Neighbor joining algorithm

- Neighbor joining works in a similar fashion to UPGMA
  - Find clusters C_1 and C_2 that minimise a function $f(C_1, C_2)$
  - Join the two clusters C_1 and C_2 into a new cluster C
  - Add a node to the tree corresponding to C
  - Assign distances to the new branches
- Differences in
  - The choice of function $f(C_1, C_2)$
  - How to assign the distances

Neighbor joining algorithm

- Recall that the distance d_{ij} for clusters C_i and C_j was

$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{p \in C_i, q \in C_j} d_{pq}$$

- Let $u(C_i)$ be the separation of cluster C_i from other clusters defined by

$$u(C_i) = \frac{1}{n-2} \sum_{C_j} d_{ij}$$

where n is the number of clusters.

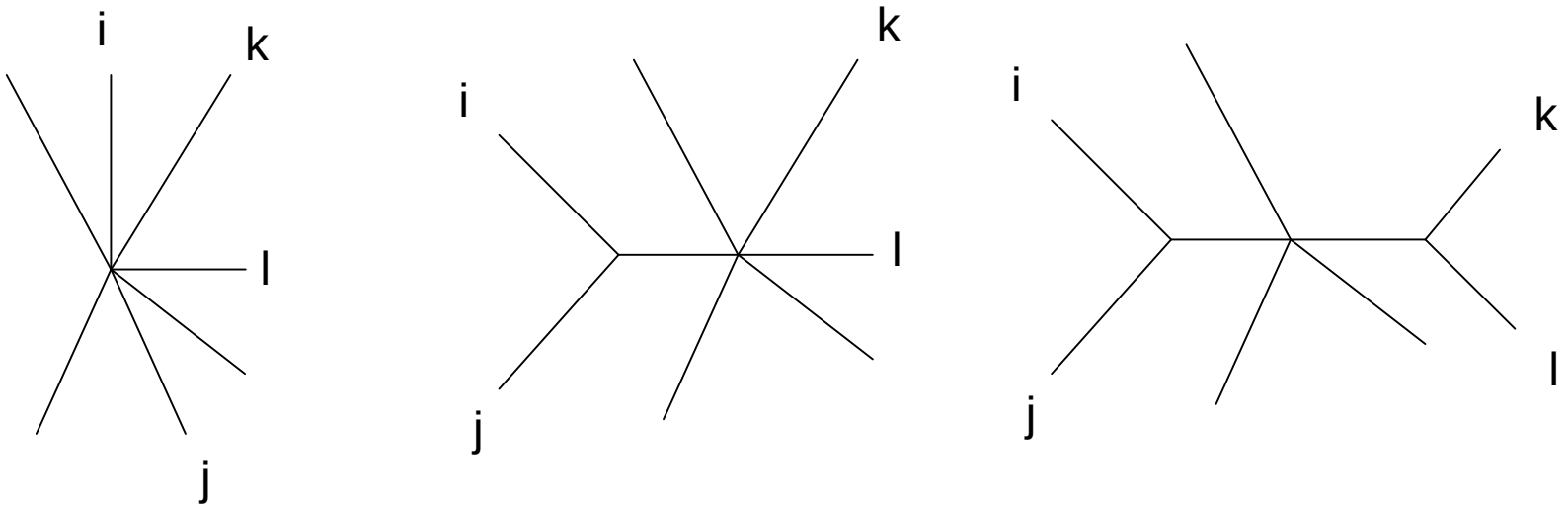
Neighbor joining algorithm

- Instead of trying to choose the clusters C_i and C_j closest to each other, neighbor joining at the same time
  - Minimises the distance between clusters C_i and C_j and
  - Maximises the separation of both C_i and C_j from other clusters

Neighbor joining algorithm

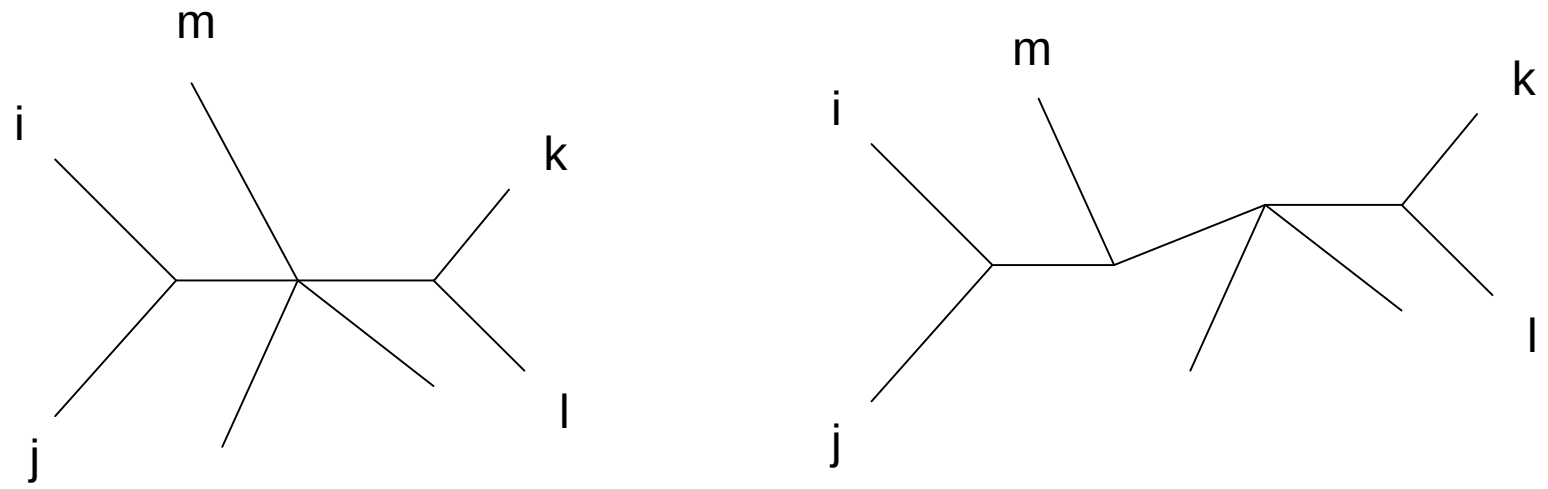
- Start with a star-shaped tree with n leaves and a hub node (see next slide), $n \geq 3$
- Iteration
  - Find nodes i and j *connected to the hub* for which $d_{ij} - u(C_i) - u(C_j)$ is minimal
  - Define new node k with edges i->k, j->k and k->hub, and define d_{kl} for all l
  - Assign length $\frac{1}{2} d_{ij} + \frac{1}{2} (u(C_i) - u(C_j))$ to the edge i -> k
  - Assign length $\frac{1}{2} d_{ij} + \frac{1}{2} (u(C_j) - u(C_i))$ to the edge j -> k
- Termination:
  - When the hub node has three edges

Creating a new branch



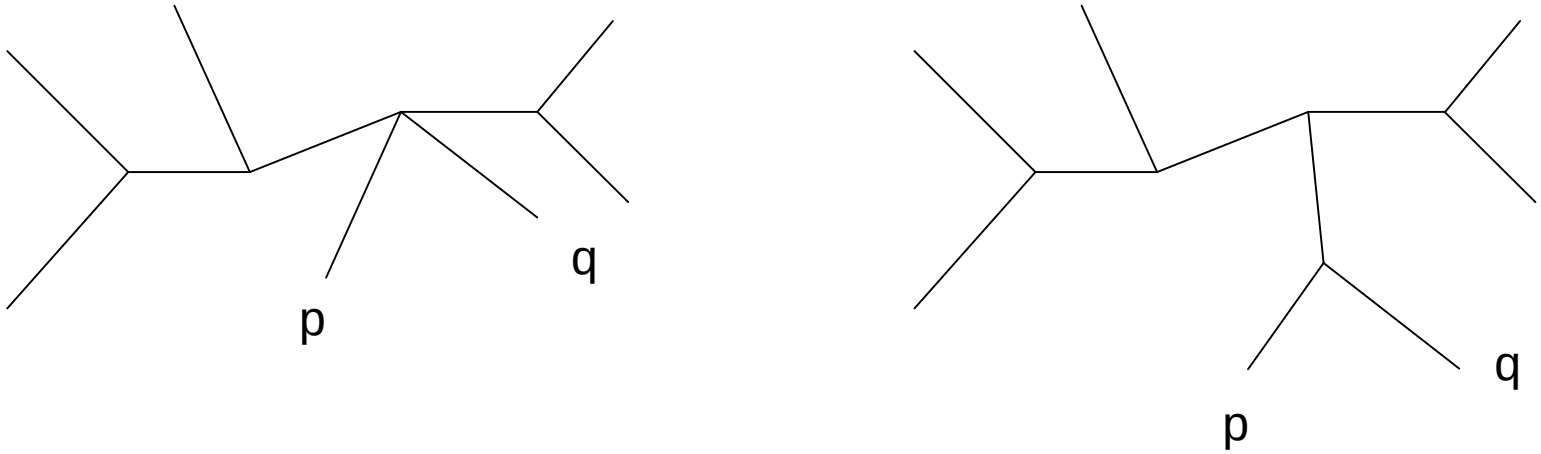
The figure shows first the merging of species i and j, and then k and l:
Each merging creates a new internal branch.

Creating a new branch



Merging (i, j) with m creates another internal branch.

Termination



Algorithm terminates when the hub node has three edges.

Assigning lengths to edges

- Distances d_{kx} from the new node k to the other nodes in the graph x are defined as $d_{kx} = ½ (d_{ix} + d_{jx} – d_{ij})$