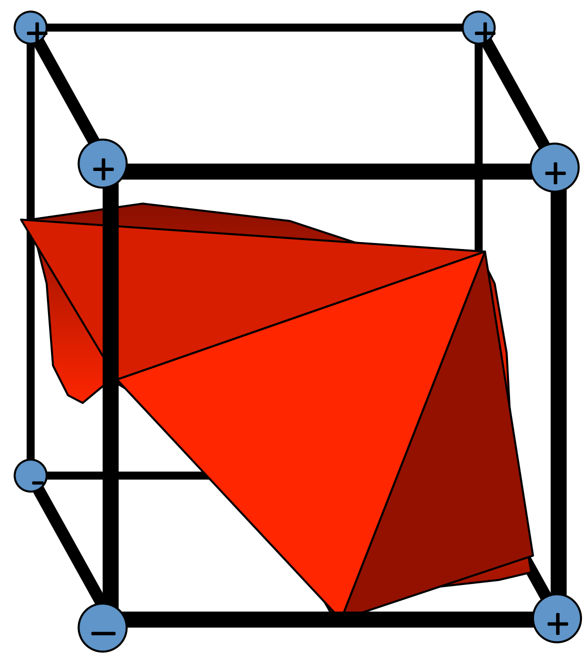
Contour Trees

CSE 788.14
Han-Wei Shen

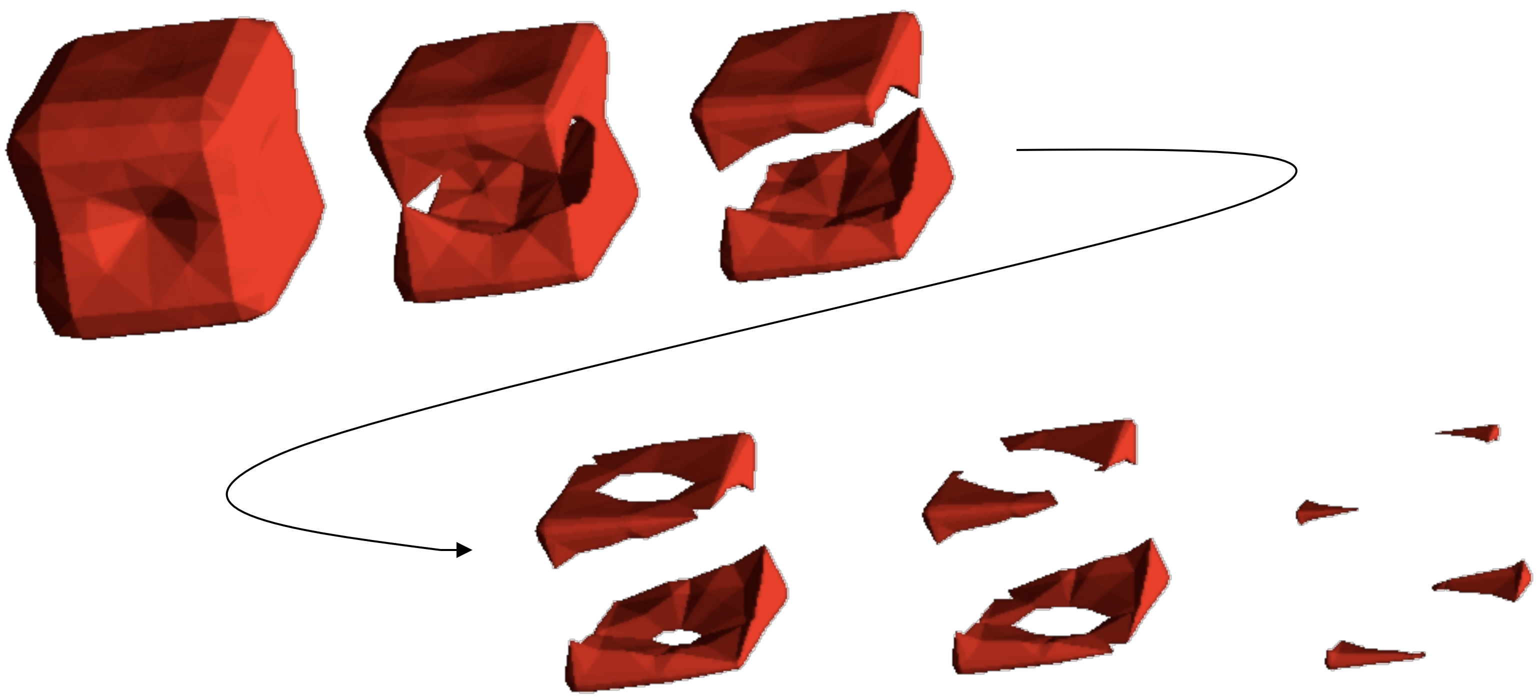
Level Sets

- Level set: $\{\mathbf{p} \in R^n | f(\mathbf{p}) = c\}$
- Level sets is also called Isolines for n=2, isosurface for n=3, or isocontours in general
- We can use the Marching Cubes algorithm to extract isosurfaces from a 3D scalar field



Evolution of Level Sets

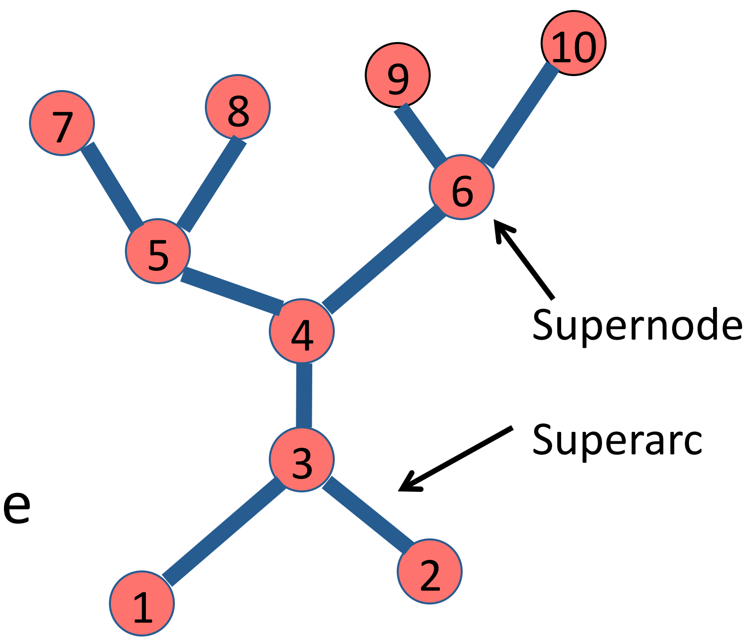
- We can watch the level set evolves as we change the contour value (isovalue) c



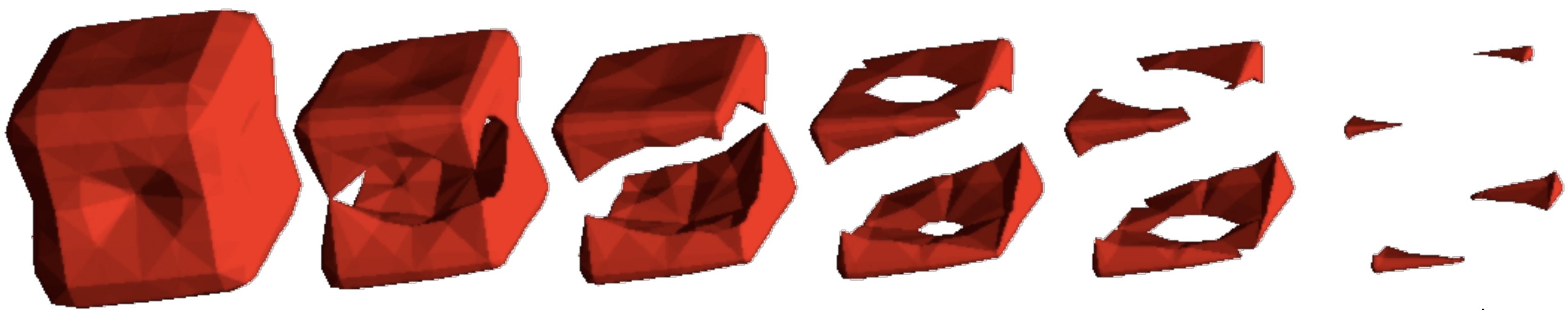
Contour Trees

- A graph-based representation to illustrate how the topology of level set changes with the scalar values

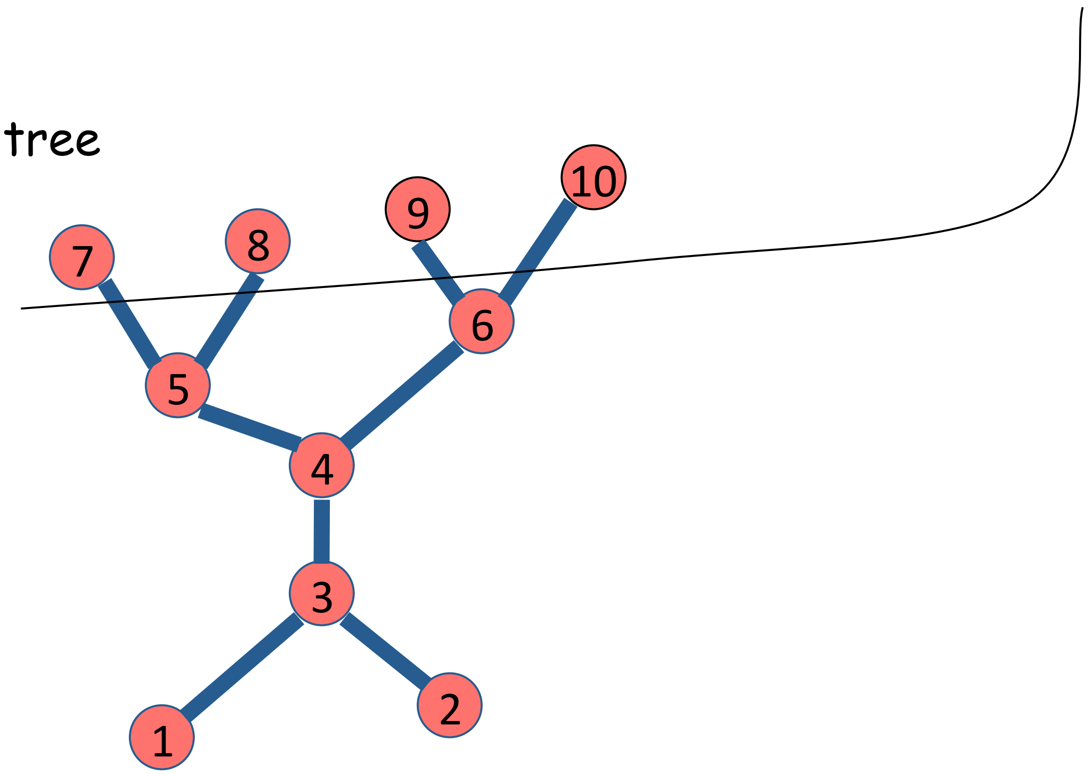
- Each leaf node represents the creation or deletion of a component
- Each interior vertex represents the joining and/or splitting of two or more components
- Each edge represents a component in the level sets for all values between the values at each end of the edge



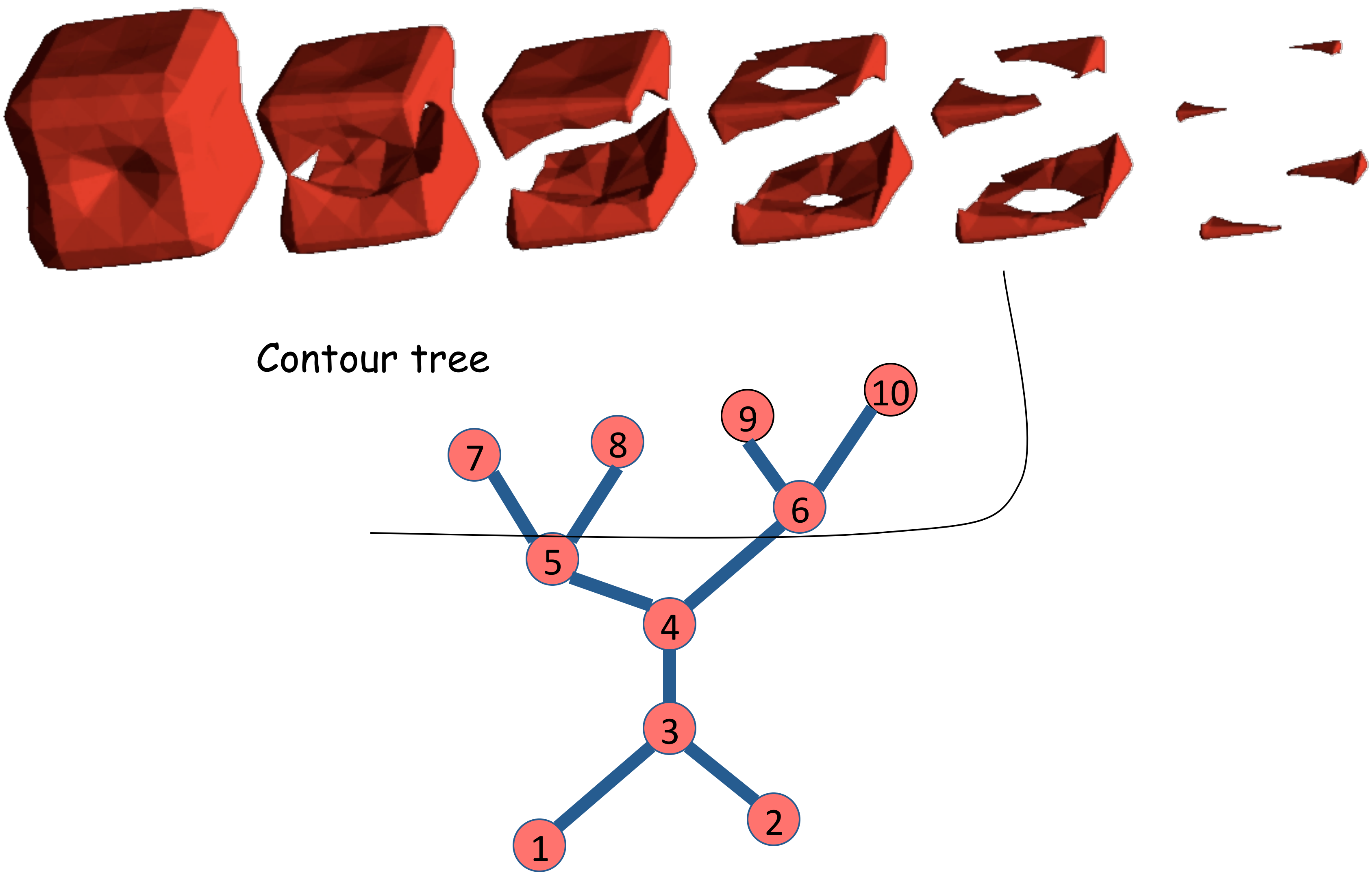
Evolution of level sets



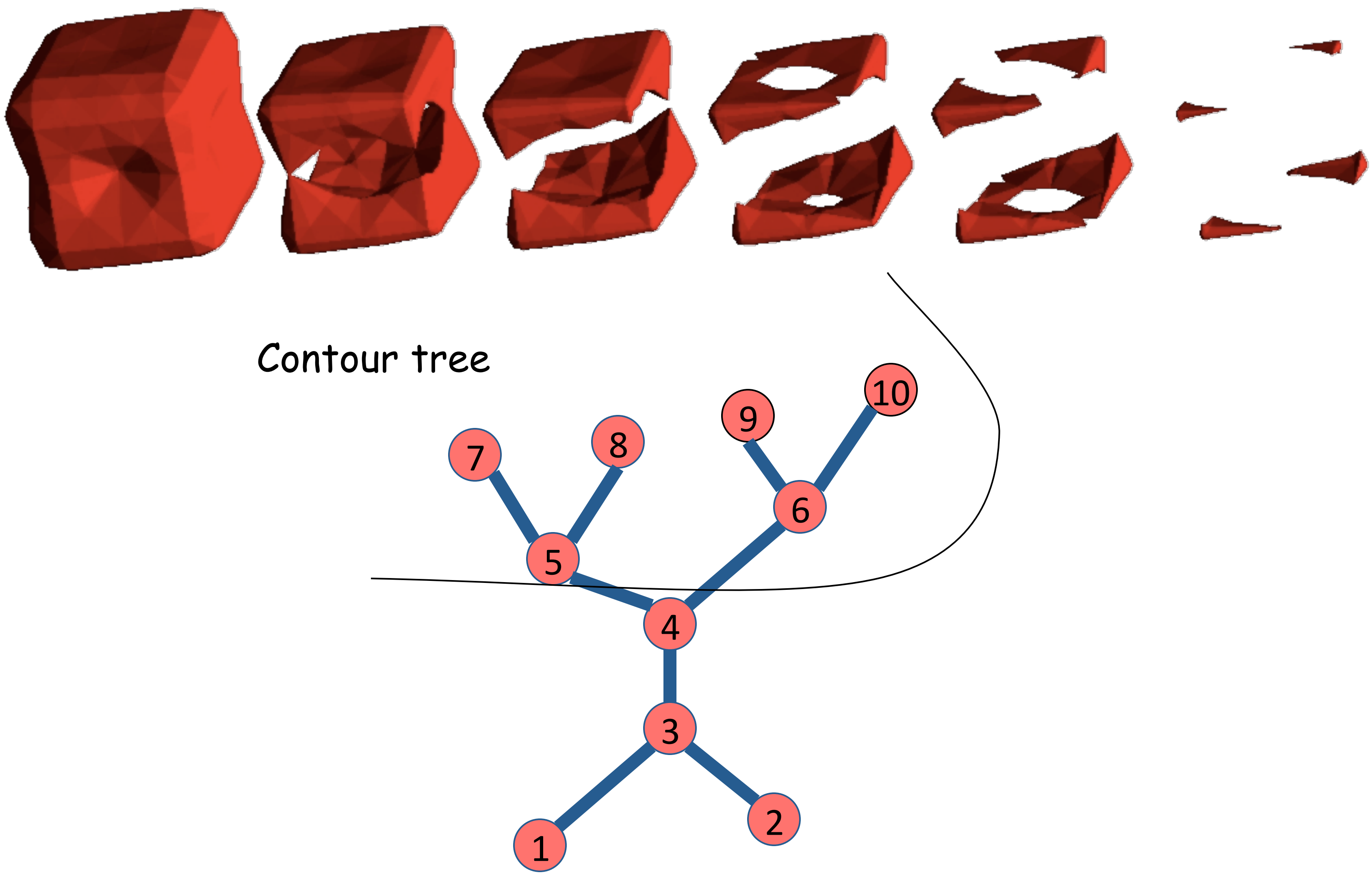
Contour tree



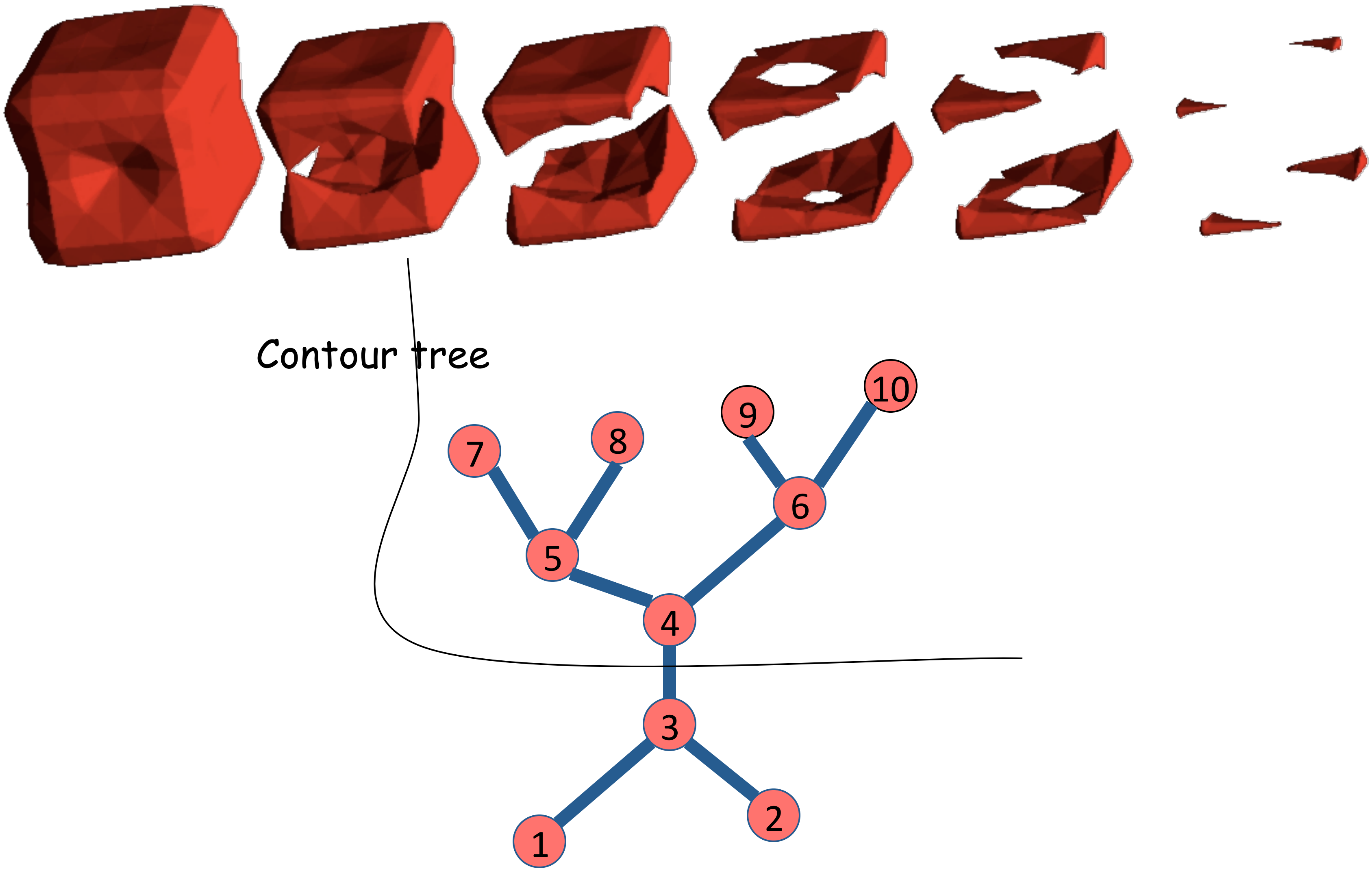
Evolution of level sets



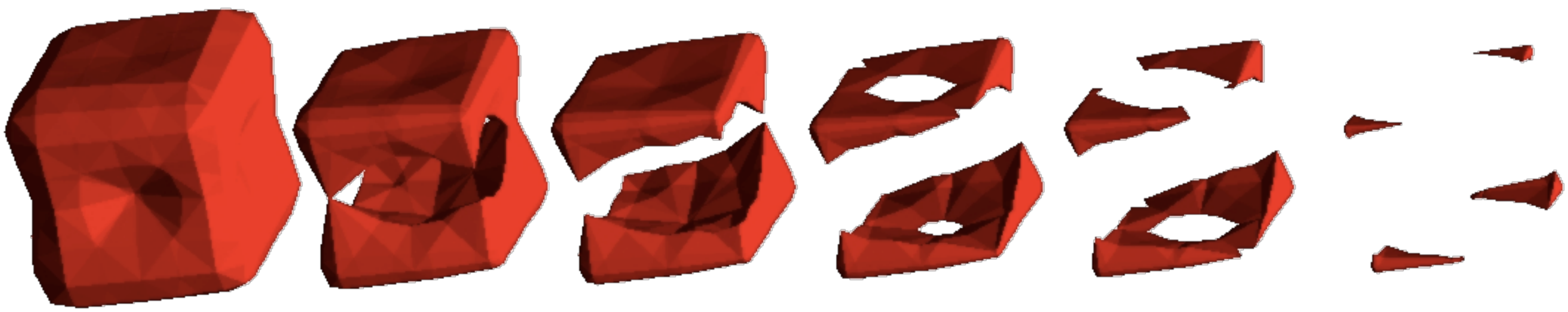
Evolution of level sets

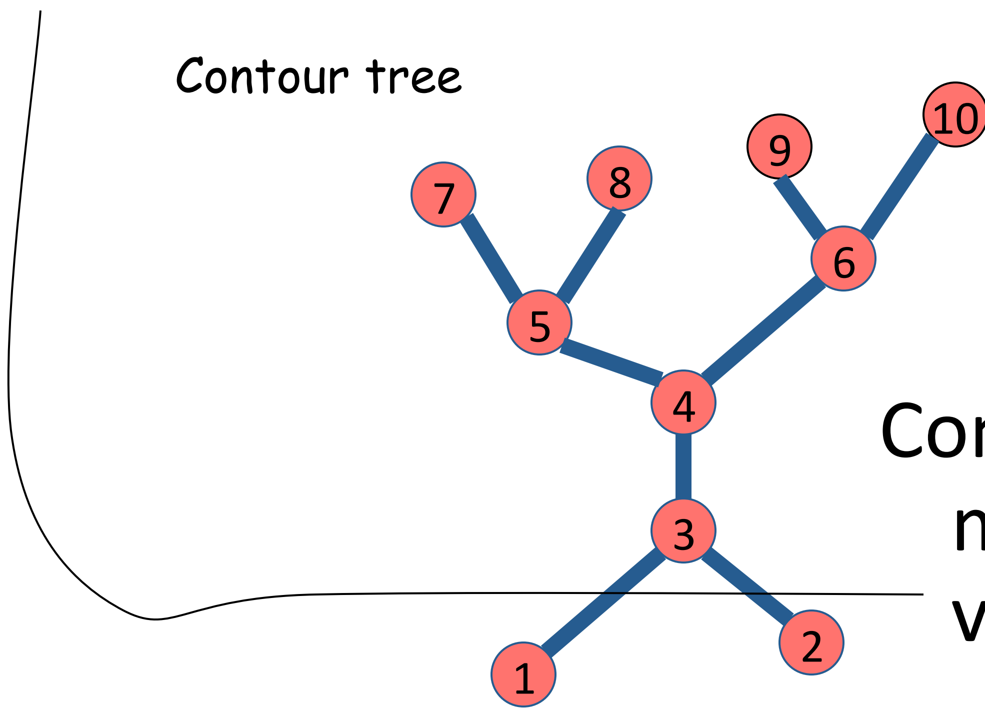


Evolution of level sets



Evolution of level sets



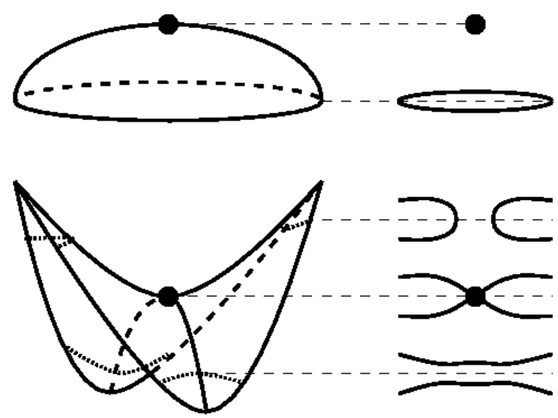


Contours appear, merge, split, & vanish

Topological Events

- The level set topology changes only at critical points

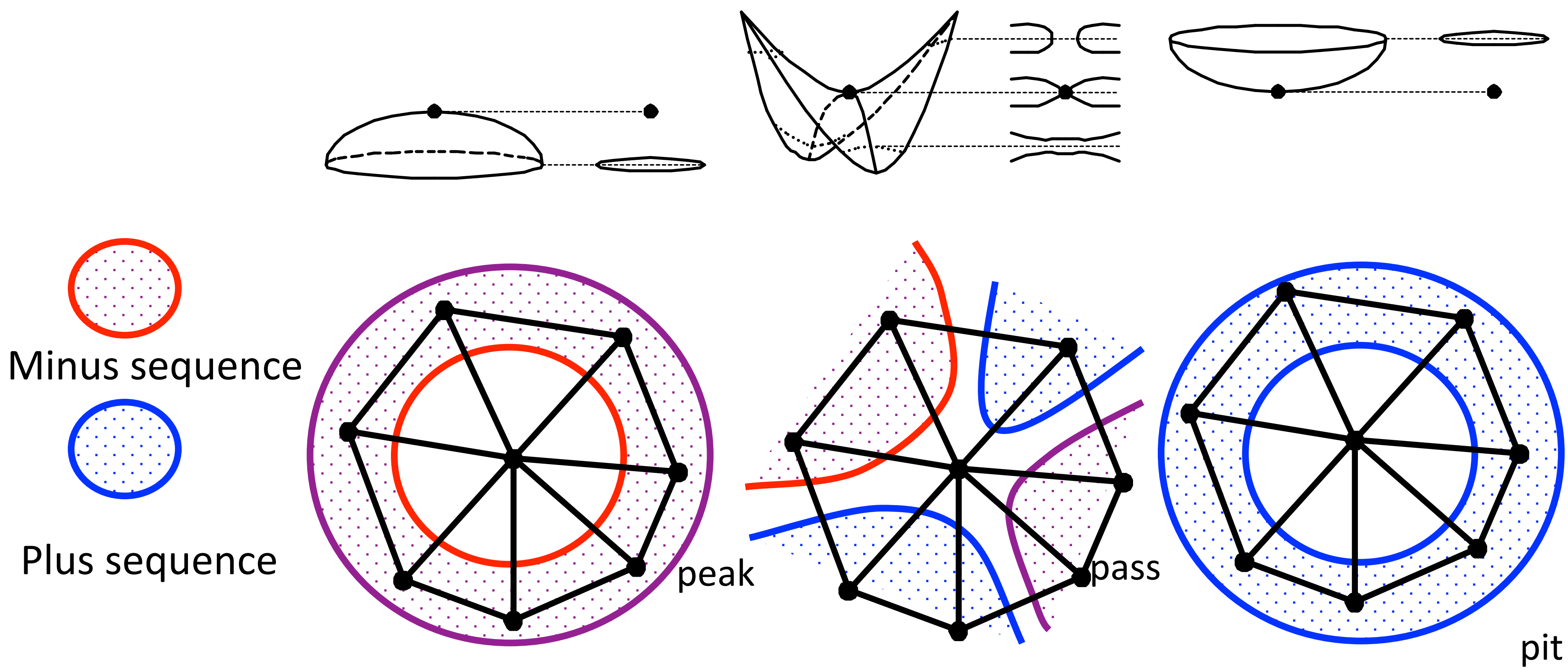
$$\frac{\partial f}{\partial x}(p) = \frac{\partial f}{\partial y}(p) = 0$$



Examples of 2D critical points

Topological Events on a Mesh

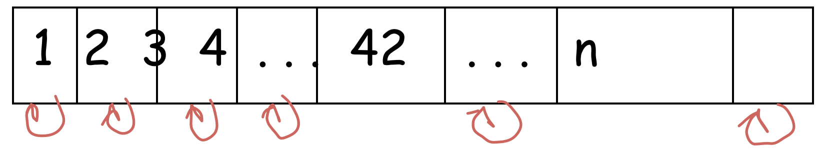
Critical Point Extraction



Extracting Contour Trees

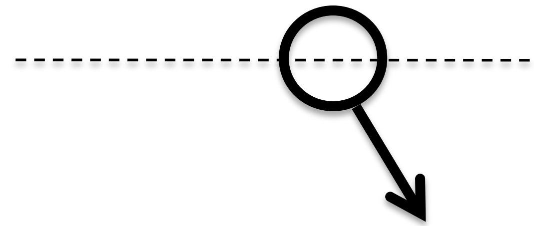
- General approach
  - Sort the scalar values at all the vertices and store into an event queue (a heap for example)
  - Scan the value C from max. to min. values in the domain
  - Track cells that are active
    - The range of the cell contains the current scalar value
  - Assign cells into one of the current components (superarcs)
  - Merge or split components at the critical topological events (local min/max and saddles)
- Use Union-Find to implement the components merge/split

Union/Find



- Data structure for integers 1..n supporting:
  - *Initialize()* each integer starts in its own group
    - *for i = 1..n, g(i) = i;*
  - *Union(i,j)* union groups of *i* and *j*
    - *g(Find(i)) = Find(j);*
  - *Find(i)* return name of group containing *i*
    - *group = i;*
    - *while group != g(group), group = g(group); // find group*
    - *while i != group,*
      {nx = g(i), g(i) = group, i = nx}   // compress path
- Does *n* union/finds in *O(n a(n))* steps

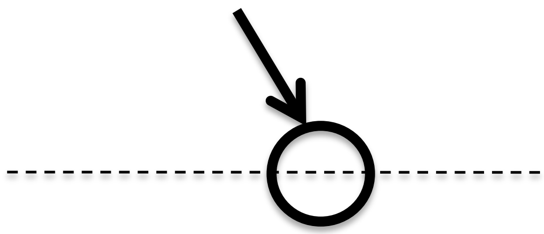
Evolution of Contour Trees



- At a local maximum:
  - a new component is born.
  - Creating a new supernode and superarc
  - Every cell incident to the local maximum vertex becomes active
  - Those cells will point to the new superarc
    - Point to the component name, which will in turn point to the superarc of the contour tree
  - Name the new component/superarc as C_a

Evolution of Contour Tree

- At a local minimum:
  - An existing component is destroyed
  - A new supernode is created
  - The end of a superarc
  - The cells incident to the local minimum are no longer active



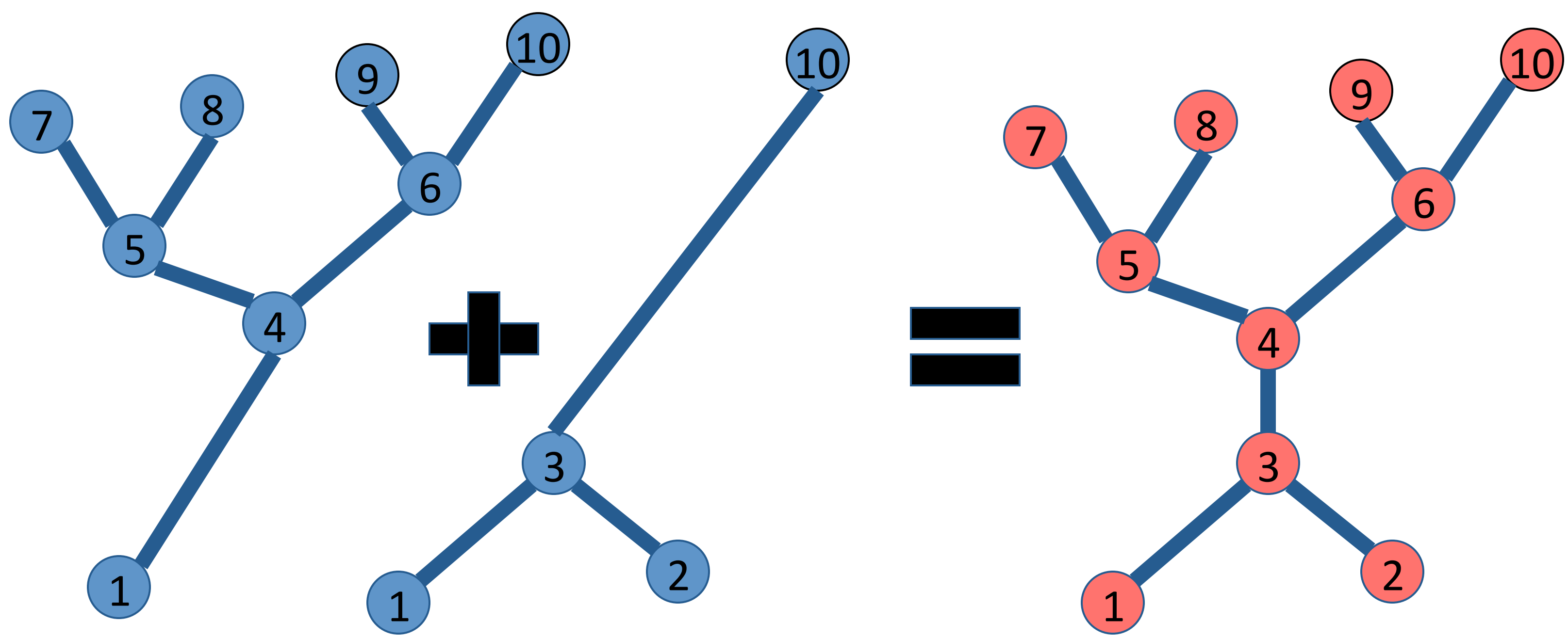
Evolution of Contour Tree

- At a saddle point:
  - (1) Two or more components merge into one, or (2) one component splits into two or more
  - need to determine which type is encountered by traversing the contours
  - Case (1): a new supernode is created; make the two/more superarcs incident to the supernode; point the still active cells to the new superarc
  - Case (2): similar but reversed actions

Creation of Contour Trees using Join/ Split Trees

- Scan the data set twice: once to create the join tree, and the other pass to create the split tree
- The Join tree has the correct down degree; and the split tree has the correct up degree
- Merge the join and split trees together into a contour tree

Join + Split Trees = Contour Tree



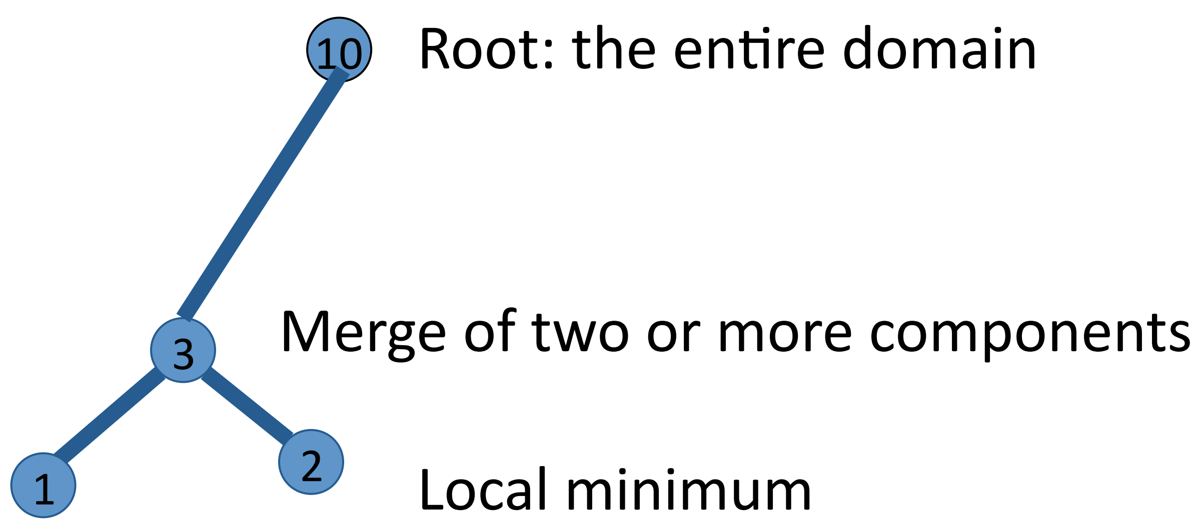
Split Tree

Join Tree

Contour Tree

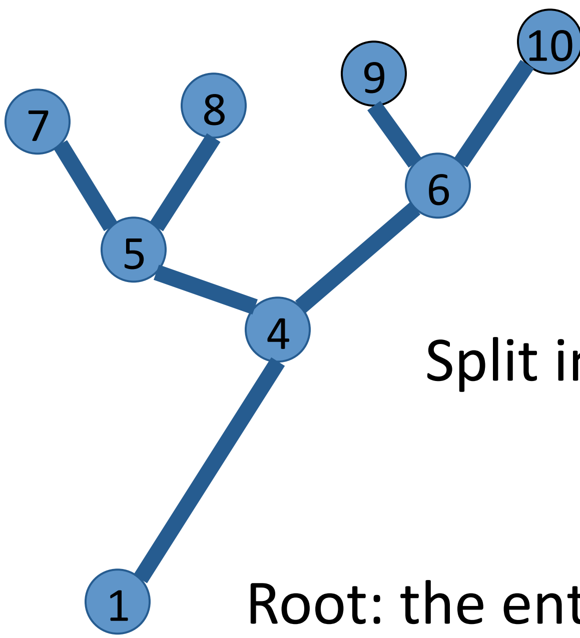
Join Tree

- A join tree is made of join components
- A join component is a connected component of the set: $$\{p \in R^d | f(p) \le x\}$$



Split Tree

- Obtained from a similar method used to construct the join tree by decreasing the parameter x: $\{p \in R^d | f(p) \geq x\}$



Split into one or more nodes

Root: the entire domain

Join and Split Trees

- Each node in the join or split tree is a node in the contour tree
- Each edge in the join or split tree represents a union of components from the contour tree
- For a node in the join (split) tree that is not in the other tree, the join (split) edge incident to this node is a single component of the contour tree

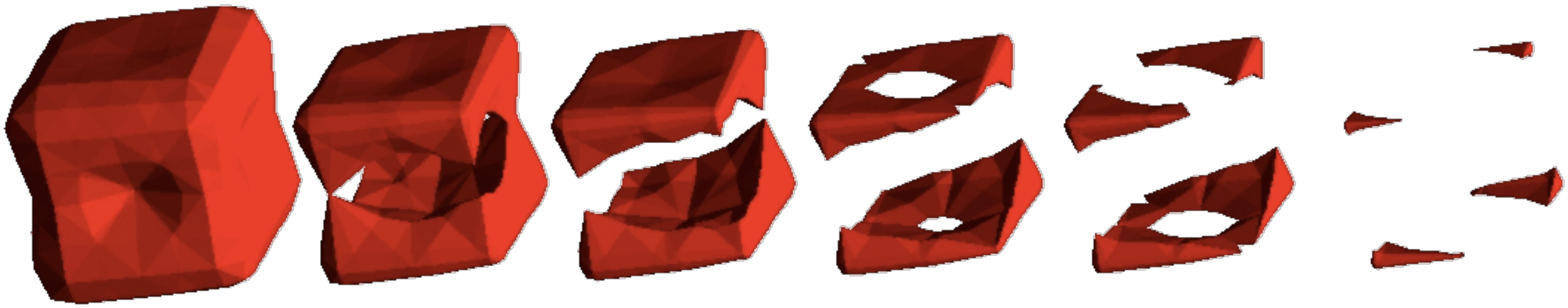
Merge Join and Split Trees

- Remember: join tree only captures the merge of components, and split tree only captures the split of components as we increase the contour value
- The down degree of the node in the join tree is correct, and the up degree of the node in the split tree is correct
- Based on the above principles, we can design a merge algorithm

Merge Algorithm

- Remove a non-root leaf node JT or ST that is not a split/join node of the other tree
  - Assume a leaf v of JT is chosen
- Move v and its incident edge from JT to CT
- If v is a node of degree 2 in ST, don't move it
- If v is a root of ST, delete v from the ST and restore the tree
- Do the above inductively

Example on Carr's paper



Contour tree