Bioinformatics Algorithms

Ankeeta Shah
Lecture 16
Spring 2016
BC3308: Genomics and Bioinformatics

Computer science is no more about computers than astronomy is about telescopes.

— Edsger Dijkstra, 1970.

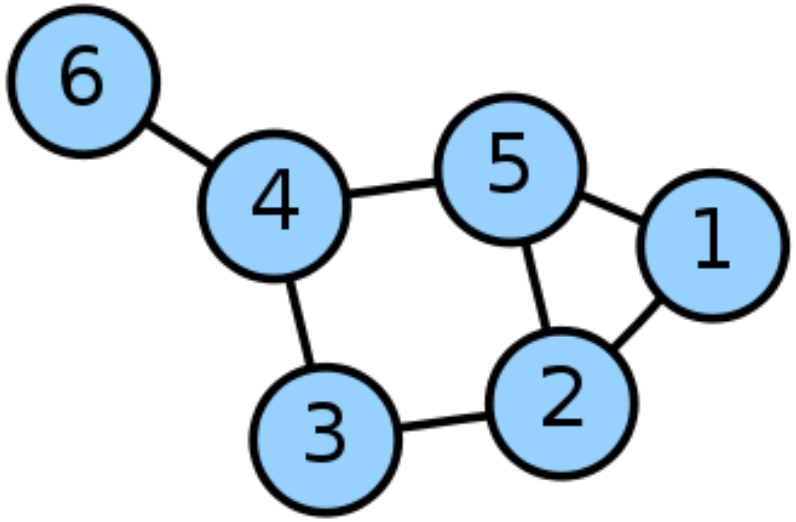
Computer science is no more about computers
~~than astronomy is about telescopes.~~

than biology is about microscopes.

— Edsger Dijkstra, 1970.

Algorithms

Definition: an algorithm is a finite sequence$_1$ of unambiguous$_2$ and effectively computable$_3$ instructions that produce some intended result$_4$



Two “Solved” Biological Problems

1. Alignment

a way of arranging the sequences of DNA, RNA, or protein to identify regions of similarity

DYNAMIC PROGRAMMING

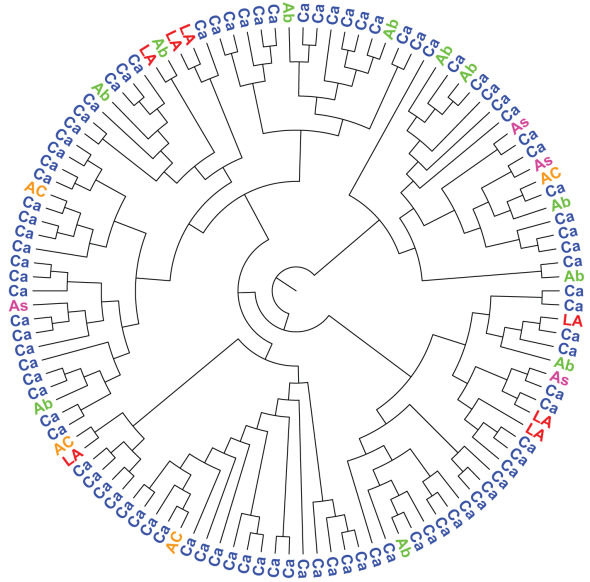
```
GAATTCAG      GAATTCAG
|  |  ||  |    |  ||  |  |
GGA-TC-G      GCAT-C-G

GAATTC-A      GAATTC-A
|  |  ||  |    |  ||  |  |
GGA-TCGA      GCAT-CGA
```

2. Clustering

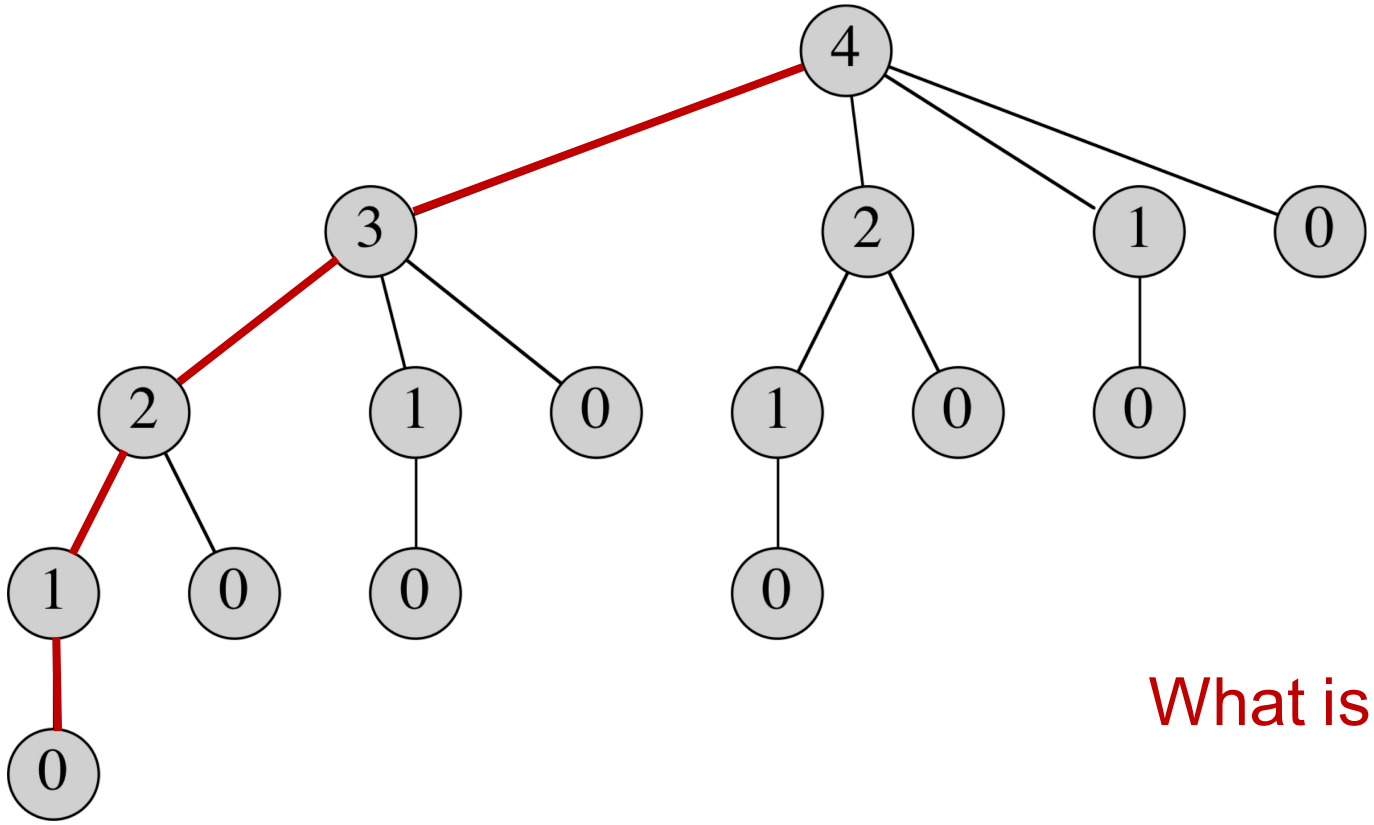
a measure of the similarity between clusters that should be combined

HIERARCHICAL CLUSTERING



Solving the alignment problem: Dynamic Programming (DP)

“Divide the problem into subproblems”

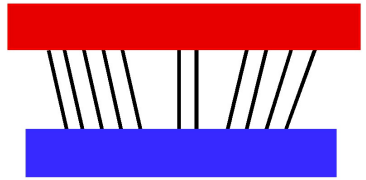


What is recursion?

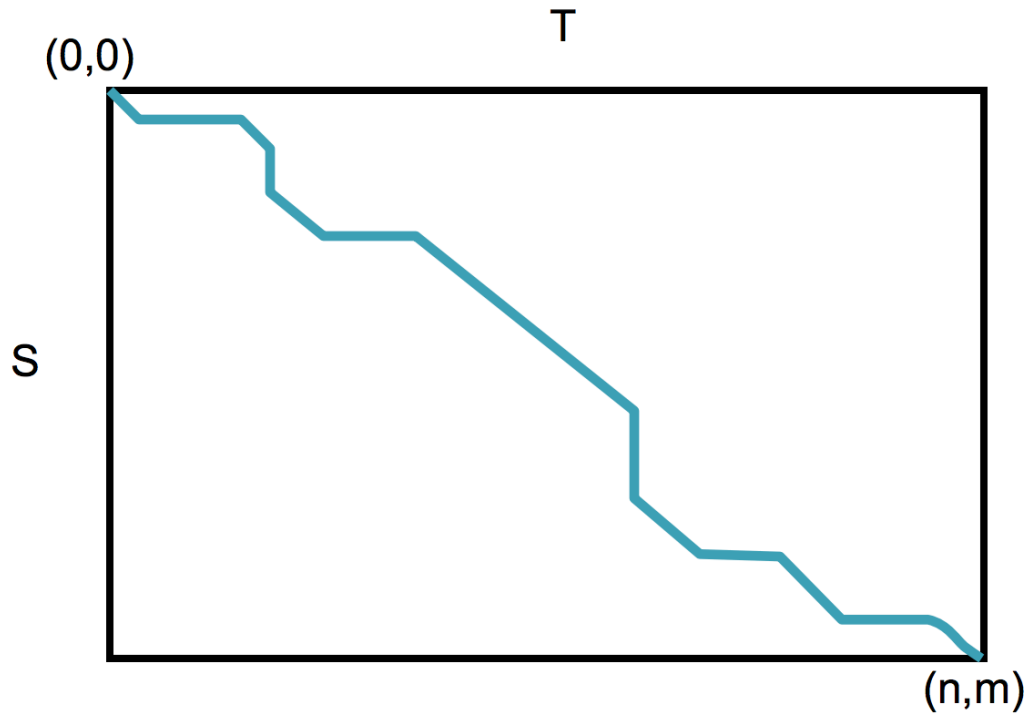
Solving the alignment problem: Scoring Matrix

		Score
Match	ACGTCTAG ACGTCTAG	+1
Mismatch (substitution)	ACGTCTAG ACGACTAA	-1
Insertion/deletion (indel/gap)	ACGTCT-G -CGTCTAG	-2

Solving the alignment problem: Algorithms

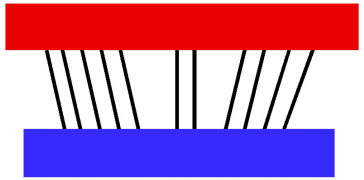


A. Global alignment: Needleman-Wunsch Algorithm



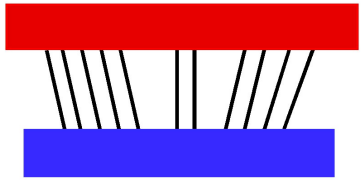
to find the best (optimal) path between vertices (0,0) and (n,m) in the matrix

Solving the alignment problem: Global Alignment



- Given two sequences, **S (length n)** and **T (length m)**, find the best end-to-end alignment of S and T
- Let **F(i,j) be the partial score** of the alignment of S[1...i] and T[1...j].
- **Maximum score** of the alignment of S and T (end-to-end) is **F(n,m)**.

Solving the alignment problem: Global Alignment Algorithm



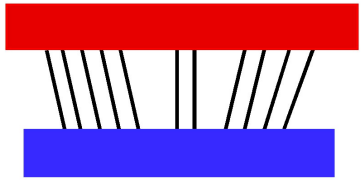
Step 0: Pre-Initialization

- Let sequence S be of length 2 (n = 2)
- Let sequence T be of length 3 (m = 3)

Draw your matrix to be (n+1)x(m+1)

		n_0	n_1
m_0			
m_1			
m_2			

Solving the alignment problem: Global Alignment Algorithm



Step 1: Initialization

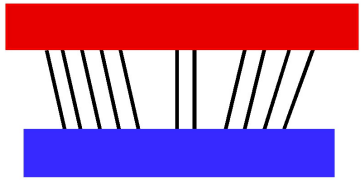
Scoring System

Match = +1
Mismatch = -1
Gap = -2

Fill the upper left corner with a 0;
Fill the matrix with the gap penalty first

		n_0	n_1
	0	-2	-4
m_0	-2		
m_1	-4		
m_2	-6		

Solving the alignment problem: Global Alignment Algorithm



Step 2: Use DP to fill the rest of the matrix

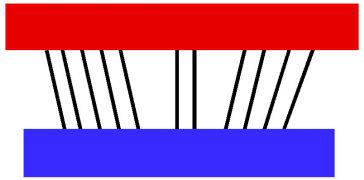
Scoring System

Match = +1
Mismatch = -1
Gap = -2

Remember to look for the **maximum** score to fill in the rest of the cells (partial scores)

		n_0	n_1
	0	-2	-4
m_0	-2	**F(i,j)**	
m_1	-4		
m_2	-6		

Solving the alignment problem: Global Alignment Algorithm



Step 3: Recurse (traceback)

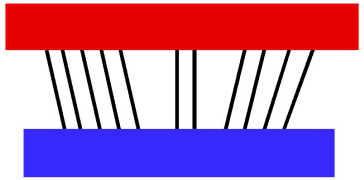
Scoring System

Match = +1
Mismatch = -1
Gap = -2

Find the optimal path via recursion

		n_0	n_1
	0	-2	-4
m_0	-2		
m_1	-4		
m_2	-6		**F(m,n)**

Solving the alignment problem: Global Alignment Algorithm



Step 4: Calculate alignment score

Scoring System

Match = +1
Mismatch = -1
Gap = -2

		n_0	n_1
	0	-2	-4
m_0	-2		
m_1	-4		
m_2	-6		**F(m,n)**

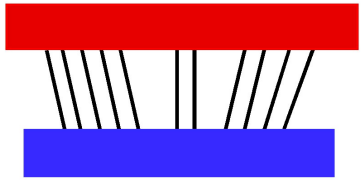
n_0 – n_1

m_0 m_1 m_2

+1 -2 -1

Score (S) = (+1) + (-2) + (-1) = -2

Solving the alignment problem: Global Alignment Example



Given two sequences

AG

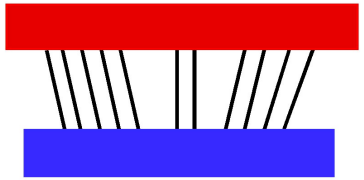
ACG

find the optimal global alignment

Scoring System

Match = +1
Mismatch = -1
Gap = -2

Solving the alignment problem: Global Alignment Example



Given two sequences

AG

ACG

find the optimal global alignment

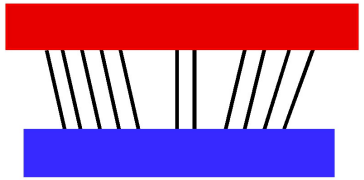
Scoring System

Match = +1
Mismatch = -1
Gap = -2

Step 0: Pre-initialization

		A	G
A			
C			
G			

Solving the alignment problem: Global Alignment Example



Given two sequences

AG

ACG

find the optimal global alignment

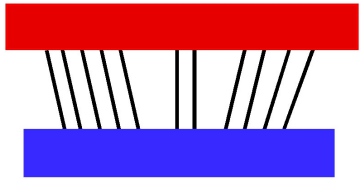
Scoring System

Match = +1
Mismatch = -1
Gap = -2

Step 1: Initialization

		A	G
	0	-2	-4
A	-2		
C	-4		
G	-6		

Solving the alignment problem: Global Alignment Example



Given two sequences

AG

ACG

find the optimal global alignment

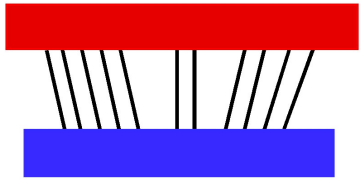
Scoring System

Match = +1
Mismatch = -1
Gap = -2

Step 2: DP to fill matrix

		A	G
	0	-2	-4
A	-2	1	-1
C	-4	-1	0
G	-6	-3	0

Solving the alignment problem: Global Alignment Example



Given two sequences

AG

ACG

find the optimal global alignment

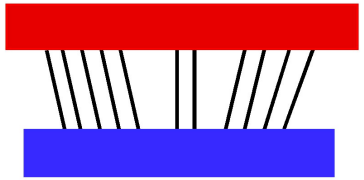
Scoring System

Match = +1
Mismatch = -1
Gap = -2

Step 3: Recurse

		A	G
	0	-2	-4
A	-2	1	-1
C	-4	-1	0
G	-6	-3	0

Solving the alignment problem: Global Alignment Example



Scoring System

Match = +1
Mismatch = -1
Gap = -2

```
A - G
A C G
```

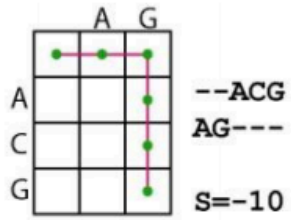
+1 -2 +1

Score (S) = (+1)+(-2)+ (+1) = 0

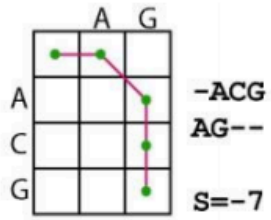
Step 4: Calculate alignment score

		A	G
	0	-2	-4
A	-2	1	-1
C	-4	-1	0
G	-6	-3	0

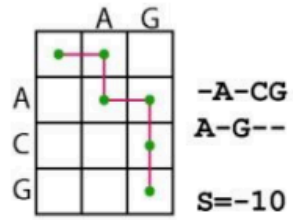
Try the global alignment example on the handout



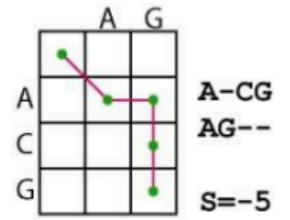
--ACG
AG---
S=-10



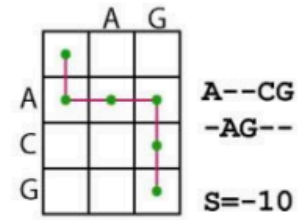
-ACG
AG--
S=-7



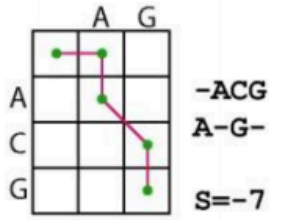
-A-CG
A-G--
S=-10



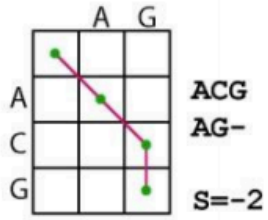
A-CG
AG--
S=-5



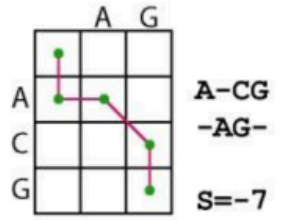
A--CG
-AG--
S=-10



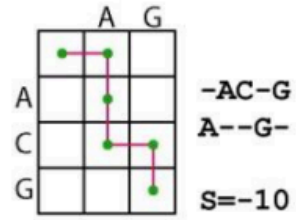
-ACG
A-G-
S=-7



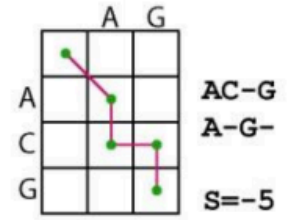
ACG
AG-
S=-2



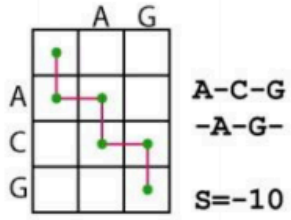
A-CG
-AG-
S=-7



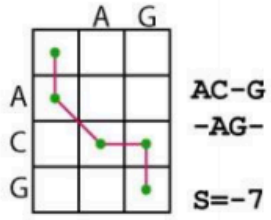
-AC-G
A--G-
S=-10



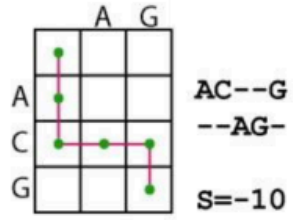
AC-G
A-G-
S=-5



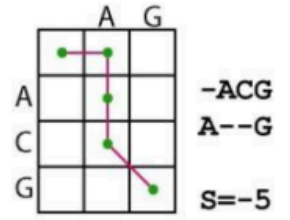
A-C-G
-A-G-
S=-10



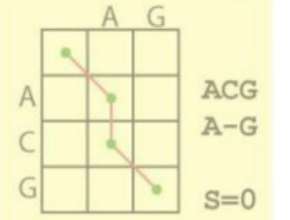
AC-G
-AG-
S=-7



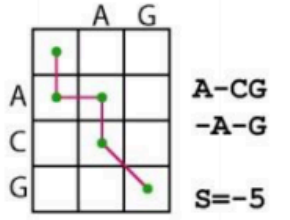
AC--G
--AG-
S=-10



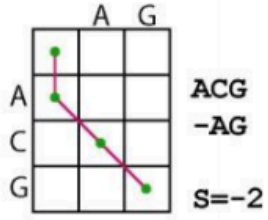
-ACG
A--G
S=-5



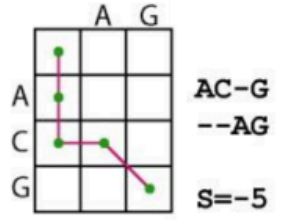
ACG
A-G
S=0



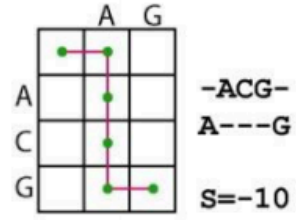
A-CG
-A-G
S=-5



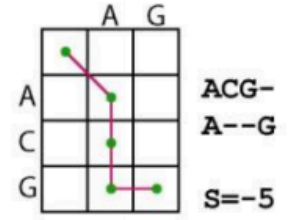
ACG
-AG
S=-2



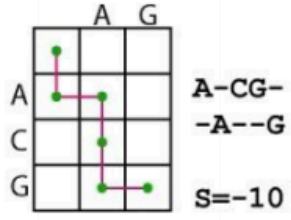
AC-G
--AG
S=-5



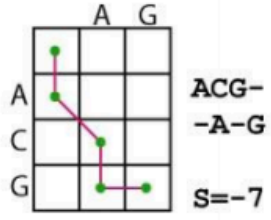
-ACG-
A---G
S=-10



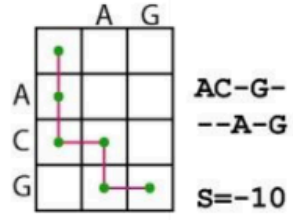
ACG-
A--G
S=-5



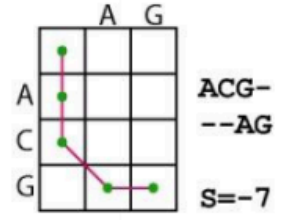
A-CG-
-A--G
S=-10



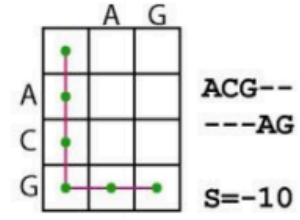
ACG-
-A-G
S=-7



AC-G-
--A-G
S=-10



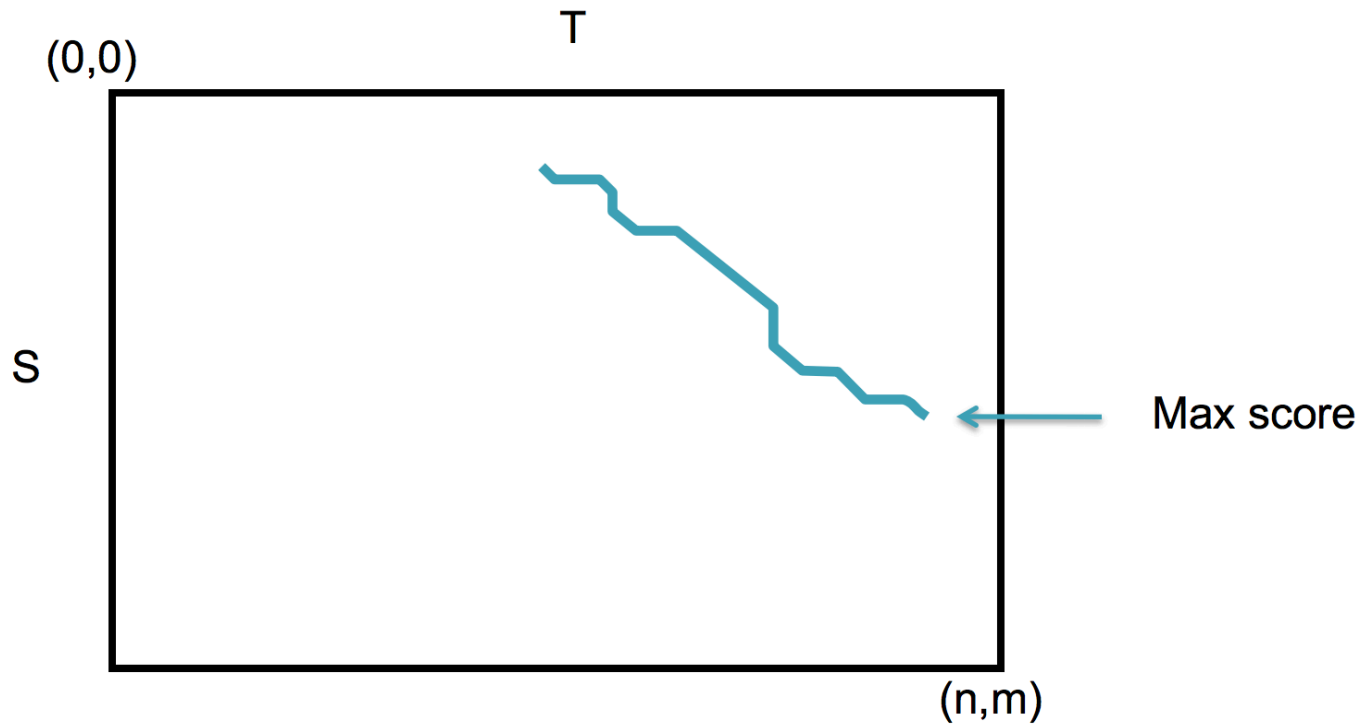
ACG-
--AG
S=-7



ACG--
---AG
S=-10

Solving the alignment problem: Algorithms

B. Local alignment: Smith-Waterman Algorithm (1981)



tries to find the best path between arbitrary vertices (i,j) and (i', j') in the matrix

Solving the alignment problem: Local Alignment Example

Scoring System

Match = +1
Mismatch = -1
Gap = -2

We don't want negative numbers in our matrix.

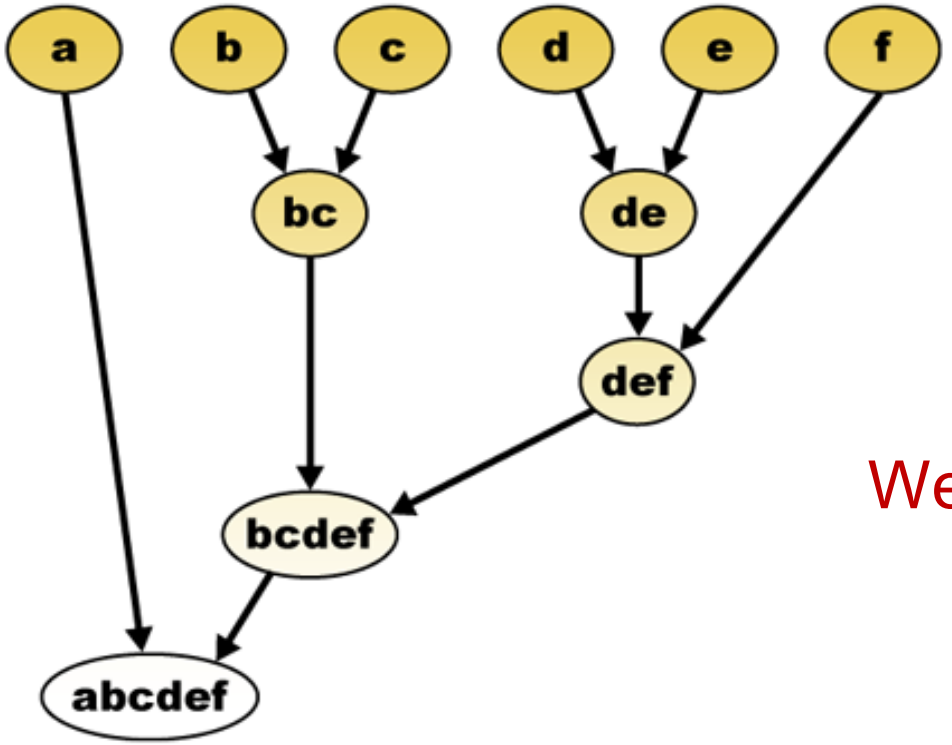
This is the matrix solution for the local alignment of these two sequences.

```
A       G
  and
A       G
```

		A	G
	0	0	0
A	0	1	0
C	0	0	0
G	0	0	1

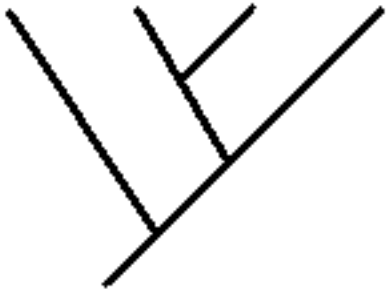
Solving the clustering problem: Hierarchical Clustering (HCA)

”Merge closest (most similar) pair of clusters and merge them into a single cluster”



We use **distance** as the metric by which to cluster

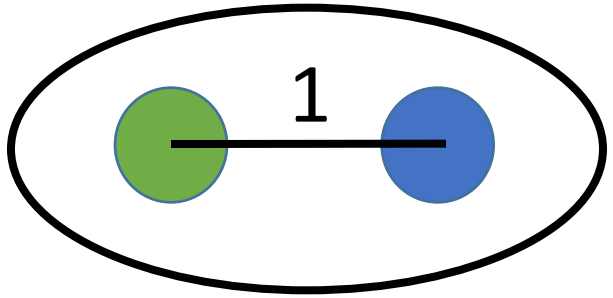
Solving the clustering problem: Two tree building algorithms



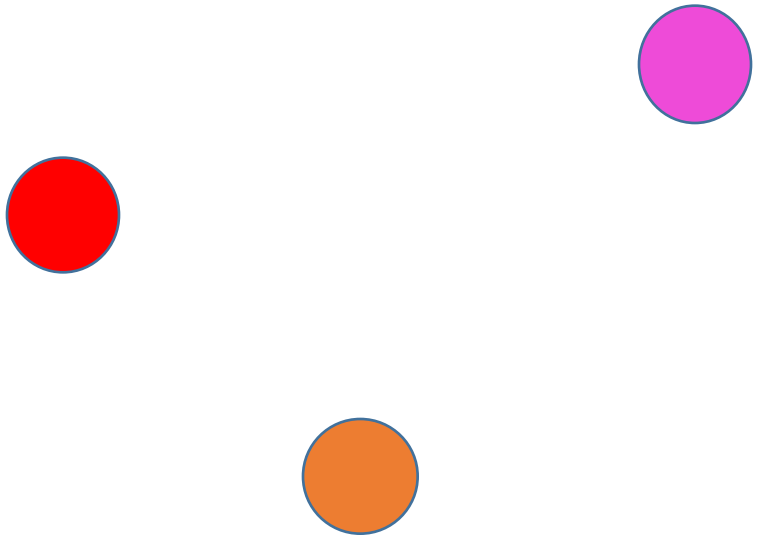
1. Unweighted Pair Group Method using Arithmetic Averages (UPGMA) (1958)

2. Neighbor Joining (NJ) (1987)

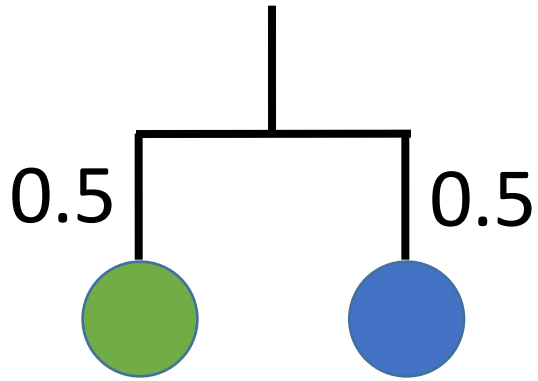
Solving the clustering problem: UPGMA



Step 1: Cluster together the two most similar sequences (i.e. the two sequences that have the shortest distance between them).

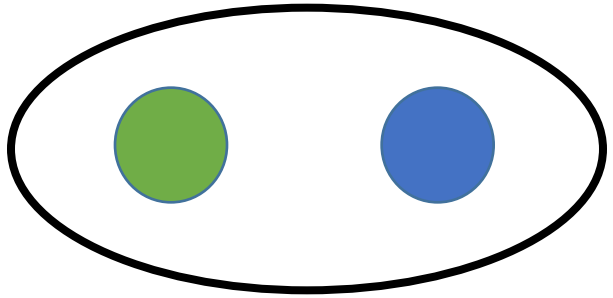


Step 2: Assign the distance between them evenly between the two branches.



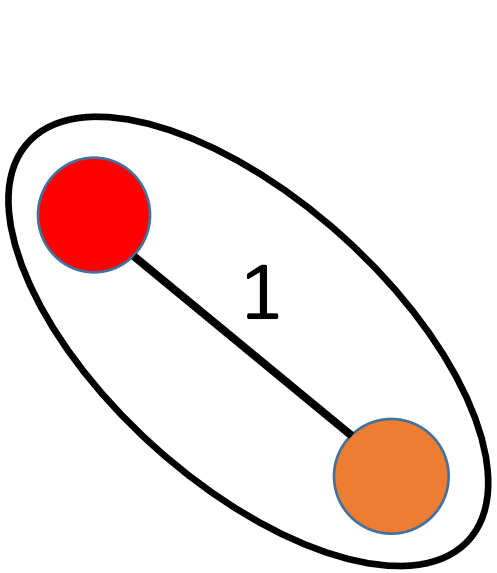
Depth = 0.5

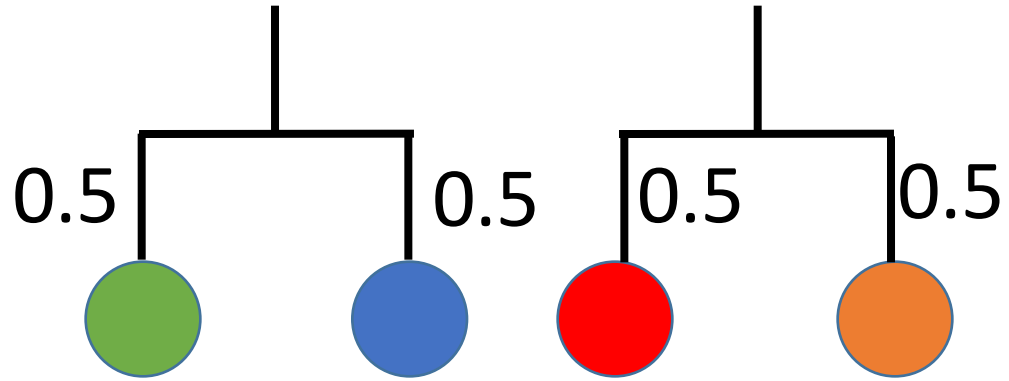
Solving the clustering problem: UPGMA



Step 3: Rewrite the distance matrix, replacing those two sequences with their average

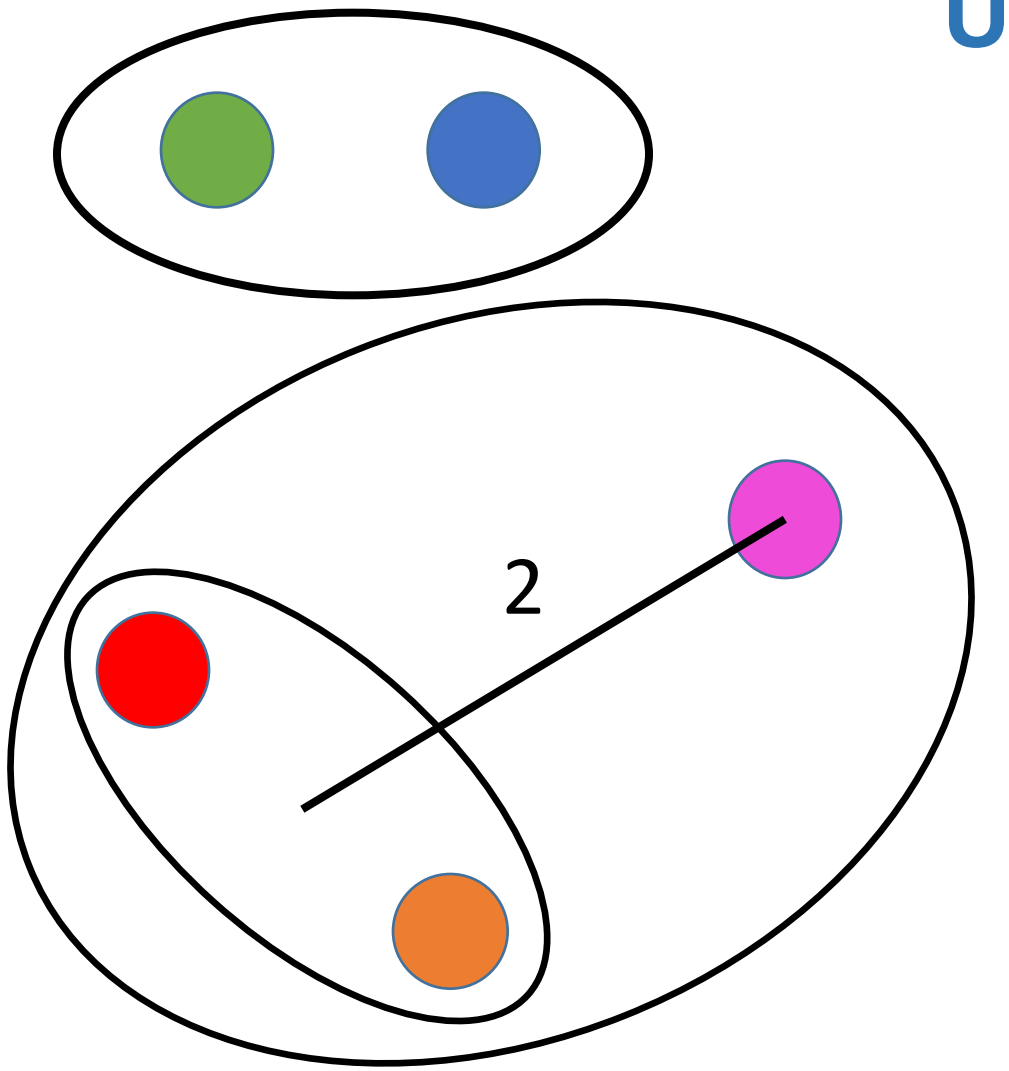
Step 4: Repeat 1, 2, and 3. Notice that the depth of the three is half the longest distance so far.



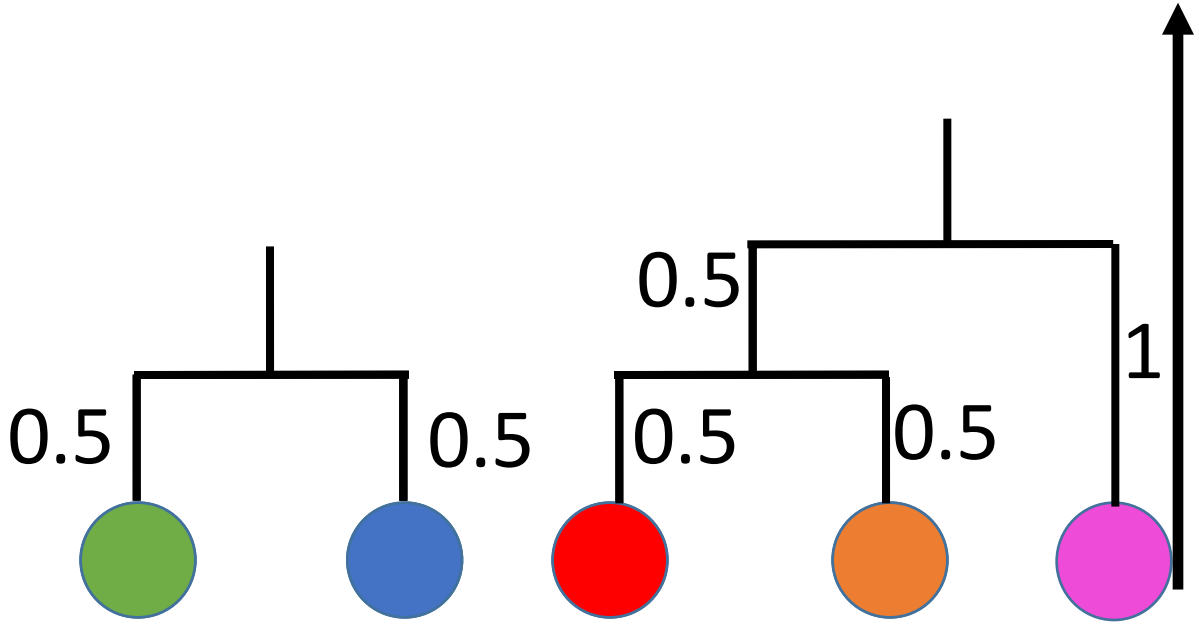


Depth = 0.5

Solving the clustering problem: UPGMA

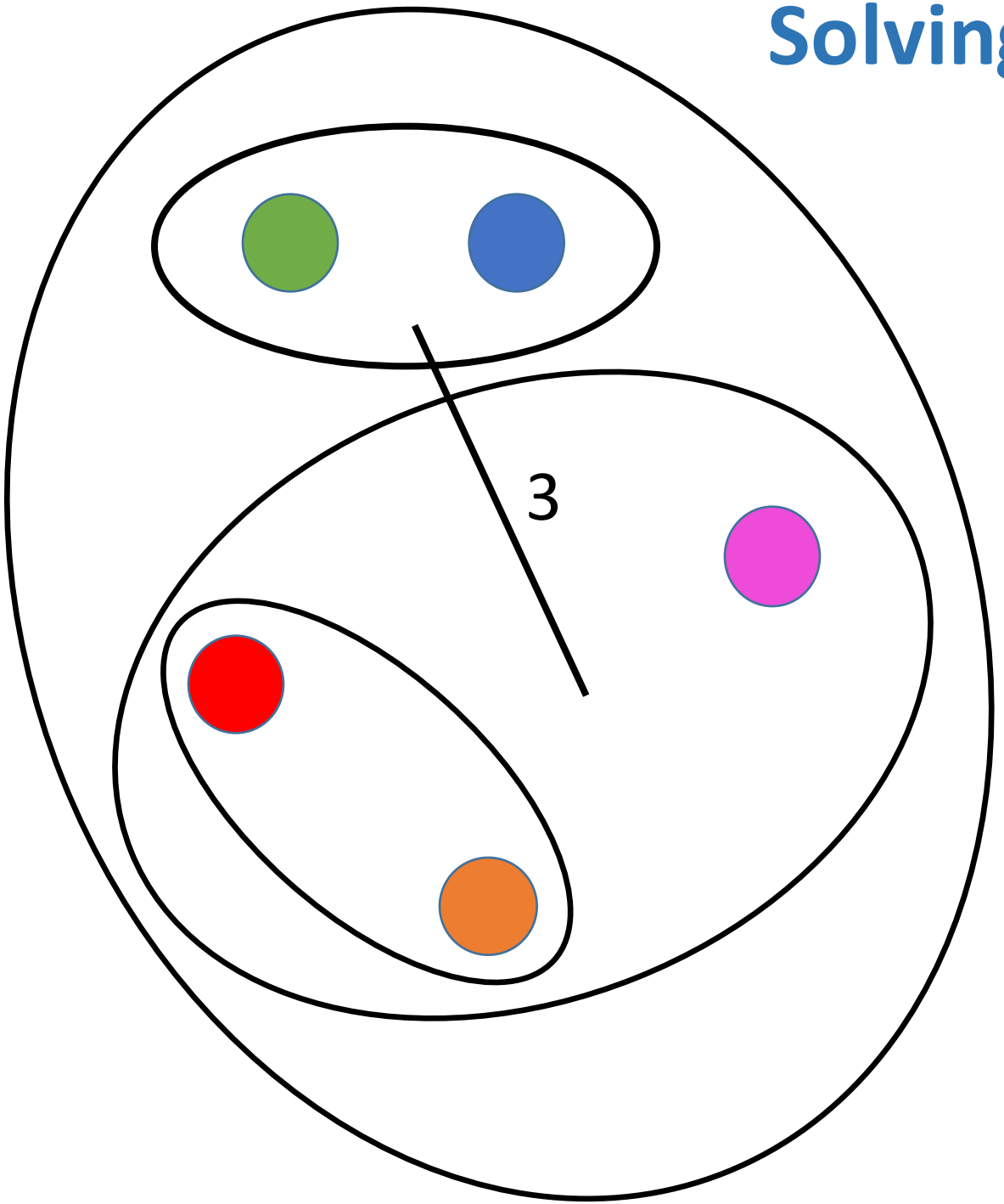


Repeat steps 1-3

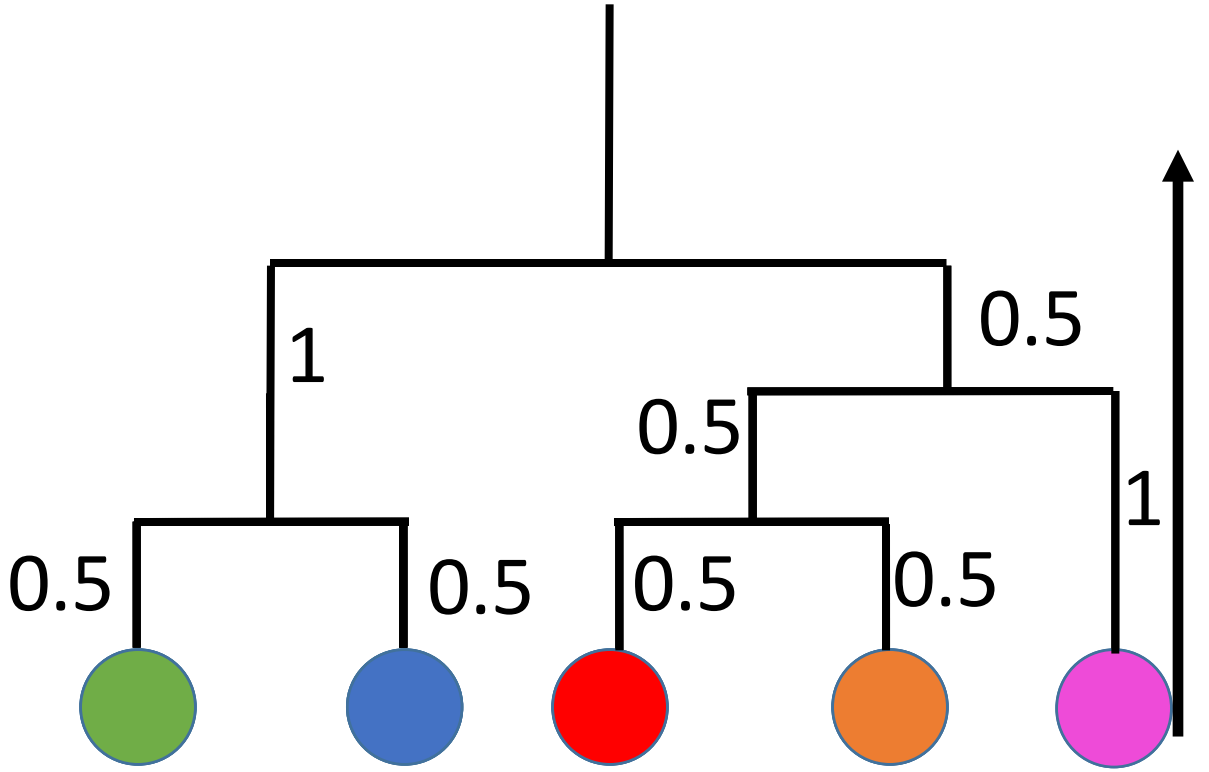


Depth = 1

Solving the clustering problem: UPGMA

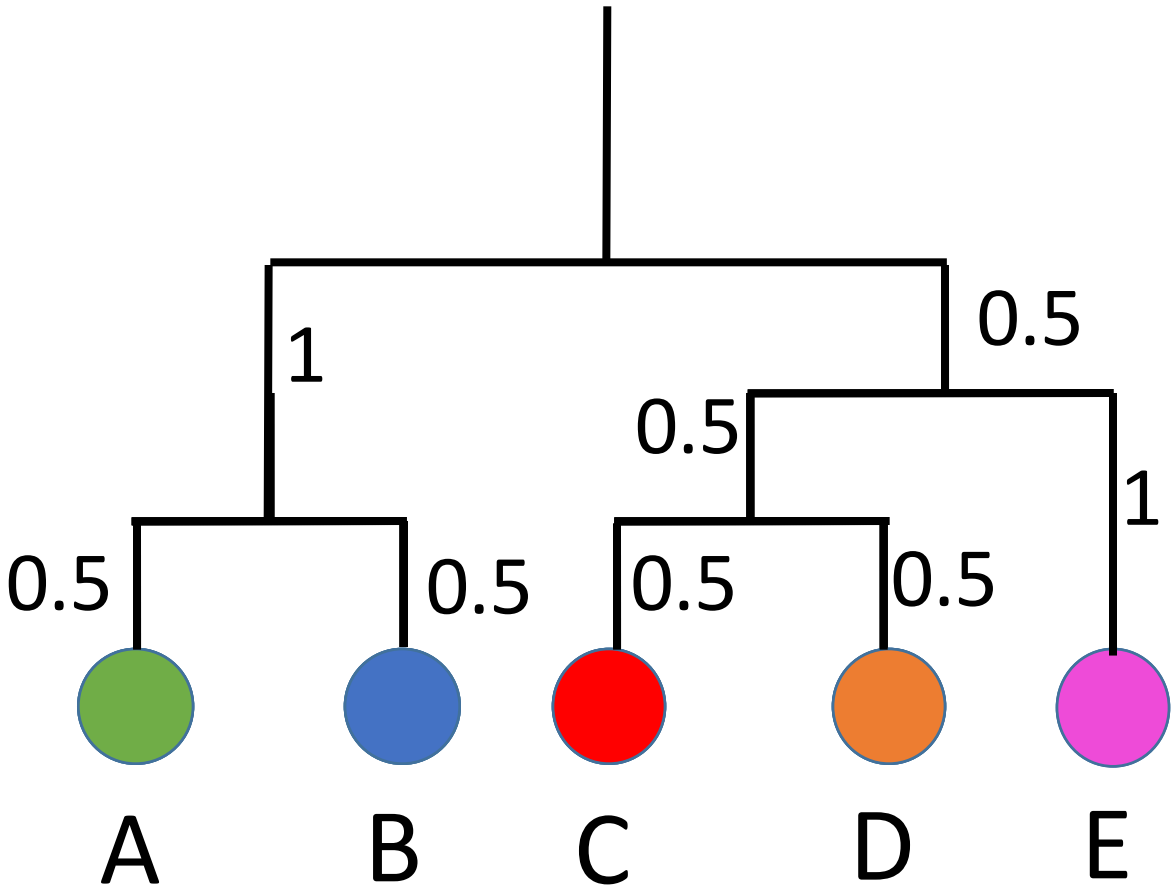


Repeat steps 1-3



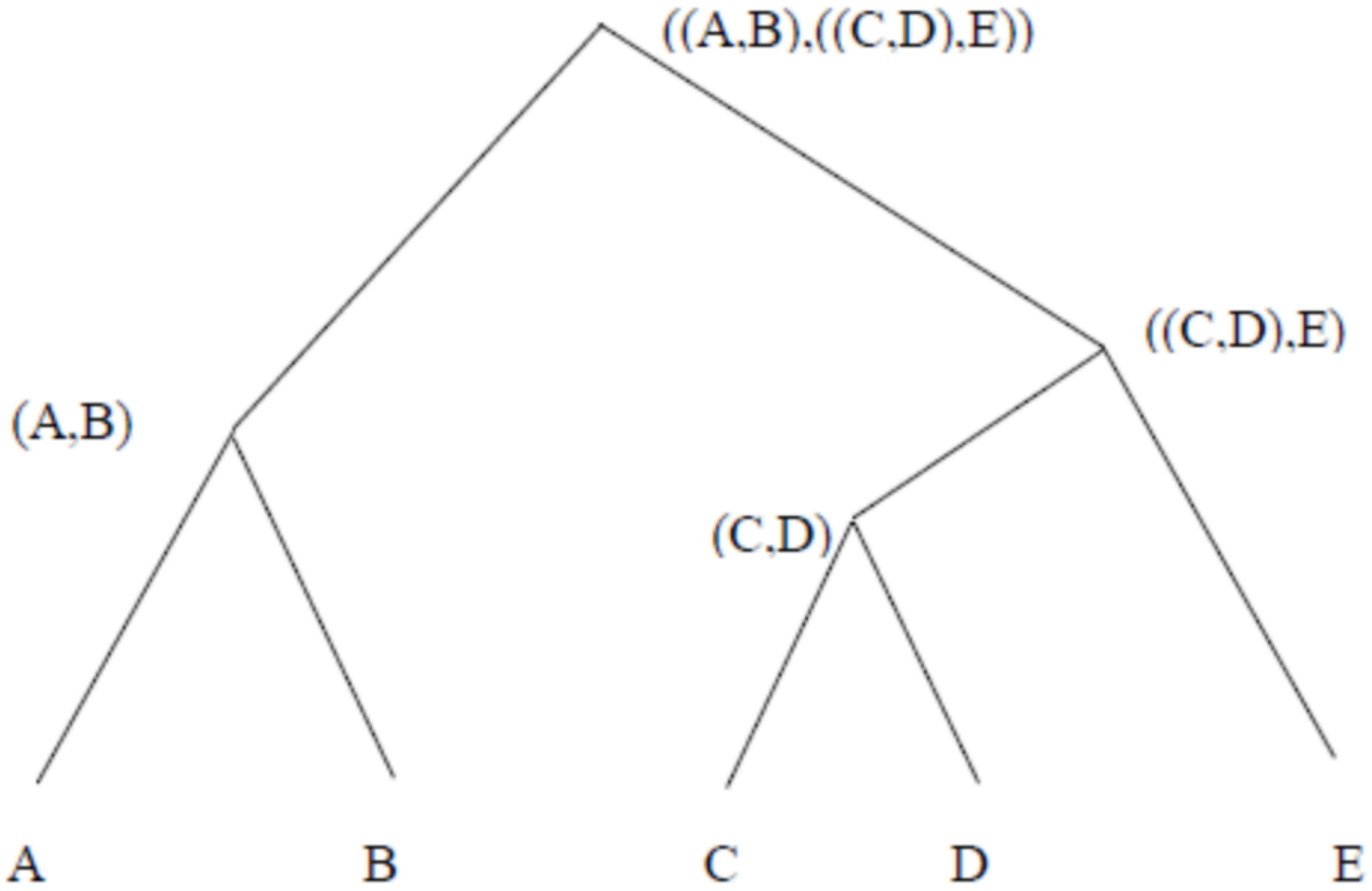
Depth = 1.5

Newick Format



((A,B),((C,D),E))

Newick Format



Solving the clustering problem: UPGMA Distance Matrix Example

	A	B	C	D	E
A	-	5	1	8	9
B	5	-	4	10	11
C	1	4	-	9	9
D	8	10	9	-	2
E	9	11	9	2	-

Solving the clustering problem: UPGMA Distance Matrix Example

	A	B	C	D	E
A	-	5	1	8	9
B	5	-	4	10	11
C	1	4	-	9	9
D	8	10	9	-	2
E	9	11	9	2	-

Step 1: Notice that A and C have the shortest pairwise distance – they should be clustered together

Solving the clustering problem: UPGMA Distance Matrix Example

	A	B	C	D	E
A	-	5	1	8	9
B	5	-	4	10	11
C	1	4	-	9	9
D	8	10	9	-	2
E	9	11	9	2	-

	AC	B	D	E
AC	-			
B		-		
D			-	
E				-

Step 2: Cluster together A and C to form AC, with branch lengths of 0.5 (1 / 2 = 0.5)

Solving the clustering problem: UPGMA Distance Matrix Example

	A	B	C	D	E
A	-	5	1	8	9
B	5	-	4	10	11
C	1	4	-	9	9
D	8	10	9	-	2
E	9	11	9	2	-

	AC	B	D	E
AC	-			
B		-		
D			-	
E				-

Step 3: In order to fill in the new matrix, take the average of the two old distances to get the new distance

Solving the clustering problem: UPGMA Distance Matrix Example

	A	B	C	D	E
A	-	5	1	8	9
B	5	-	4	10	11
C	1	4	-	9	9
D	8	10	9	-	2
E	9	11	9	2	-

	AC	B	D	E
AC	-			
B		-		
D			-	
E				-

Step 3: In order to fill in the new matrix, take the average of the two old distances to get the new distance

Solving the clustering problem: UPGMA Distance Matrix Example

	A	B	C	D	E
A	-	5	1	8	9
B	5	-	4	10	11
C	1	4	-	9	9
D	8	10	9	-	2
E	9	11	9	2	-

	AC	B	D	E
AC	-			
B		-		
D			-	
E				-

Step 3: In order to fill in the new matrix, take the average of the two old distances to get the new distance

Solving the clustering problem: UPGMA Distance Matrix Example

	A	B	C	D	E
A	-	5	1	8	9
B	5	-	4	10	11
C	1	4	-	9	9
D	8	10	9	-	2
E	9	11	9	2	-

(5 + 4) / 2 = 4.5

	AC	B	D	E
AC	-	4.5		
B	4.5	-		
D			-	
E				-

Step 3: In order to fill in the new matrix, take the average of the two old distances to get the new distance

Solving the clustering problem: UPGMA Distance Matrix Example

	A	B	C	D	E
A	-	5	1	8	9
B	5	-	4	10	11
C	1	4	-	9	9
D	8	10	9	-	2
E	9	11	9	2	-

	AC	B	D	E
AC	-	4.5	8.5	9
B	4.5	-	10	11
D	8.5	10	-	2
E	9	11	2	-

Step 4: Repeat this process for all cells in the distance matrix

Solving the clustering problem: UPGMA Distance Matrix Example

	AC	B	D	E
AC	-	4.5	8.5	9
B	4.5	-	10	11
D	8.5	10	-	2
E	9	11	2	-

Cluster together D and E to form DE, with branches of length 1

Solving the clustering problem: UPGMA Distance Matrix Example

	AC	B	D	E
AC	-	4.5	8.5	9
B	4.5	-	10	11
D	8.5	10	-	2
E	9	11	2	-

	AC	B	DE
AC	-		
B		-	
DE			-

Cluster together D and E to form DE, with branches of length 1

Solving the clustering problem: UPGMA Distance Matrix Example

	AC	B	D	E
AC	-	4.5	8.5	9
B	4.5	-	10	11
D	8.5	10	-	2
E	9	11	2	-

	AC	B	DE
AC	-	4.5	8.75
B	4.5	-	10.5
DE	8.75	10.5	-

Fill new distance matrix as before

Solving the clustering problem: UPGMA Distance Matrix Example

	AC	B	DE
AC	-	4.5	8.75
B	4.5	-	10.5
DE	8.75	10.5	-

Cluster B and AC to form ABC, with branches of length 2.25

Solving the clustering problem: UPGMA Distance Matrix Example

	AC	B	DE
AC	-	4.5	8.75
B	4.5	-	10.5
DE	8.75	10.5	-

	ABC	DE
ABC	-	9.625
DE	9.625	-

Cluster B and AC to form ABC, with branches of length 2.25

Solving the clustering problem: UPGMA Distance Matrix Example

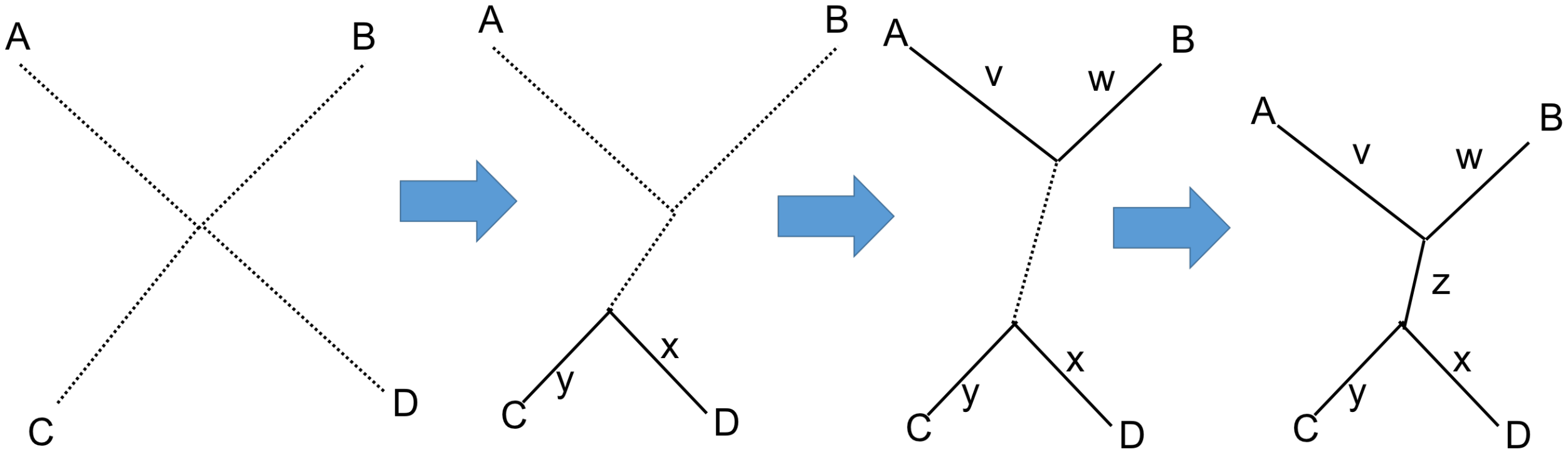
	AC	B	DE
AC	-	4.5	8.75
B	4.5	-	10.5
DE	8.75	10.5	-

	ABC	DE
ABC	-	9.625
DE	9.625	-

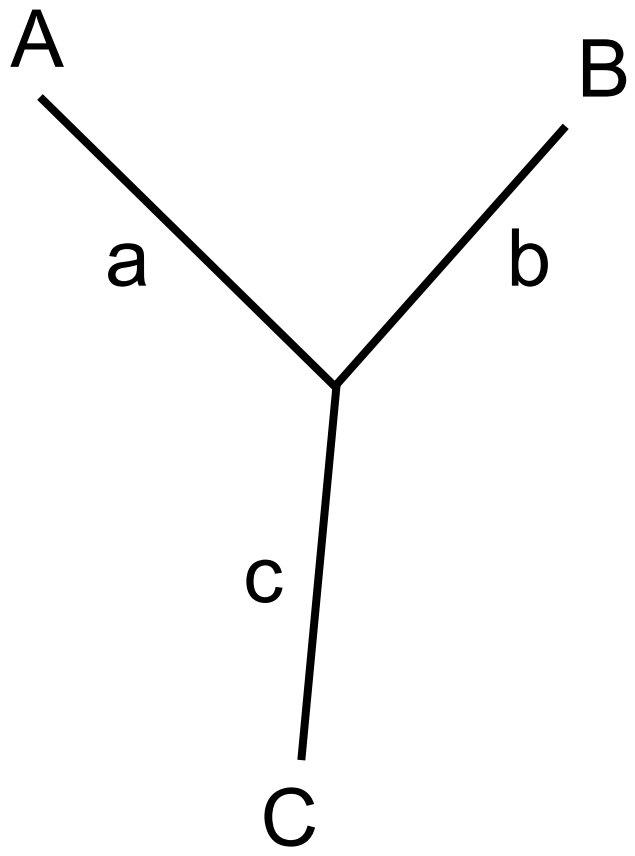
Finally, cluster ABC with DE, with branches of length 4.80

Try the UPGMA example on the handout

Solving the clustering problem: Neighbor Joining (NJ)



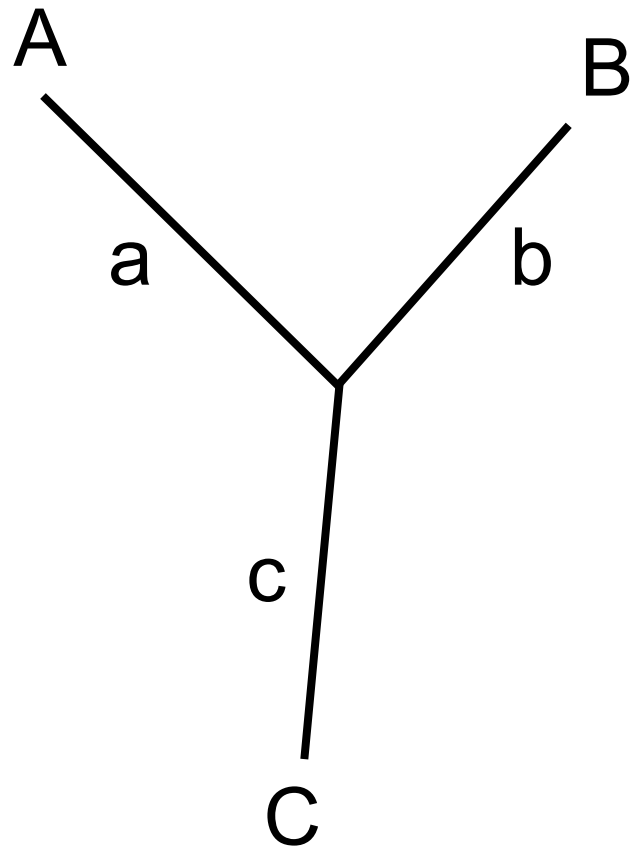
Solving the clustering problem: Neighbor Joining (NJ)



System of equations:

- d(A,B) = a + b
- d(B,C) = b + c
- d(A,C) = a + c

Solving the clustering problem: Neighbor Joining (NJ)



System of equations:

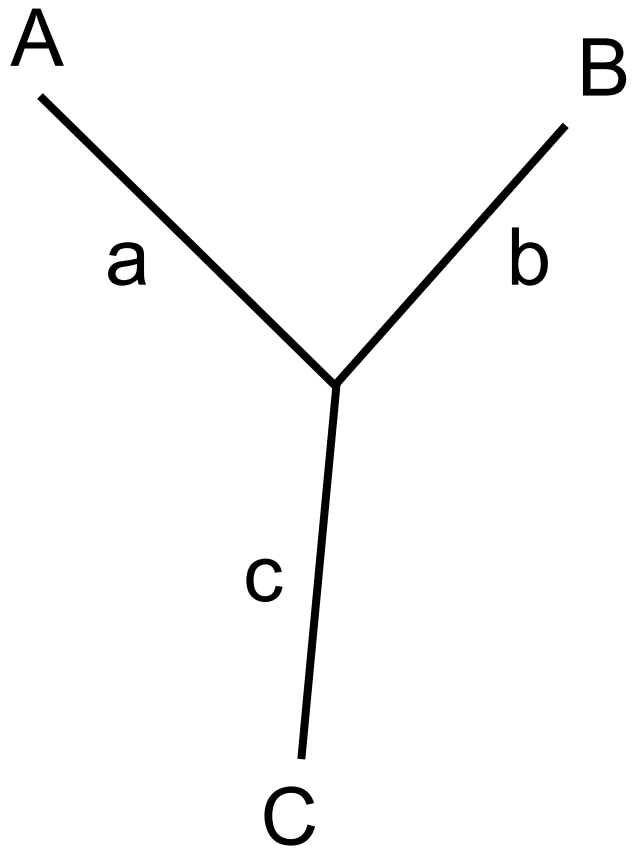
- d(A,B) = a + b
- d(B,C) = b + c
- d(A,C) = a + c

0. Let us solve for *a*
1. We take the sum of all distances that contain *a*. Then we algebraically manipulate until we get to *a*

$$d(A,B)+d(A,C) = a + b + a + c = 2a + b + c$$
$$d(A,B)+d(A,C) - d(B,C) = 2a + b + c - b - c$$
$$= 2a$$

$$\tfrac{1}{2}\, d(A,B)+d(A,C) - d(B,C) = \tfrac{1}{2}\,(2a)$$
$$\tfrac{1}{2}\, d(A,B)+d(A,C) - d(B,C) = a$$

Neighbor Joining



System of equations:

- d(A,B) = a + b
- d(B,C) = b + c
- d(A,C) = a + c

Solution:

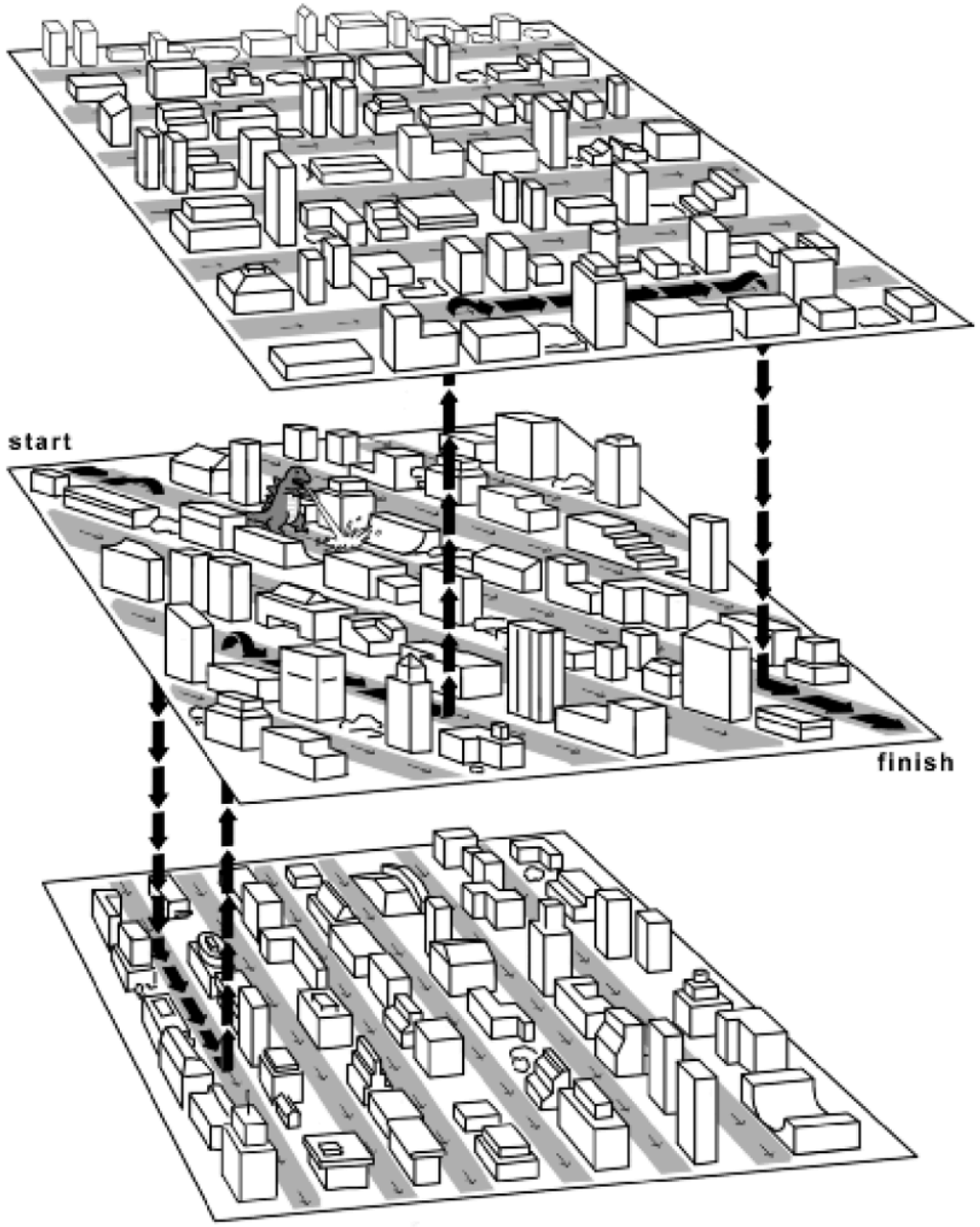
- a = ½ [d(A,B)+d(A,C)−d(B,C)]
- b = ½ [d(A,B)+d(B,C)−d(A,C)]
- c = ½ [d(A,C)+d(B,C)−d(A,B)]

Given a distance matrix, you could solve for a, b, and c

Multiple Sequence Alignment (MSA)

and

Phylogenetics



Multiple Sequence Alignment (MSA)

and

Phylogenetics