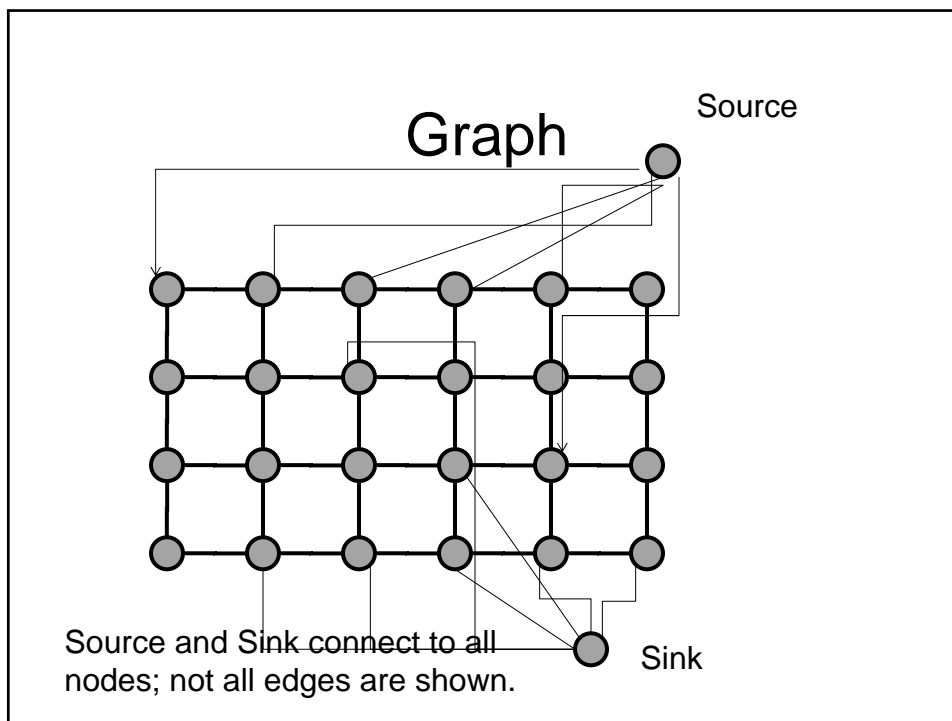
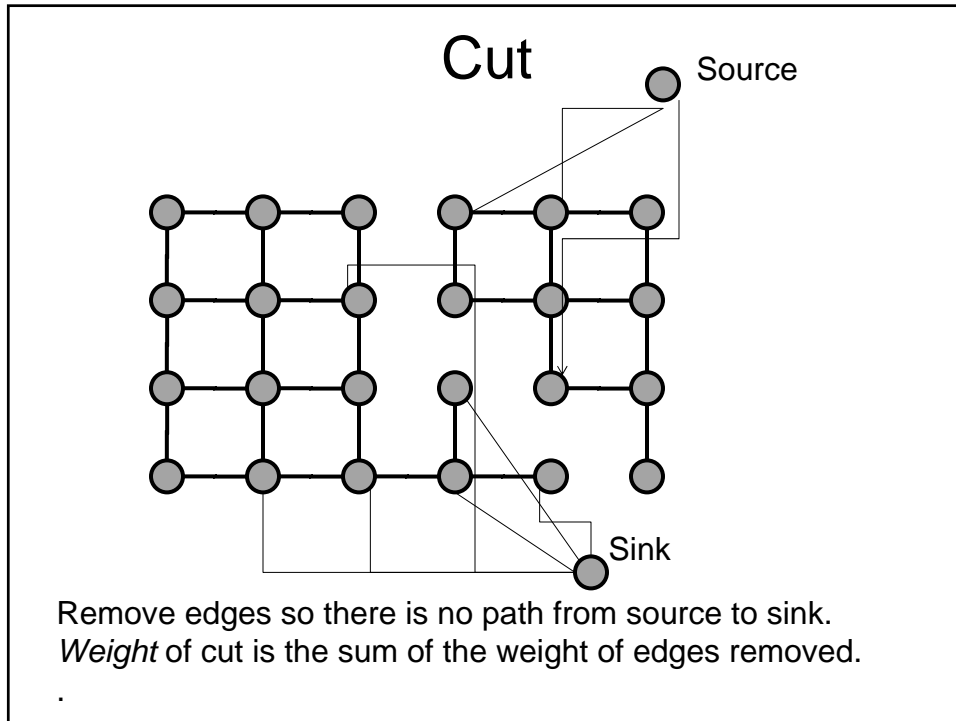


Graph Cuts for Segmentation

- Seek division of image into *foreground* and *background*.
- Turn image into graph, each pixel connected to neighbors and special source (foreground) and sink (background nodes).
- A *cut* of the graph divides it into foreground and background.
- Edge weights determine:
 - Is a pixel likely to be foreground or background?
 - Is a pixel likely to have same label as neighbors?





Min Cut

- *Min Cut* is the cut with the lowest weight
- Well studied problem with many practical applications.

Min Cut for Interactive Segmentation

- Assume user has specified some pixels as foreground/background.
- Identify a cut as a segmentation:
 - Pixels connected to source are foreground.
 - Pixels connected to sink are background.
 - The weight of edges in the cut should reflect knowledge of foreground and background.

Hard constraints

- Let S be source, T be sink, $w(p,q)$ is weight of edge between nodes p & q .
- If pixel p definitely is foreground, make:
 - $w(p,S)$ very big, $w(p,T) = 0$.
 - Edge from p to S ($E(p,S)$) will never be cut.
 - $E(q,T)$ must therefore always be cut so there's no path from S to T

Data Term

- Let F be set of pixels known to be foreground. Let B be background pixels.
- What about $p \notin F \cup B$?
- Compare properties of p to foreground and background pixels.

Color Histogram Comparison

1. Compute color histograms for foreground and background, h_f, h_b
2. *Smooth* histograms by adding a constant to each bin.
3. Normalize histograms so they sum to 1 (like probabilities).
4. Find $\Pr(p|\text{Foreground})$, $\Pr(p|\text{Background})$ by finding bin p belongs to, and looking up values in normalized histograms.
5. $w(p,S) = -\log(\Pr(p|\text{Background}))$
6. $w(p,T) = -\log(\Pr(p|\text{Foreground}))$

Histograms with Graph Cut

- Why $-\log$?
 - We are adding weights. We multiply probabilities, so add logs.
 - We maximize probabilities, so minimize $-\log$.
- Example: if p has a color that rarely appears in foreground, edge to source will have low weight.
- Why smooth? We only have a small sample.

Graph Cut with Data Term

- Suppose we compute mincut with just these edges to source and sink.
- Segmentation respects user input.
- Other pixels classified based on whether they resemble foreground or background.
- Results can be quite spotty.

Smoothness term

- If a pixel, p , is foreground, its neighbor, q , is likely to be foreground.
 - Especially if p and q are similar.

$$w(p, q) = e^{-\frac{(I(p) - I(q))^2}{2\sigma^2}}$$

- This is gradient, normalized in ad-hoc way.
- Note, gradient is taken between pixels, not on one pixel.

Results



User Input



No Edge Weights
Just Data Term



Full segmentation