

CS181: Computer Vision

Image Segmentation

Graph-based segmentations (Felzenswalb, Normalized Cut)

Thursday, October 15th, 2026



Md Alimoor Reza

Associate Professor of Computer Science

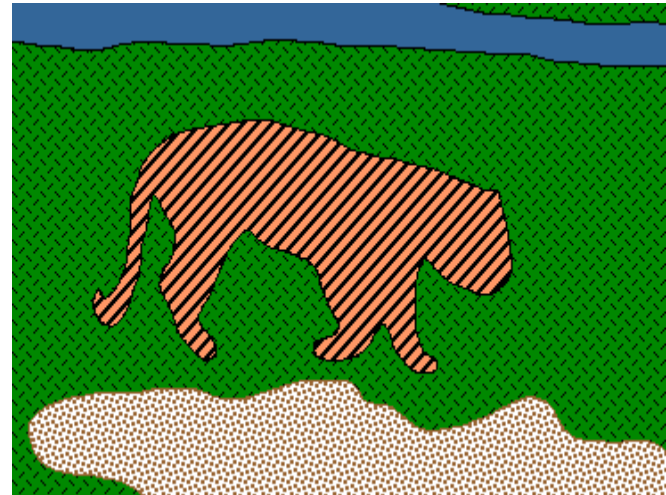
Today's agenda

- Image Segmentation
- Image Segmentation Approaches
 - Algorithmic approach
 - Clustering-based algorithms
 - k-means clustering
 - mean-shift clustering
 - SLIC clustering
 - Graph-based algorithms
 - Learning-based approach

Image segmentation

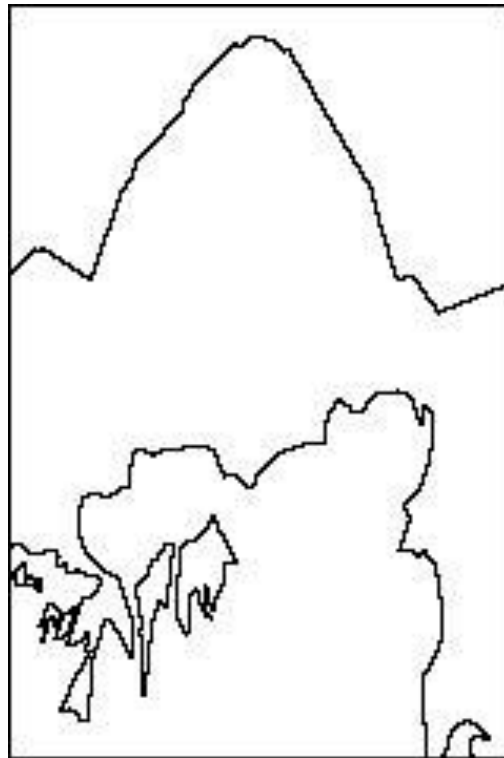
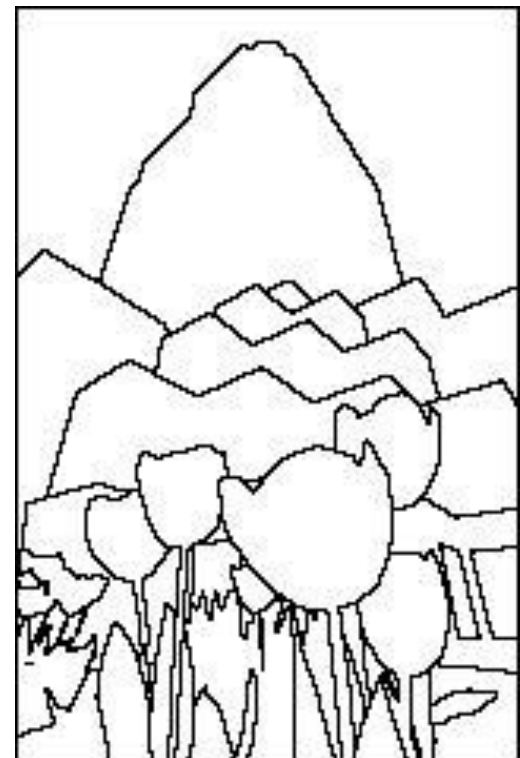
- **Goal:**

- break apart an image into simpler components
- reduce an image to a small number of homogeneous regions (“segments”)



Hard to judge success

- Which of these segmentations is “correct”?

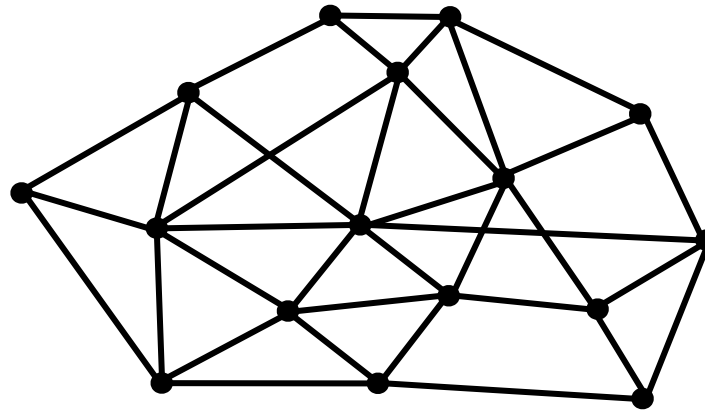
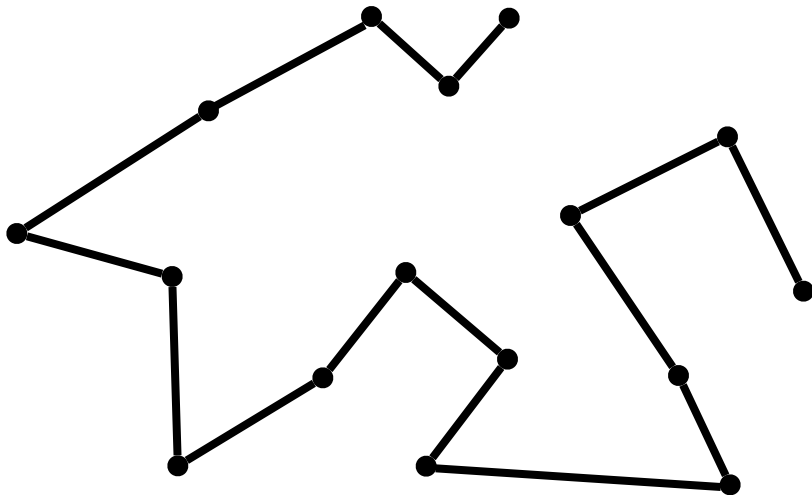


[Martin 2001]

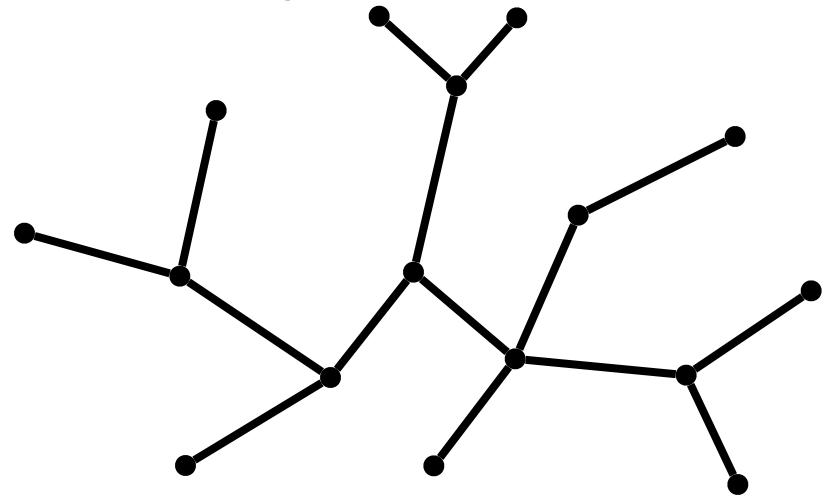
Today's agenda

- Image Segmentation
- Image Segmentation Approaches
 - Algorithmic approach
 - Clustering-based algorithms
 - k-means clustering
 - mean-shift clustering
 - SLIC clustering
 - Graph-based algorithms
 - Learning-based approach

An alternative approach: View image as a graph



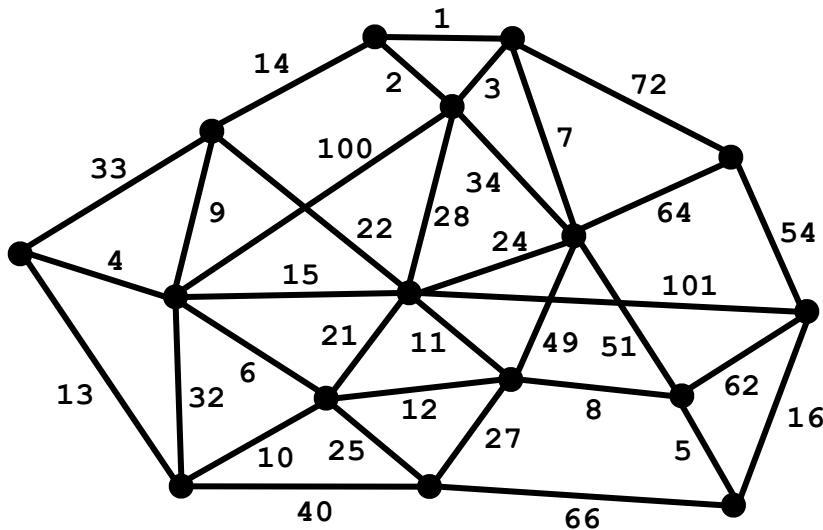
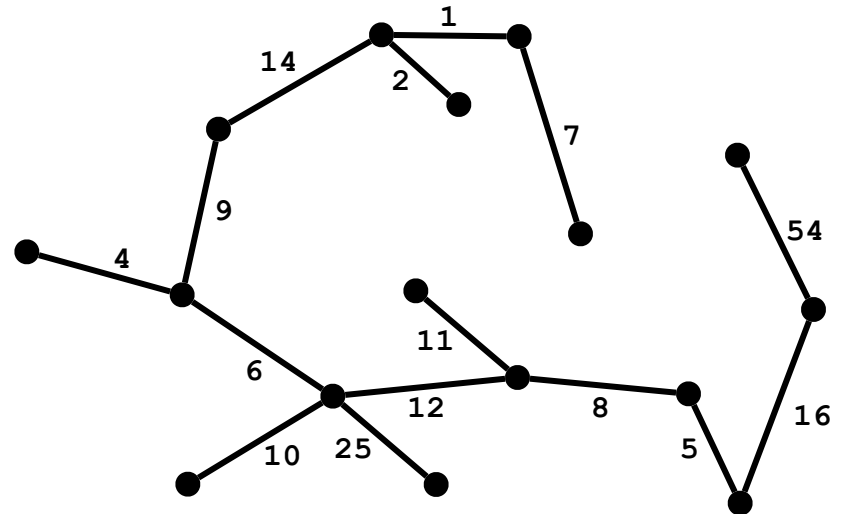
Spanning tree 1



Spanning tree 2

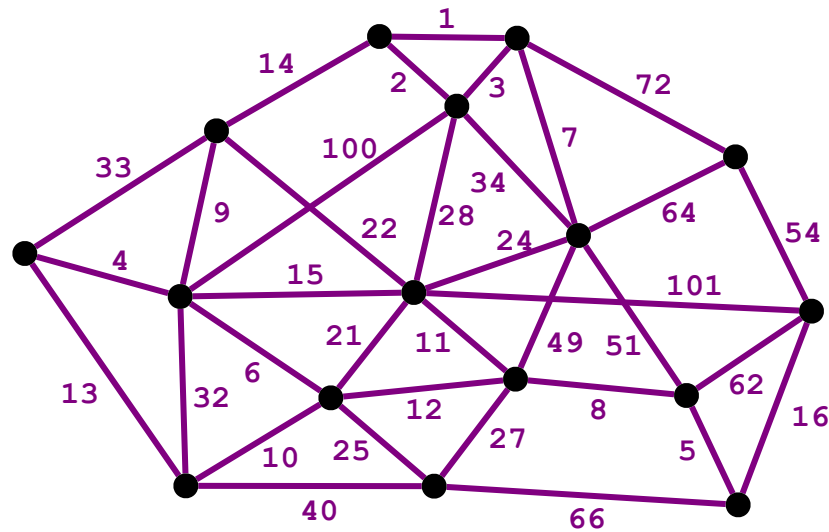
Minimum Spanning Trees

- Suppose edges are weighted, and we want a spanning tree of *minimum cost* (sum of edge weights)

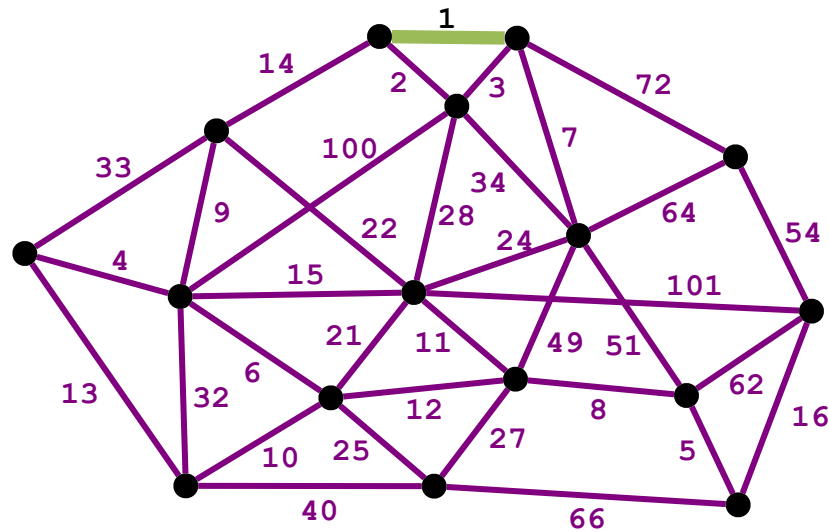


Kruskal's Algorithm

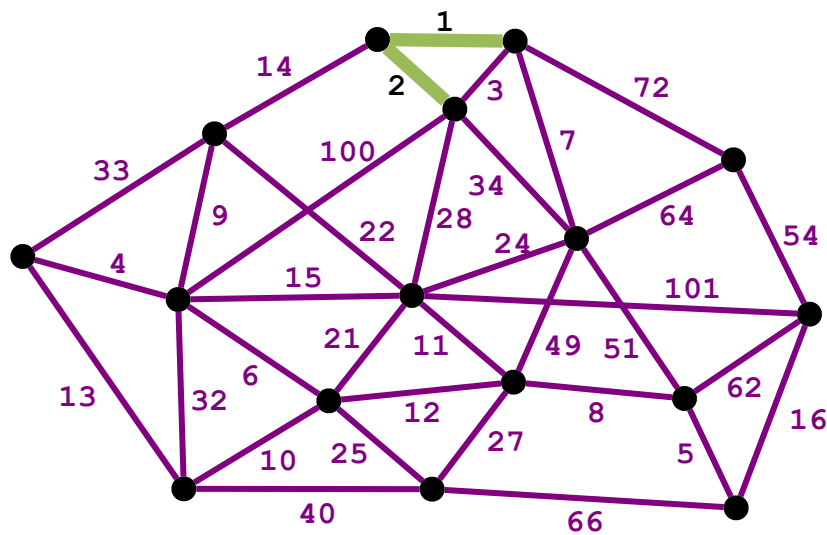
Find a min weight edge – if it forms a cycle with edges already taken, throw it out, otherwise keep it



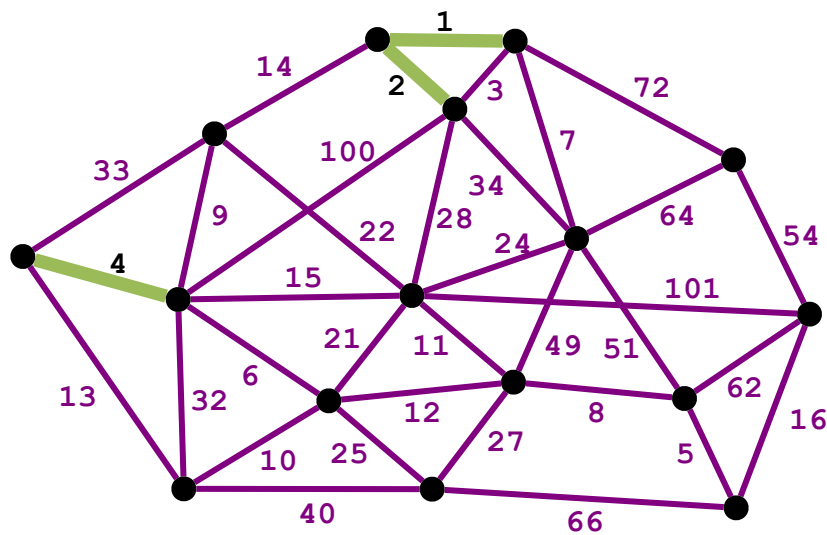
Kruskal's Algorithm



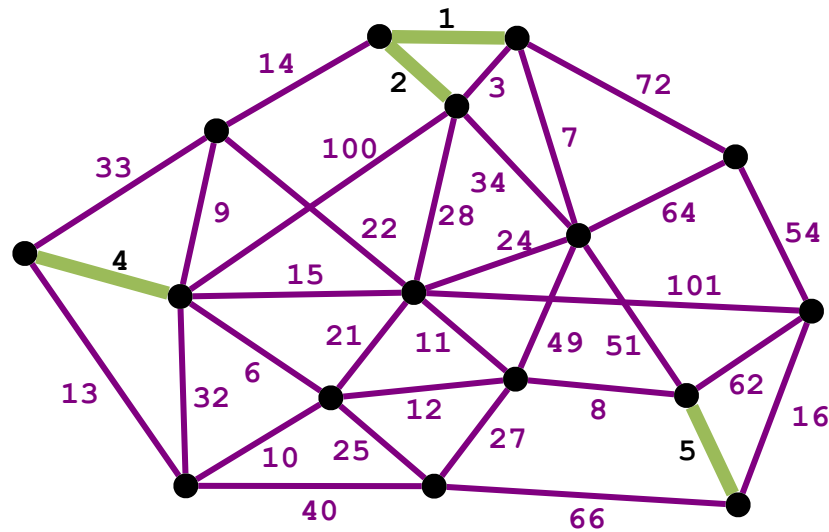
Kruskal's Algorithm



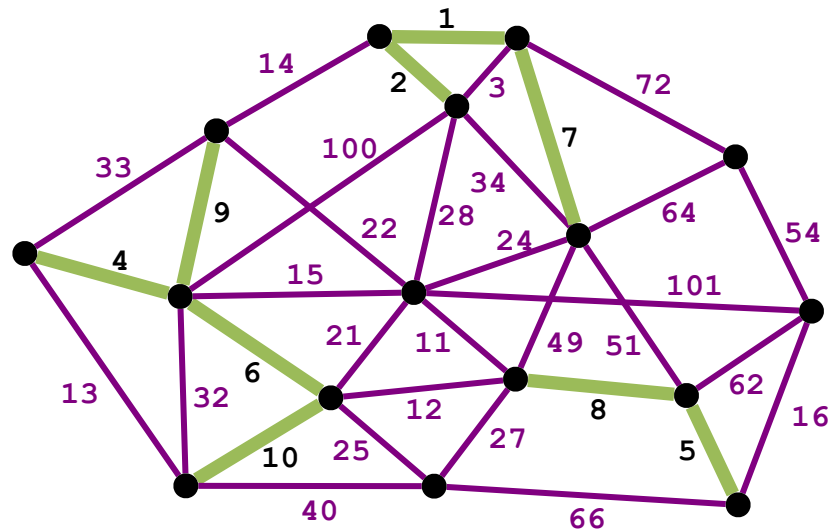
Kruskal's Algorithm



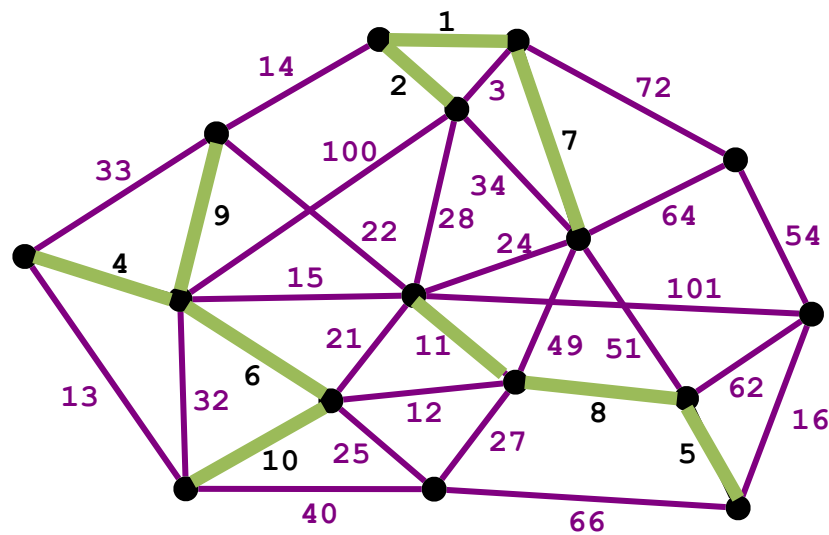
Kruskal's Algorithm



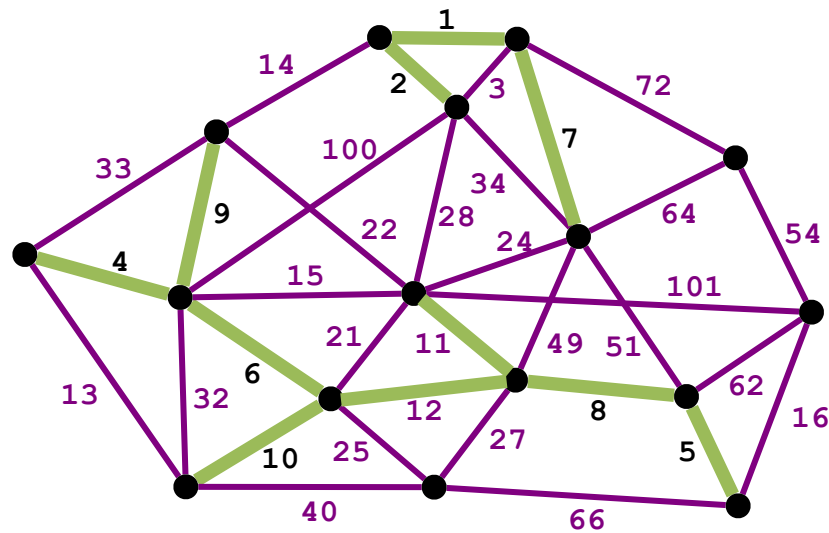
Kruskal's Algorithm



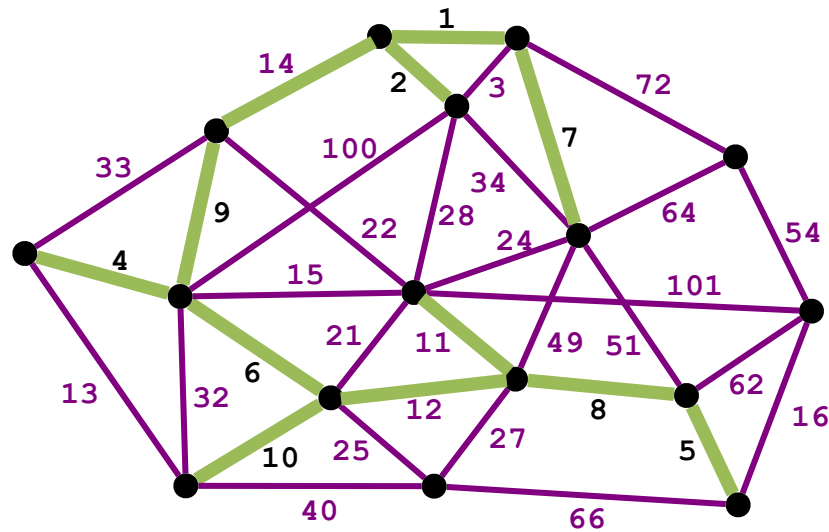
Kruskal's Algorithm



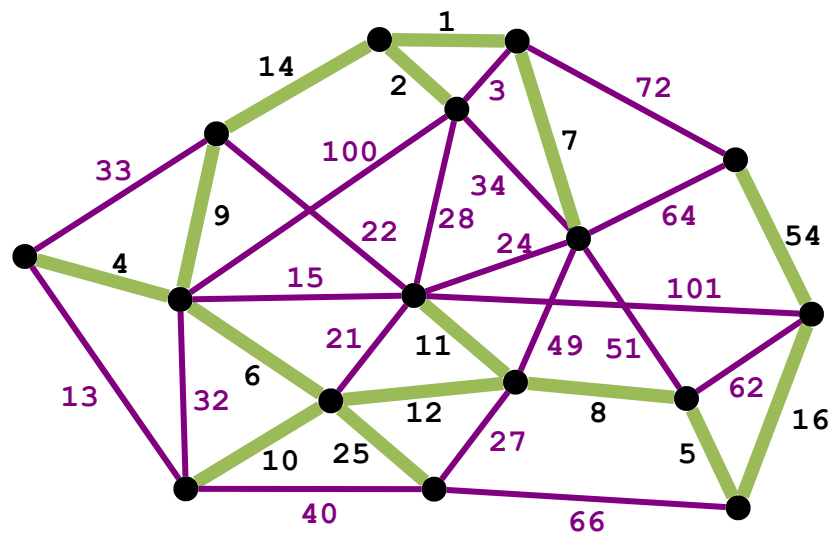
Kruskal's Algorithm



Kruskal's Algorithm

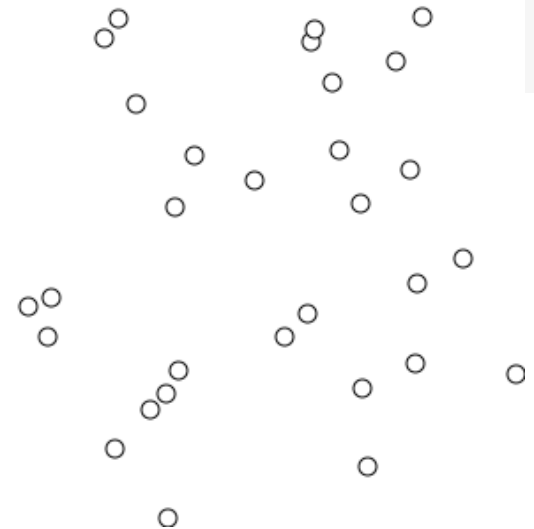


Kruskal's Algorithm



Kruskal's Algorithm (Pseudocode)

```
algorithm Kruskal( $G$ ) is  
   $F := \emptyset$   
  for each  $v \in G.V$  do  
    MAKE-SET( $v$ )  
  for each  $(u, v)$  in  $G.E$  ordered by  $\text{weight}(u, v)$ , increasing do  
    if FIND-SET( $u$ )  $\neq$  FIND-SET( $v$ ) then  
       $F := F \cup \{(u, v)\} \cup \{(v, u)\}$   
      UNION(FIND-SET( $u$ ), FIND-SET( $v$ ))  
  return  $F$ 
```



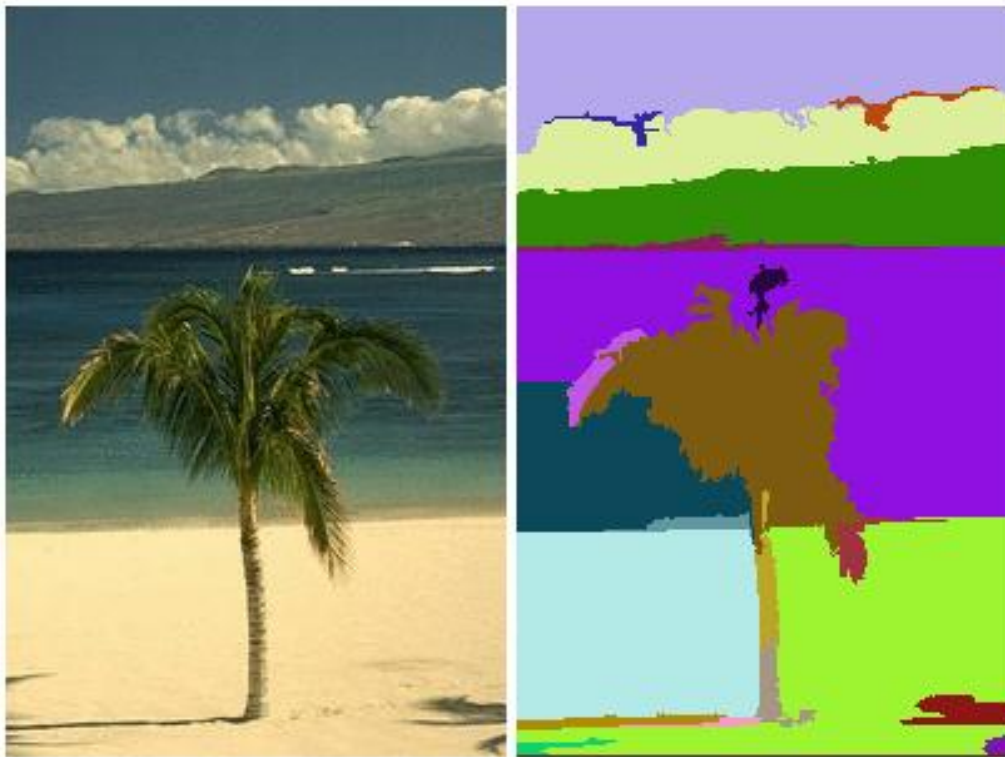
Today's agenda

- Image Segmentation
- Image Segmentation Approaches
 - Algorithmic approach
 - Clustering-based algorithms
 - k-means clustering
 - mean-shift clustering
 - SLIC clustering
 - Graph-based algorithms
 - Felzenswalb segmentation
 - Learning-based approach

Back to image segmentation...

- **Goal:**

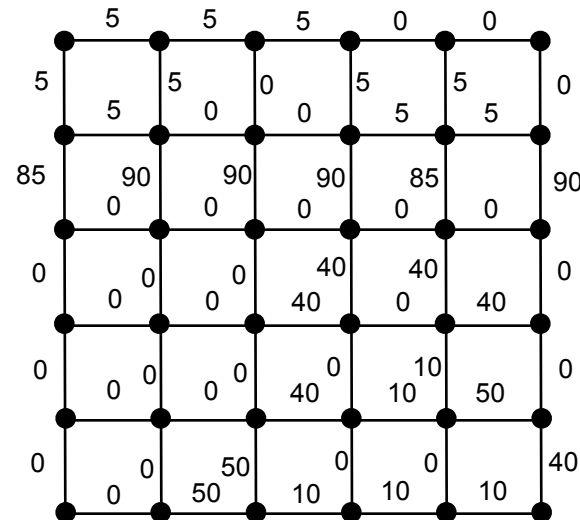
- break apart an image into simpler components
- reduce an image to a small number of homogeneous regions (“segments”)



Segmentation as a graph problem

- Represent an image as a graph
 - Vertices represent image pixels
 - Edges between adjacent pixels
 - Edge weights give difference in color between pixels

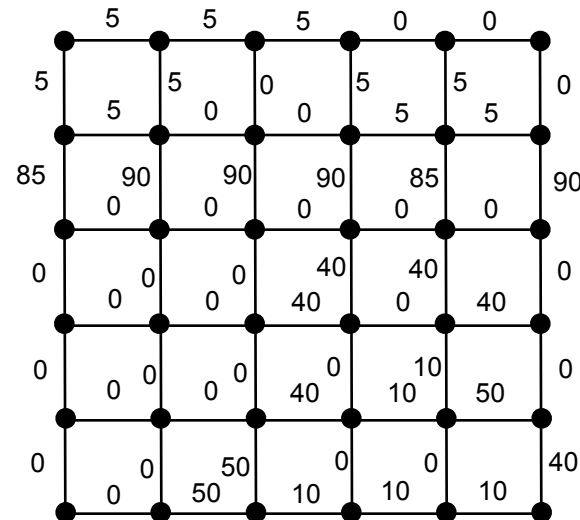
10	15	10	15	10	10
15	10	10	10	15	10
100	100	100	100	100	100
100	100	100	60	60	100
100	100	100	60	50	100
100	100	50	60	50	60



Segmentation as a graph problem

- Goal: find a small number of homogeneous regions
 - find a set of connected components in the graph, such that the sum of edge weights in each component is low
 - we can do this by finding a minimum spanning tree, and then removing a few high-weight edges

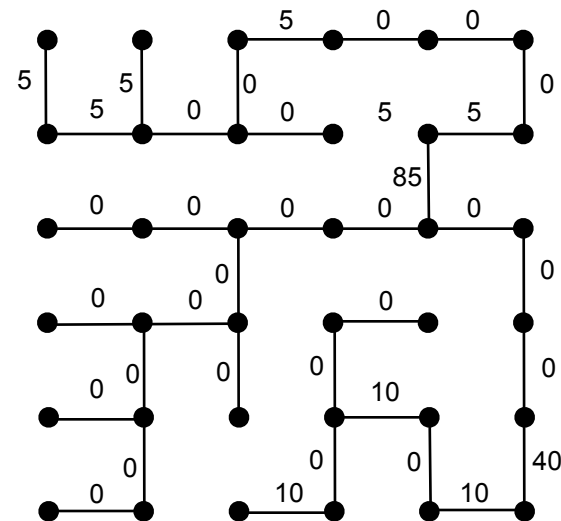
10	15	10	15	10	10
15	10	10	10	15	10
100	100	100	100	100	100
100	100	100	60	60	100
100	100	100	60	50	100
100	100	50	60	50	60



Segmentation as a graph problem

- Goal: find a small number of homogeneous regions
 - find a set of connected components in the graph, such that the sum of edge weights in each component is low
 - we can do this by finding a minimum spanning tree, and then removing a few high-weight edges

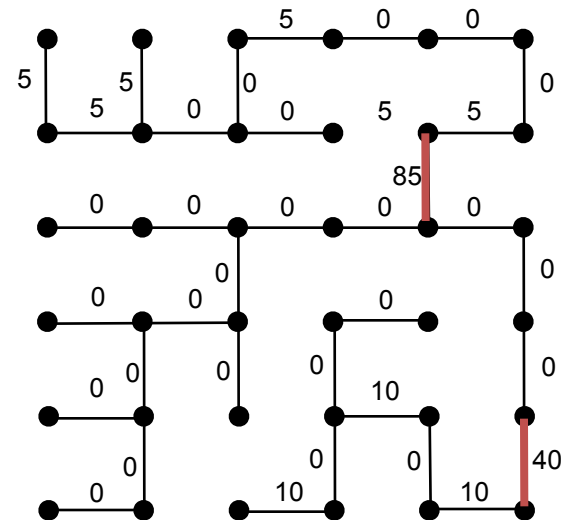
10	15	10	15	10	10
15	10	10	10	15	10
100	100	100	100	100	100
100	100	100	60	60	100
100	100	100	60	50	100
100	100	50	60	50	60



Segmentation as a graph problem

- Goal: find a small number of homogeneous regions
 - find a set of connected components in the graph, such that the sum of edge weights in each component is low
 - we can do this by finding a minimum spanning tree, and then removing a few high-weight edges

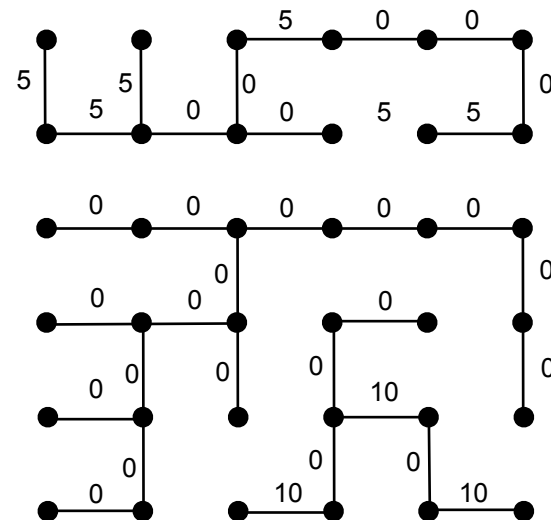
10	15	10	15	10	10
15	10	10	10	15	10
100	100	100	100	100	100
100	100	100	60	60	100
100	100	100	60	50	100
100	100	50	60	50	60



Segmentation as a graph problem

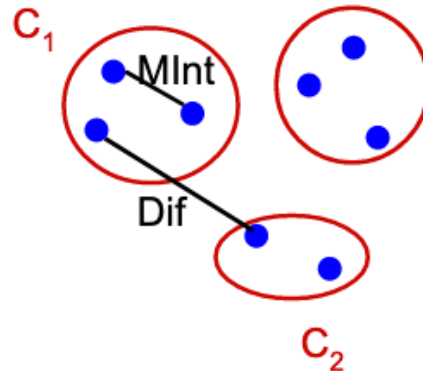
- Goal: find a small number of homogeneous regions
 - find a set of connected components in the graph, such that the sum of edge weights in each component is low
 - we can do this by finding a minimum spanning tree, and then removing a few high-weight edges

10	15	10	15	10	10
15	10	10	10	15	10
100	100	100	100	100	100
100	100	100	60	60	100
100	100	100	60	50	100
100	100	50	60	50	60



Felzenswalb & Huttenlocher algorithm

- Consider properties of components, rather than edges, when adding an edge
 - Only link components if the difference between them is much less than difference within them



- Sort edges into $(e_1, \dots, e_n)$ by increasing weight
- Initialize *the data-structure* S with one component per pixel
- For each e_q in $(e_1, \dots, e_n)$ do:
 - If **weight of e_q** small relative to **internal difference of components** it connects then merge components

Felzenswalb & Huttenlocher algorithm

- Constructs a graph on the pixels and then merges them based on whether there exists any boundary between them

$$D(C_1, C_2) = \begin{cases} \text{true} & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ \text{false} & \text{otherwise} \end{cases}$$

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w(v_i, v_j).$$

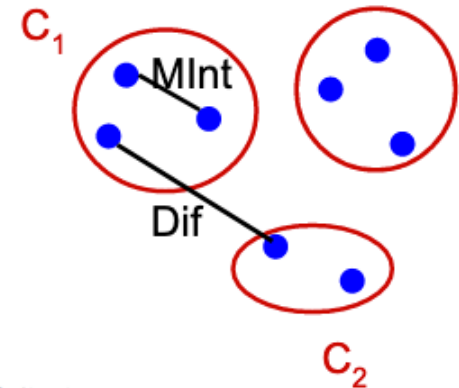
$$Int(C) = \max_{e \in MST(C, E)} w(e).$$

$$MInt(C_1, C_2)$$

$$= \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)).$$

where

$$\tau(C) = k/|C|$$



- $Dif(C_1, C_2)$ is the difference between two components
- $MInt(C_1, C_2)$ is the value of internal difference in the two components C_1 and C_2
 - it is minimum value between the maximum-valued edges of the two MSTs

Some results



Today's agenda

- Image Segmentation
- Image Segmentation Approaches
 - Algorithmic approach
 - Clustering-based algorithms
 - k-means clustering
 - mean-shift clustering
 - SLIC clustering
 - **Graph-based algorithms**
 - Felzenswalb segmentation
 - Normalized Cut segmentation
 - Learning-based approach

Normalized cuts - Shi and Malik'00

1. Build complete graph with affinity weights
2. Run eigen decomposition
3. Partition graph according to second smallest eigenvector
4. Recursively perform Step#1 on each partition

Activity: Graph-based Segmentation

▼ Felzenswalb graph-based algorithm

- scale – higher means larger clusters.
- sigma – width (standard deviation) of Gaussian kernel used in preprocessing
- min_size – minimum component size. Enforced using postprocessing.

```
▶ from skimage.segmentation import felzenszwalb
img = skimage.io.imread('sample_data/' + image_names[1])
# ----- Felzenswalb segmentation algorithm -----
segments = felzenszwalb(img, scale=3.0, sigma=0.95, min_size=800)
#skimage.io.imsave('coffee_felz.png', segments)
#plot.imshow(np.asarray(segments))
plt.imshow(mark_boundaries(img, segments))
plt.title('Felzenswalb graph-based segmentation')
```

```
☐ Text(0.5, 1.0, 'Felzenswalb graph-based segmentation')
```



Recognition through segmentation

- Many vision systems have tried to use segmentation as an intermediate step, e.g.
 - Do segmentation on an image
 - Compute features (shape, color, etc.) for each segment
 - Decide whether each segment is an object or not



Segmentation: Caveats

- Image segmentation is not a bottom-up problem
 - Needs to be done with recognition, simultaneously
- Segmentation makes hard decisions
 - Making the wrong decision could be catastrophic for later steps of a system
- Difficult to evaluate; when is a segmentation successful?

