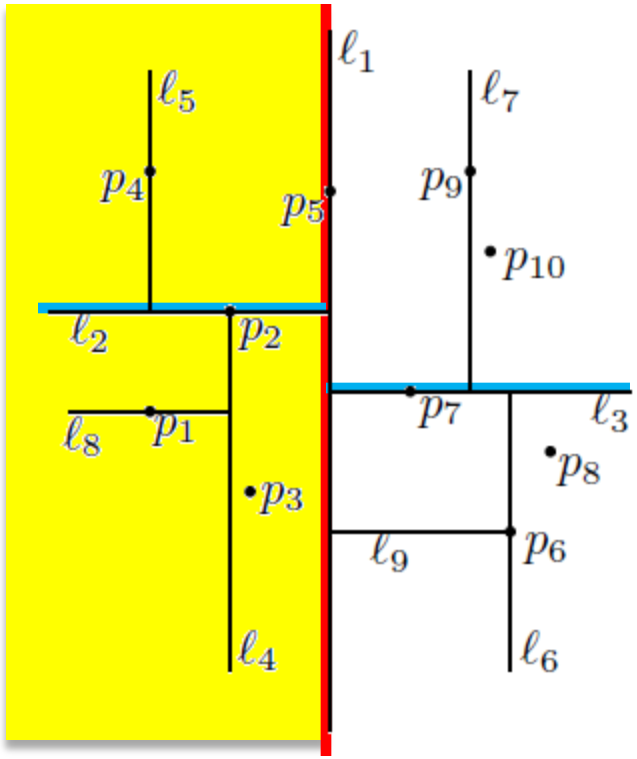


After the splitting, the two sides contain (almost) same number of points

Please see this website for a demo:
http://donar.umiacs.umd.edu/quadtree/points/kdtree.html

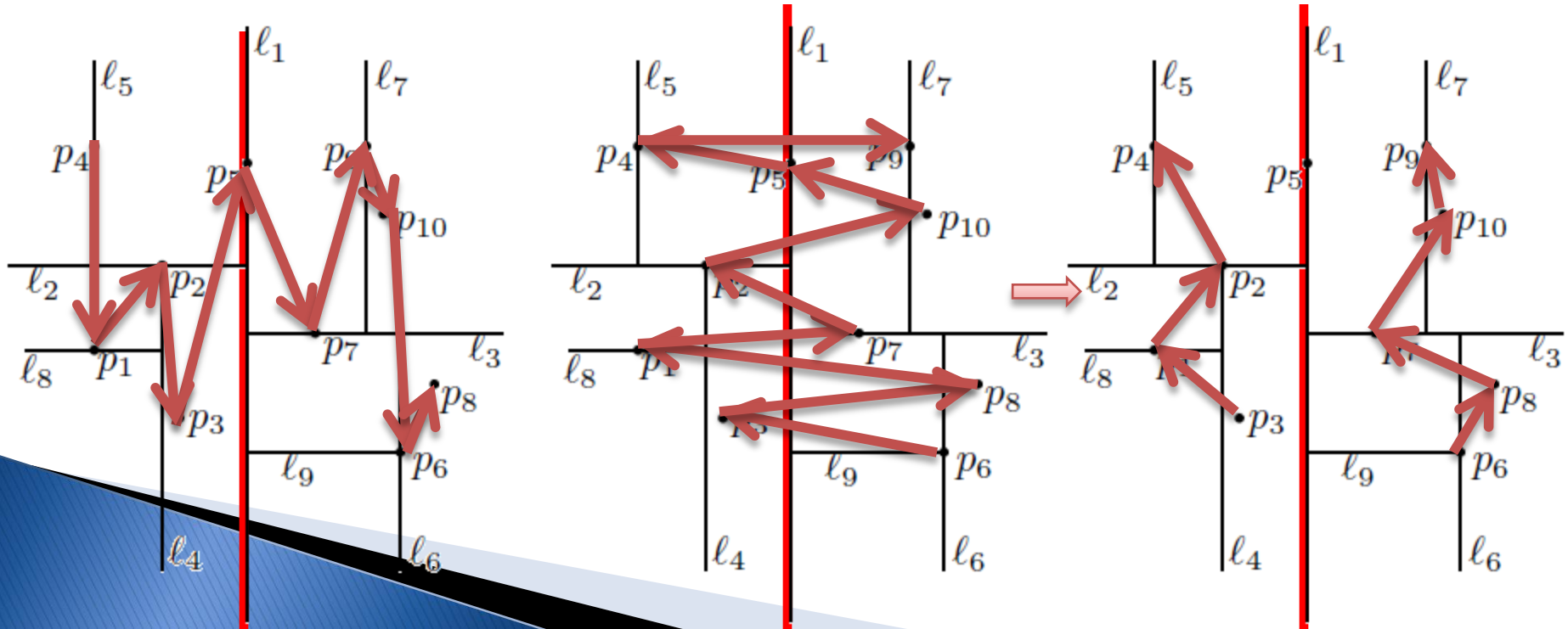
kd-tree Complexity

- At each node, splitting a set of point by their median, in either x or y
- Tricks
  - Two sorted linklists in two directions
  - Reverse of merge sort



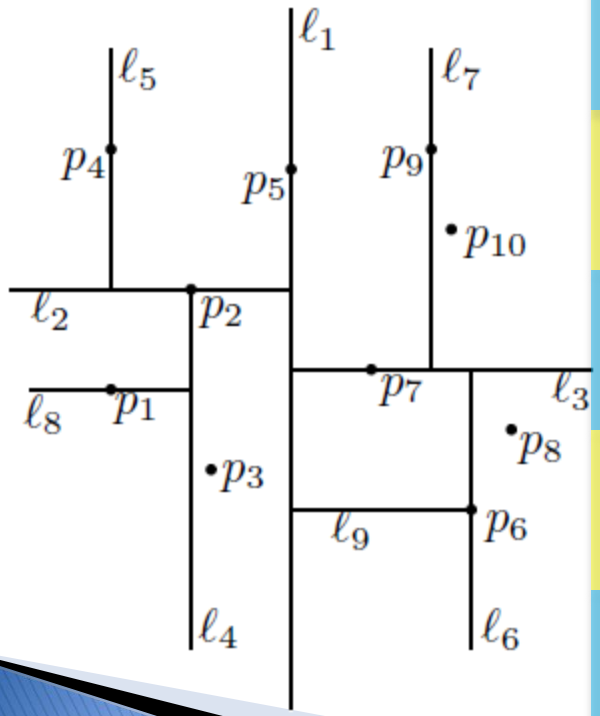
Kd-tree Complexity

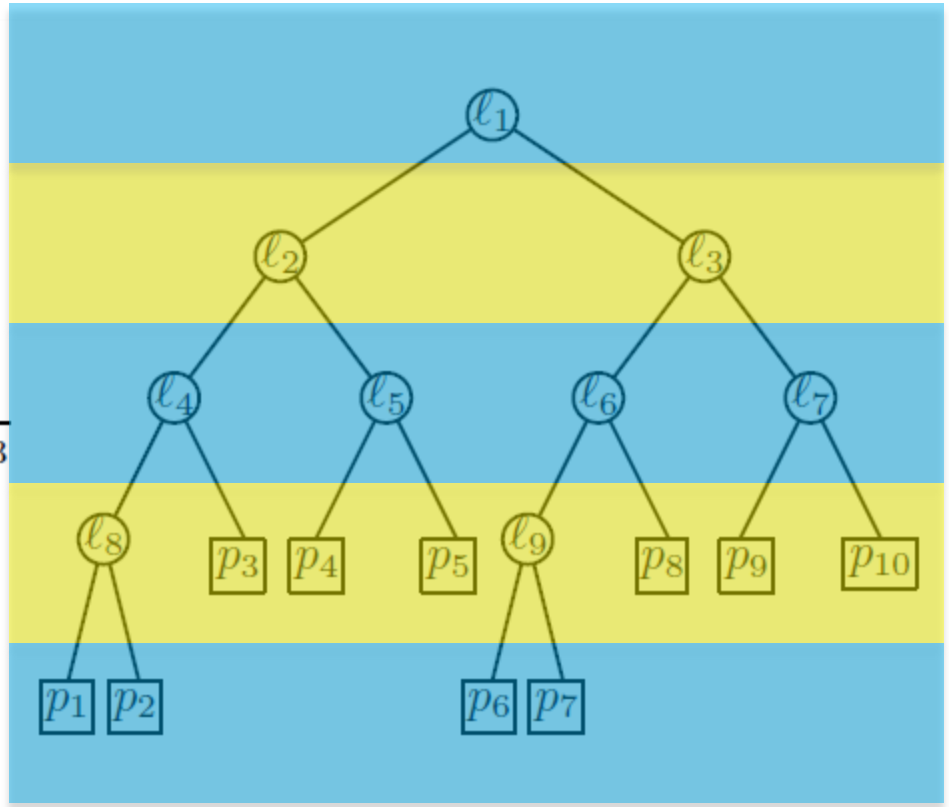
- For example, separating points by x with a vertical line
  - x-list: Simple
  - y-list: Scan through the list and separate into two



kd-tree Complexity

- Processing altogether at most n points in each horizontal level of the tree
  - $O(n \log n)$



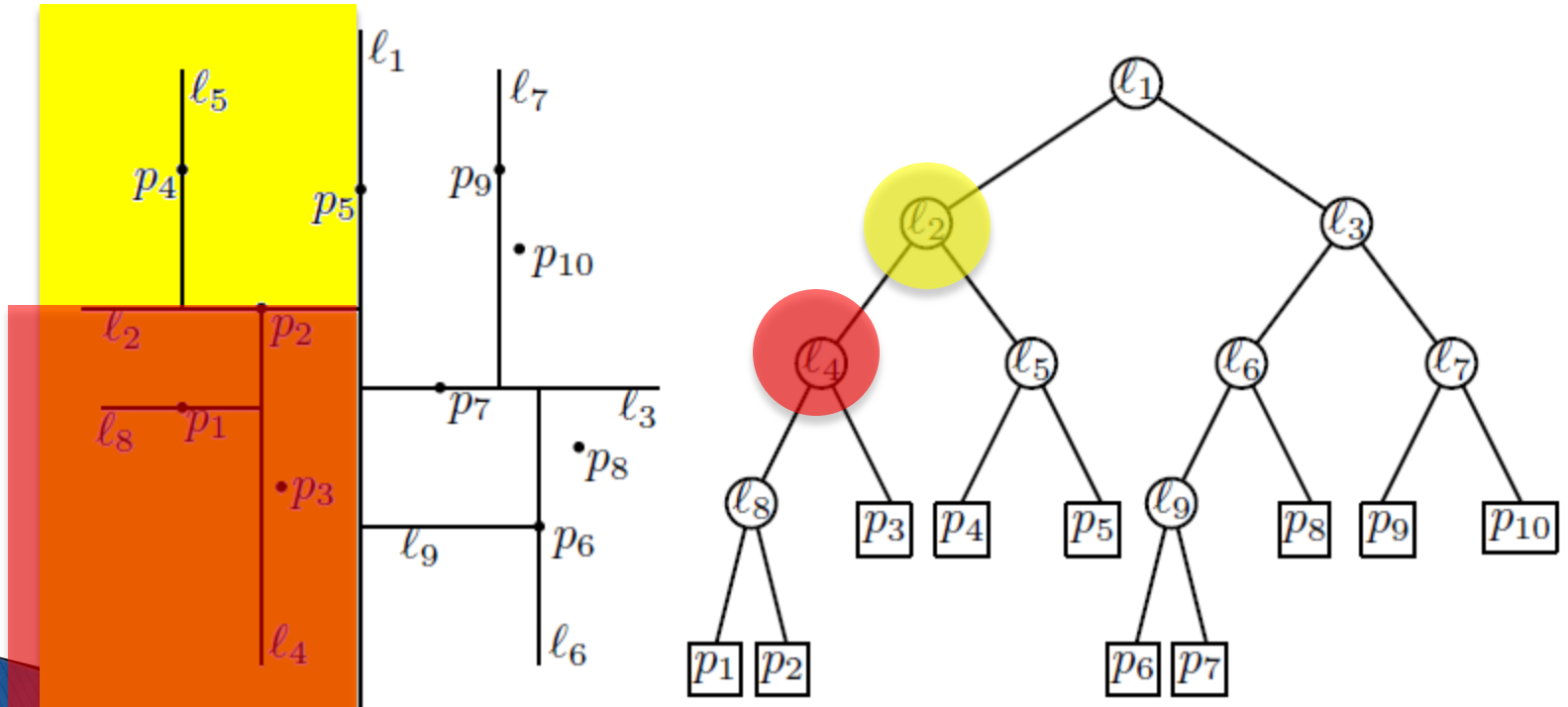


2D *kd*-tree Range Query

And its complexity

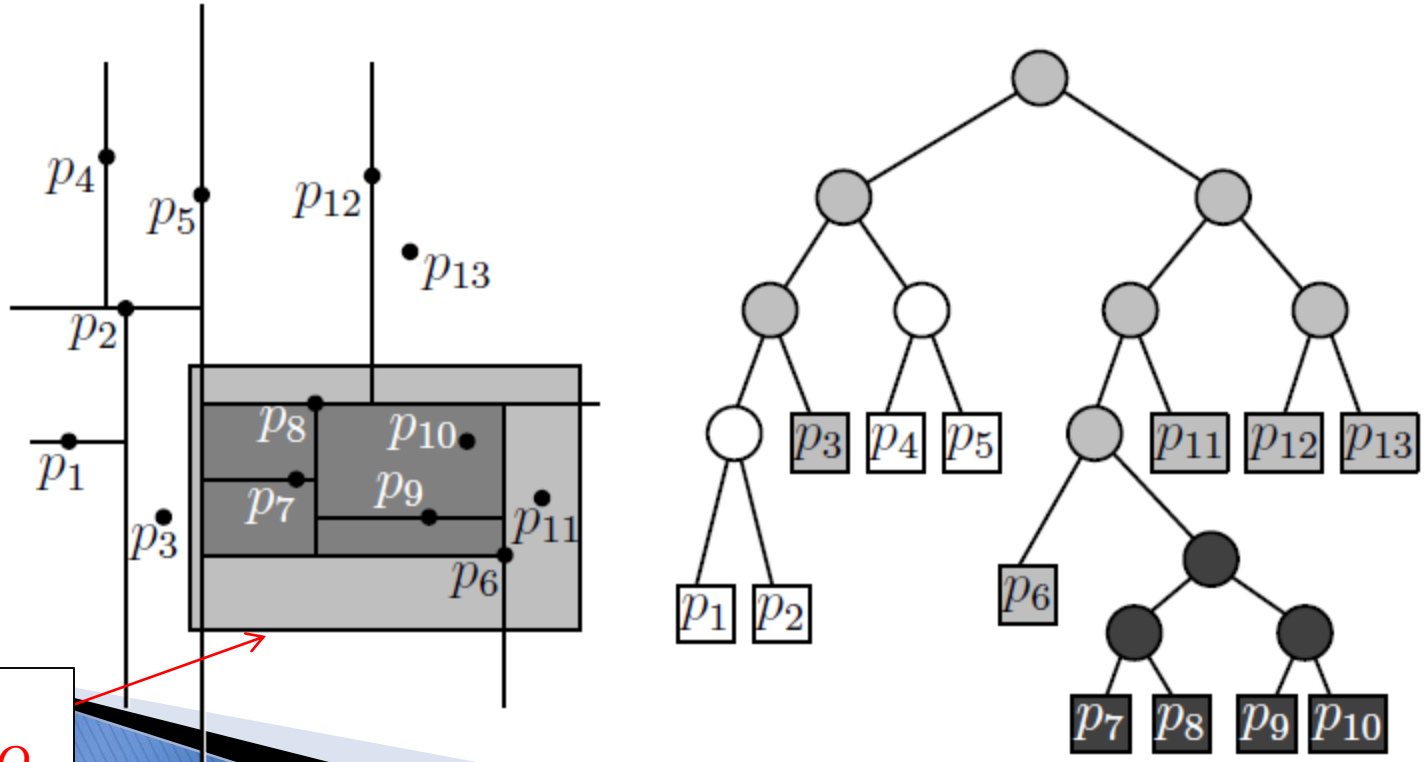
kd-tree Region

- For each node, it represents a region in the space



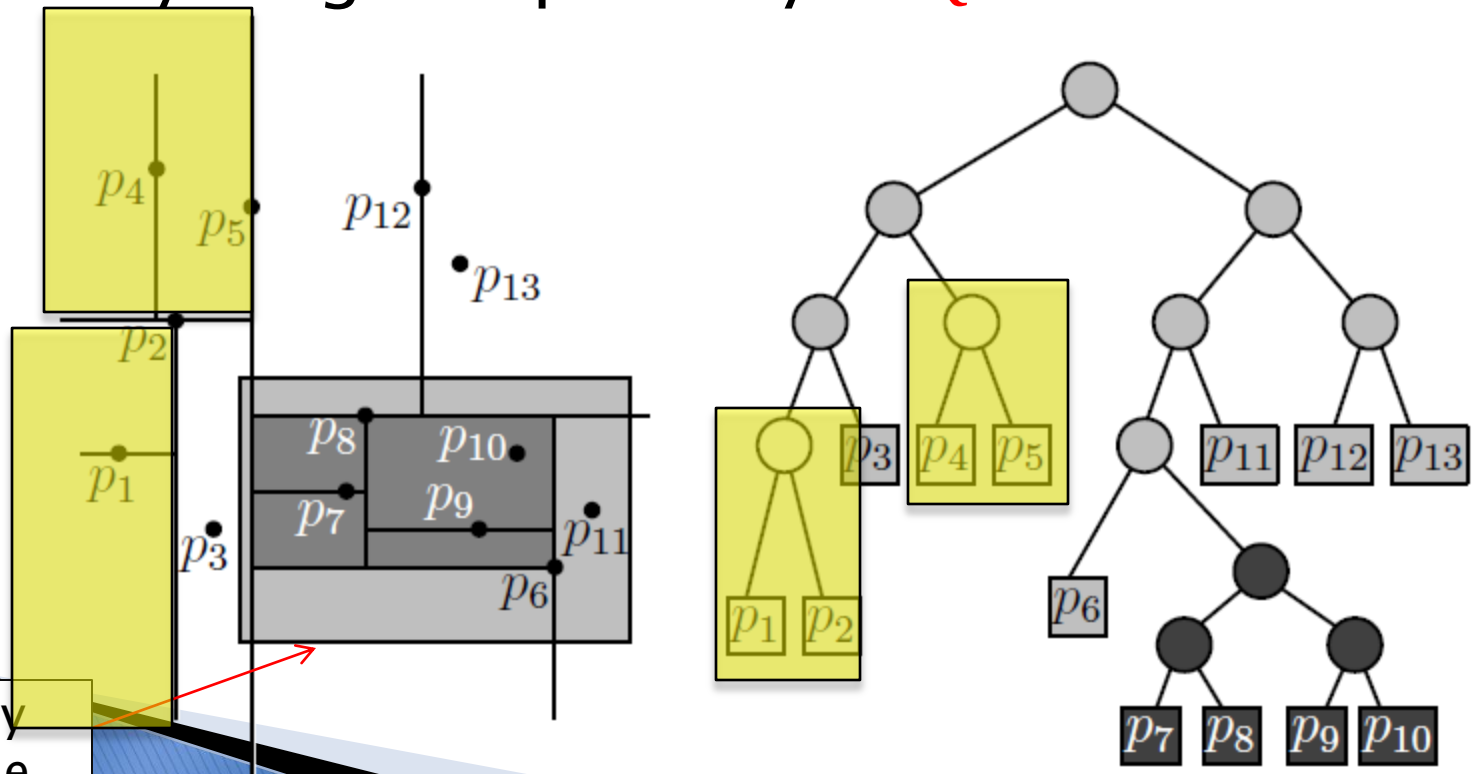
2D Range Query

- Let the area of query be a rectangular area Q
  - Again, three types of nodes (regions) in the tree
    - Namely, black, grey and white



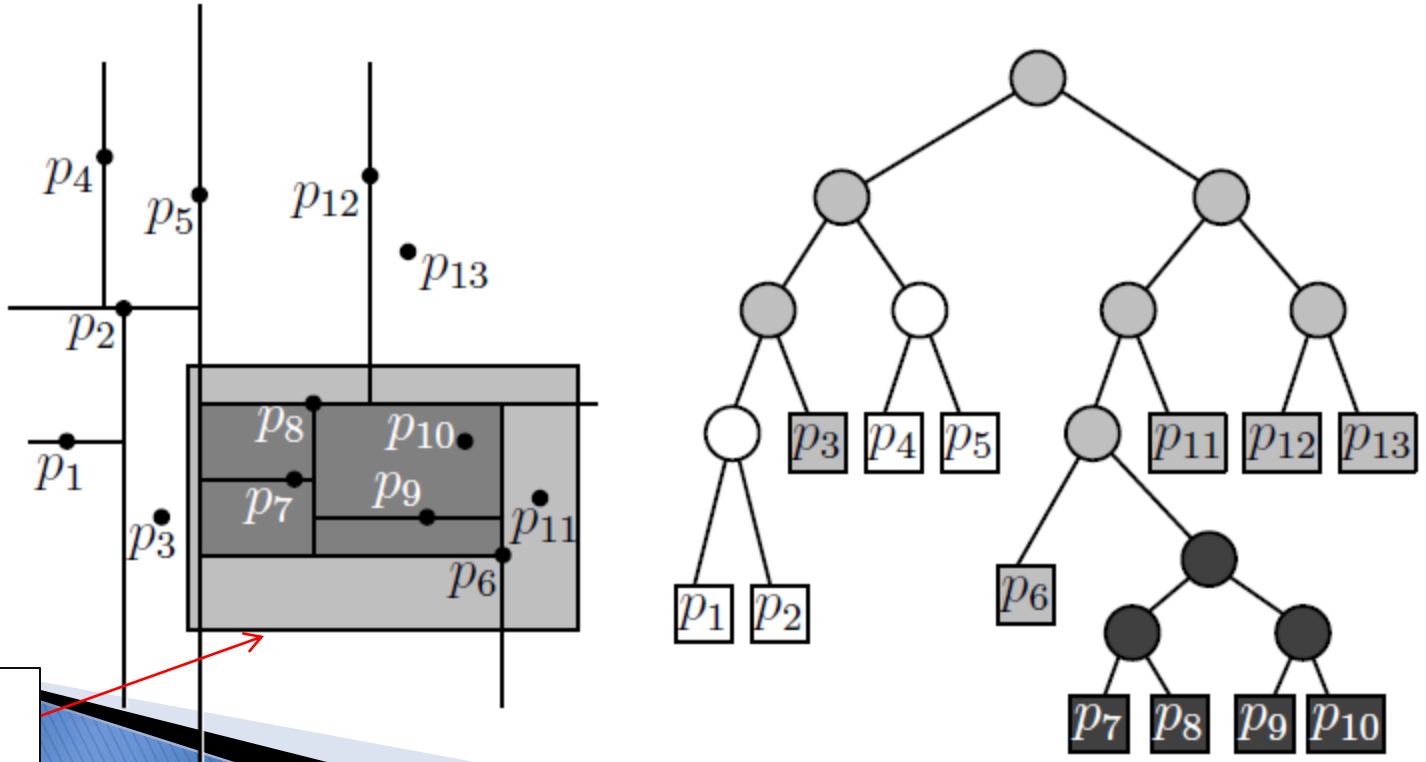
2D Range Query

- Black: Regions completely in Q
- White: Regions completely out of Q
- Grey: Regions partially in Q



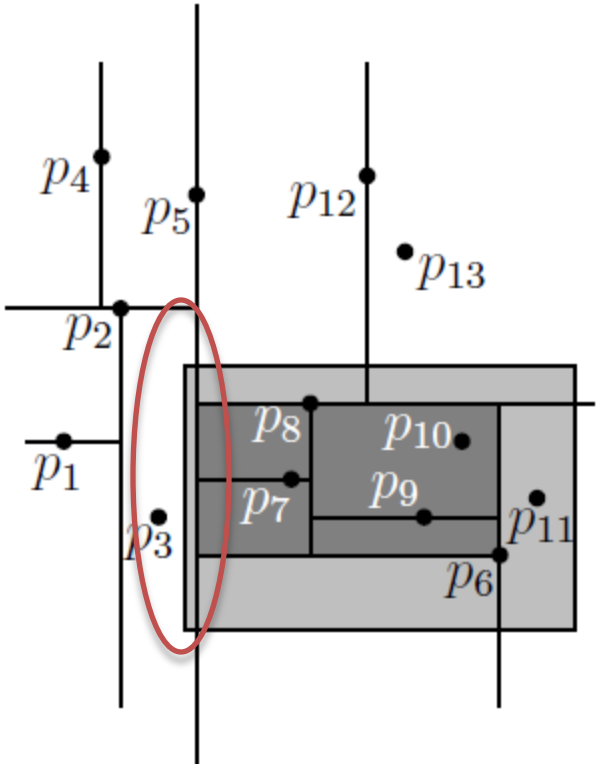
2D Range Query

- Black: Report the whole subtree
- White: Stop recursion
- Grey: Recursively search the two subtrees



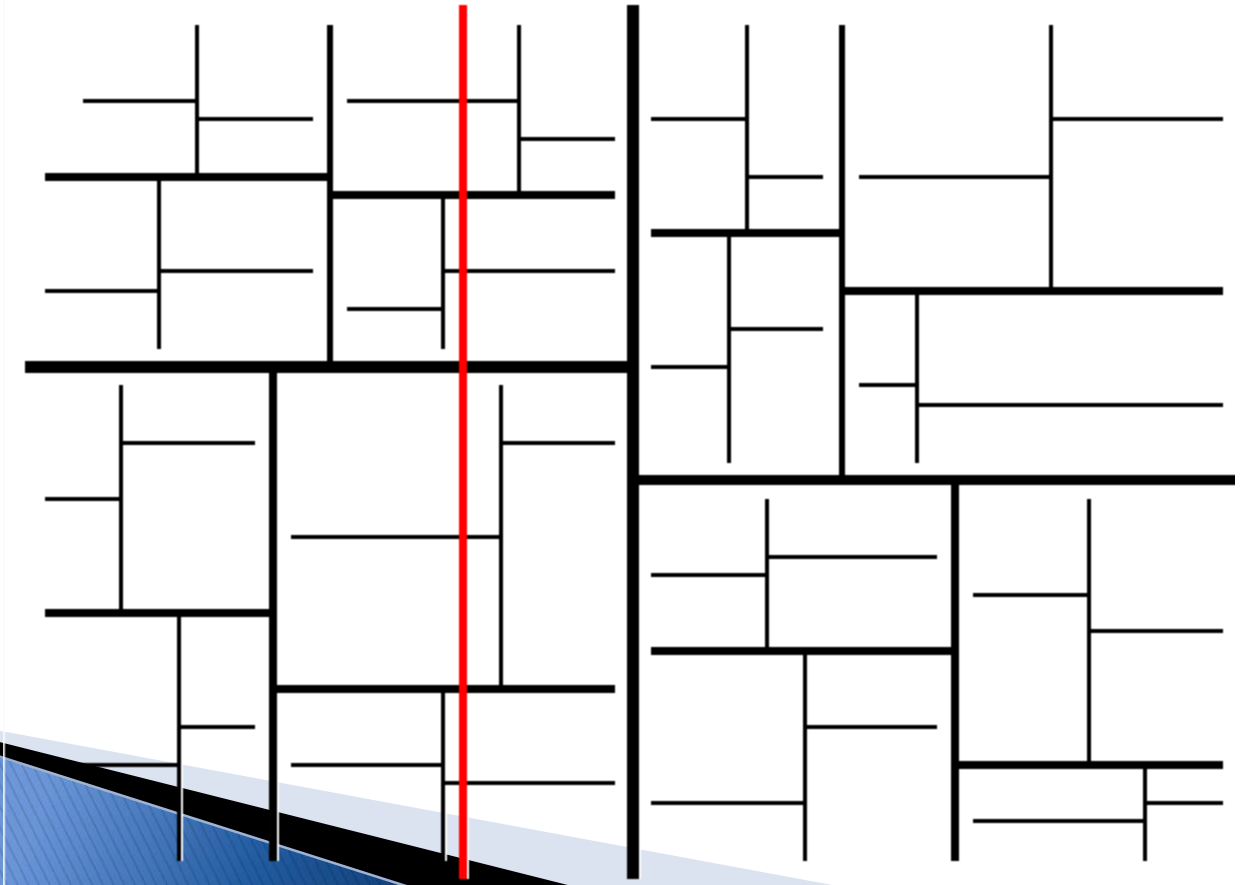
2d Range Query Complexity

- Same question: How many black and grey nodes are there?
  - We do not need to care about the white nodes
  - Black = number of output = k
  - Grey
    - When a region is partially intersecting Q, it means that the region is intersecting the boundary of Q.
    - Worst case if a side of Q spans all the vertical or horizontal cells



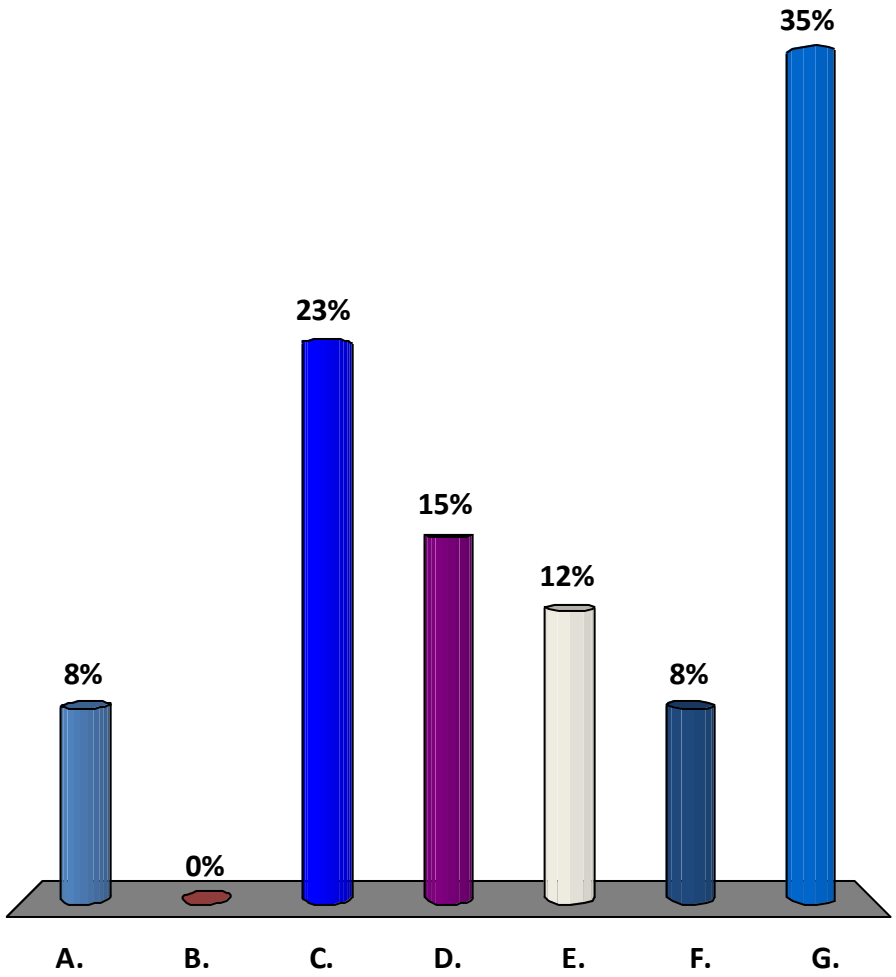
2d Range Query Complexity

- Same question
  - The worst case: How many cells touch an infinite horizontal or vertical line (acts as one side of Q)



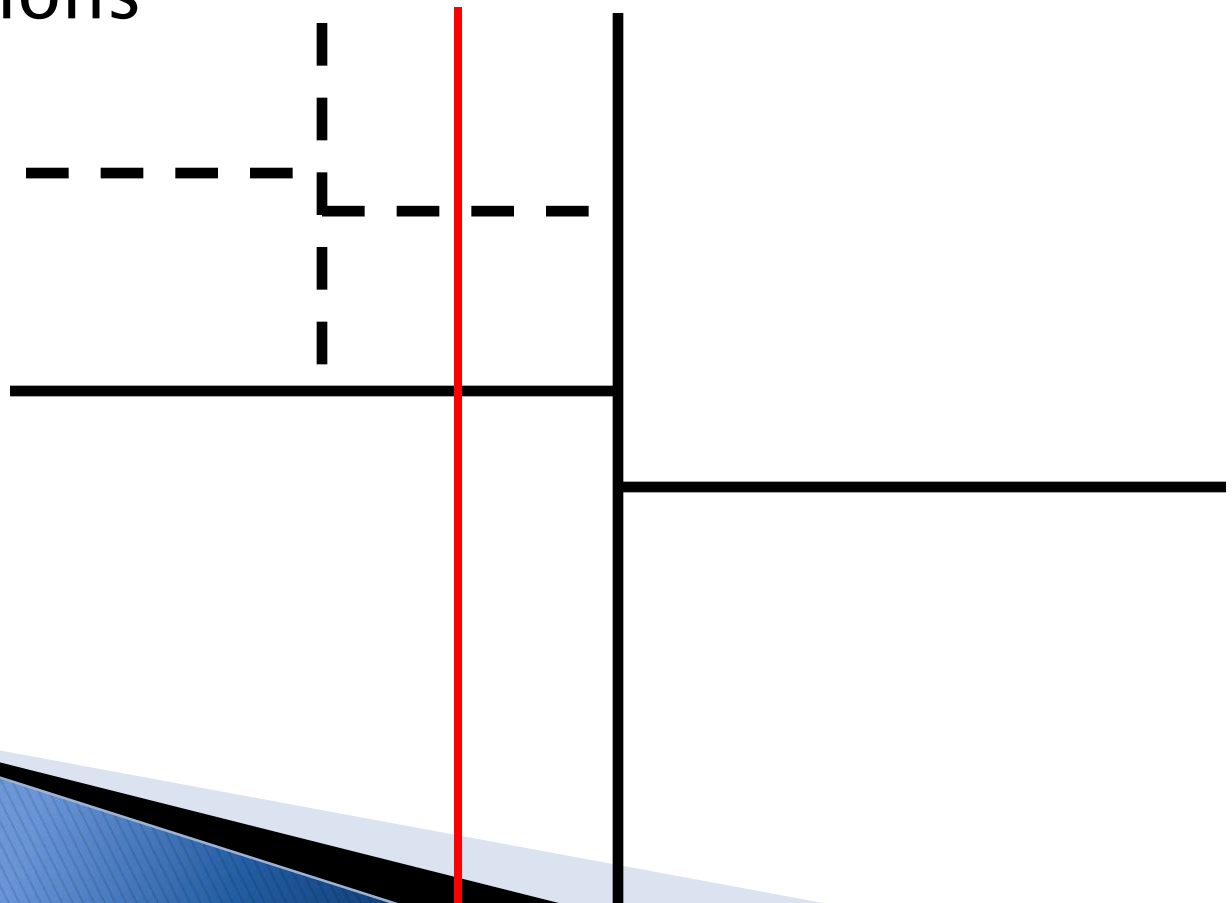
The worst case: How many cells touch an infinite horizontal or vertical line (acts as one side of Q)

A. n
B. $2n$
C. $\log n$
D. $n^{1/2}$ ✓
E. n^2
F. 2^n
G. Wait, my head pain…



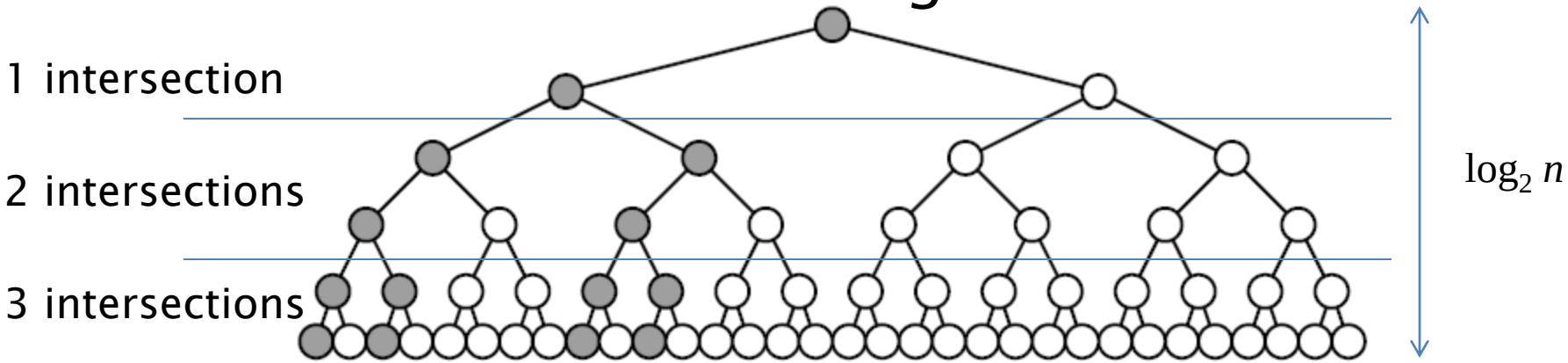
2d Range Query Complexity

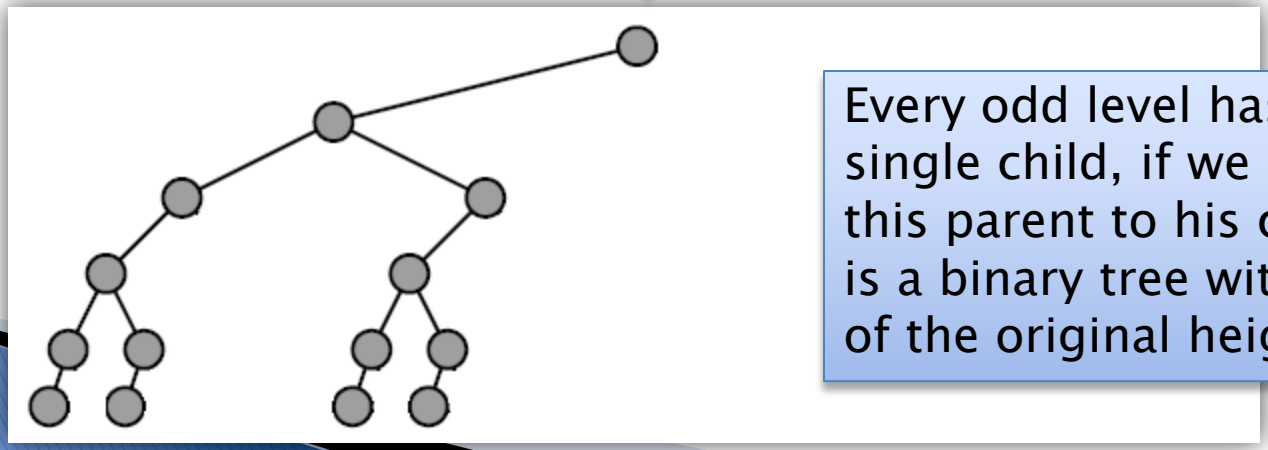
- Given an infinite line, every two splits (x, then y), the line will cut double number of the regions



#Grey Nodes

- Every two splits (x, then y), the line will cut double number of the regions





Every odd level has only a single child, if we merge this parent to his child. It is a binary tree with half of the original height

Grey Nodes

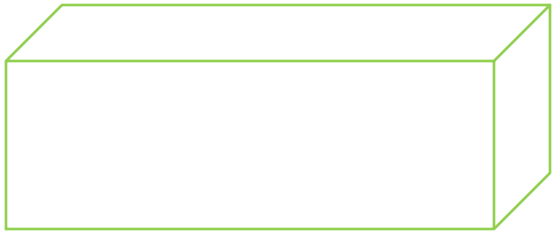
- Assume the original binary tree is complete, the half heighted (after merging nodes) one will also be complete
- The number of nodes with height $\frac{1}{2}\log_2 n$ is

$$2^{\frac{1}{2}\log_2 n} = 2^{\log_2 \sqrt{n}} = \sqrt{n}$$

- Reverse the effect of merging the single parent with its child, then count the four sides, the number of grey nodes is $O(8\sqrt{n})$
- Finally, the complexity is $O(\sqrt{n}+k)$

#Grey nodes #Black nodes

Higher Dimensional Range Query

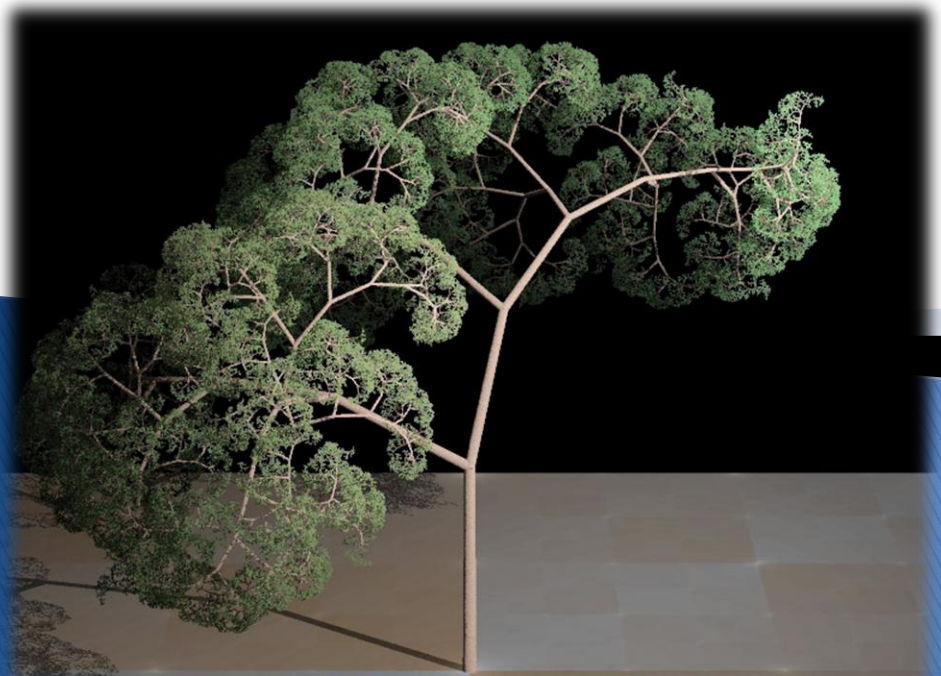


- For example, 3D query
  - Q is a rectangular cuboid
- Similar construction, split x, y, then z
- Complexity, every 3 splits, a plane cut 4 regions

$$O(n^{1/3} + k)$$

- d-dimensional range query?

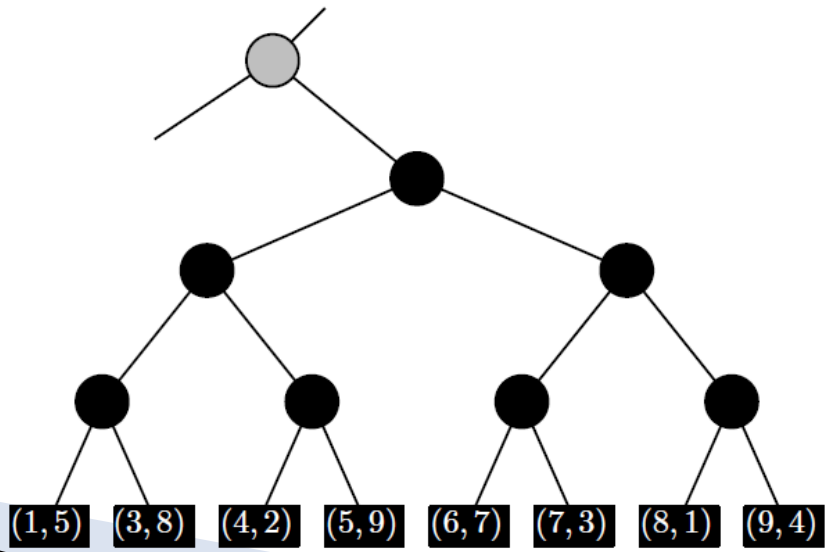
2D Range Tree



2D Range Trees

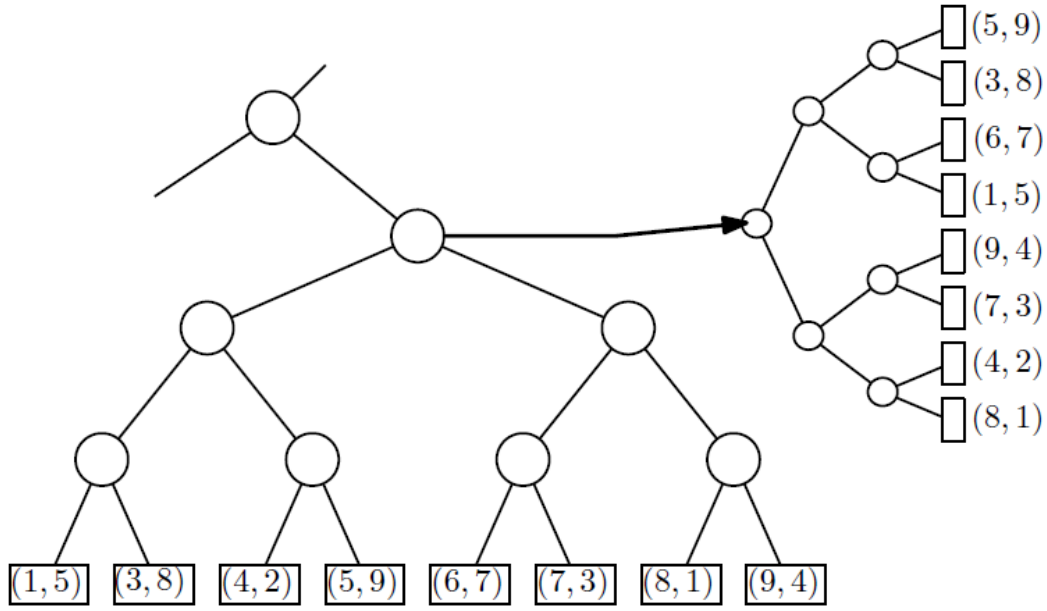
- Can we do better?
  - $O(\sqrt{n})$ is not cheap
- First, build a BST according to x

n	$\log n$	$\sqrt{n}$
4	2	2
16	4	4
64	6	8
256	8	16
1024	10	32
4096	12	64
1.000.000	20	1000



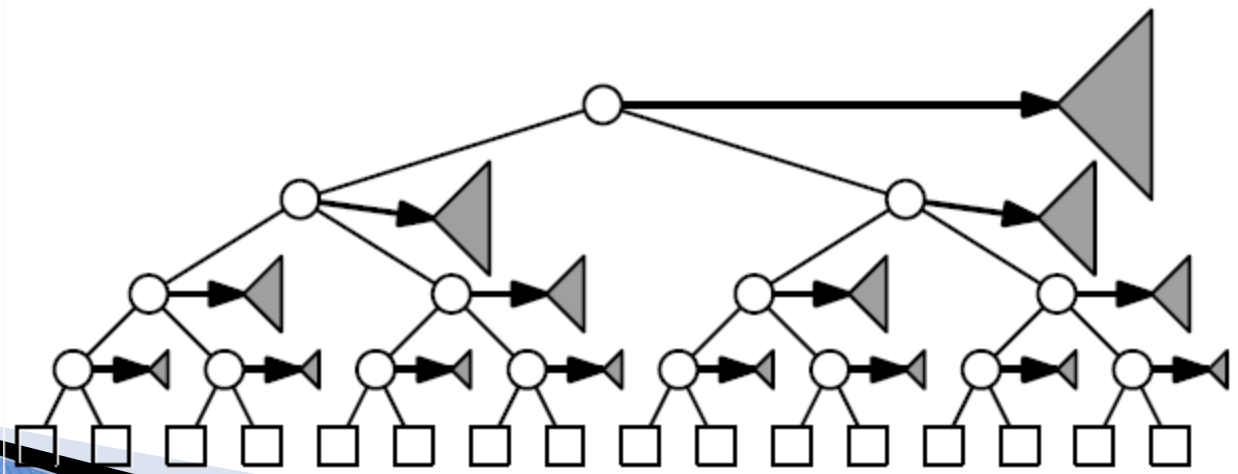
2D Range Trees

- For each node, there is a set of points in its subtree
- We build another BST in the direction of y for EACH NODE



2D Range Trees

- How much storage does this crazy tree cost?
  - $O(n)$ for each level
  - $O(n \log n)$ for all levels
- However, the construction time for EACH level is $O(n \log n)$
  - So overall construction needs $O(n \log^2 n)$

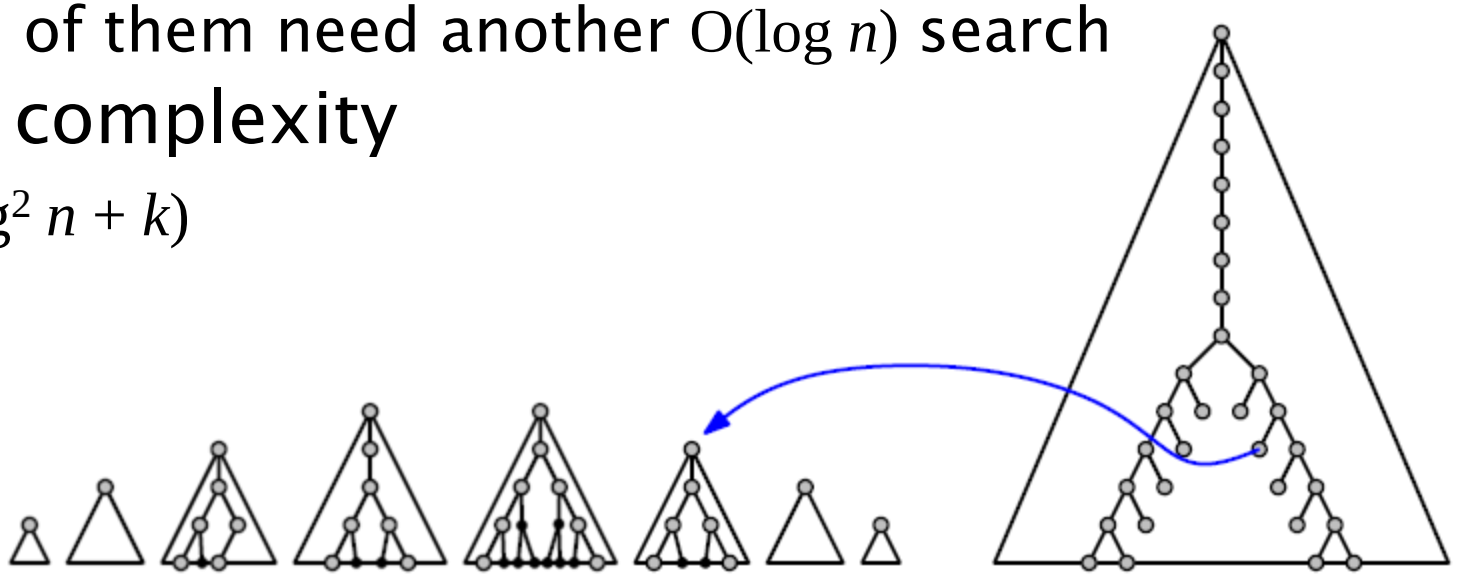


2D Range Trees

- But if we reuse the linklist trick in the previous kd-tree construction
  - For every y-tree, we know the sorted order
  - Time complexity to construct a BST with given sorted order?
    - $O(n)$
- So overall construction needs $O(n \log n)$

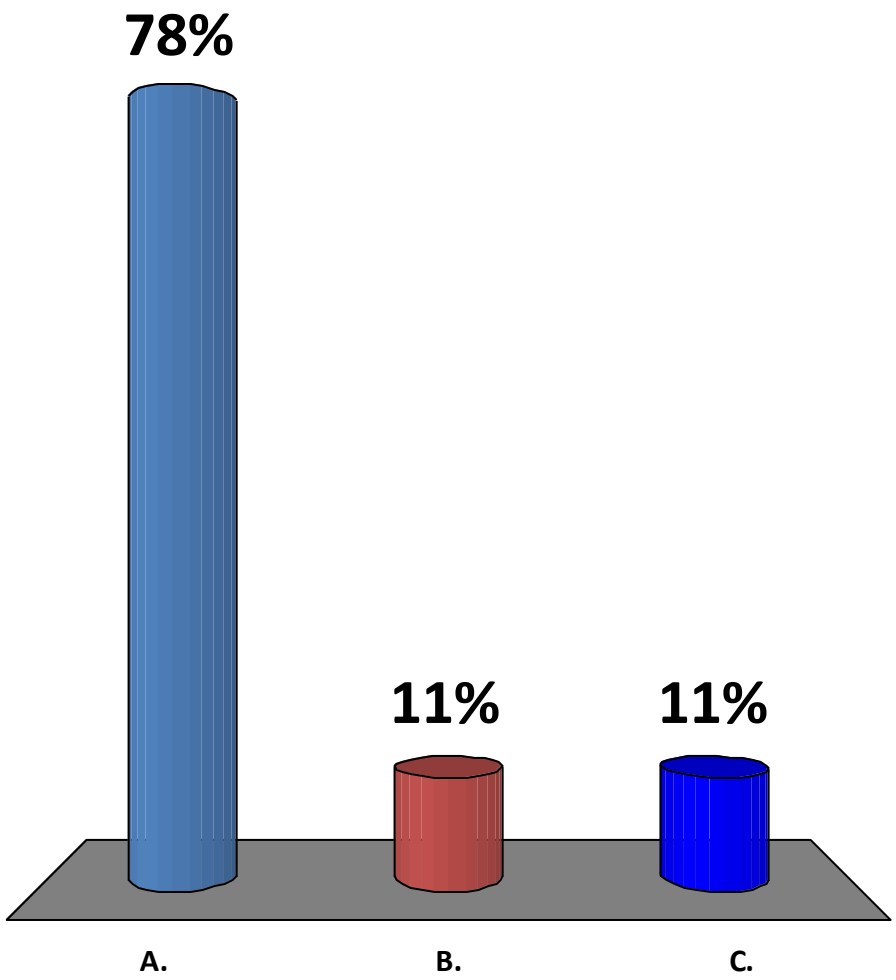
2D Rectangular Query

- Again, we ask how many grey nodes
  - We know it is $O(\log n)$ for the first dimension
- It means, the number of trees we need to search in the second dimension is $O(\log n)$
  - Each of them need another $O(\log n)$ search
- Total complexity
  - $O(\log^2 n + k)$



If n is big, which one is better

A. $O(\log^2 n)$
B. $O(n^{1/2})$
C. Same

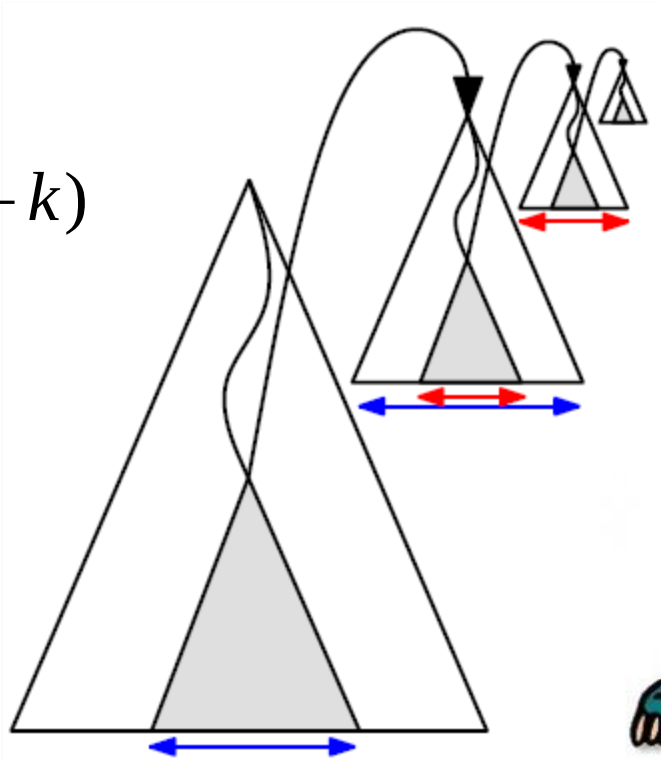


A Brief Idea of Efficiency

n	$\log n$	$\log^2 n$	$\sqrt{n}$
4	2	4	2
16	4	16	4
64	6	36	8
256	8	64	16
1024	10	100	32
4096	12	144	64
16384	14	196	128
65536	16	256	256
1M	20	400	1K

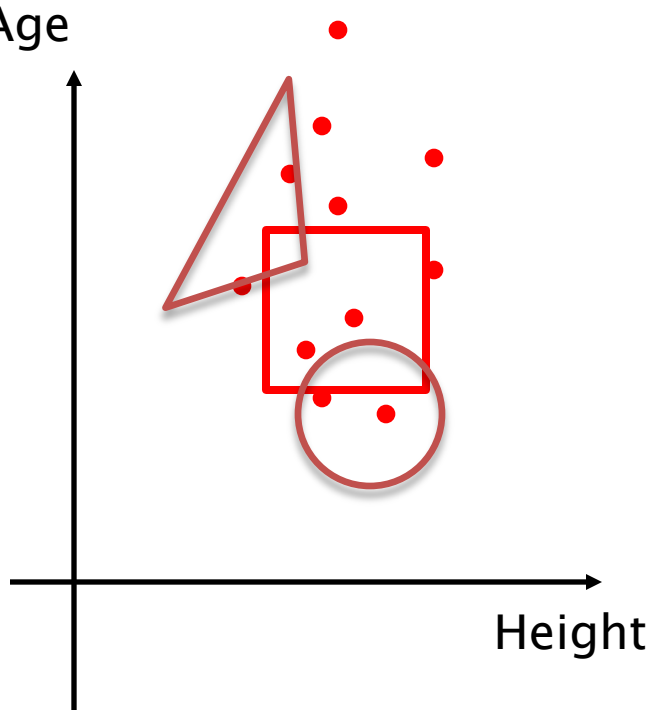
N-dimensions

- Recursively construct BST in a node, then another BST in a node , and……
- Complexity
  - Storage $O(n \log^d n)$
  - Range query $O(\log^d n + k)$
  - Counting $O(\log n)$
    - By *fractional cascading*



Other Range Search

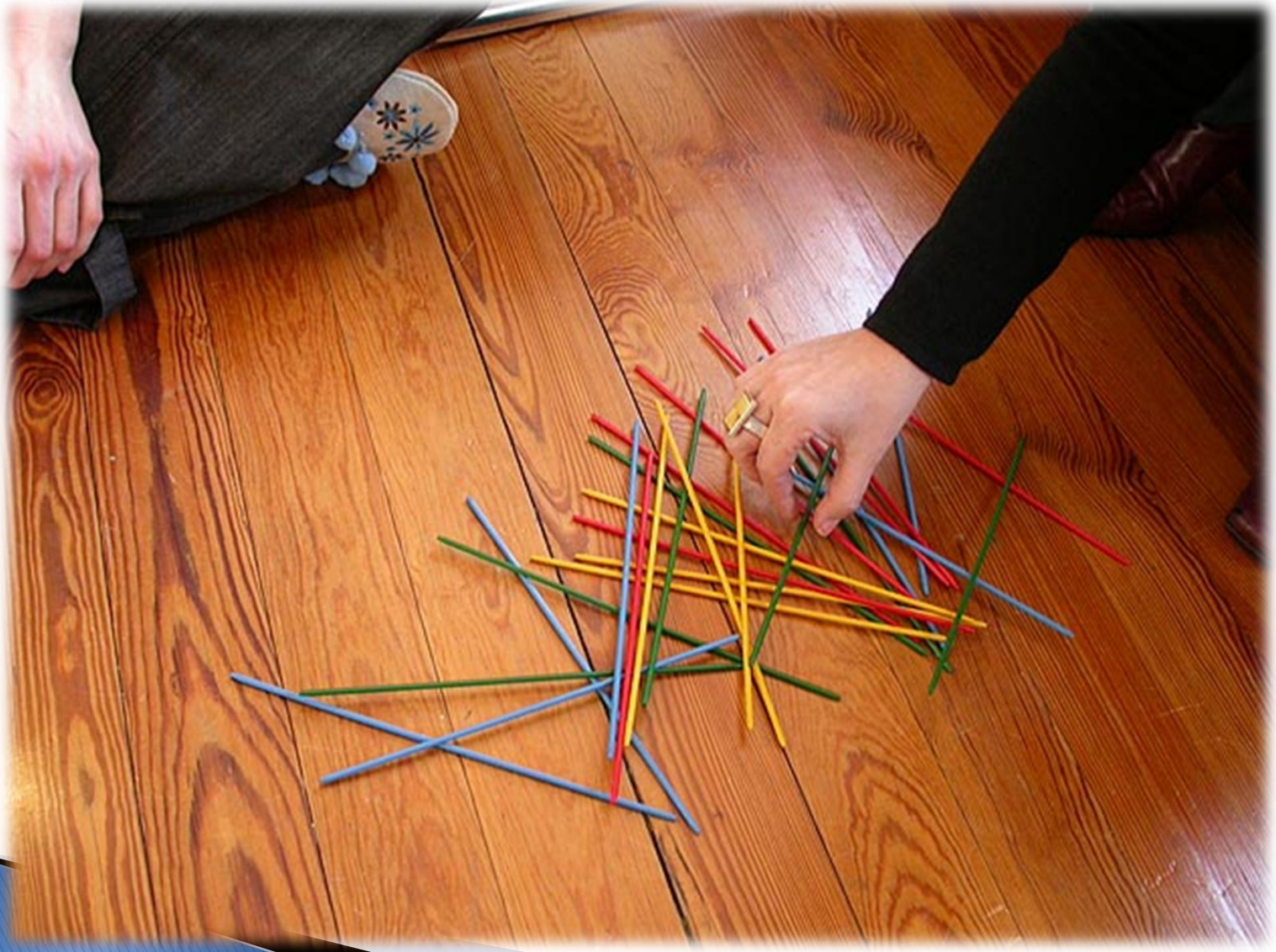
- In this lecture, we are actually doing orthogonal range search
- There are other types of range query rather than a rectangular cuboid
  - E.g. simplex, sphere, etc.



Geometric range searching and its relatives
Pankaj K. Agarwal. , Jeff Erickson
Advances in Discrete and Computational Geometry , American Mathematical Society Press, 1999, pages 1-56.
http://www.cs.uiuc.edu/~jeffe/pubs/survey.html

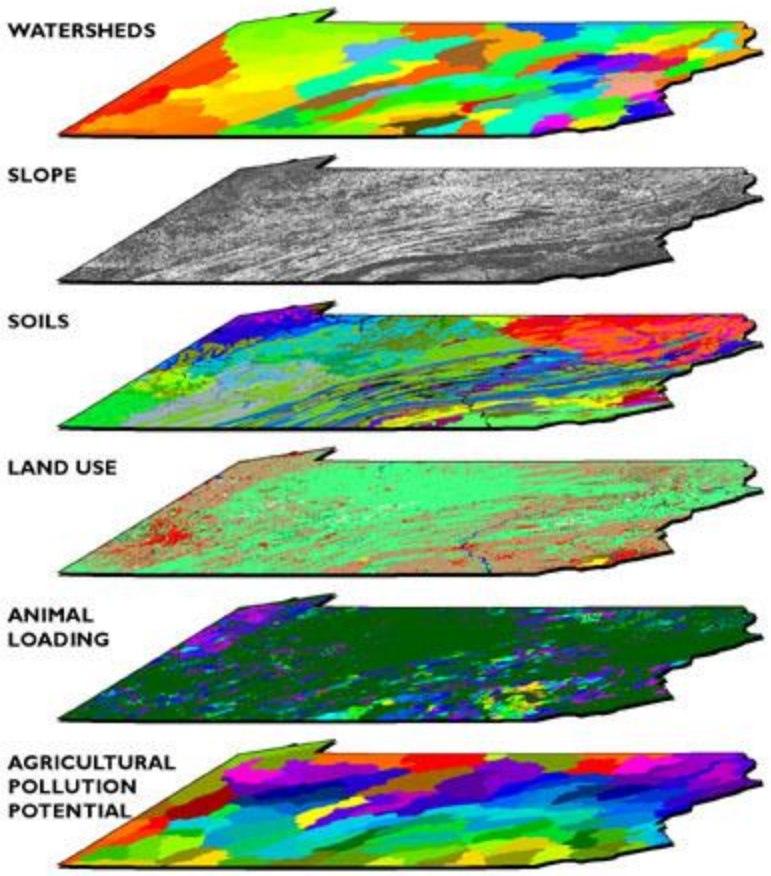
Line Segment Intersection

Pick-up-sticks



Thematic Map

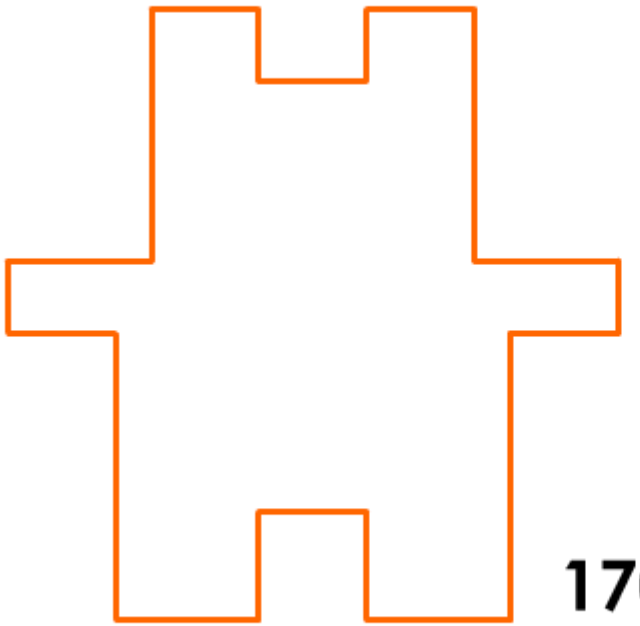
- A thematic map shows the spatial distribution of one or more specific data themes for standard geographic areas.
  - May be qualitative in nature (e.g., predominant farm types)
  - Or, quantitative (e.g., percentage population change).



Examples

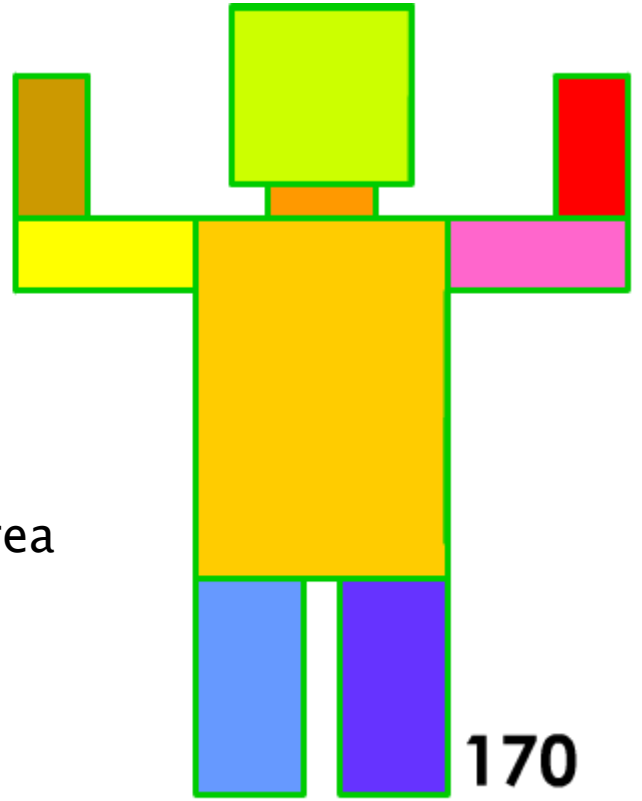
- GIS:
  - Only displaying the information within an area, e.g. a city or a county
  - Geographers study the intersection of pine forest and the annual precipitation is between 1000mm and 1500mm
- Graphic design
  - Intersection of two shapes
- Transformer

Initialization



170

Mesh Area

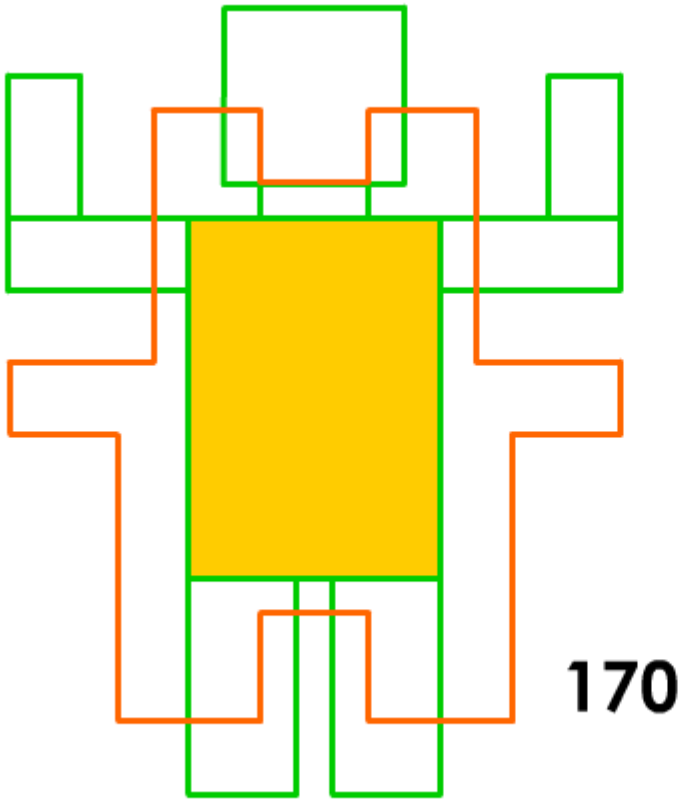


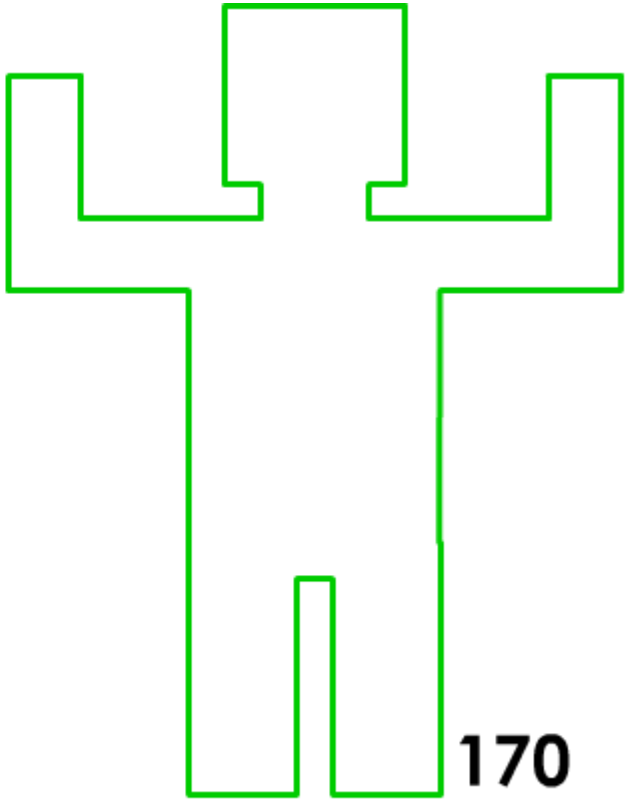
170

Target Mesh

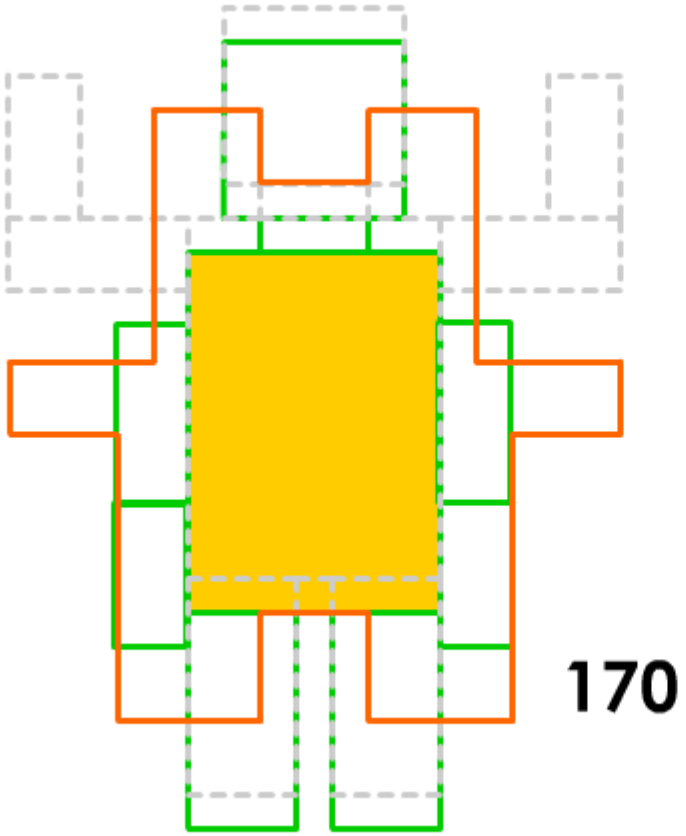
Source Mesh

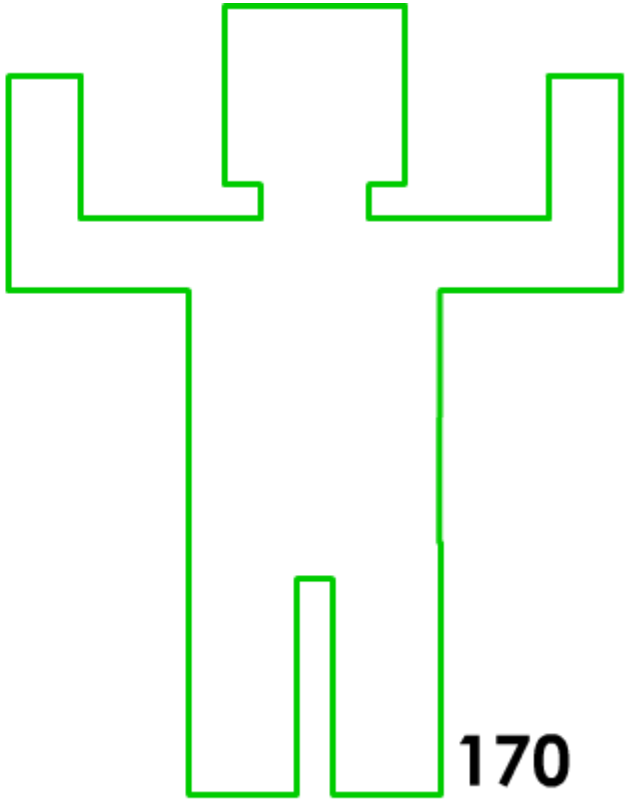
Initialization





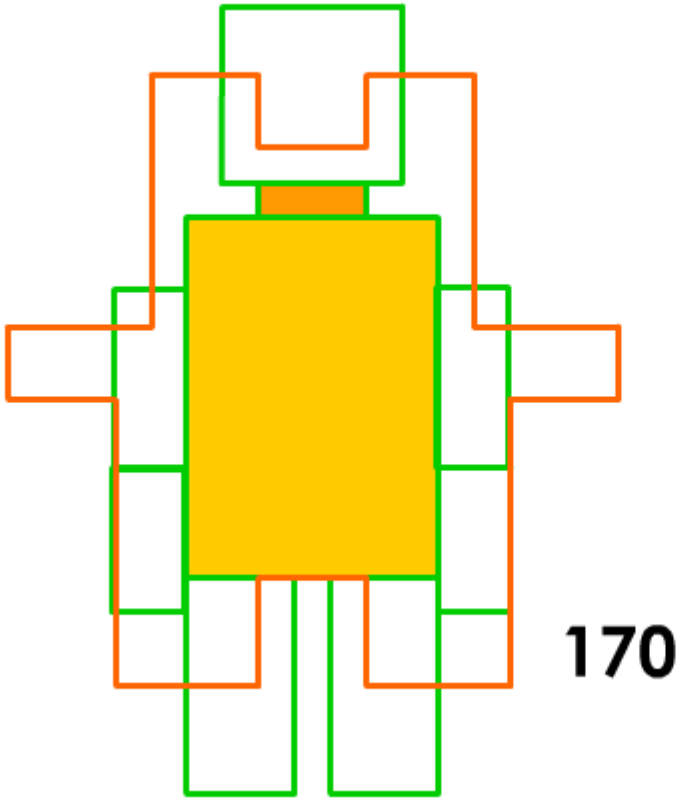
Rotate to Fit





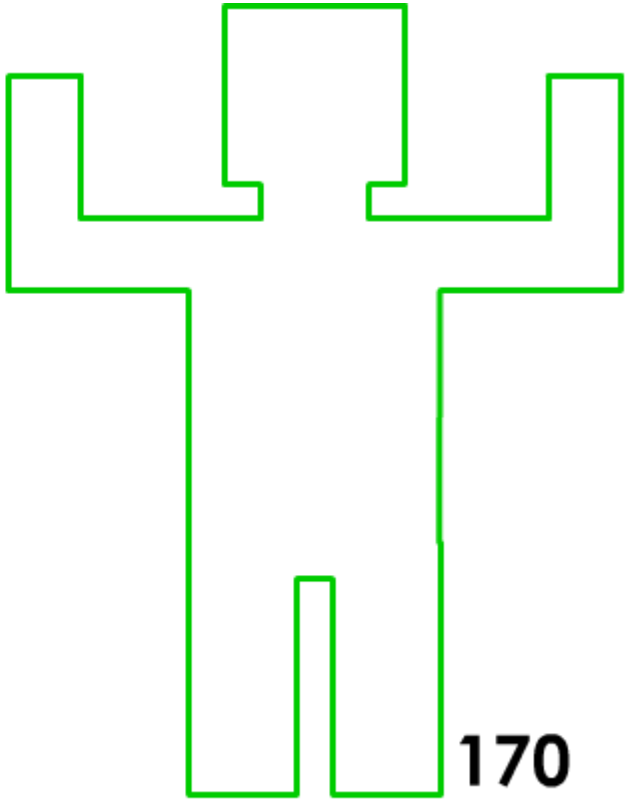
Attempt to fit source into target by rotating exist joints only

Neck



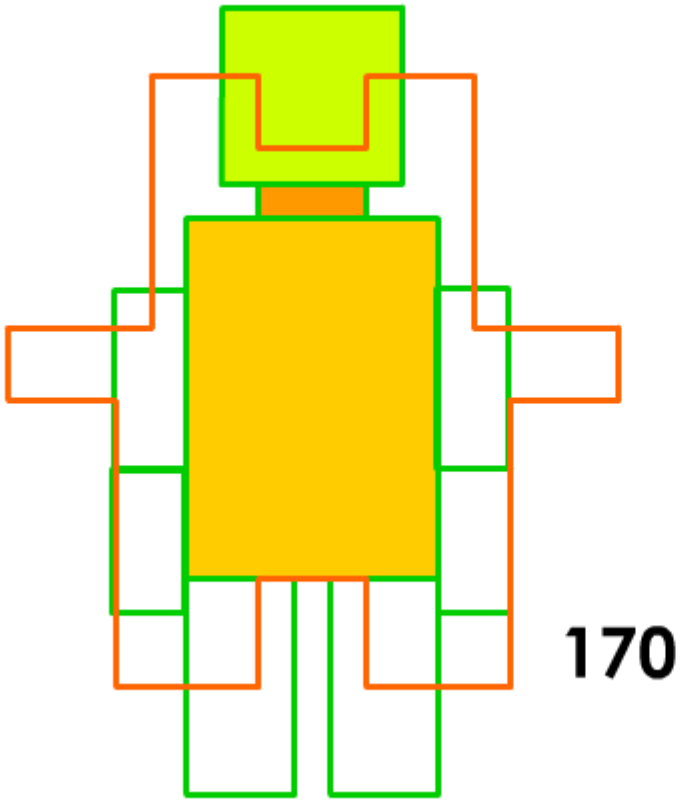
170

OK!

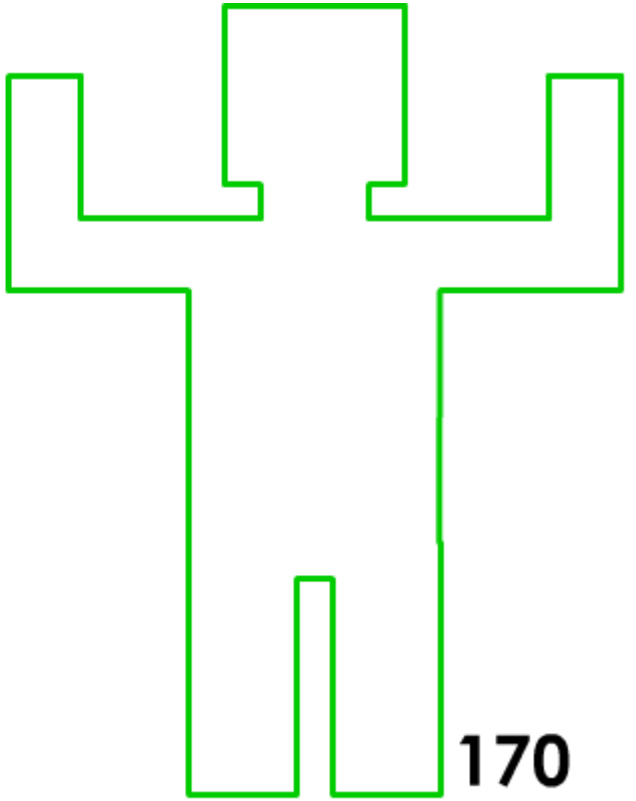


170

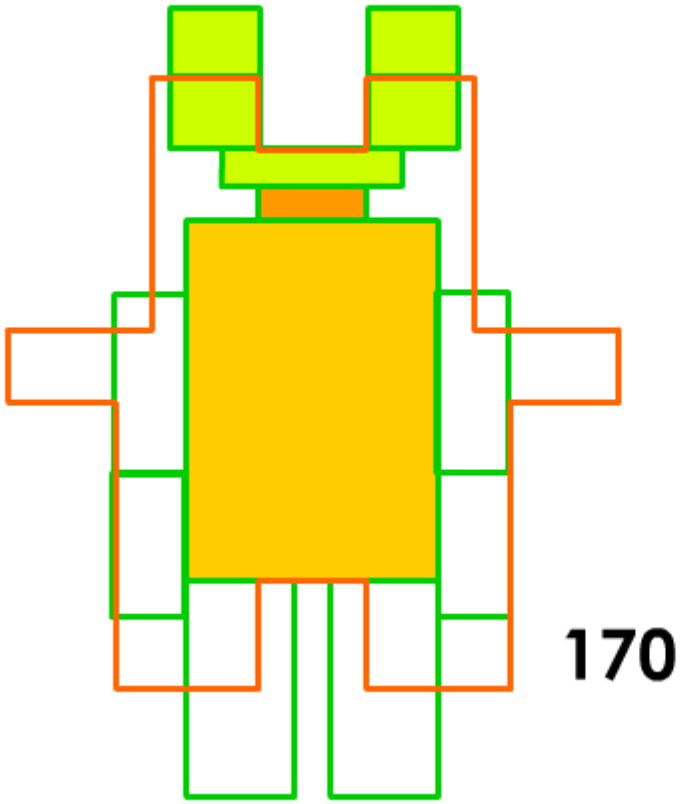
Head

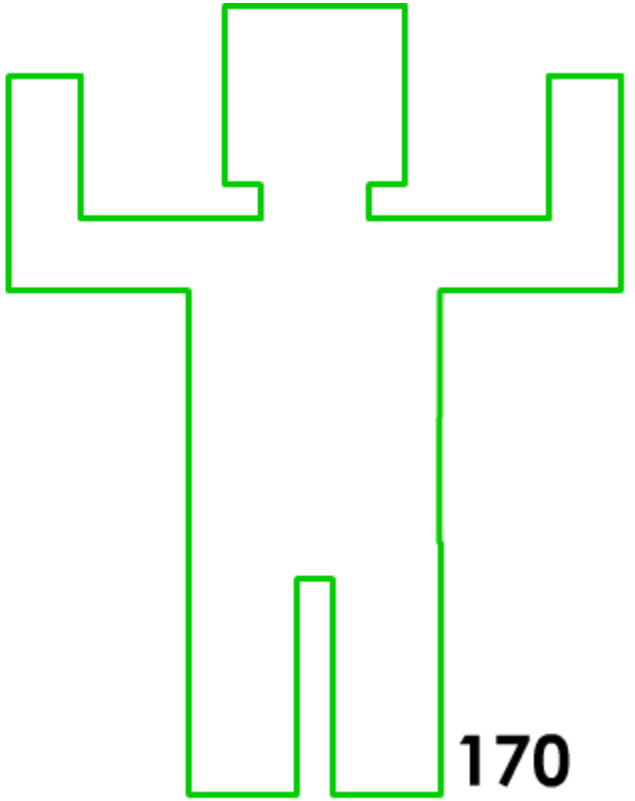


Protrusion Detected



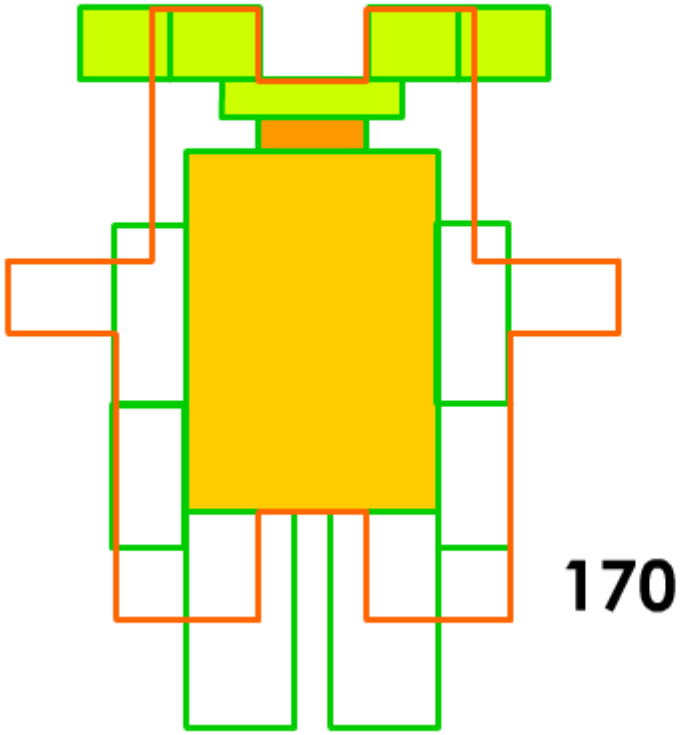
Head

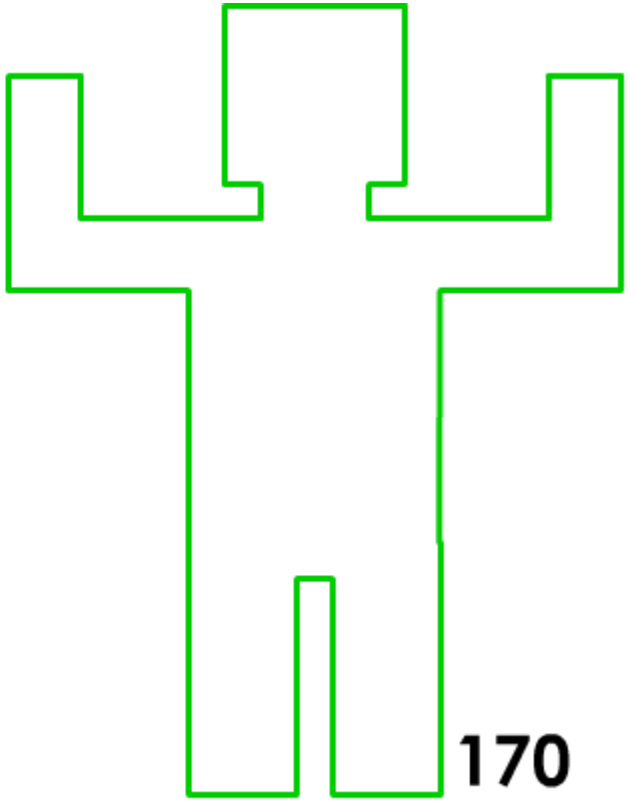




Split to attempt fitting

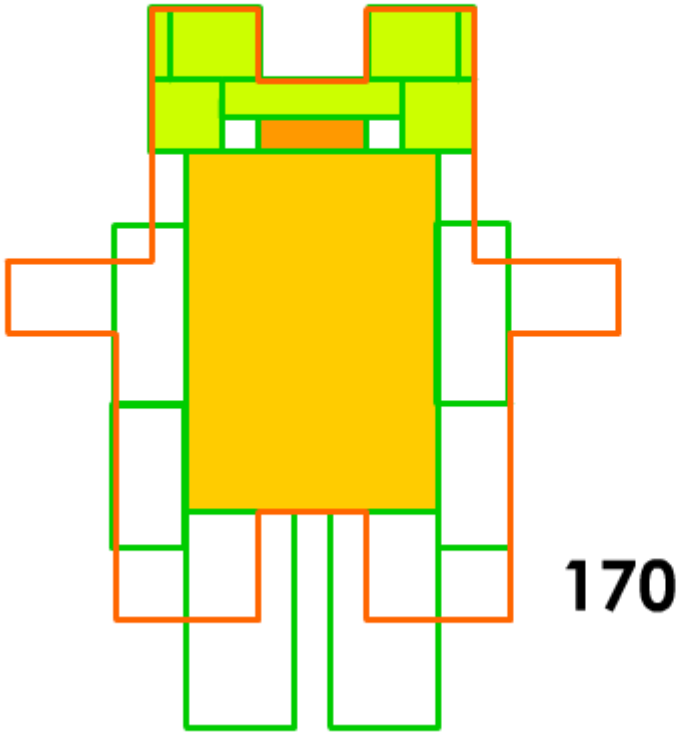
Head

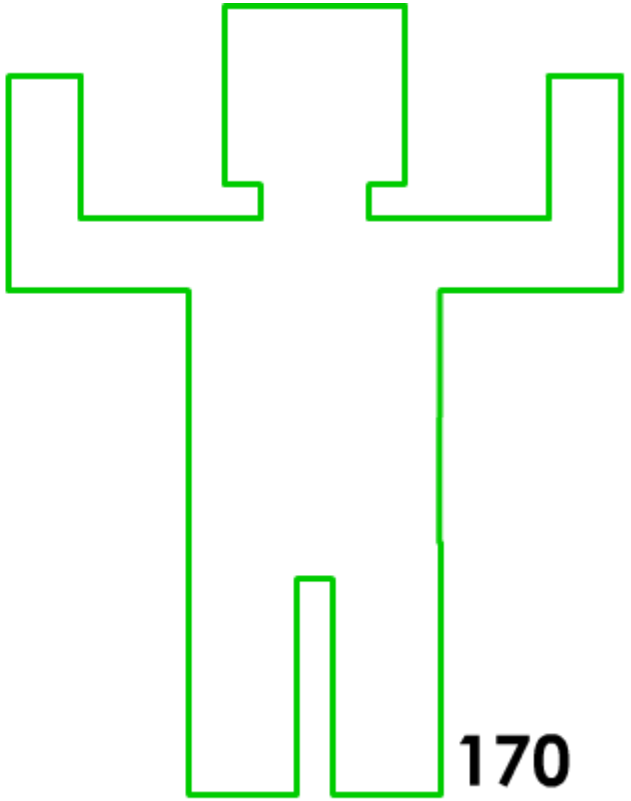




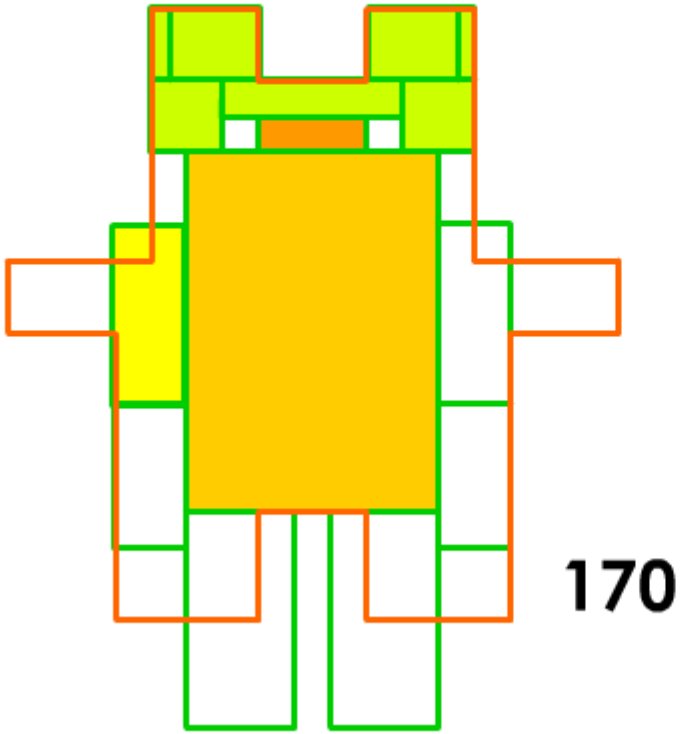
Perform until no protrusion or no other options possible

Head

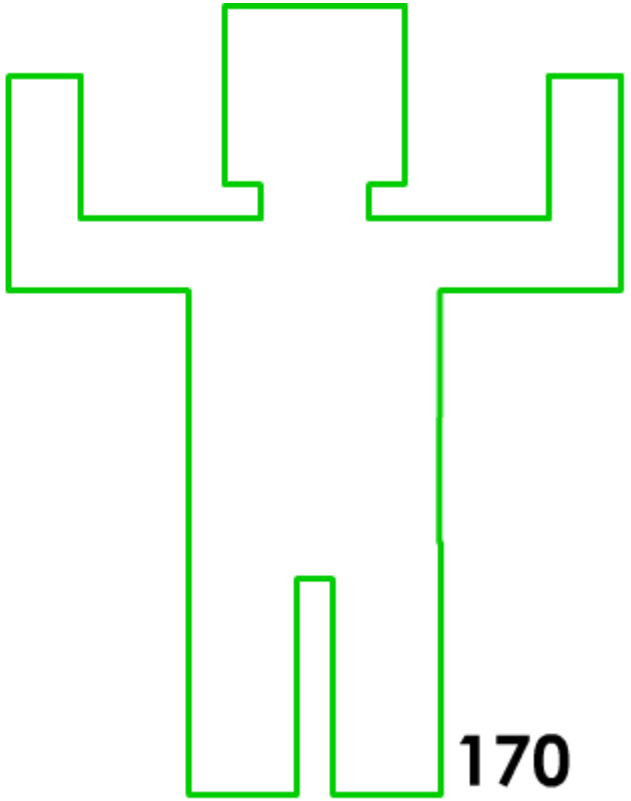




Left Arm

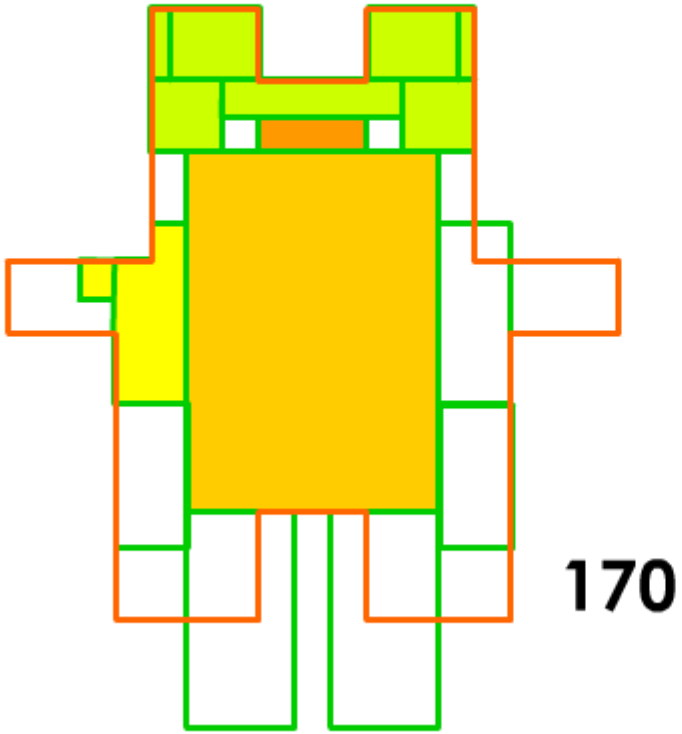


170

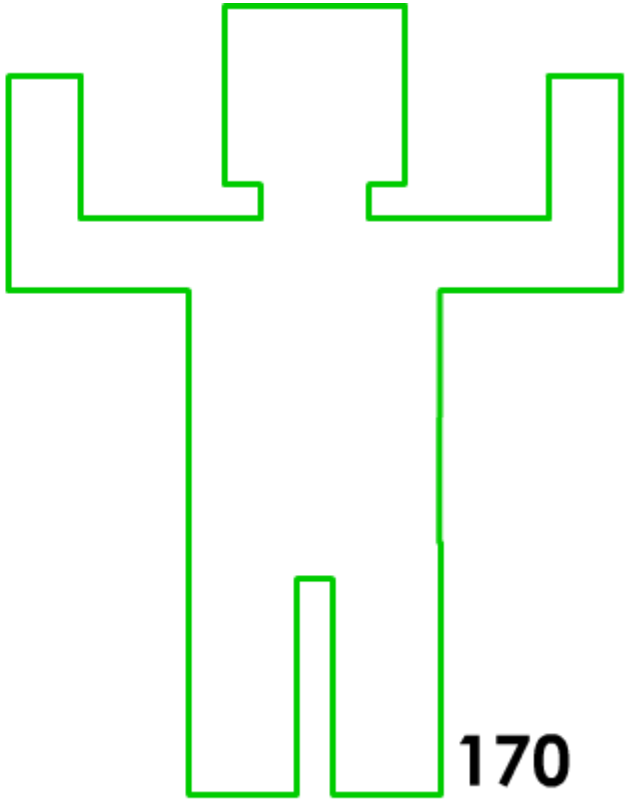


170

Left Arm

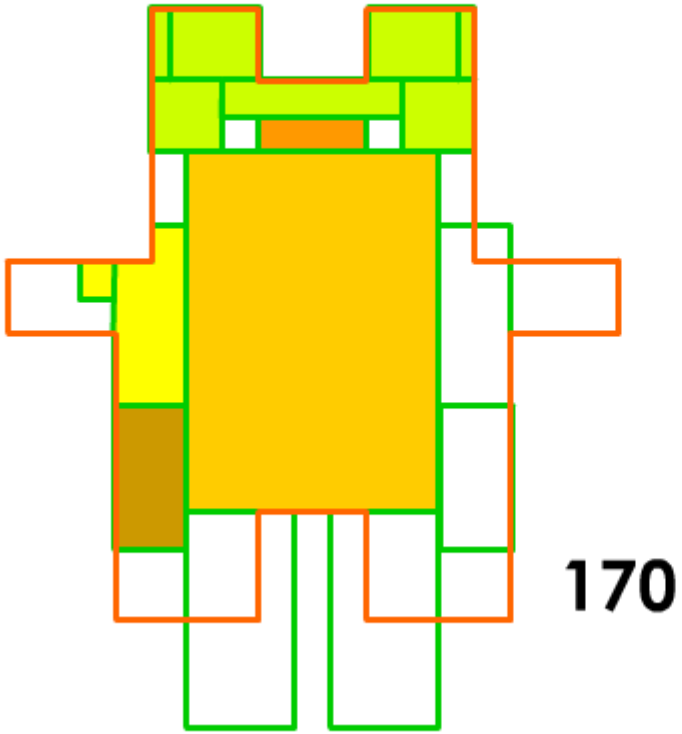


170

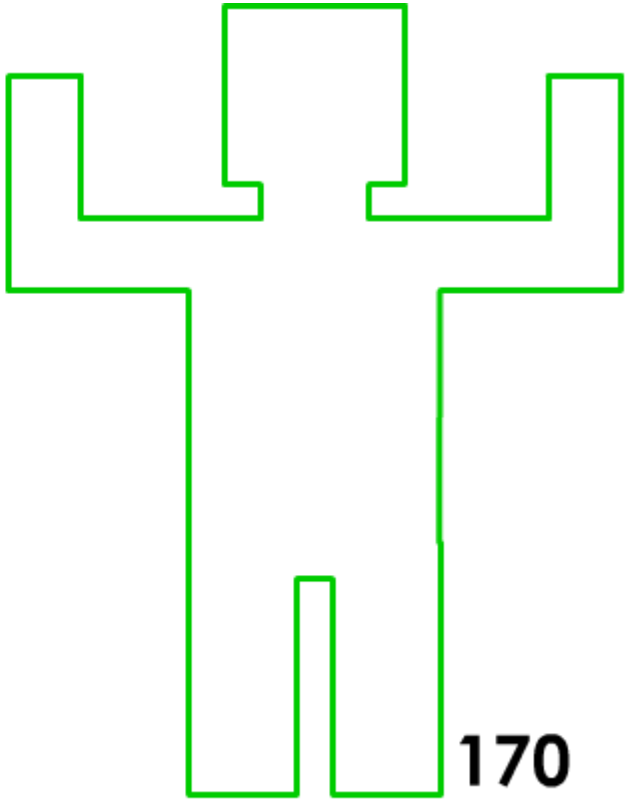


170

Left Arm

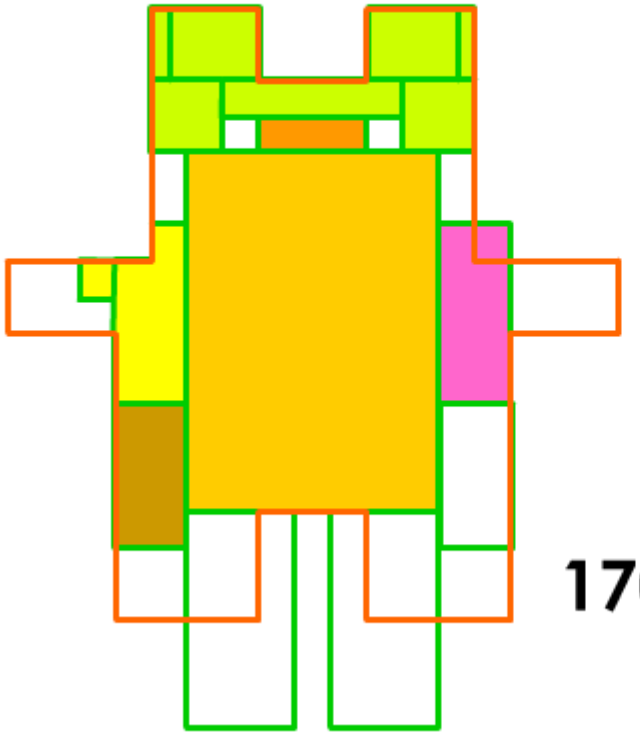


170

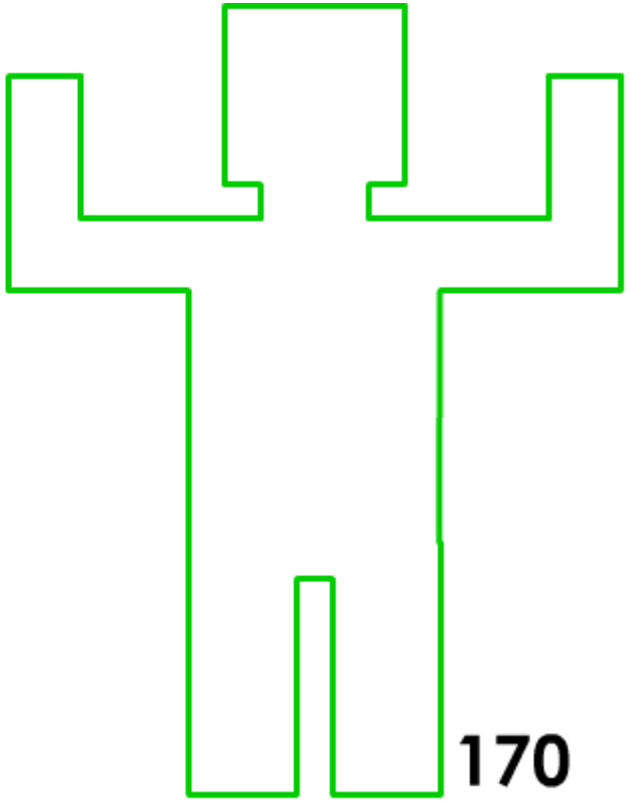


170

Right Arm

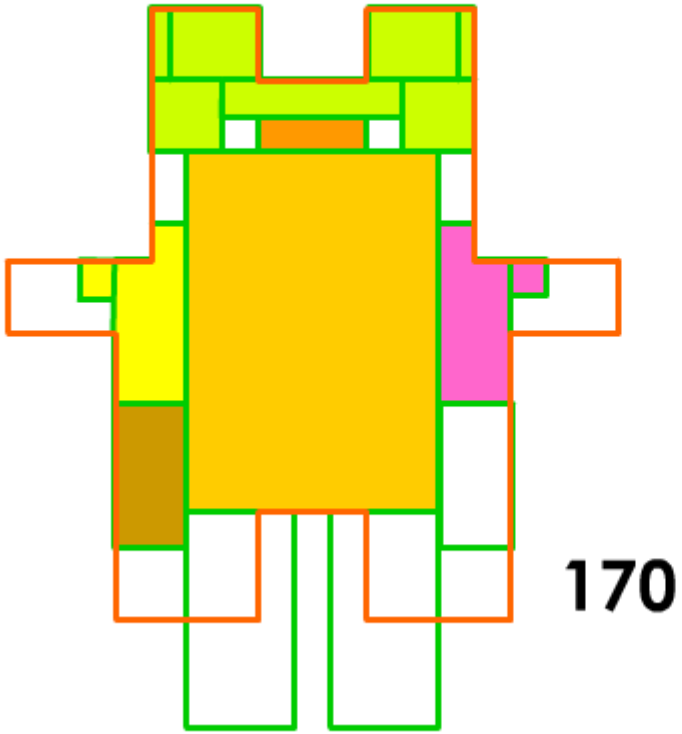


170

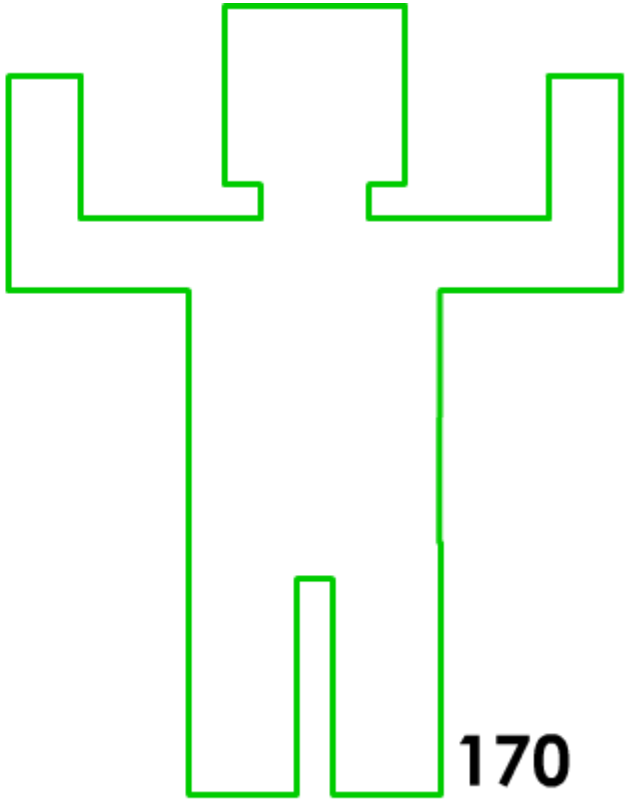


170

Right Arm

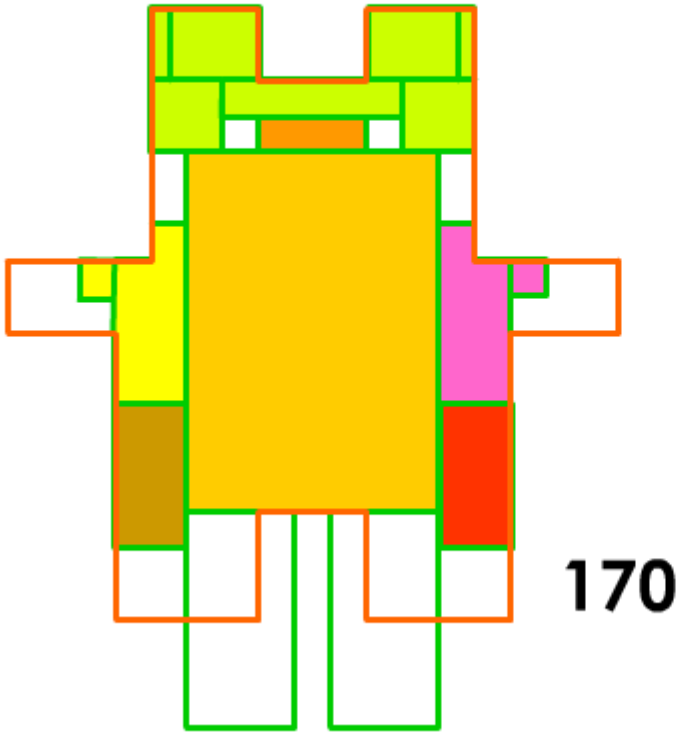


170

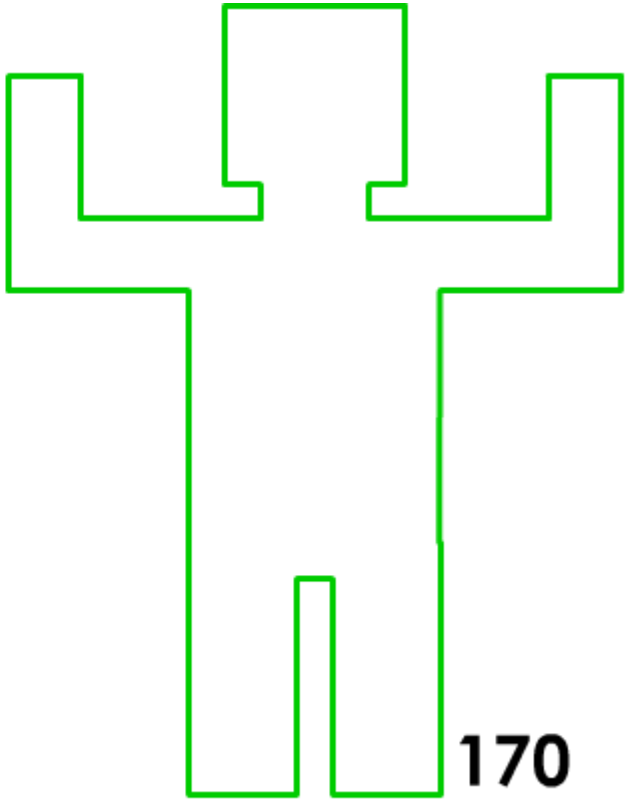


170

Right Arm

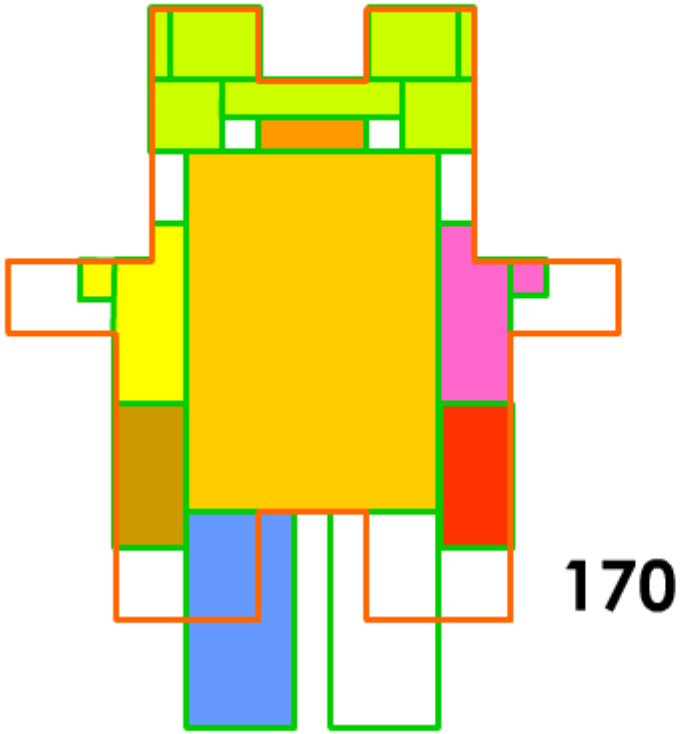


170

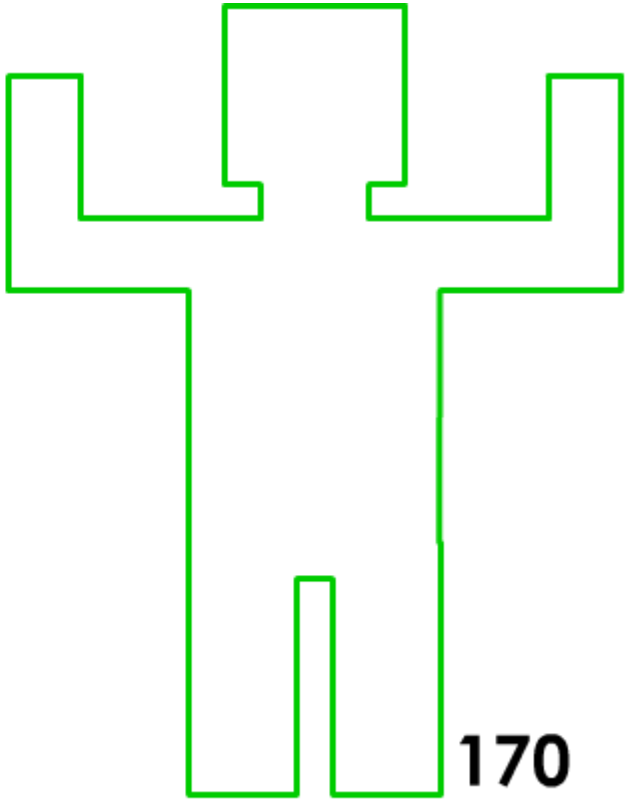


170

Left Leg

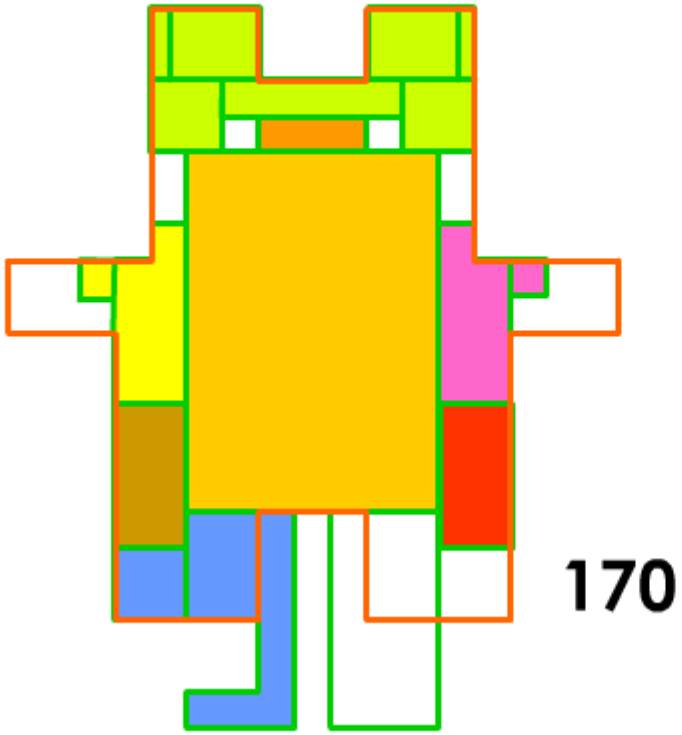


170

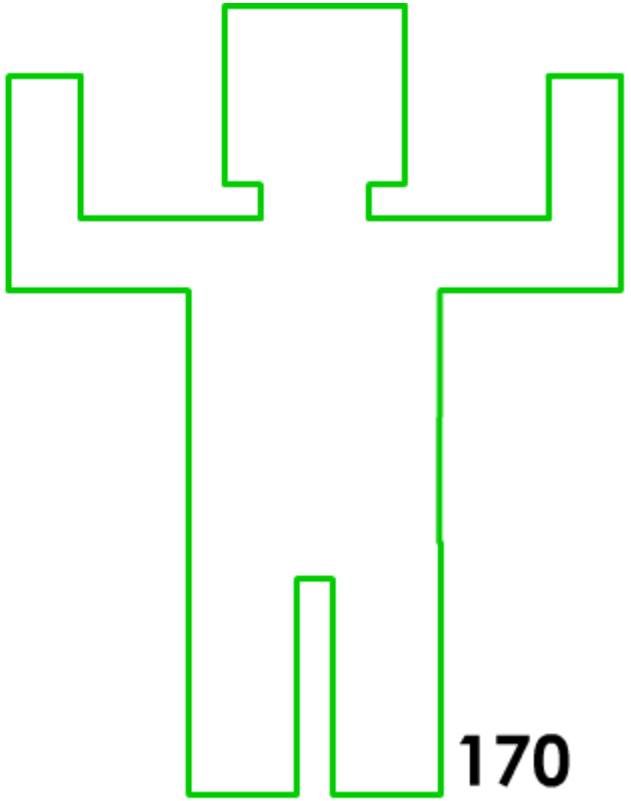


170

Left Leg

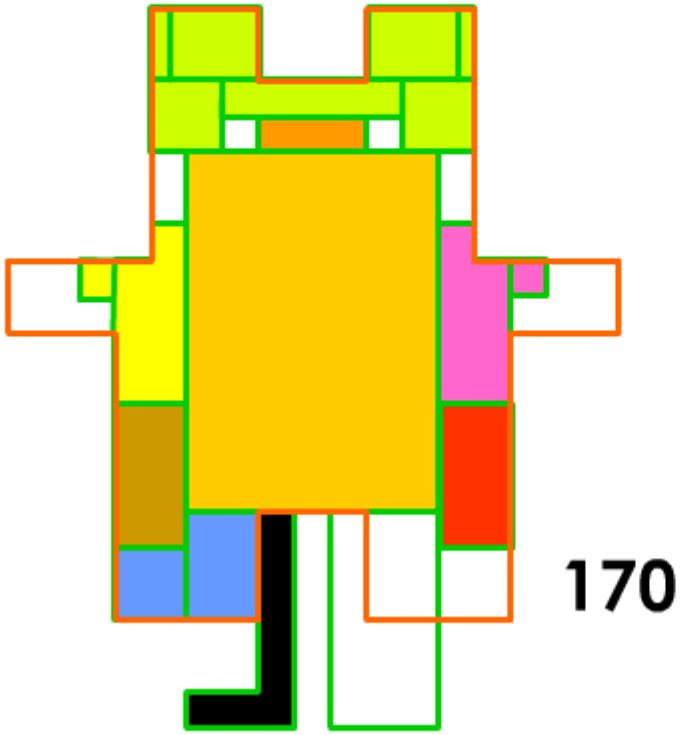


170

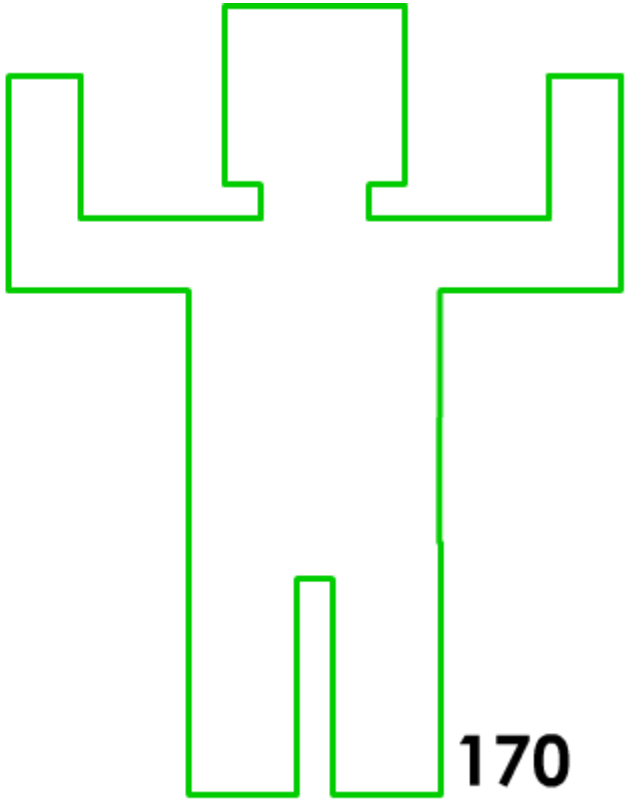


170

Left Leg

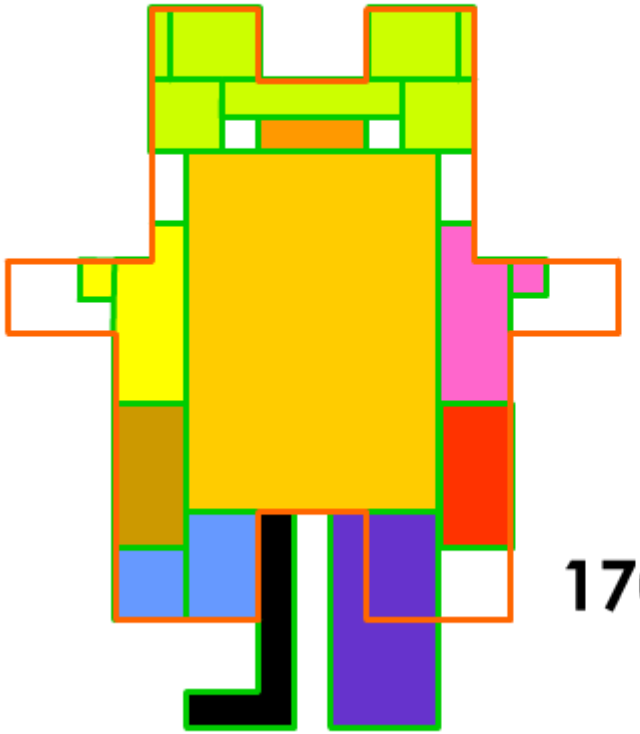


170

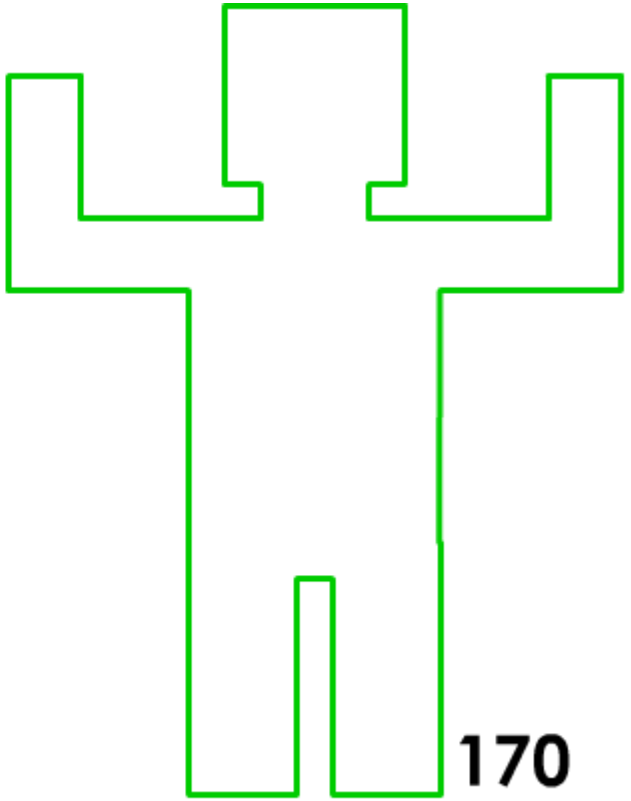


170

Right Leg

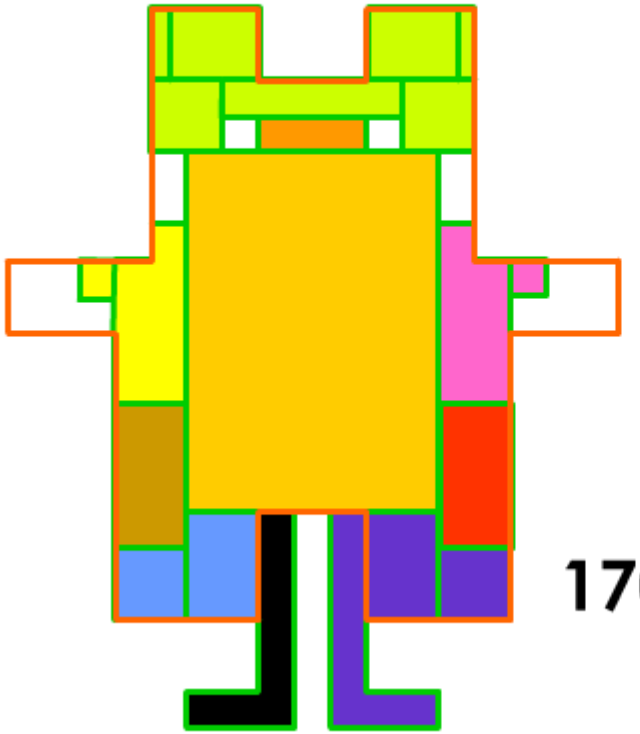


170

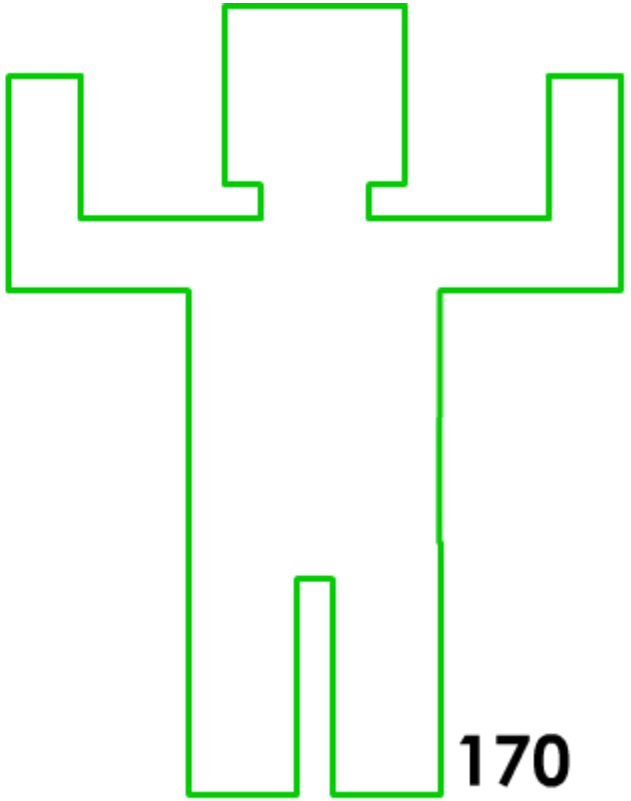


170

Right Leg

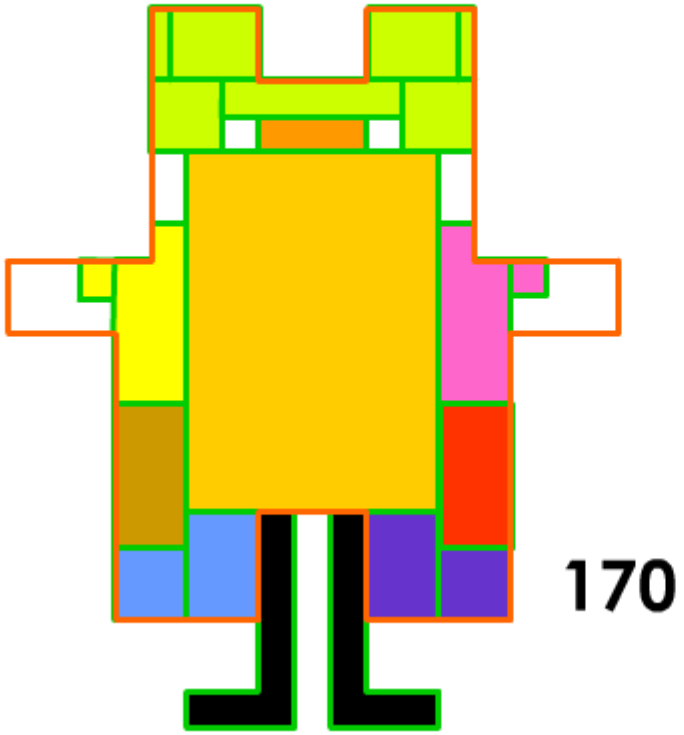


170

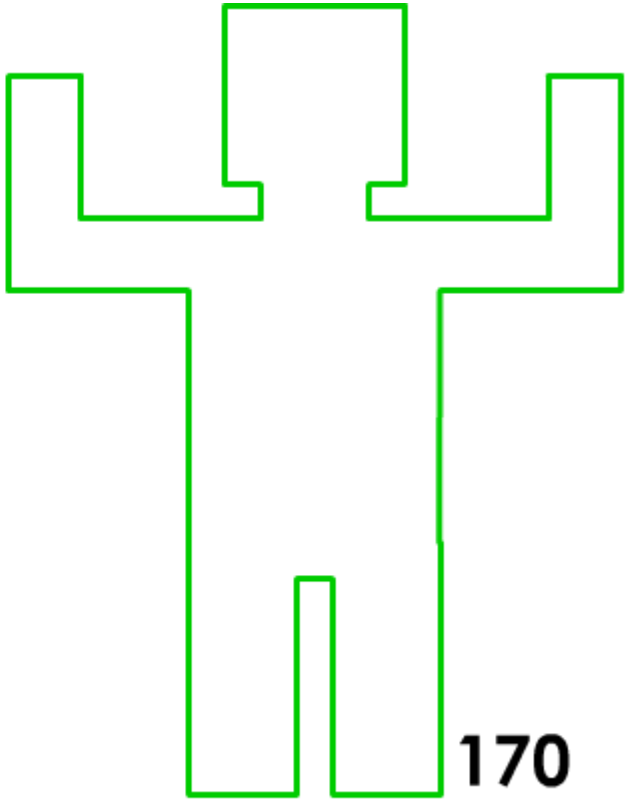


170

Right Leg



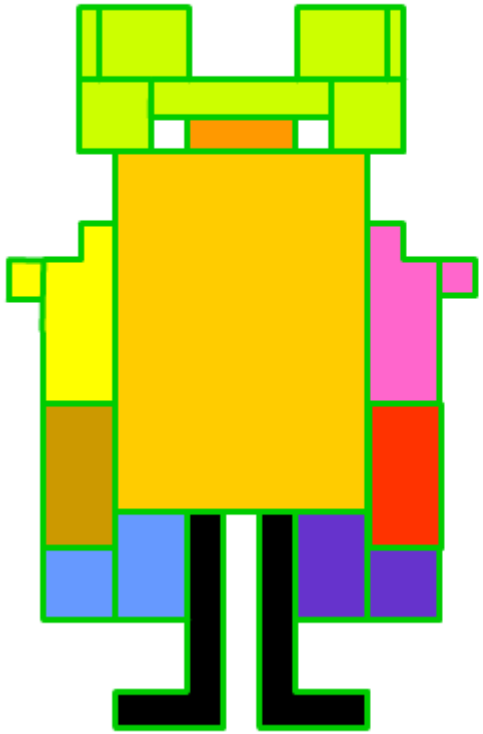
170

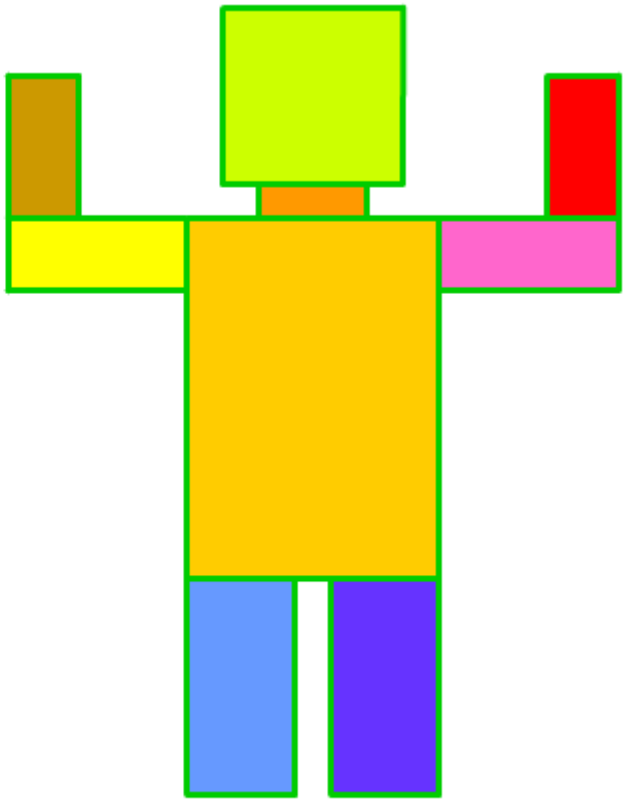


170

Cavities and protrusions still exist

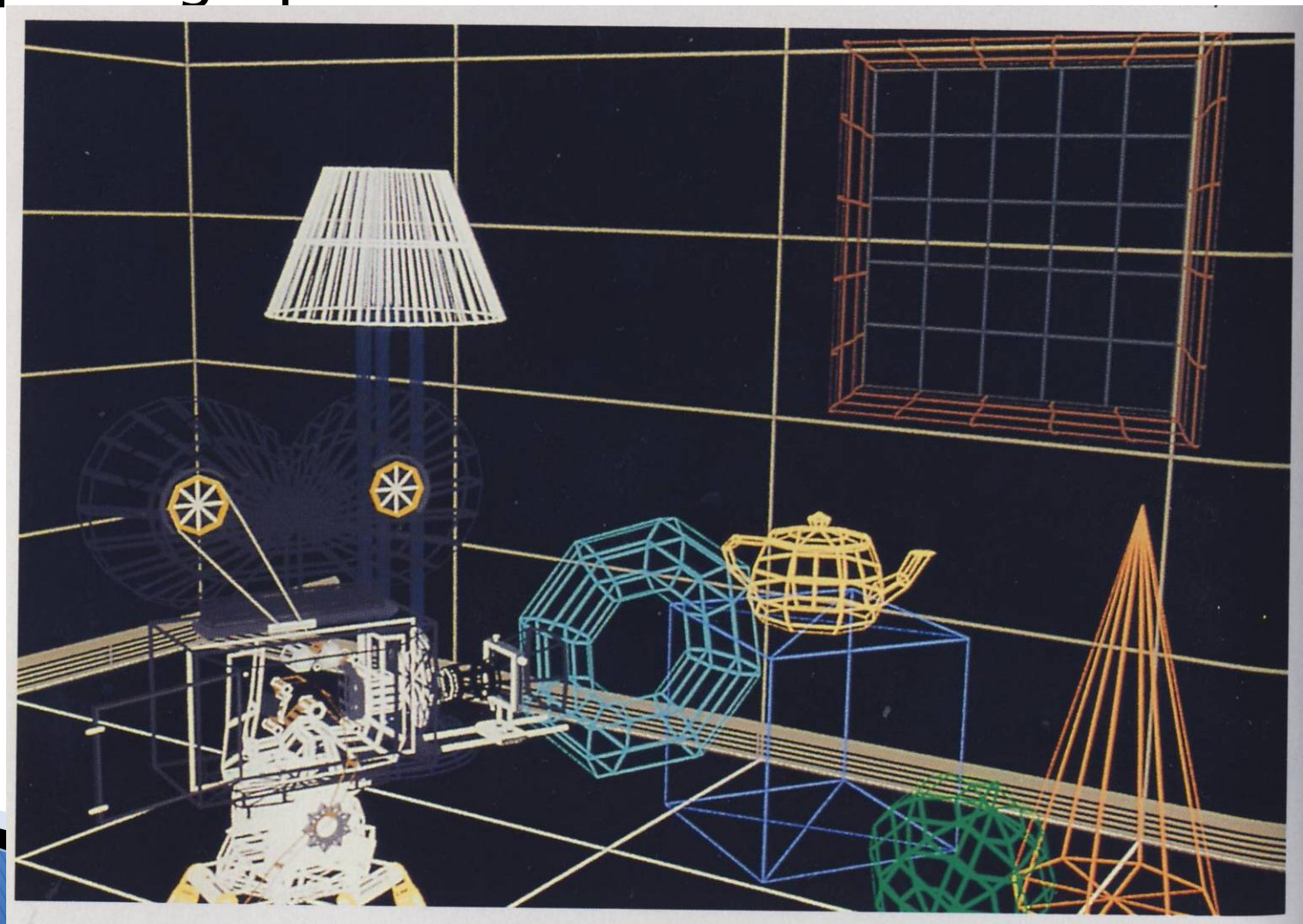
Before and After





Another Application: The Painter Algorithm

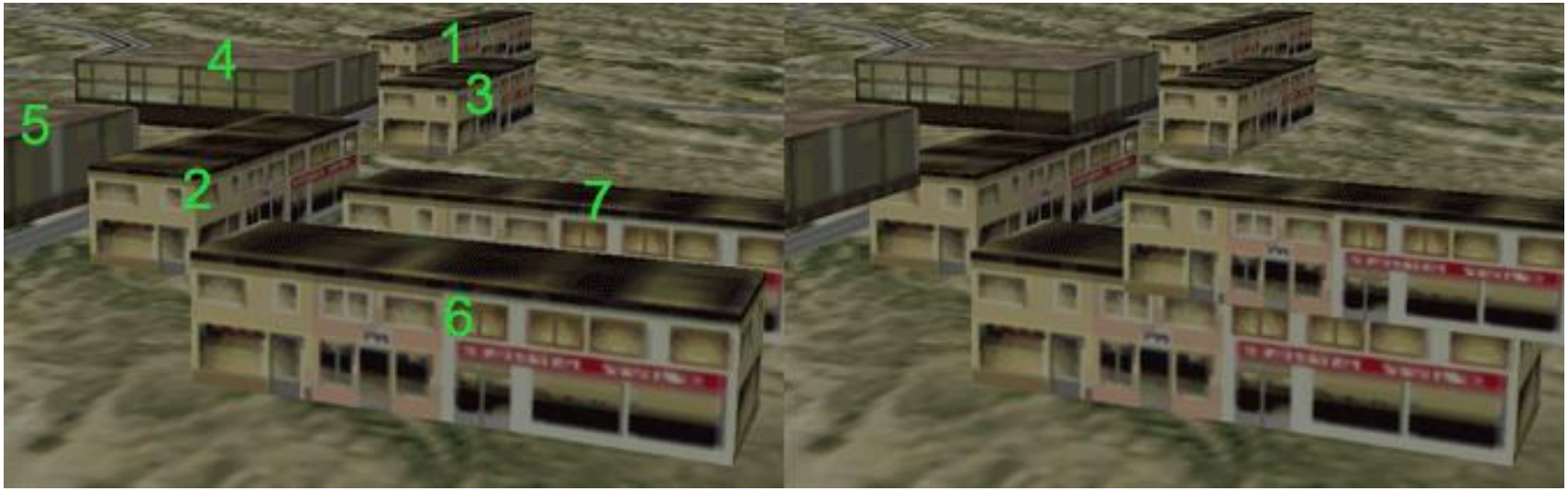
- In computer graphics



Drawing Order

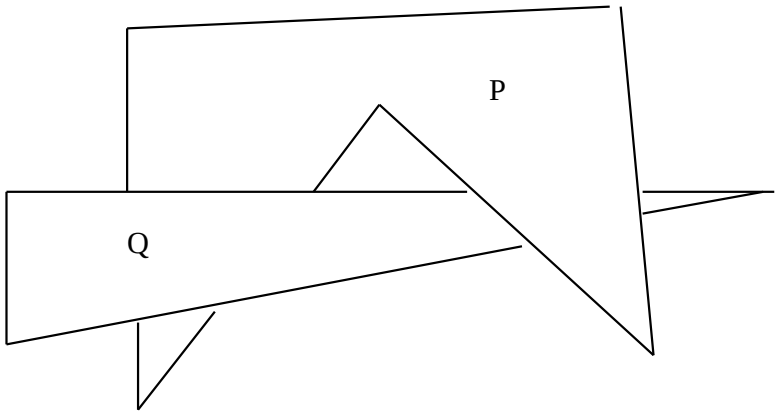
- From back to front (to the screen)

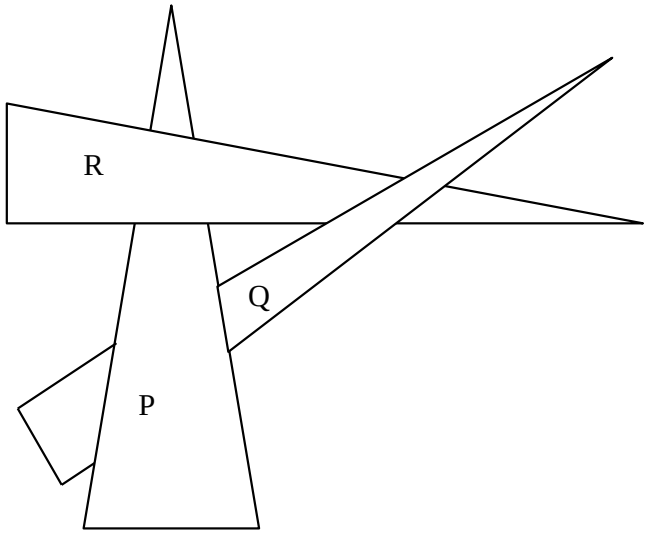
Wrong Order



However

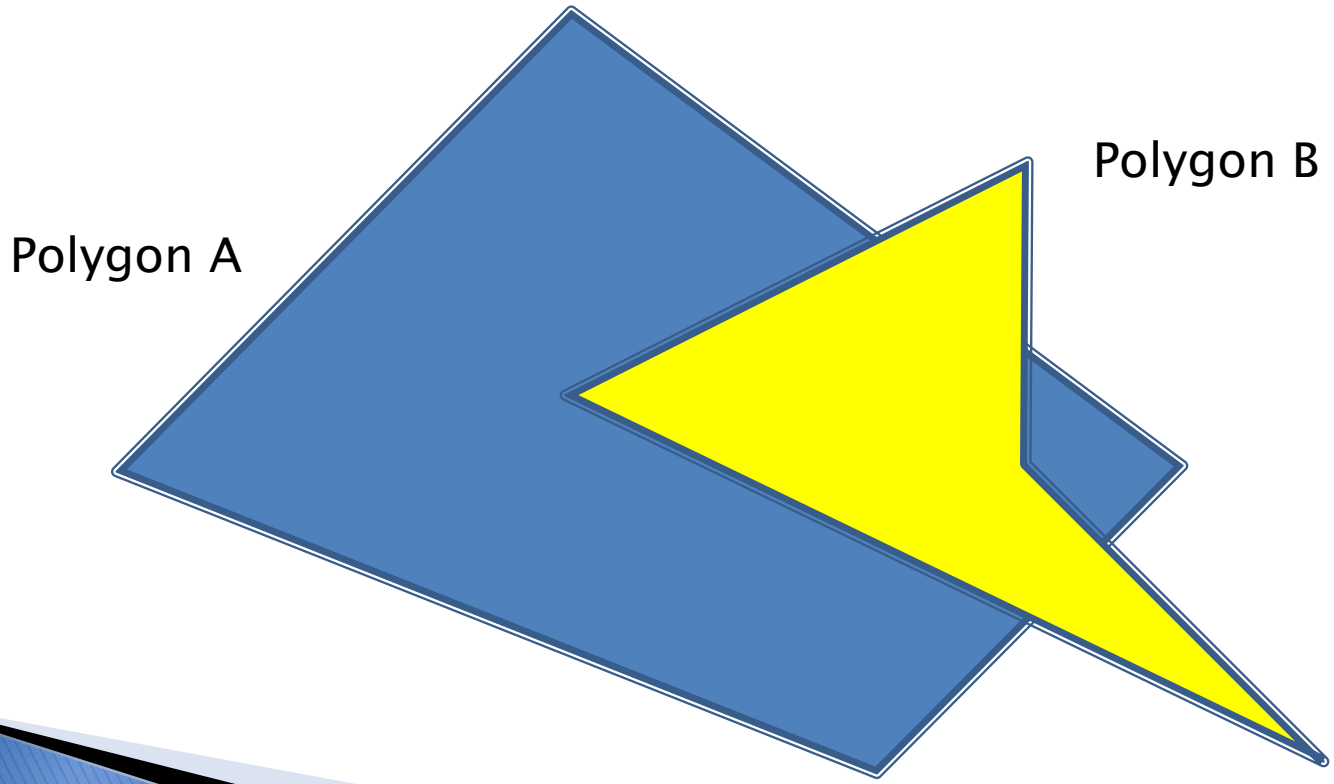
- Sometime, the drawing order does not exist





Hidden Surface Removal

- Depth Sort

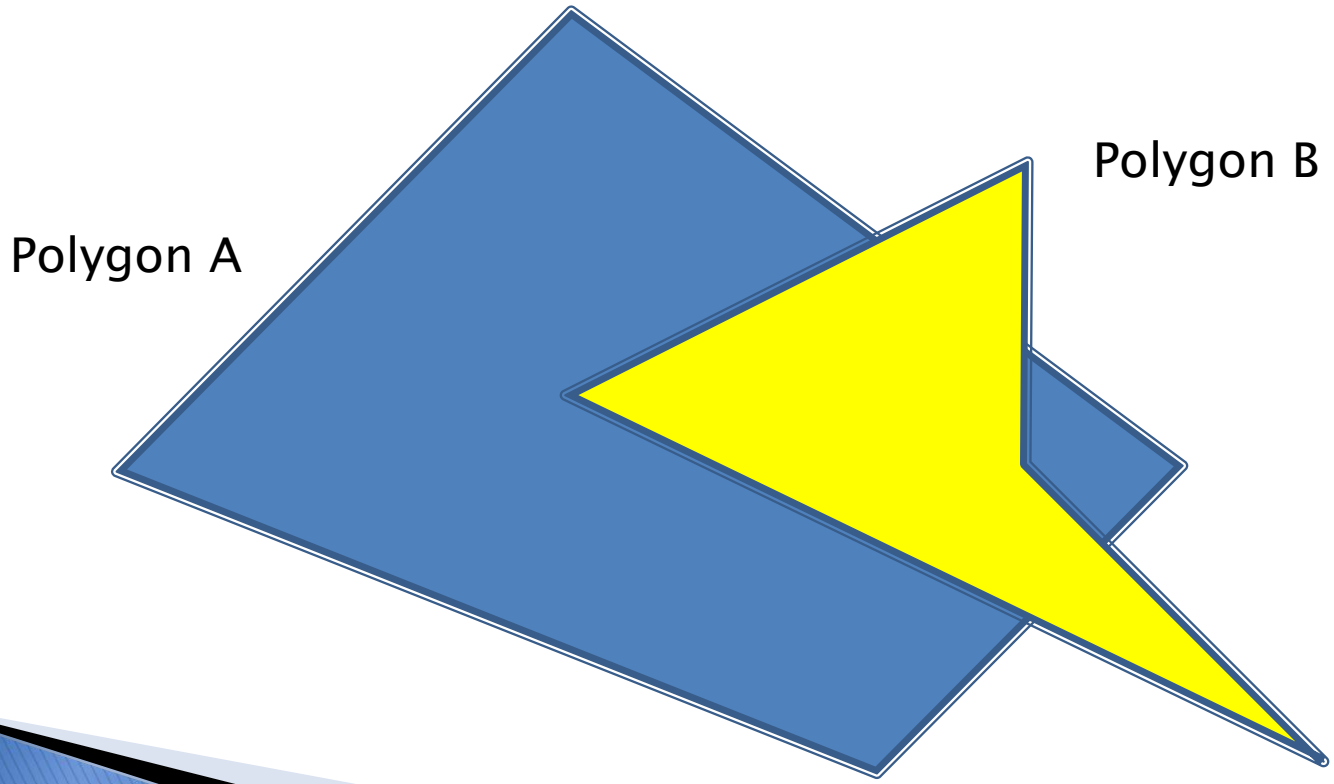


Depth Sort

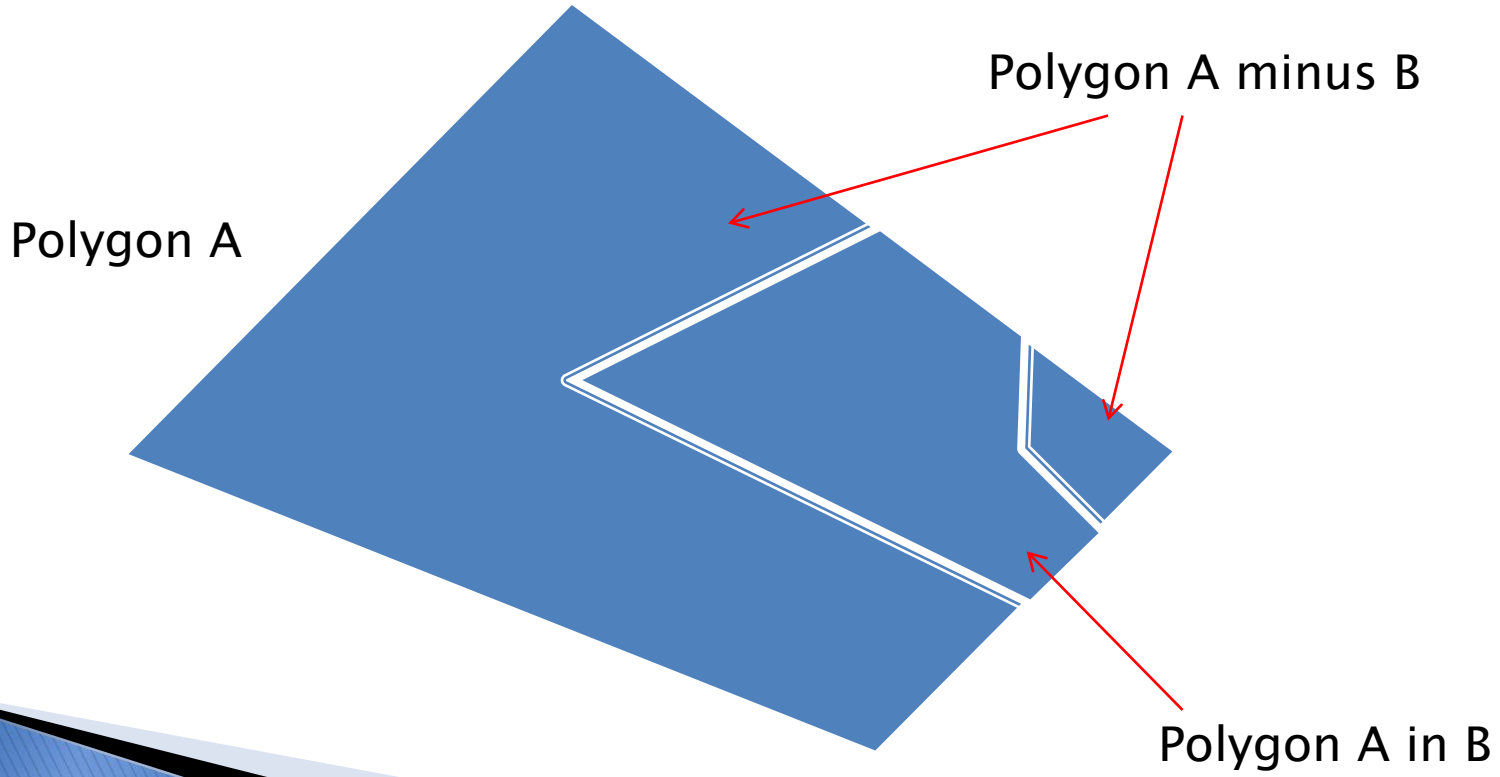
- For two polygons and B, we can compute
  - the union of A and B
  - the intersection of A and B
  - A minus B
  - B minus A

Hidden Surface Removal

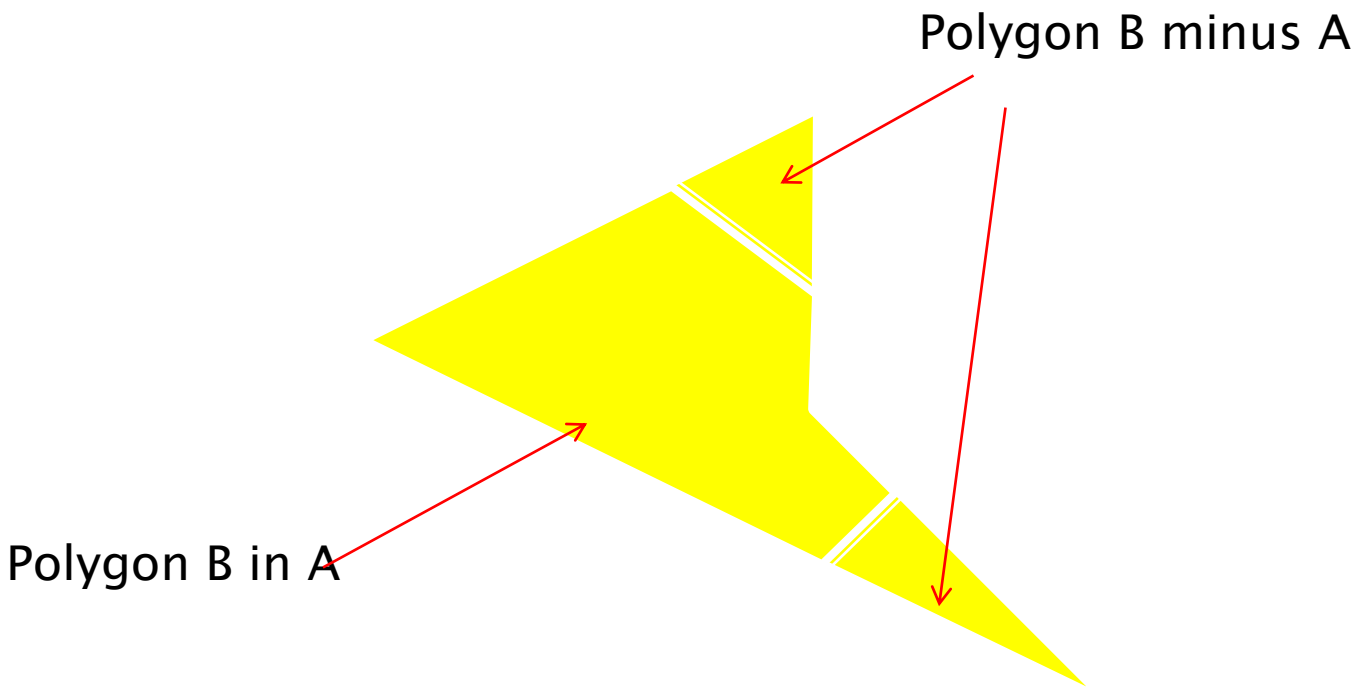
- Depth Sort



Depth Sort



Depth Sort



Type 1a: Depth Sorting

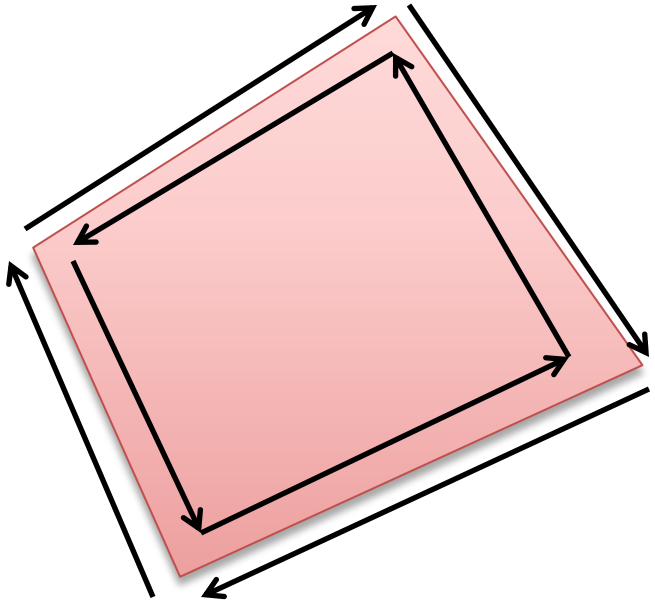
- Drawing order:
  - Determine which one to draw first
    - Polygon “A in B” or Polygon “B in A”
      - In this case, draw “A in B” first then draw “B in A” if polygon B is transparent
      - If B is not transparent, we can ignore or skip drawing “A in B” directly
  - Do not need to care about “A-B” or “B-A”
    - If they do not intersect other polygons in the screen

Boolean Operation

- A−B, B−A, A∩B, A∪B are called *the Boolean operations* of two polygons
  - Weiler-Atherton Algorithm
    - Computing intersections of polygons
    - A generic polygon clipping algorithm

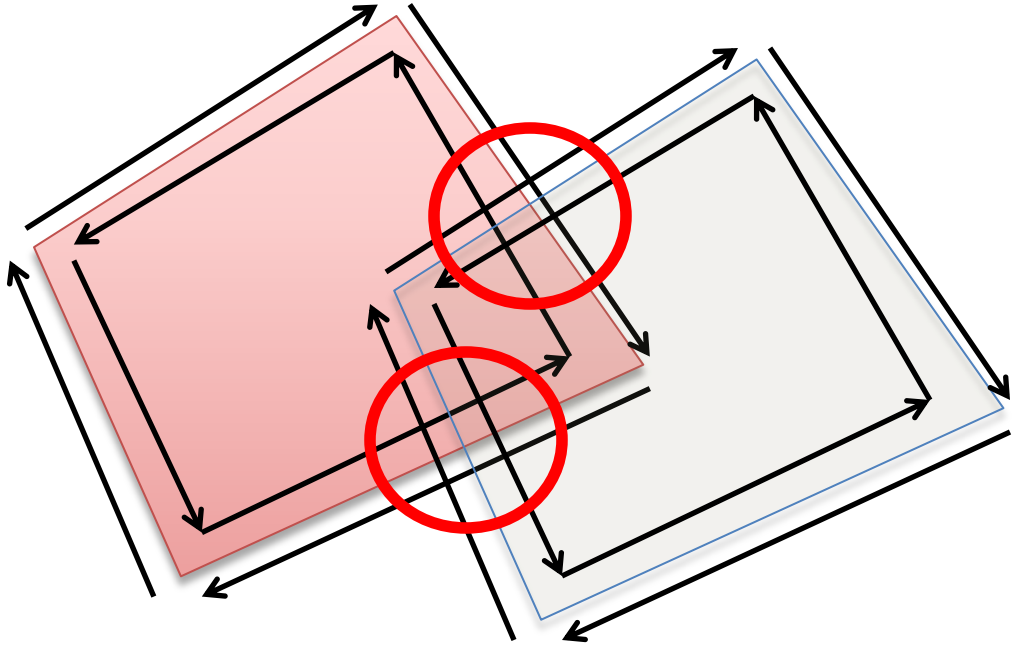
Weiler-Atherton Algorithm

- First, represent any polygon by a double-connected edge list
- Simply



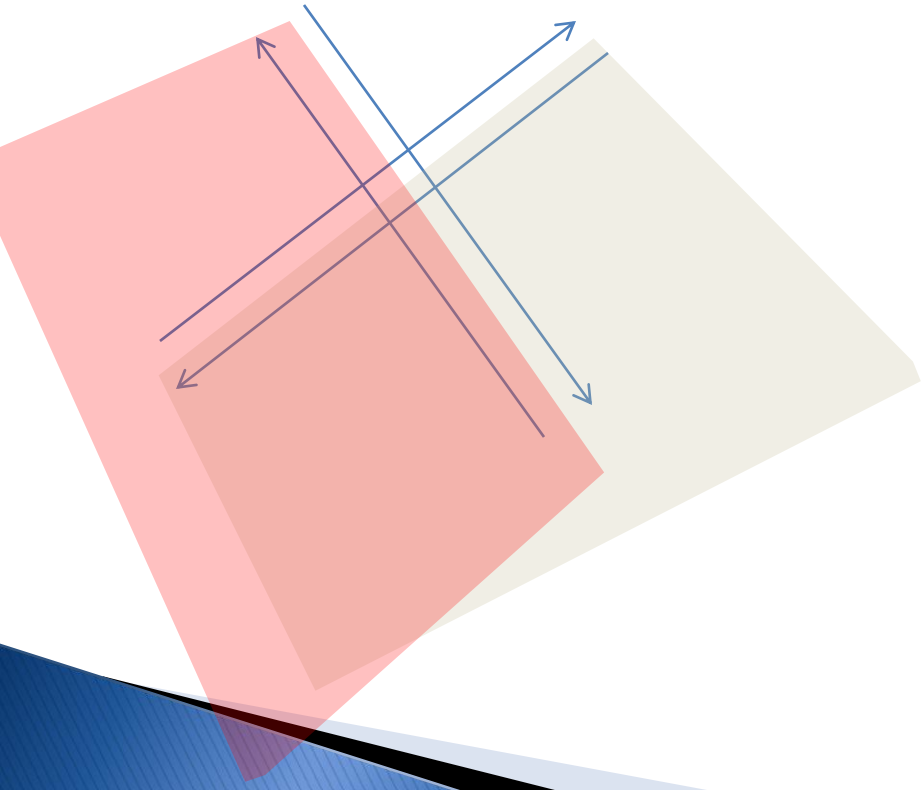
Weiler-Atherton Algorithm

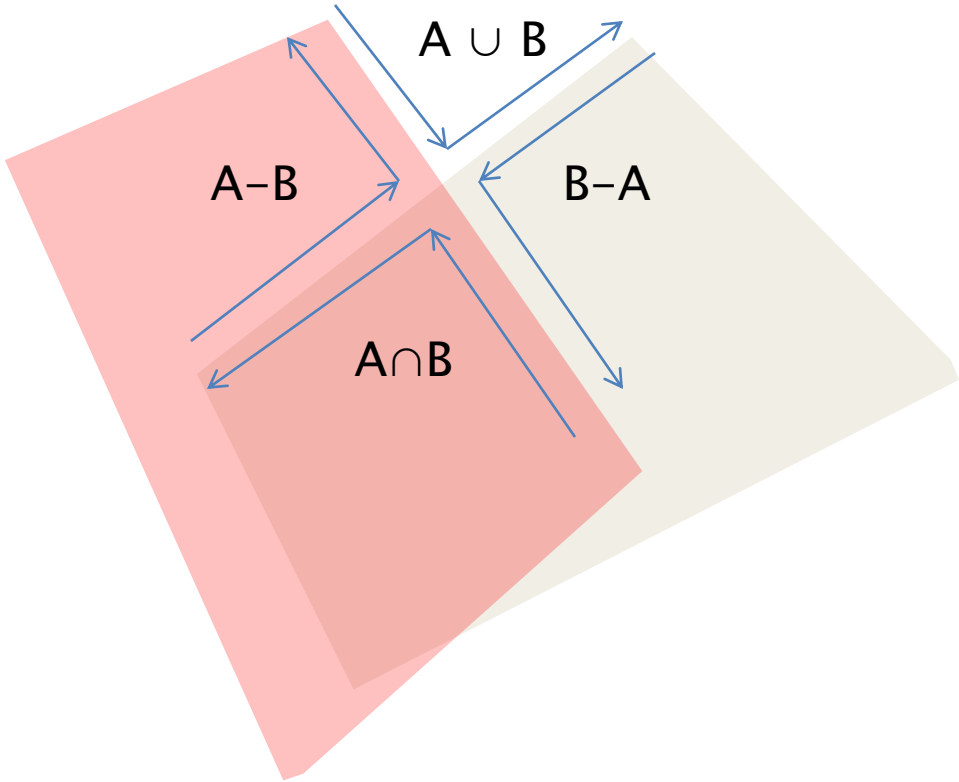
- Then, use line segment intersection to compute all the intersections



Weiler-Atherton Algorithm

- For each intersection, simple reroute the edge list

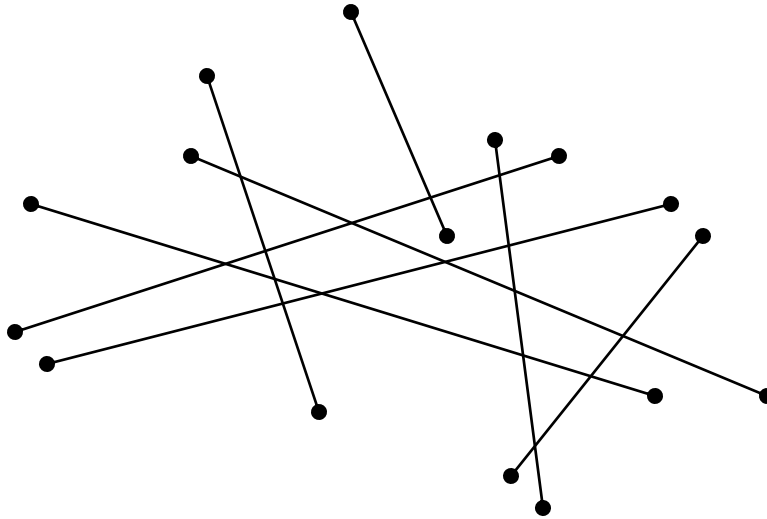




Line Intersection Algorithm

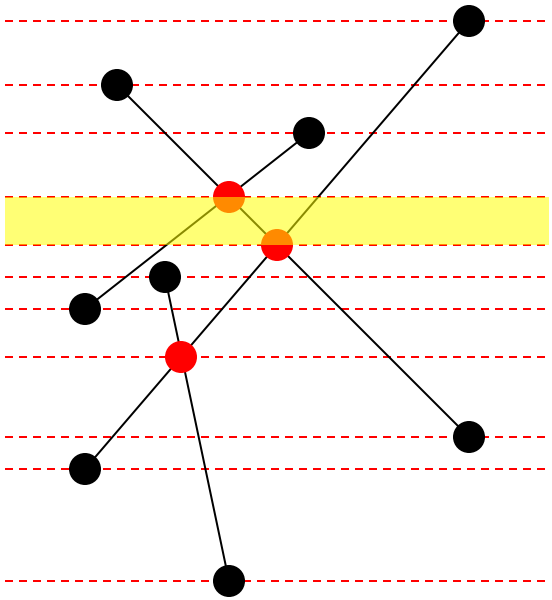
Line Segment Intersection

- Given a finite number of (finite) line segments
- Compute all of their intersections
- Naïve solution
  - Compare every pair
  - ${}_nC_2 = \Theta(n^2)$
- However the number of intersection could be exactly ${}_nC_2$
  - Or Could be less
- If it has less intersection
  - Can it run faster?



Plane Sweep Algorithm

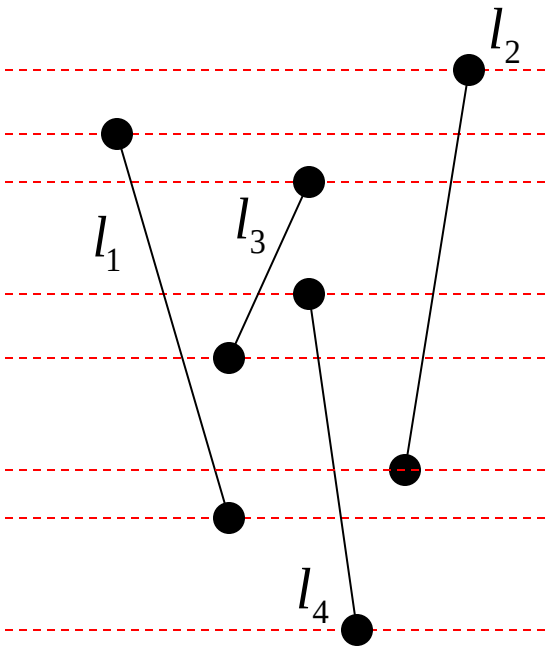
- A plane is swept through a direction
  - E.g. from the top to the bottom
- Notice the left to right sequence of segments S remain the same within two consecutive points (in the sweeping direction)
  - Black plus red
- However, we only know the black, but not the red



For this time being, let's assume that there is no two points with the same y coordinates

Events (Assuming No Intersection)

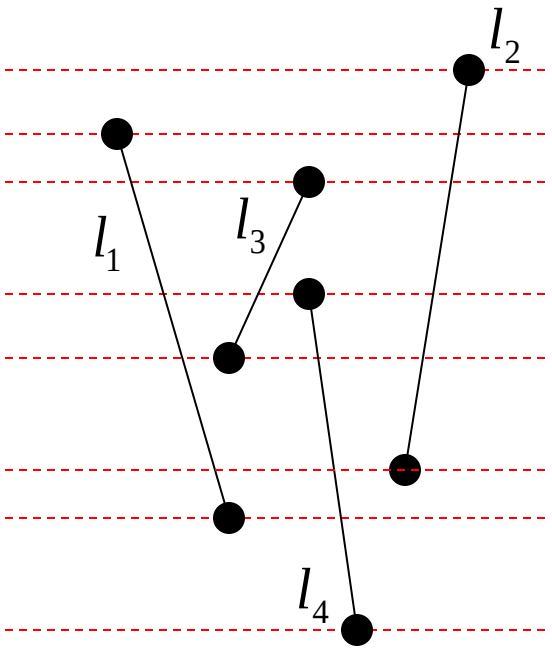
- Suppose there is NO intersection
- Put all the vertices into a priority queue Q, according to the y coordinate
- Each time, dequeue the vertex with the highest y value in the queue
  - If it is a upper vertex of a segment, it will be inserted into the horizontal order
  - If it is a lower vertex, it will be deleted from the order



Events (Assuming No Intersection)

- The sequence S

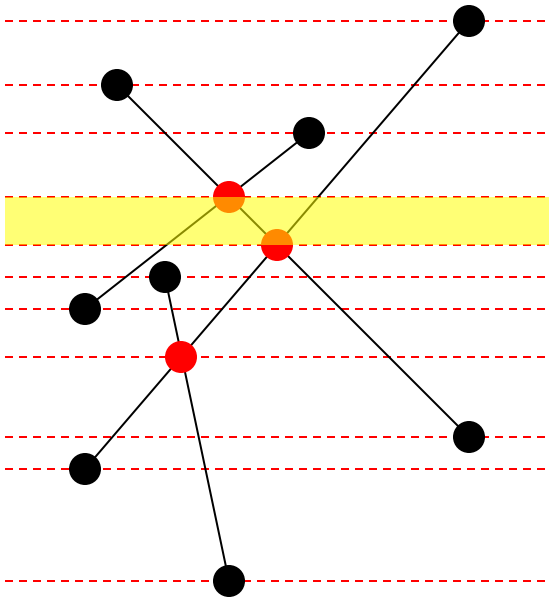
1. $\emptyset$
2. (l_2)
3. (l_1, l_2)
4. (l_1, l_3, l_2)
5. (l_1, l_3, l_4, l_2)
6. $(l_1, l_3, l_4, l_2) = (l_1, l_4, l_2)$
7. $(l_1, l_4, l_2) = (l_1, l_4)$
8. $(l_4, l_2) = (l_4)$
9. $(l_4) = \emptyset$



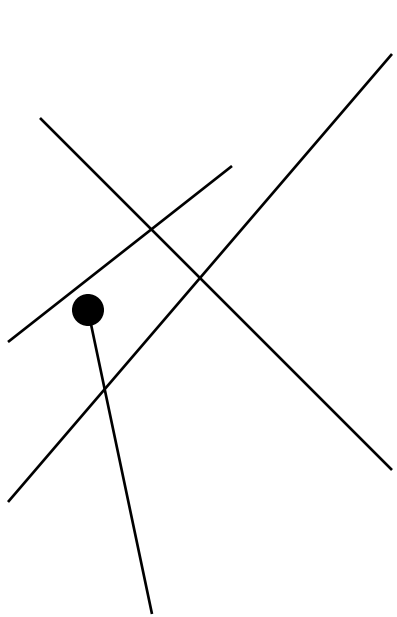
Each insertion or deletion of a segment is handled by a balanced binary search tree, i.e. O(log n)

Detecting Intersection

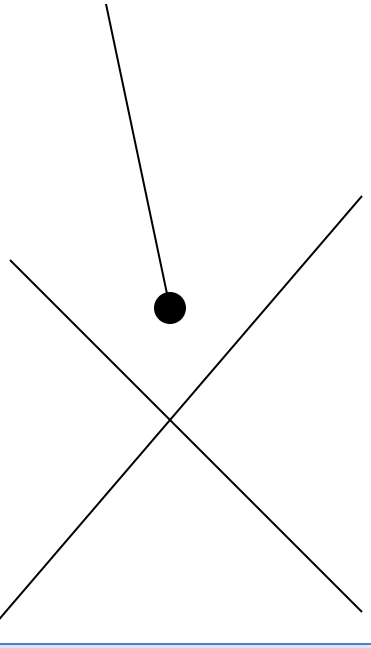
- For every horizontal slab
- The sequence S do NOT change
- Any intersection will ONLY occur between *two neighboring* segments
- **Main idea**: check for intersections between two neighbors whenever changes (= an event happens)
  - And only need to check the changing ones



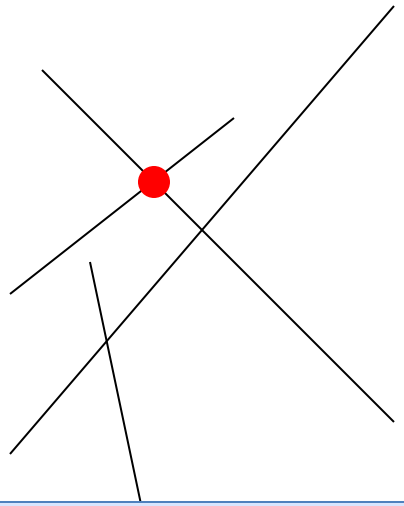
Three types of “Events”



Upper Vertex



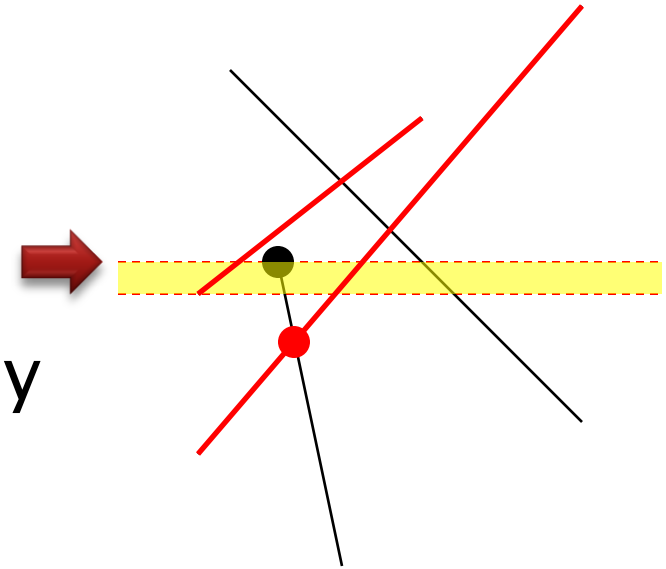
Lower Vertex



Intersection

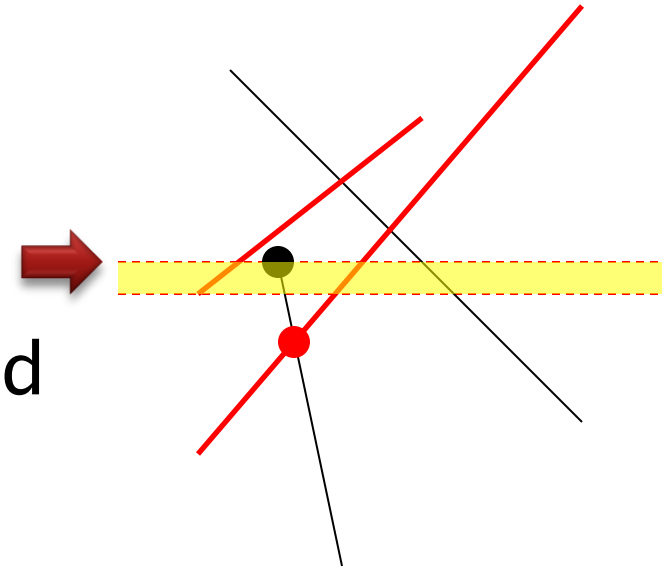
Event Type 1: the Upper Vertex

- When a new segment is introduced (the event of its upper vertex)
- The first intersection of this new segment will occur with its **IMMEDIATE** neighbors only
  - Before the next event



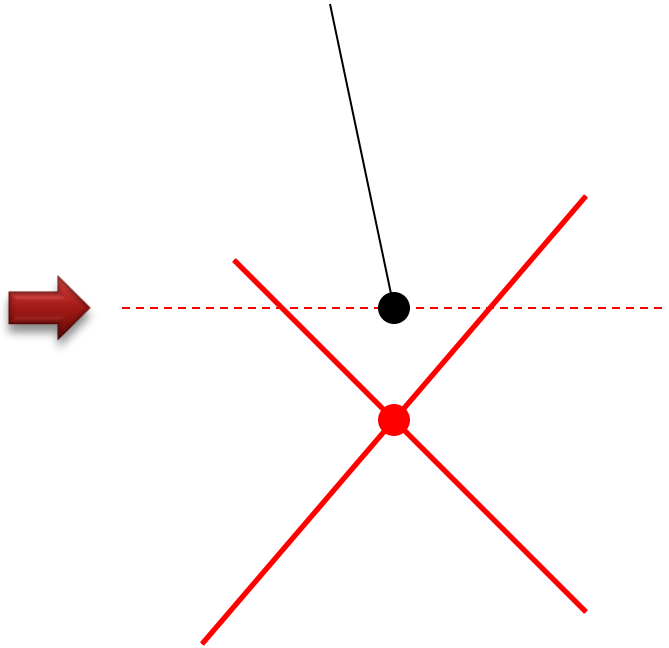
Event Type 1: the Upper Vertex

- Therefore, when it is a event of “an upper vertex”, check for intersection with its two immediate neighbors
  - O(1)
- If there is an intersection, add this intersection as an event into Q



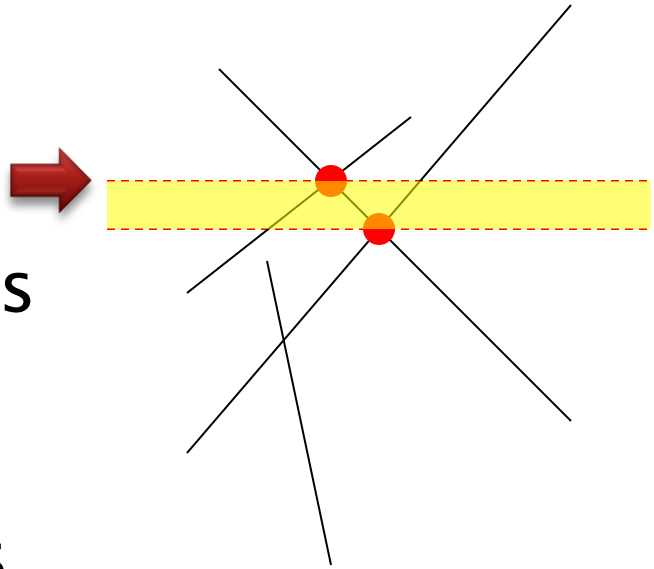
Event Type 2: the Lower Vertex

- A segment l is leaving the sequence S
- So, the two neighbors of l may intersect each other
- If the neighbors of intersect each other, add this intersection as an event into Q

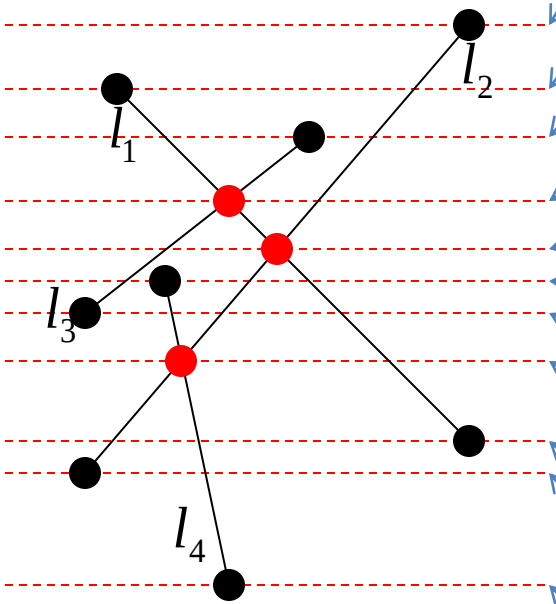


Event Type 3: The Intersection

- When an intersection happens, the two segments that intersect change their order in S
- So again, check the neighbors of these two in the new S
- And queue any event if there is some intersection happens



An Example



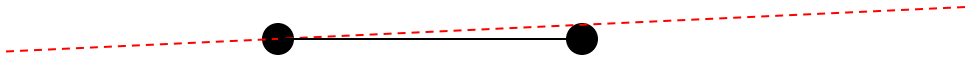
Q	S	Check
$a_2,a_1,a_3,a_4,b_3,b_1,b_2,b_4$	l_2	N/A
$a_1,a_3,a_4,b_3,b_1,b_2,b_4$	l_1,l_2	$l_1+l_2\rightarrow i_{12}$
a_3, i_{12}, a_4,b_3,b_1,b_2,b_4	l_1,l_3,l_2	$l_1+l_3\rightarrow i_{13}$ l_3+l_2
i_{13}, i_{12}, a_4,b_3,b_1,b_2,b_4	l_3, l_1,l_2	(l_1+l_2)
$i_{12},a_4,b_3,b_1,b_2,b_4$	l_3, l_2,l_1	l_3+l_2
a_4,b_3,b_1,b_2,b_4	l_3, l_4,l_2,l_1	l_4+l_3 $l_4+l_2\rightarrow i_{42}$
b_3, i_{42},b_1,b_2,b_4	l_4,l_2,l_1	
i_{42},b_1,b_2,b_4	l_2,l_4,l_1	l_4+l_1
b_1,b_2,b_4	l_2,l_4	
b_2,b_4	l_4	
b_4	$\varnothing$	

Data Structure

- Priority Queue Q
  - Heap
- Binary tree for S
- $O(\log N)$ for each event, for N is the event number, not the segment number
- Let n = # of line segment, and k = # of intersection
  - $N = n + k$
- Overall $O((n+k) \log (n+k))$
  - Even $k = O(n^2)$, $O(\log k) = O(\log n)$
  - So $O((n+k) \log n)$

Some Final Note

- What if there are two points with the same y coordinate?
  - Priority are given to the x coordinate as a second key
  - Deem it as rotating the sweeping line a very small angle



- What if there are more than two points with the same x AND y coordinates…

Some Final Note

- What if there are more than two points with the same x AND y coordinates…
  - Actually easier
  - Intersection points
    - Reverse the order in S
  - Lower points
    - Treat as normal
  - Upper points
    - Move down the sweeping line a little bit down