Gene expression & Clustering

Determining gene function

- Sequence comparison tells us if a gene is similar to another gene, e.g., in a new species
  - Dynamic programming
  - Approximate pattern matching
- Genes with similar sequence likely to have similar function
- Doesn't always work.
  - "Homologous" genes may not be similar enough at the sequence level, to be detected this way
- New method to determine gene function: directly measure gene activity (DNA arrays)

DNA Arrays--Technical Foundations

- An array works by exploiting the ability of a given mRNA molecule to hybridize to the DNA template.

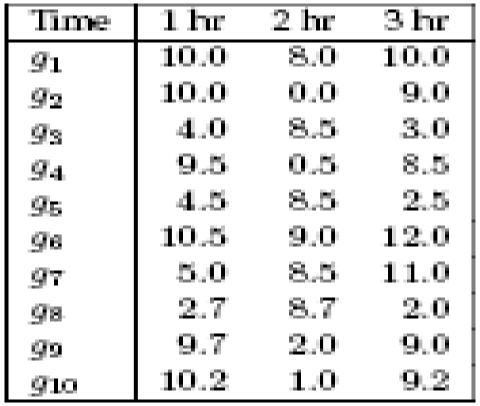
- Using an array containing many DNA samples in an experiment, the expression levels of hundreds or thousands genes within a cell by measuring the amount of mRNA bound to each site on the array.

- With the aid of a computer, the amount of mRNA bound to the spots on the microarray is precisely measured, generating a profile of gene expression in the cell.

Gene expression

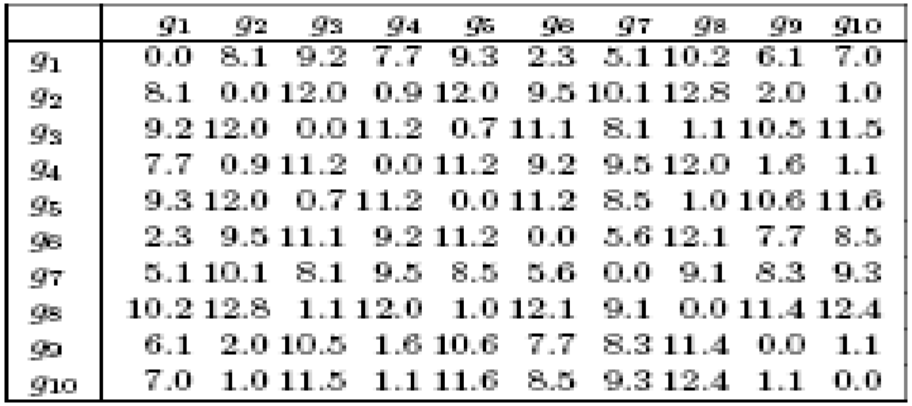
- Microarray gives us an *n* x *m* expression matrix ***I***
  - Each of n rows corresponds to a gene
  - Each of m columns corresponds to a condition or time point
  - Each column comes from one microarray
- ***I***(*j*,*k*) is the expression level of gene *j* in condition/experiment *k*
- If two genes (rows) have similar "expression profiles", then
  - they may be related in function
  - they may be "co-regulated"

Clustering of Microarray Data



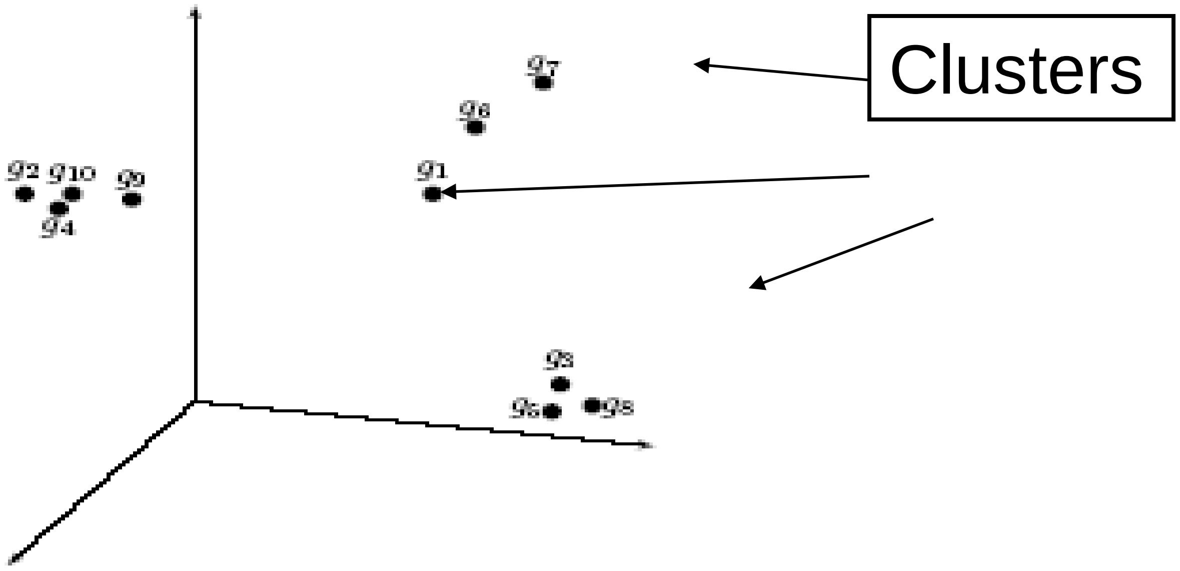
Time	1 hr	2 hr	3 hr
g_1	10.0	8.0	10.0
g_2	10.0	0.0	9.0
g_3	4.0	8.5	3.0
g_4	9.5	0.5	8.5
g_5	4.5	8.5	2.5
g_6	10.5	9.0	12.0
g_7	5.0	8.5	11.0
g_8	2.7	8.7	2.0
g_9	9.7	2.0	9.0
g_{10}	10.2	1.0	9.2

(a) Intensity matrix, I



	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0
g_3	9.2	12.0	0.0	11.2	0.7	11.1	8.1	1.1	10.5	11.5
g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1
g_5	9.3	12.0	0.7	11.2	0.0	11.2	8.5	1.0	10.6	11.6
g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0

(b) Distance matrix, d



(c) Expression patterns as points in three-dimentsional space.

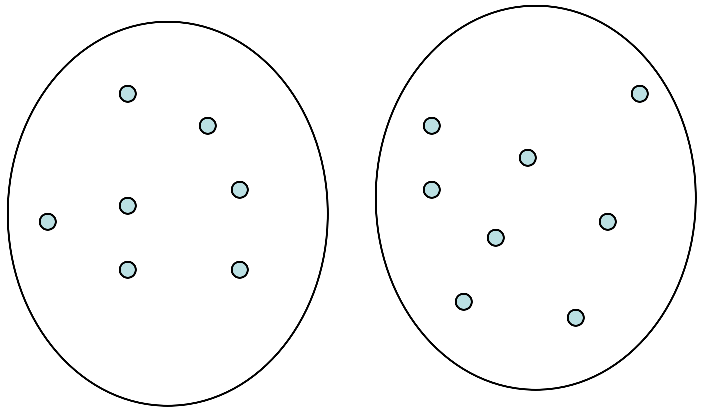
Clustering

- Find groups of genes that have similar expression profiles to one another
- Such groups may be functionally related, and/or co-regulated
- Compute pairwise distance metric $d(i,j)$ for every pair of genes i and j
- This gives an n x n “distance matrix” $\boldsymbol{d}$

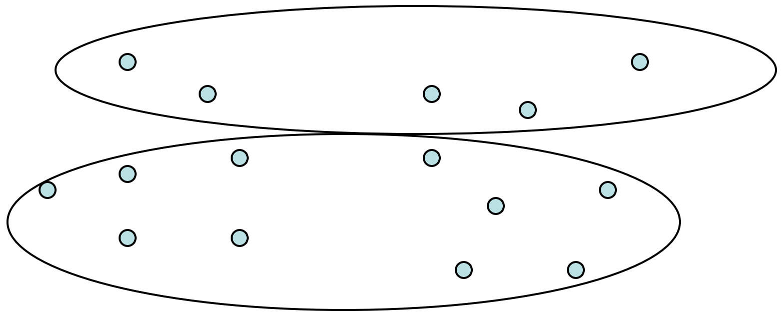
Goal of clustering

- To group together genes into clusters such that
  - Genes within a cluster have highly similar expression profiles (small $d(i,j)$): “**homogeneity”**
  - Genes in different clusters have very different expression profiles (large $d(i,j)$): “**separation”**
- “Good” clustering is one that adheres to these goals
- A really “good” clustering is decided by biological interpretation of the clusters

The Points are in some multi-dimensional space



Good Clustering



Bad Clustering

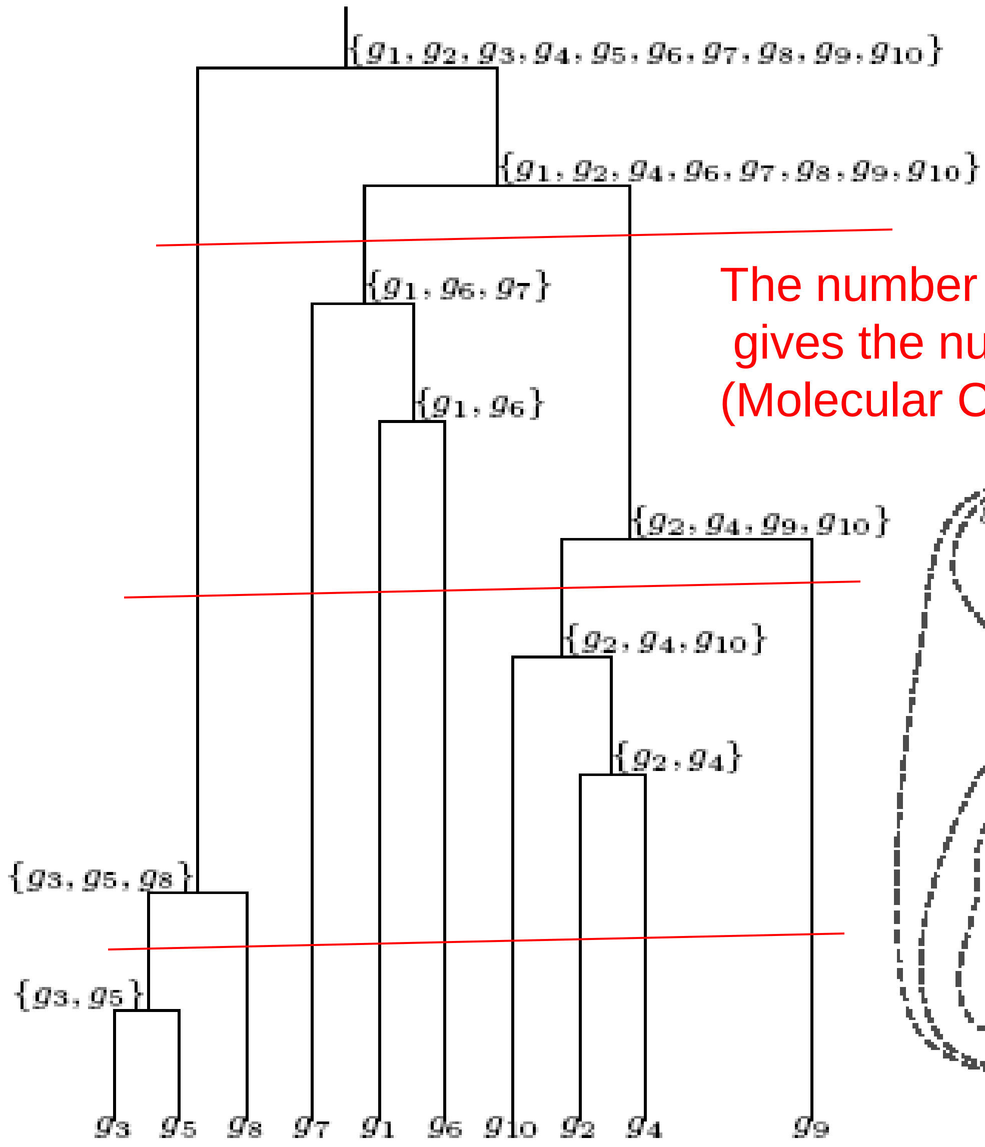
Clustering problems

- How to measure distance/similarity ?
- How many clusters ?
- Very large data sets: ~10,000 gene, ~100 conditions create computational difficulties

Hierarchical clustering

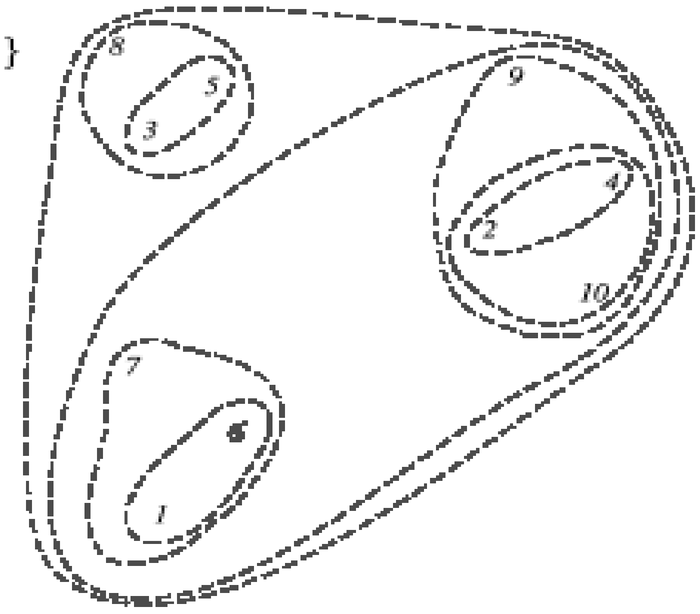
- One approach to clustering
- Does not explicitly partition genes into groups
- Organizes genes into a tree; genes are at the leaves of the tree
- Edges have lengths
- Total path length between two genes (leaves) correlates with the distance between the genes

Hierarchical Clustering



$\{g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, g_9, g_{10}\}$
$\{g_1, g_2, g_4, g_6, g_7, g_8, g_9, g_{10}\}$
$\{g_1, g_6, g_7\}$
$\{g_1, g_6\}$
$\{g_2, g_4, g_9, g_{10}\}$
$\{g_2, g_4, g_{10}\}$
$\{g_2, g_4\}$
$\{g_3, g_5, g_8\}$
$\{g_3, g_5\}$
g_3 g_5 g_8 g_7 g_1 g_6 g_{10} g_2 g_4 g_9

The number of vertical lines intersecting a red line gives the number of sub-clusters at that point.
(Molecular Clock : length of vertical lines



Hierarchical Clustering Algorithm

```
1.  Hierarchical Clustering ($d$ , $n$)
2.    Form $n$ clusters each with one element
3.    Construct a graph $T$ by assigning one vertex to each cluster
4.    **while** there is more than one cluster
5.      Find the two closest clusters $C_1$ and $C_2$
6.      Merge $C_1$ and $C_2$ into new cluster $C$ with $|C_1| + |C_2|$ elements
7.      **Compute distance from $C$ to all other clusters**
8.      Add a new vertex $C$ to $T$ and connect to vertices $C_1$ and $C_2$
9.      Remove rows and columns of $d$ corresponding to $C_1$ and $C_2$
10.     Add a row and column to $d$ corresponding to the new cluster $C$
11.   return $T$
```

Different ways to define distances between clusters may lead to different clusterings

Hierarchical Clustering: Recomputing Distances

- $d_{min}(C, C^*) = \min\ d(x,y)$
  *for all elements x in C and y in C**

  – Distance between two clusters is the **smallest** distance between any pair of their elements

- $d_{avg}(C, C^*) = (1 / |C^*||C|) \sum d(x,y)$
  *for all elements x in C and y in C**

  – Distance between two clusters is the **average** distance between all pairs of their elements

K-means clustering

- Another popular solution to the clustering problem
- Guess a number *k*, which is the number of clusters that will be reported
- Finds explicit clusters (unlike hierarchical clustering)

The Problem

- Given a set of *n* data points $V = (v_1, v_2, \dots v_n)$
  - Each data point is a vector in an m-dimensional space- and an integer *k,* the problem is to determine a set of *k* points or *centers* in m-dimensional space that minimizes the squared error distortion as defined below.
- Let $X=(x_1,x_2,..,x_k)$ be a set of *k* points in the same vector space
- For each v_i, find $x \in X$ that is closest to it, i.e., the Euclidian distance $d(v_i,x)$ is least
- Sum the square of this Euclidian distance, over all v_i
- Formally,

$$d(v_i, X) = \min_{x \in X} d(v_i, x)$$

$$d(V, X) = \frac{\sum_{i}^{n} d(v_i, X)^2}{n}$$

Objective function

- Find *X* such that *d*(*V*,*X*) is minimized
- **Input**: A set, ***V***, consisting of *n* points and a parameter *k*
- **Output**: A set ***X*** consisting of *k* points (*cluster centers*) that minimizes the *d(**V**,**X**)* over all possible choices of ***X***
- An NP-complete problem for k > 1

K-Means Clustering: Lloyd Algorithm

```
1. Lloyd Algorithm
2.    Arbitrarily assign the $k$ cluster centers
3.    while the cluster centers keep changing
4.       Assign each data point to the cluster $C_i$
            corresponding to the closest cluster
               representative (center)  $(1 \le i \le k)$
5.       After the assignment of all data points,
         compute new cluster representatives
         according to the center of gravity of each
            cluster, that is, the new cluster
            representative is
         $\Sigma v \,/\, |C|$  for all $v$ in $C$   for every cluster $C$
```

*This may lead to merely a locally optimal clustering.

Conservative K-Means Algorithm

- Lloyd algorithm is fast but in each iteration it moves many data points, not necessarily causing better convergence.
- A more conservative method would be to move one point at a time only if it improves the overall **clustering cost**
    - The smaller the clustering cost of a partition of data points is, the better that clustering is
    - Different methods (e.g.,d(V,X) we saw earlier) can be used to measure this clustering cost

K-Means "Greedy" Algorithm

```
1.  ProgressiveGreedyK-Means($k$)
2.  Select an arbitrary partition $P$ into $k$ clusters
3.  while forever
4.     $bestChange \leftarrow 0$
5.     for every cluster $C$
6.        for every element $i$ not in $C$
7.           if moving $i$ to cluster $C$ reduces its clustering cost
8.              if $(cost(P) - cost(P_{i \rightarrow C}) > bestChange$
9.                 $bestChange \leftarrow cost(P) - cost(P_{i \rightarrow C})$
10.                $i^* \leftarrow i$
11.                $C^* \leftarrow C$
12.    if $bestChange > 0$
13.       Change partition $P$ by moving $i^*$ to $C^*$
14.    else
15.       return $P$
```

UPGMA: Unweighted Pair Group Method with Arithmetic Mean

- UPGMA is a clustering algorithm that:
  - computes the distance between clusters using average pairwise distance
  - assigns a *height* to every vertex in the tree
- Does not require an additive distance matrix. Output tree does not necessarily match the distance matrix for every pair of nodes

Clustering in UPGMA

- Given two disjoint clusters C_i, C_j of sequences,

$$d_{ij} = \frac{1}{|C_i| \times |C_j|} \Sigma_{\{p \in Ci,\ q \in Cj\}} d_{pq}$$

- Algorithm is a variant of the hierarchical clustering algorithm

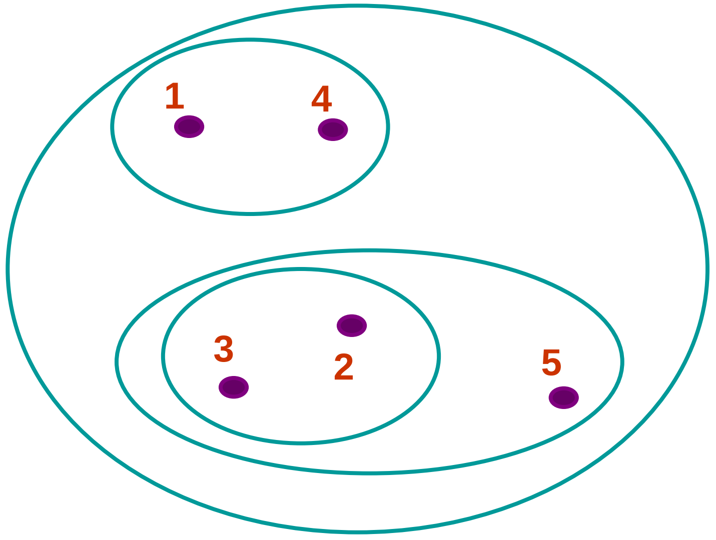
UPGMA Algorithm

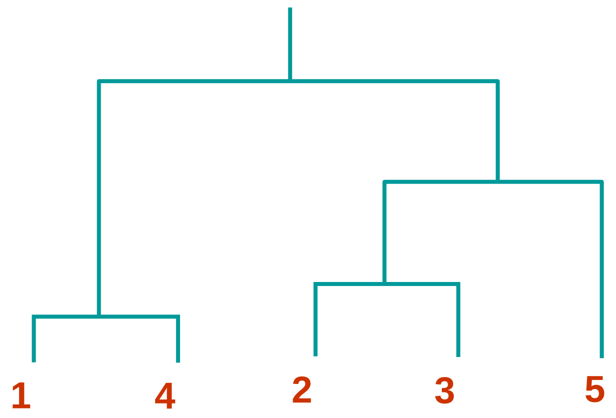
```
UPGMA ($D$, $n$)
1.  Form $n$ clusters each with a single element
2.  Construct a graph $T$ by assigning one vertex to each cluster
3.  Assign height $h(v)=0$ to every vertex $v$ in the graph
4   while there is more than one cluster
5.      Find the two closest clusters $C_1$ and $C_2$
6.      Merge $C_1$ and $C_2$ into new cluster $C$ with $|C_1| + |C_2|$ elements
7.      for every cluster C* ≠ C compute

            $$D(C,C^*) = \frac{1}{|C||C^*|} \sum_{i \text{ in C}} \sum_{j \text{ in C}^*} D(i,j)$$

8.      Add a new vertex $C$ to $T$ and connect to vertices $C_1$ and $C_2$
9.          $$h(C) \leftarrow \frac{D(C_1,C_2)}{2}$$
10.     Assign length $h(C) - h(C_1)$ to edge $(C_1,C)$
11.     Assign length $h(C) - h(C_2)$ to edge $(C_2,C)$
12.     Remove rows and columns of $D$ corresponding to $C_1$ and $C_2$
13.     Add a row and column to $D$ corresponding to the new cluster $C$
14. return $T$
```

UPGMA Algorithm (cont'd)





UPGMA’s Weakness

- The algorithm produces an **ultrametric** tree : the distance from the root to any leaf is the same
- UPGMA assumes a constant molecular clock: all species represented by the leaves in the tree are assumed to accumulate mutations (and thus evolve) at the same rate. This is a major pitfall of UPGMA.

Neighbor Joining Algorithm

The neighbor joining algorithm works well with additive trees, does not assume any molecular clock and generates clusters that satisfy homogeneity and separation criteria .
For a cluster *C* , define *u*(*C*) as

$$u(C) = \frac{1}{\text{number of clusters - 2}} \sum_{\text{all clusters } C'} D(C, C')$$

To choose which two clusters to merge, the algorithm minimizes $D(C_1, C_2) - u(C_1) - u(C_2)$

(The theory behind this algorithm is complex and is left out here.)

NeighborJoining (*D*,*n*)

```
1.  Form $n$ clusters, each with a single element
2.  Construct a graph $T$ by assigning a vertex to each cluster
3.  while there is more than one cluster
4.      Find clusters $C_1,C_2$ minimizing $D(C_1,C_2)-u(C_1)-u(C_2)$
5.      Merge $C_1,C_2$ into a new cluster $C$ of size $|C_1|,+|C_2|$
6       Compute $D(C,C^*)=\frac{D(C_1,C)+D(C_2,C)}{2}$ to every other cluster C*
7       Add vertex $C$ to $T$ and connect to vertices $C_1$ and $C_2$
8.      Assign length $\frac{1}{2}D(C_1,C_2)+\frac{1}{2}(u(C_1)-u(C_2))$ to edge $(C_1,C)$
9.      Assign length $\frac{1}{2}D(C_1,C_2)+\frac{1}{2}(u(C_2)-u(C_1))$ to edge $(C_2,C)$
10.     Remove rows and columns of $D$ corresponding to $C_1$ and $C_2$
11.     Add a row and column to $D$ for the new cluster $C$
12. Return $T$
```

Sources

- http://www.math.tau.ac.il/~rshamir/ge/02/scribes/lec01.pdf
- http://bioinformatics.oupjournals.org/cgi/screenpdf/20/3/340.pdf
- http://www.absoluteastronomy.com/encyclopedia/M/Mi/Minimum_spanning_tree.htm
- Serafim Batzoglou (UPGMA slides) http://www.stanford.edu/class/cs262/Slides
- Watkins, W.S., Rogers A.R., Ostler C.T., Wooding, S., Bamshad M. J., Brassington A.E., Carroll M.L., Nguyen S.V., Walker J.A., Prasas, R., Reddy P.G., Das P.K., Batzer M.A., Jorde, L.B.: Genetic Variation Among World Populations: **Inferences From 100 *Alu* Insertion Polymorphisms**