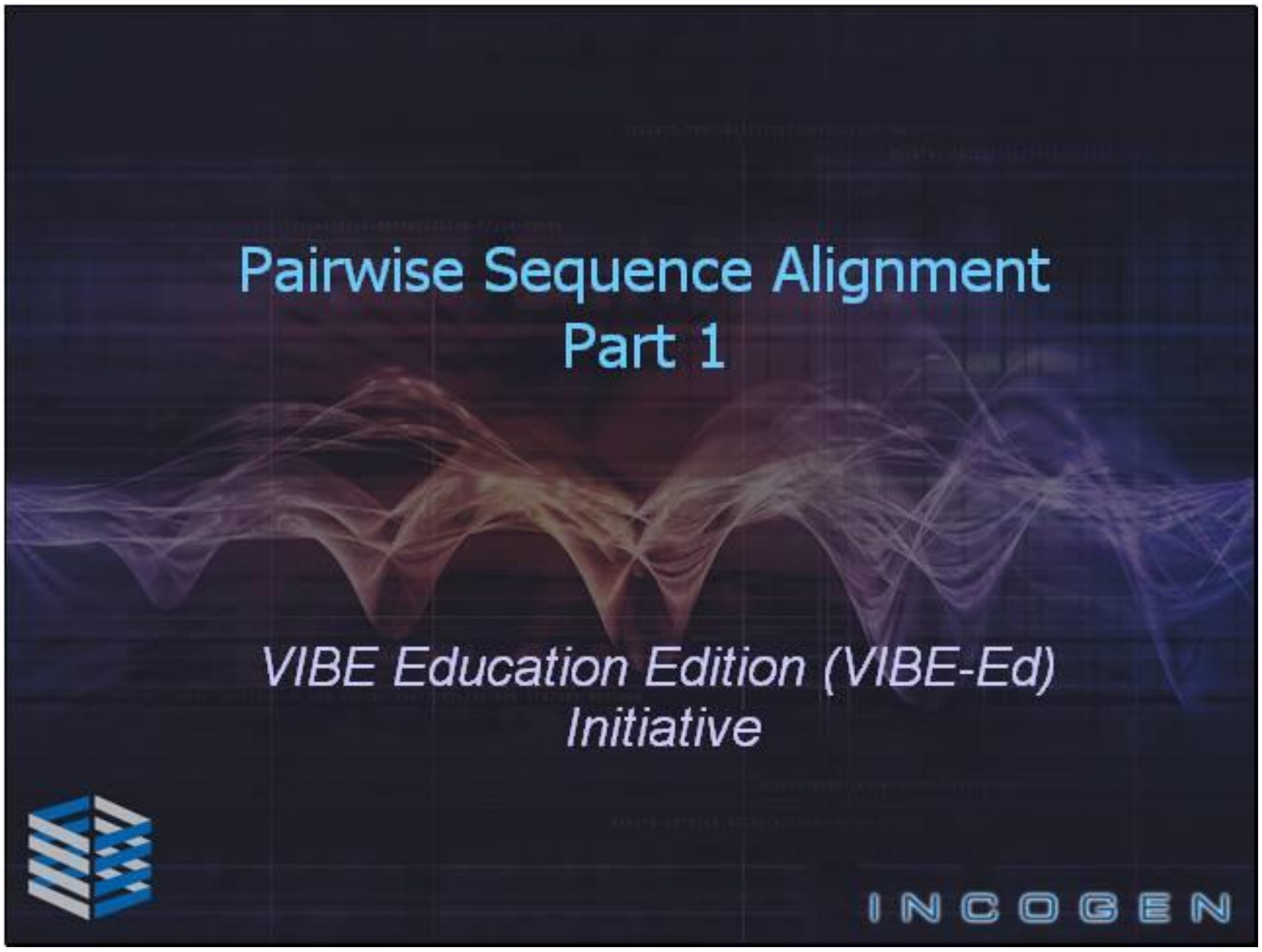
Slide 1 - Pairwise Sequence Alignment Part 1



Pairwise Sequence Alignment Part 1

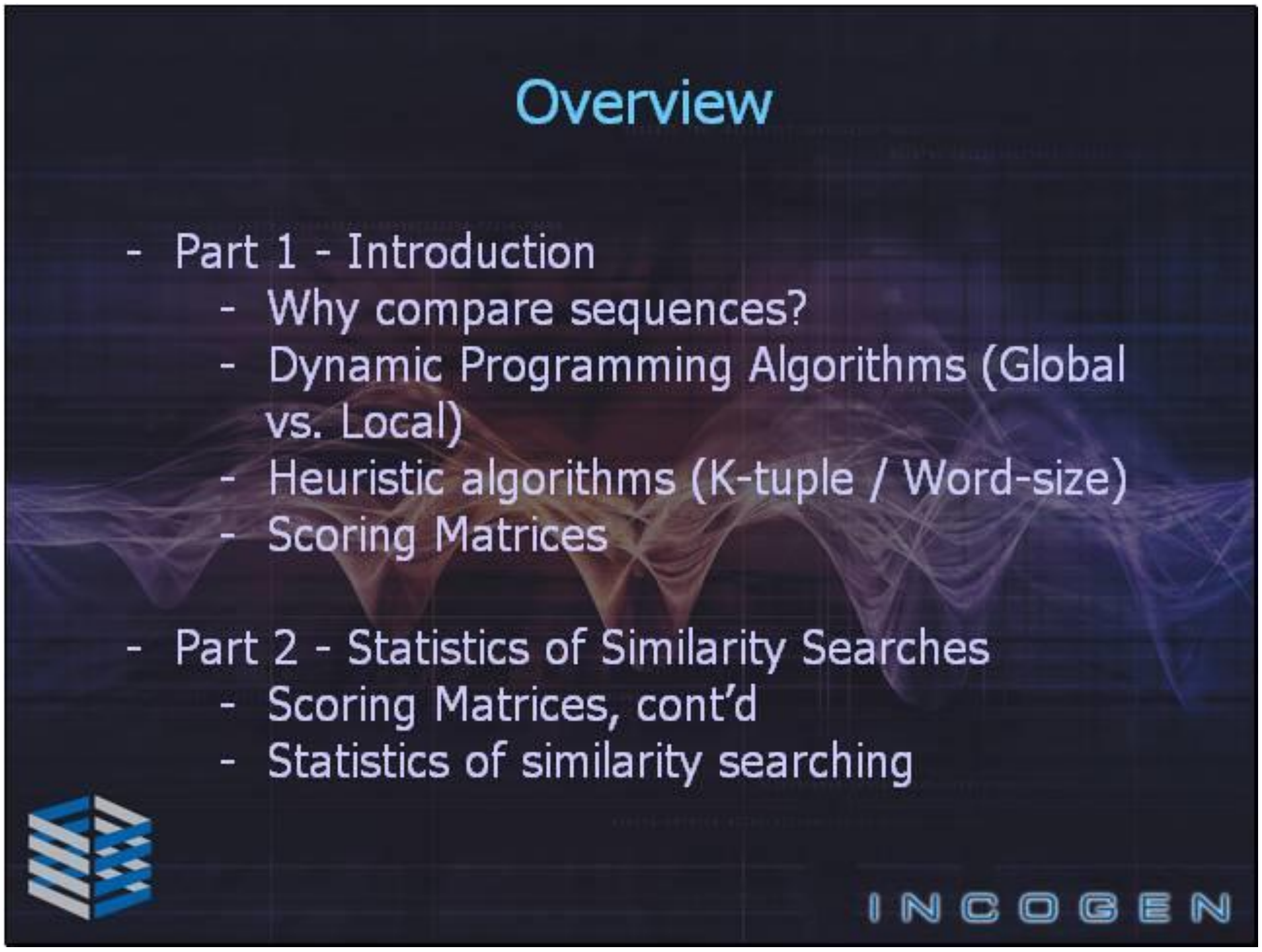
VIBE Education Edition (VIBE-Ed) Initiative

INCOGEN

Slide notes

This presentation and the next one will explain the process of finding similar regions between two sequences, called pairwise sequence alignment.

Slide 2 - Overview



Overview

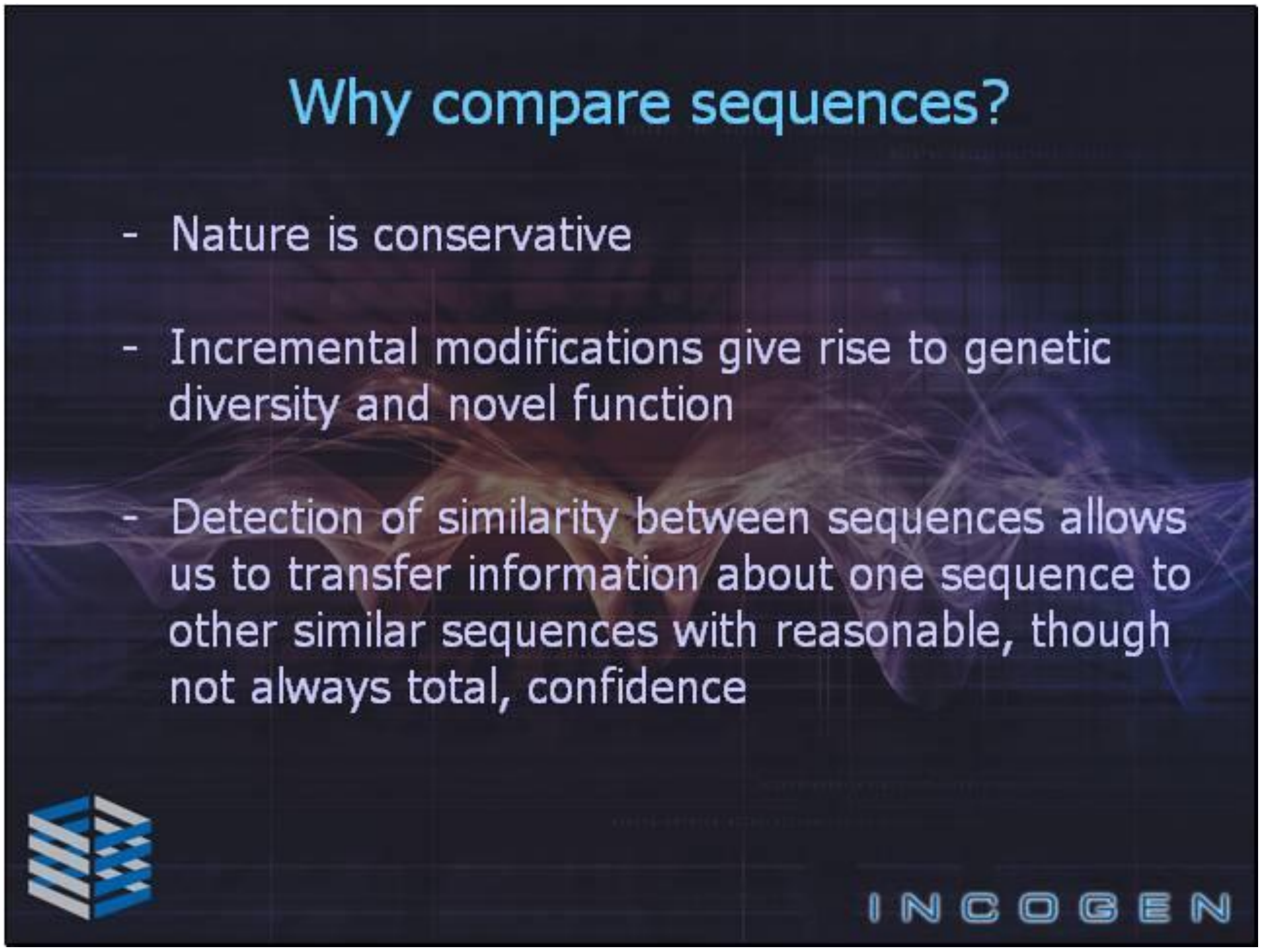
- Part 1 - Introduction
  - Why compare sequences?
  - Dynamic Programming Algorithms (Global vs. Local)
  - Heuristic algorithms (K-tuple / Word-size)
  - Scoring Matrices
- Part 2 - Statistics of Similarity Searches
  - Scoring Matrices, cont'd
  - Statistics of similarity searching

INCOGEN

Slide notes

In this presentation, we will discuss why we want to compare sequences. I will then briefly describe both the dynamic programming algorithms, global and local, and the heuristic algorithms. Included in this will be a brief discussion of scoring matrices and their effect on algorithm results. The second presentation will include a more detailed discussion of scoring matrices and an explanation of the statistics of similarity searches.

Slide 3 - Why compare sequences?



Why compare sequences?

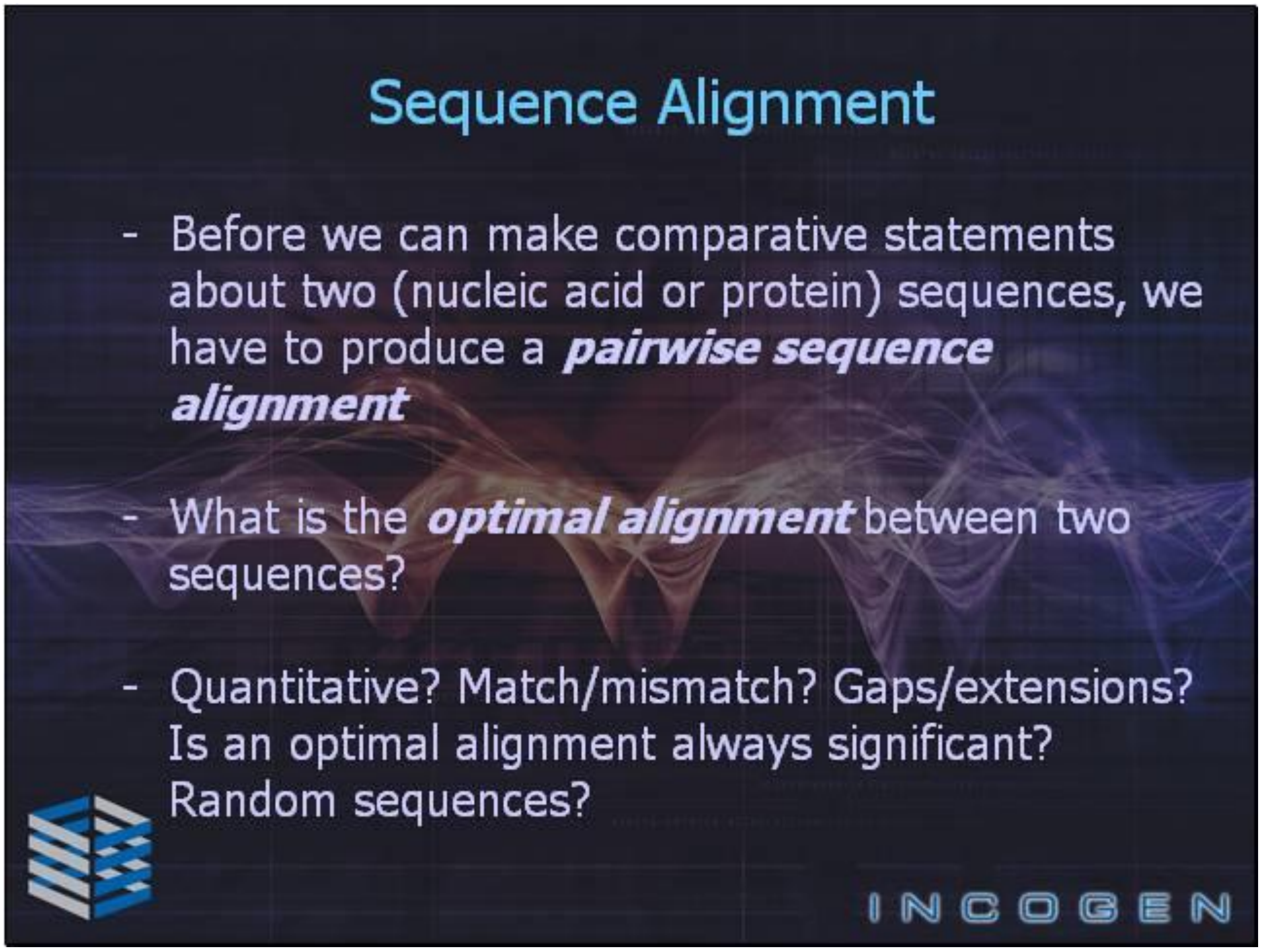
- Nature is conservative
- Incremental modifications give rise to genetic diversity and novel function
- Detection of similarity between sequences allows us to transfer information about one sequence to other similar sequences with reasonable, though not always total, confidence

INCOGEN

Slide notes

Why do we want to compare two sequences? First, nature is conservative. Incremental modifications, not large-scale changes, give rise to the genetic diversity and novel function that we see. The detection of similarities between sequences allows us to infer the roles and functions of newly isolated sequences using well-known, characterized sequences. For example, if we find that a sequence we have isolated from Human is very similar to a sequence from Mouse that is known to be involved in digestion, it is reasonable to infer that the sequence we found may also be involved in digestion.

Slide 4 - Sequence Alignment



Sequence Alignment

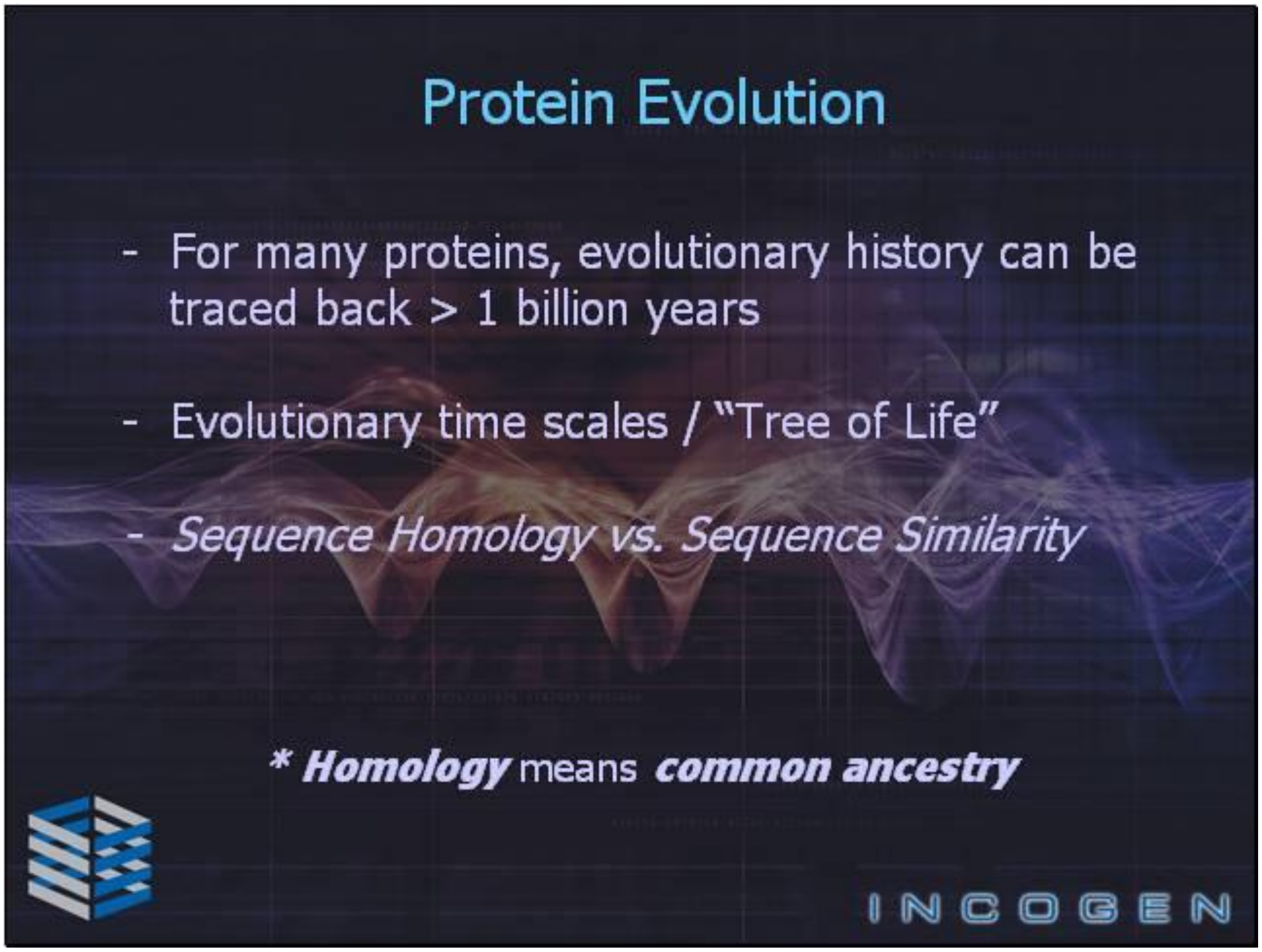
- Before we can make comparative statements about two (nucleic acid or protein) sequences, we have to produce a ***pairwise sequence alignment***
- What is the ***optimal alignment*** between two sequences?
- Quantitative? Match/mismatch? Gaps/extensions? Is an optimal alignment always significant? Random sequences?

INCOGEN

Slide notes

Before we can make comparative statements about two sequences, we have to produce a pairwise sequence alignment; in other words, we need to find the optimal alignment between the two sequences. So, how do we define “optimal”? Which algorithm should we use? How should we score matches and mismatches? Should we allow gaps? If so, how should we score them? Should we score protein sequences differently than we score DNA sequences? Another thought: are all alignments biologically significant? What are the chances that two random, non-related sequences can produce a “good” alignment?

Slide 5 - Protein Evolution



Protein Evolution

- For many proteins, evolutionary history can be traced back > 1 billion years
- Evolutionary time scales / "Tree of Life"
- *Sequence Homology vs. Sequence Similarity*

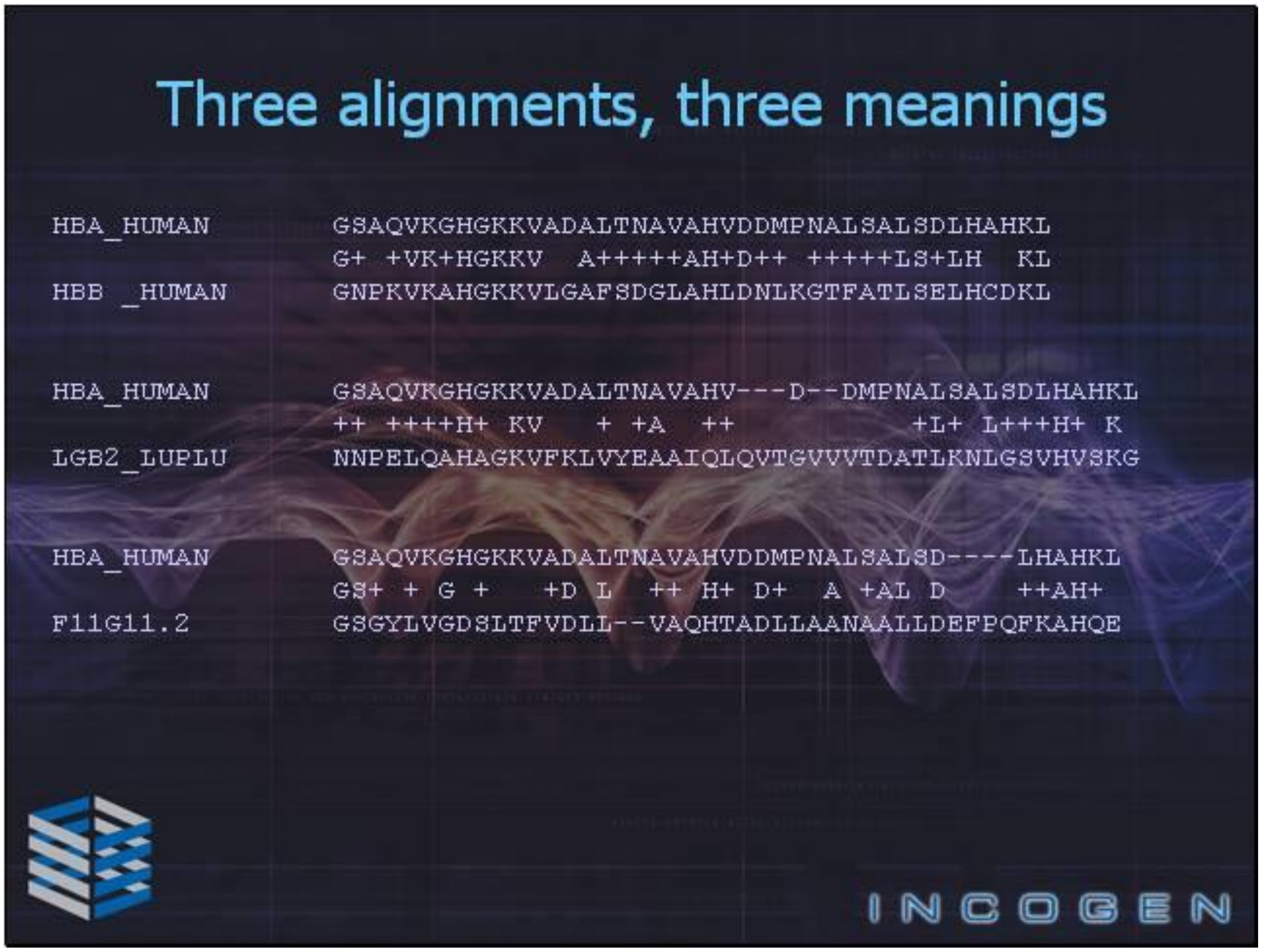
* ***Homology*** means ***common ancestry***

INCOGEN

Slide notes

The premise behind most sequencing projects is to determine the sequence first and then infer function. We can trace the evolutionary history of many proteins back more than one billion years. Over this time frame, an enormous amount of information is preserved, making it possible to find similarities in proteins that are separated by hundreds of millions of years. Unfortunately, it isn't as simple as comparing sequences and assuming that they are related because they can be aligned. Sequences can be very similar and contain good alignments without sharing a common ancestor. In other words, similar sequences are not necessarily homologous.

Slide 6 - Three alignments, three meanings



Three alignments, three meanings

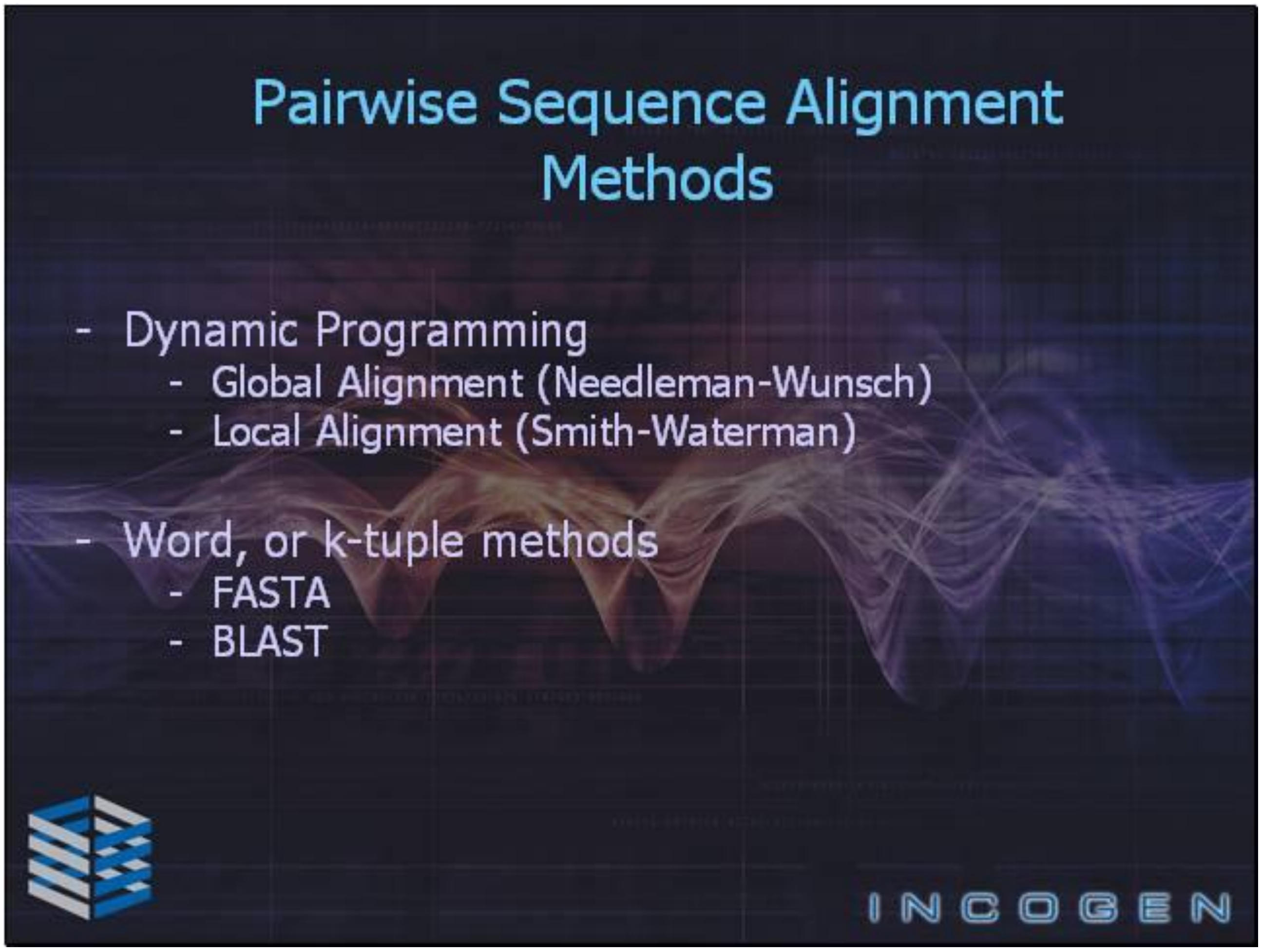
```
HBA_HUMAN      GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKL
               G+ +VK+HGKKV  A+++++AH+D++ +++++LS+LH  KL
HBB _HUMAN     GNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKL

HBA_HUMAN      GSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL
               ++ ++++H+ KV   + +A  ++          +L+ L+++H+ K
LGB2_LUPLU     NNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG

HBA_HUMAN      GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSD----LHAHKL
               GS+ + G +   +D L  ++ H+ D+  A +AL D    ++AH+
F11G11.2       GSGYLVGDSLTFVDLL--VAQHTADLLAANAALLDEFPQFKAHQE
```

INCOGEN

Slide notes

For example, on this slide, we show the human alpha globin sequence aligned with three other sequences. The first alignment is obviously quite good; there are clear similarities between human alpha globin and human beta globin. The alignment between leghaemoglobin from yellow lupin and human alpha globin is not as good, but the 3-D structures of the proteins are identical, the proteins share the same function and they are known to be evolutionarily related. The third alignment, between human alpha globin and nematode glutathioine S-transferase appears to be of similar quality to the second alignment, but in this case, the alignment is spurious. The two proteins have completely different functions and structures. So, how do we tell the difference between alignments that represent real evolutionary relationships, such as alignment 2, and coincidental similarities, such as alignment 3?

Slide 7 - Pairwise Sequence Alignment Methods



Pairwise Sequence Alignment Methods

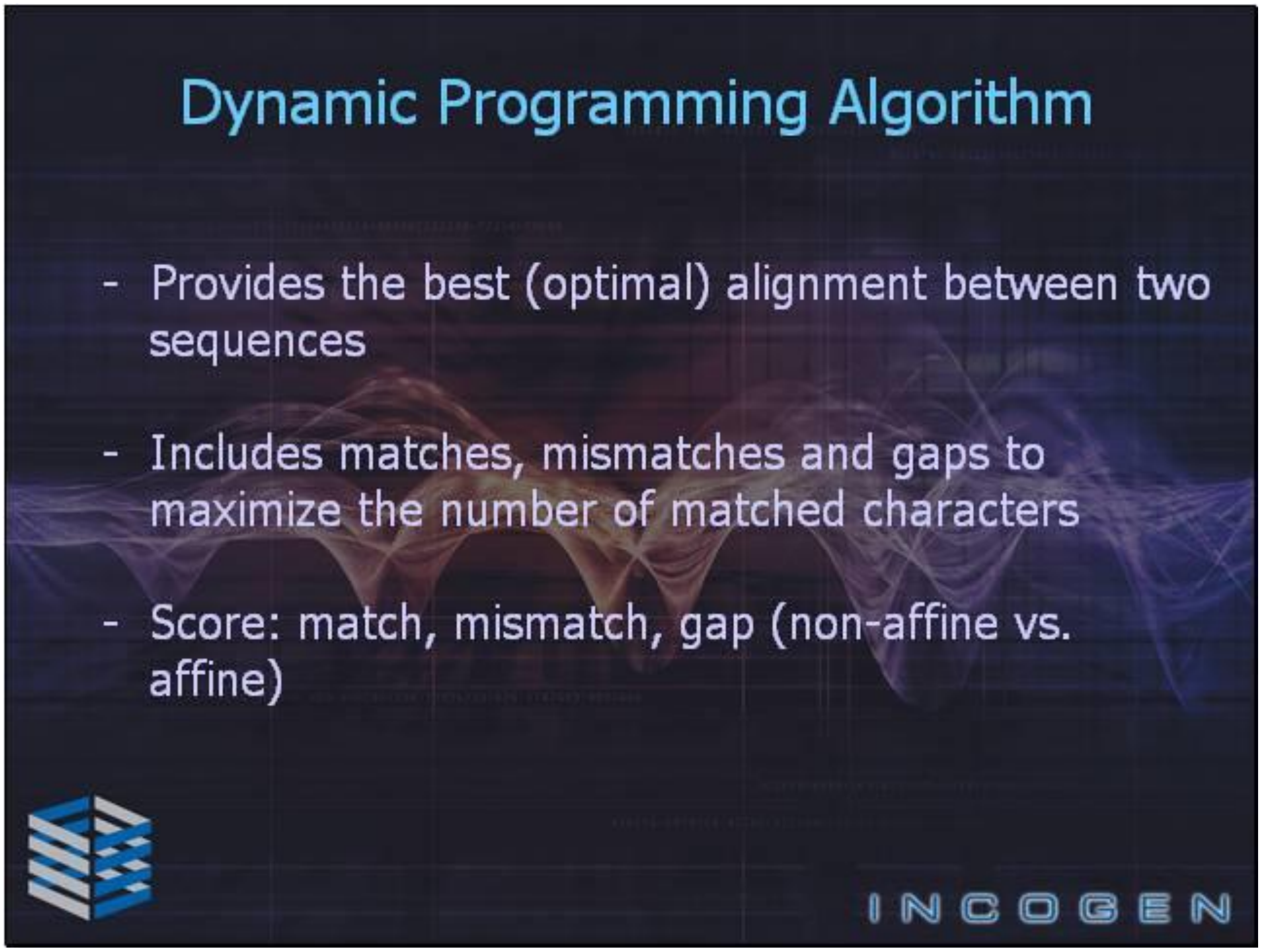
- Dynamic Programming
  - Global Alignment (Needleman-Wunsch)
  - Local Alignment (Smith-Waterman)
- Word, or k-tuple methods
  - FASTA
  - BLAST

INCOGEN

Slide notes

First, we have to align the sequences. There are two methods used to align sequences. The first is dynamic programming. Alignments found using dynamic programming can either be global, using algorithms such as Needleman-Wunsch, or local, using algorithms such as Smith-Waterman. The second method used is word, or k-tuple, methods. In this case, the algorithm compares small sections, with some minimum word length, of each sequence to begin the alignment instead of comparing single letters. BLAST and FASTA both use word, or k-tuple, methods. Both methods have their advantages and disadvantages.

Slide 8 - Dynamic Programming Algorithm



Dynamic Programming Algorithm

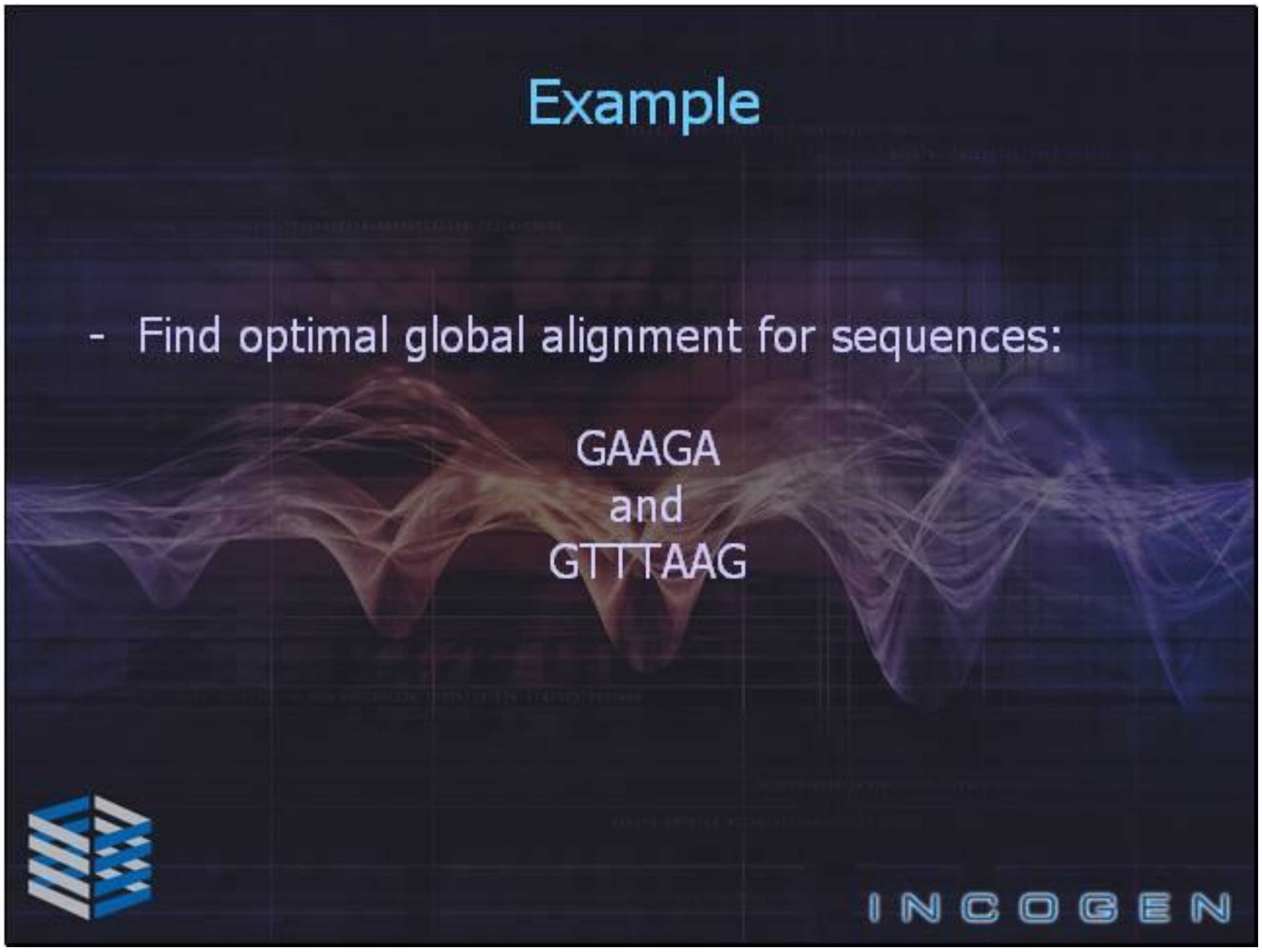
- Provides the best (optimal) alignment between two sequences
- Includes matches, mismatches and gaps to maximize the number of matched characters
- Score: match, mismatch, gap (non-affine vs. affine)

INCOGEN

Slide notes

Dynamic Programming methods are guaranteed to provide the best alignment between any two sequences. Matches, mismatches, and gaps can be included into the score calculated by the algorithm; this maximizes the number of matched characters.

Slide 9 - Example



Example

- Find optimal global alignment for sequences:

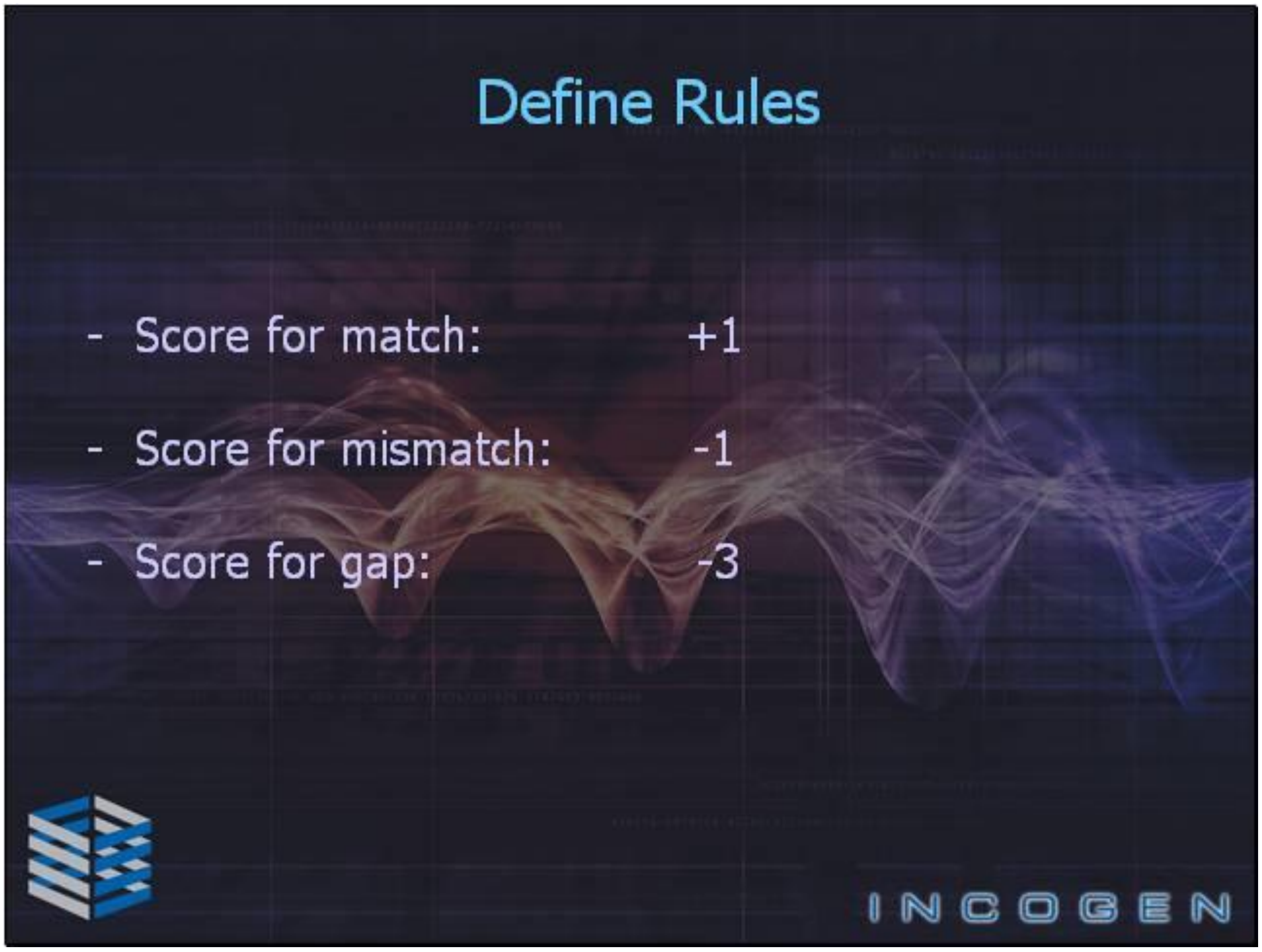
GAAGA
and
GTTTAAG

INCOGEN

Slide notes

For example, let’s say we wanted to find the best global alignment between these two sequences.

Slide 10 - Define Rules



Define Rules

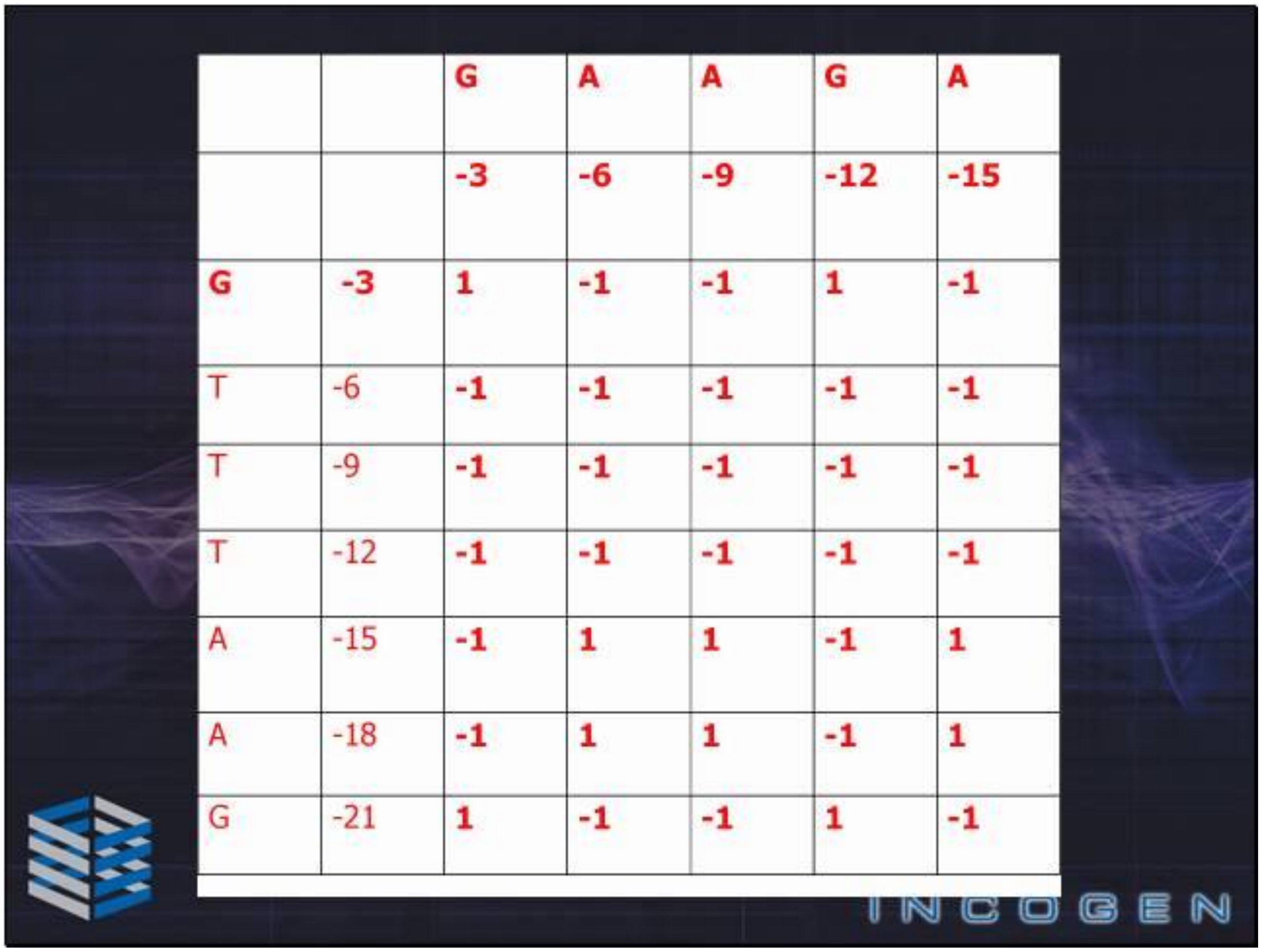
- Score for match:	+1
- Score for mismatch:	-1
- Score for gap:	-3

INCOGEN

Slide notes

First, we have to set up the scoring rules. In this case, a match has a score of 1, a mismatch has a score of -1, and a gap has a score of -3.

Slide 11 - Slide 11



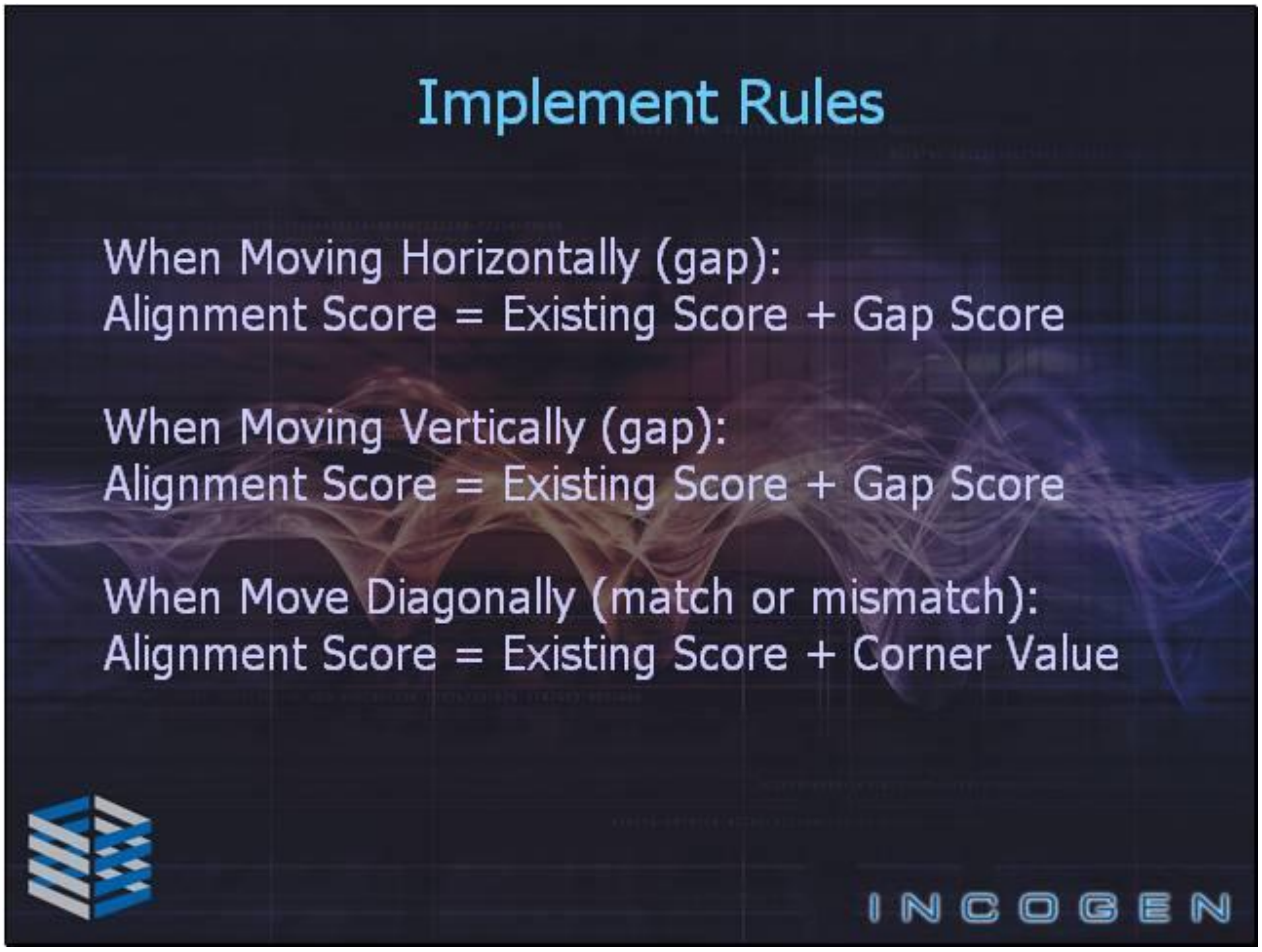
		G	A	A	G	A
		-3	-6	-9	-12	-15
G	-3	1	-1	-1	1	-1
T	-6	-1	-1	-1	-1	-1
T	-9	-1	-1	-1	-1	-1
T	-12	-1	-1	-1	-1	-1
A	-15	-1	1	1	-1	1
A	-18	-1	1	1	-1	1
G	-21	1	-1	-1	1	-1

INCOGEN

Slide notes

Next, we set up a table with one sequence across the top and one sequence along the left side; a gap, or empty space, is included at the beginning of each sequence. Then we include in the table the value of aligning the character at the top with the character at the left. Because we are performing a global alignment, gaps at the beginning and end of the alignment also score -3.

Slide 12 - Slide 12



Implement Rules

When Moving Horizontally (gap):
Alignment Score = Existing Score + Gap Score

When Moving Vertically (gap):
Alignment Score = Existing Score + Gap Score

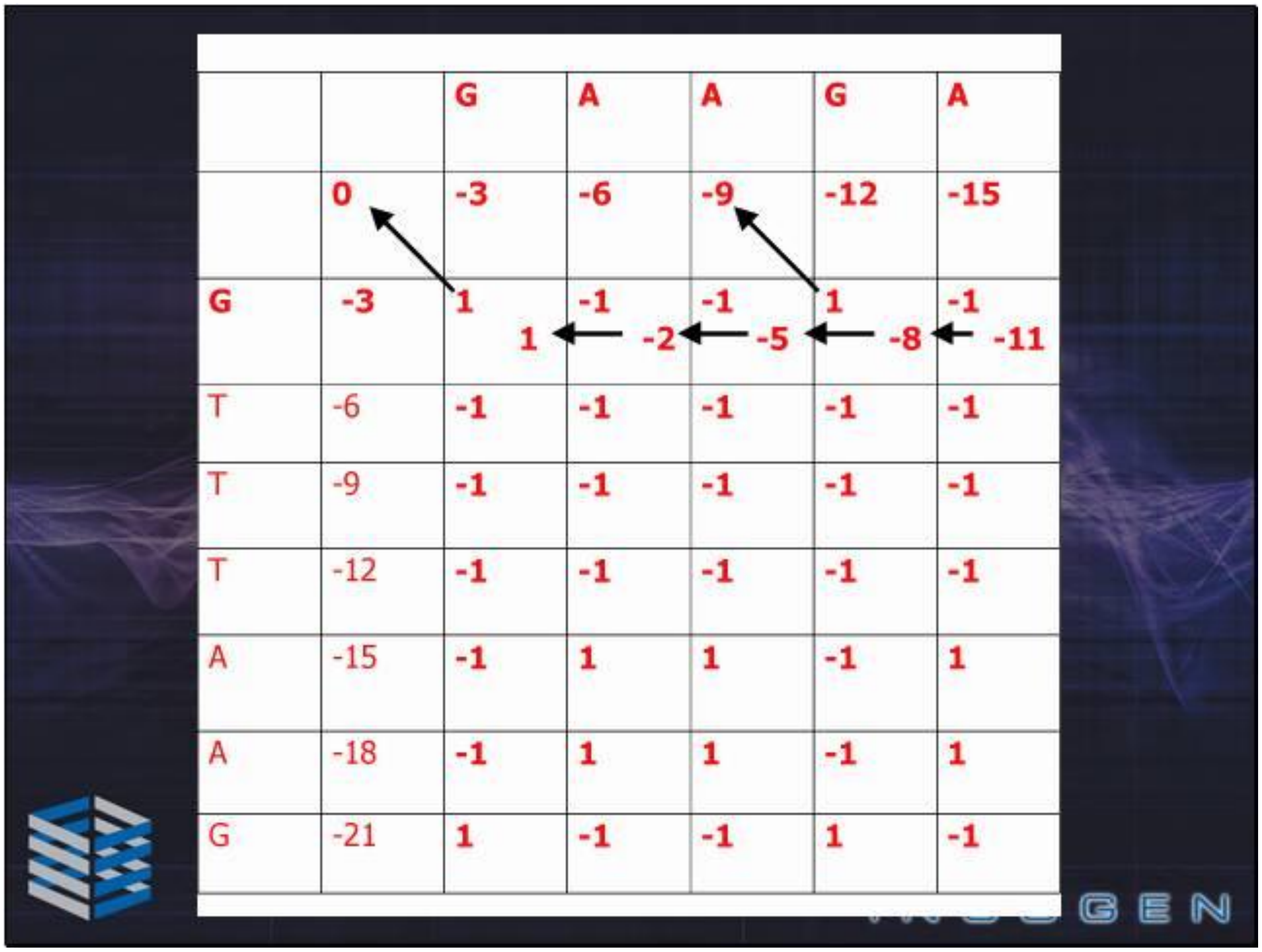
When Move Diagonally (match or mismatch):
Alignment Score = Existing Score + Corner Value

INCOGEN

Slide notes

Now, we need to set up the rules for filling out the table. When we move horizontally along the table, which represents adding a gap to the top sequence, or vertically along the table, which represents adding a gap to the left sequence, we add the gap score, -3, to the existing score. When we move diagonally along the table, we add the existing score to the corner value in the table. To fill out each cell, we move vertically, horizontally, and diagonally into the cell and keep the highest score.

Slide 13 - Slide 13

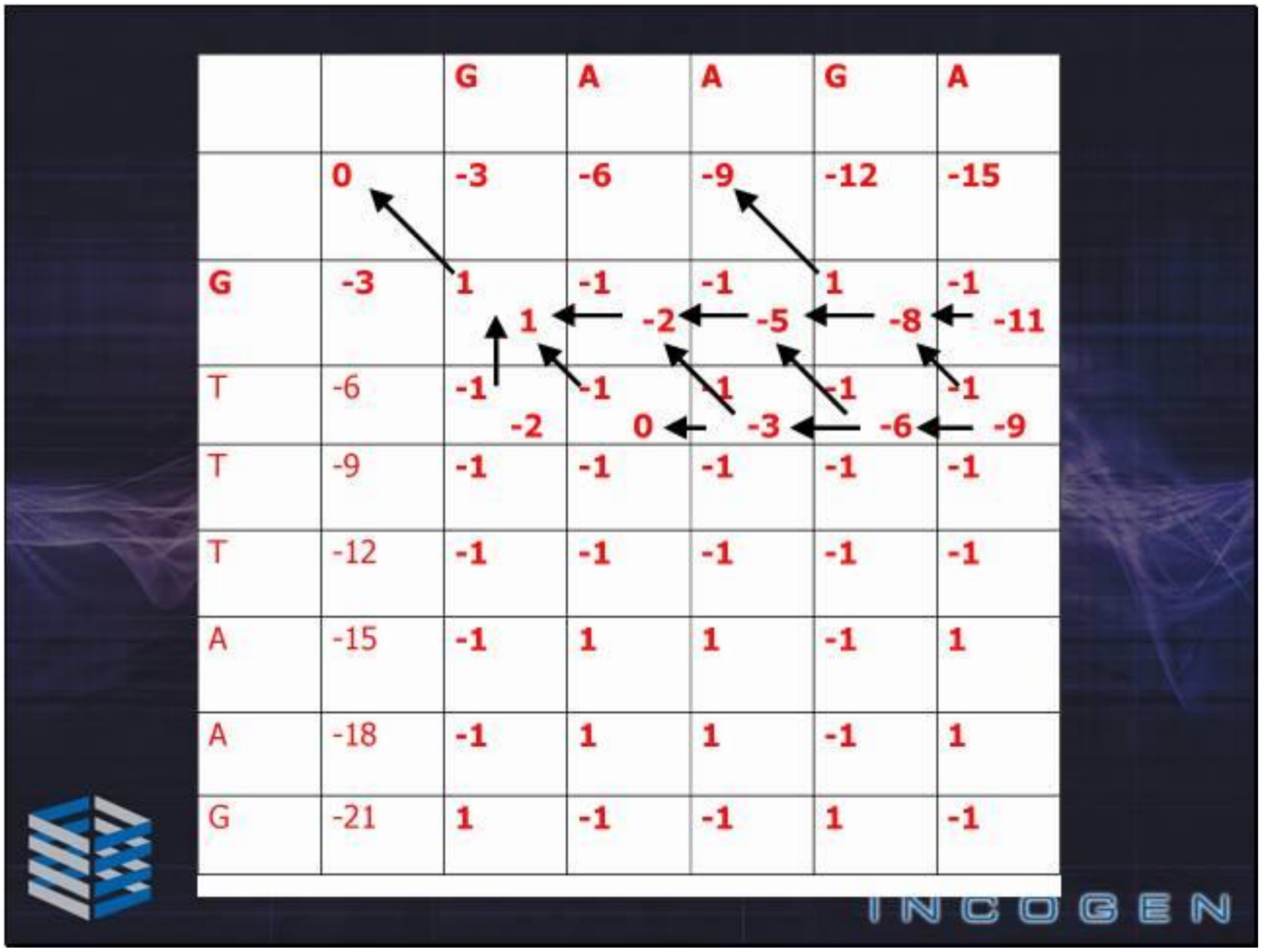


		G	A	A	G	A
	0	-3	-6	-9	-12	-15
G	-3	1 1	-1 ← -2	-1 ← -5	1 ← -8	-1 ← -11
T	-6	-1	-1	-1	-1	-1
T	-9	-1	-1	-1	-1	-1
T	-12	-1	-1	-1	-1	-1
A	-15	-1	1	1	-1	1
A	-18	-1	1	1	-1	1
G	-21	1	-1	-1	1	-1

Slide notes

This slide shows how we fill out the first row. Notice that we draw arrows in the table to indicate from which cell we calculated this maximum score. These arrows will be used later to find the optimum alignment.

Slide 14 - Slide 14



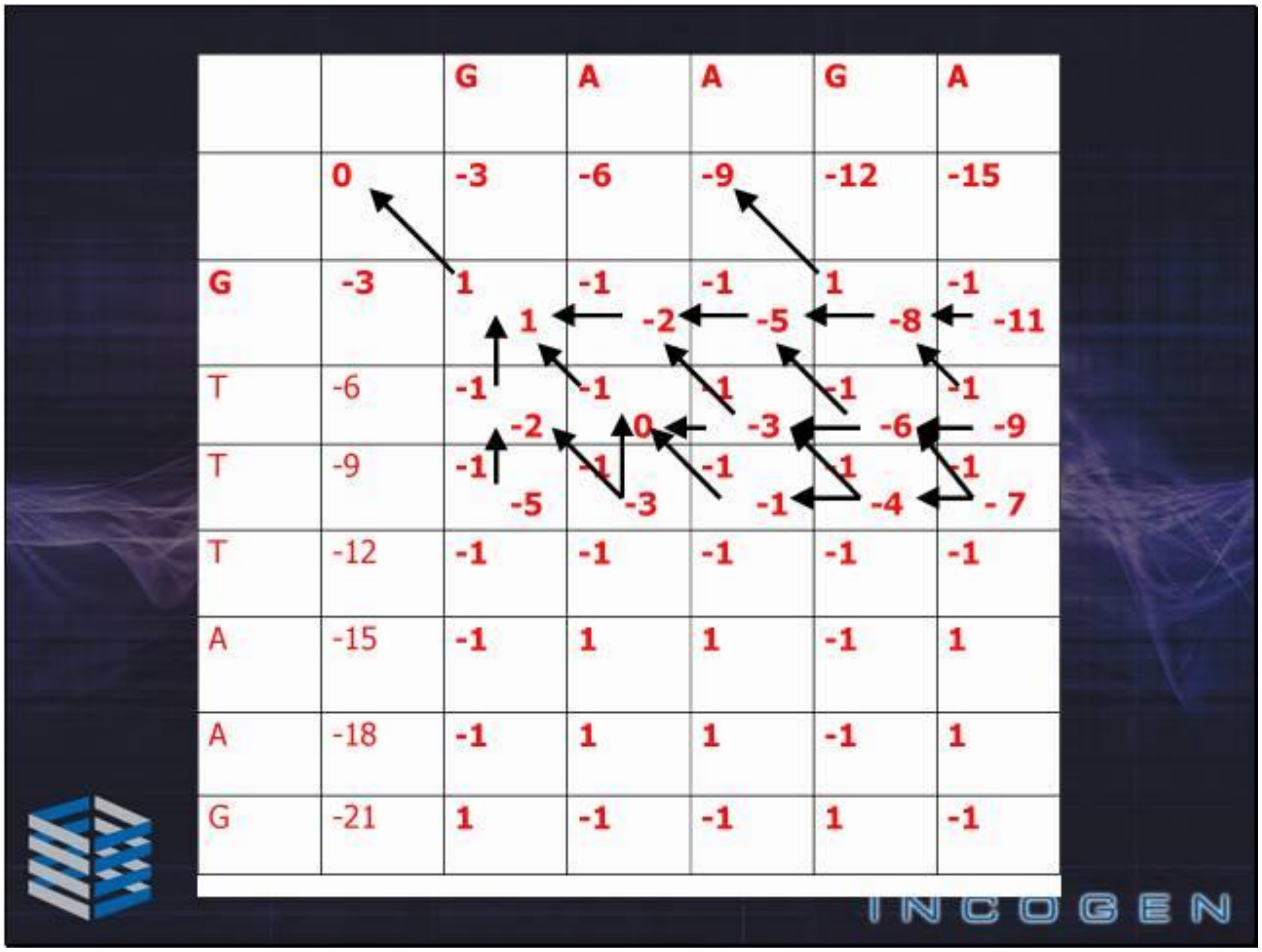
		G	A	A	G	A
	0	-3	-6	-9	-12	-15
G	-3	1 1	-1 -2	-1 -5	1 -8	-1 -11
T	-6	-1 -2	-1 0	-1 -3	-1 -6	-1 -9
T	-9	-1	-1	-1	-1	-1
T	-12	-1	-1	-1	-1	-1
A	-15	-1	1	1	-1	1
A	-18	-1	1	1	-1	1
G	-21	1	-1	-1	1	-1

INCOGEN

Slide notes

In this slide, we have filled out the second row of the table. Notice that some of the cells have arrows leading to two previous cells instead of one. In these cases, there was more than one way to get the same highest score, and we mark both ways.

Slide 15 - Slide 15

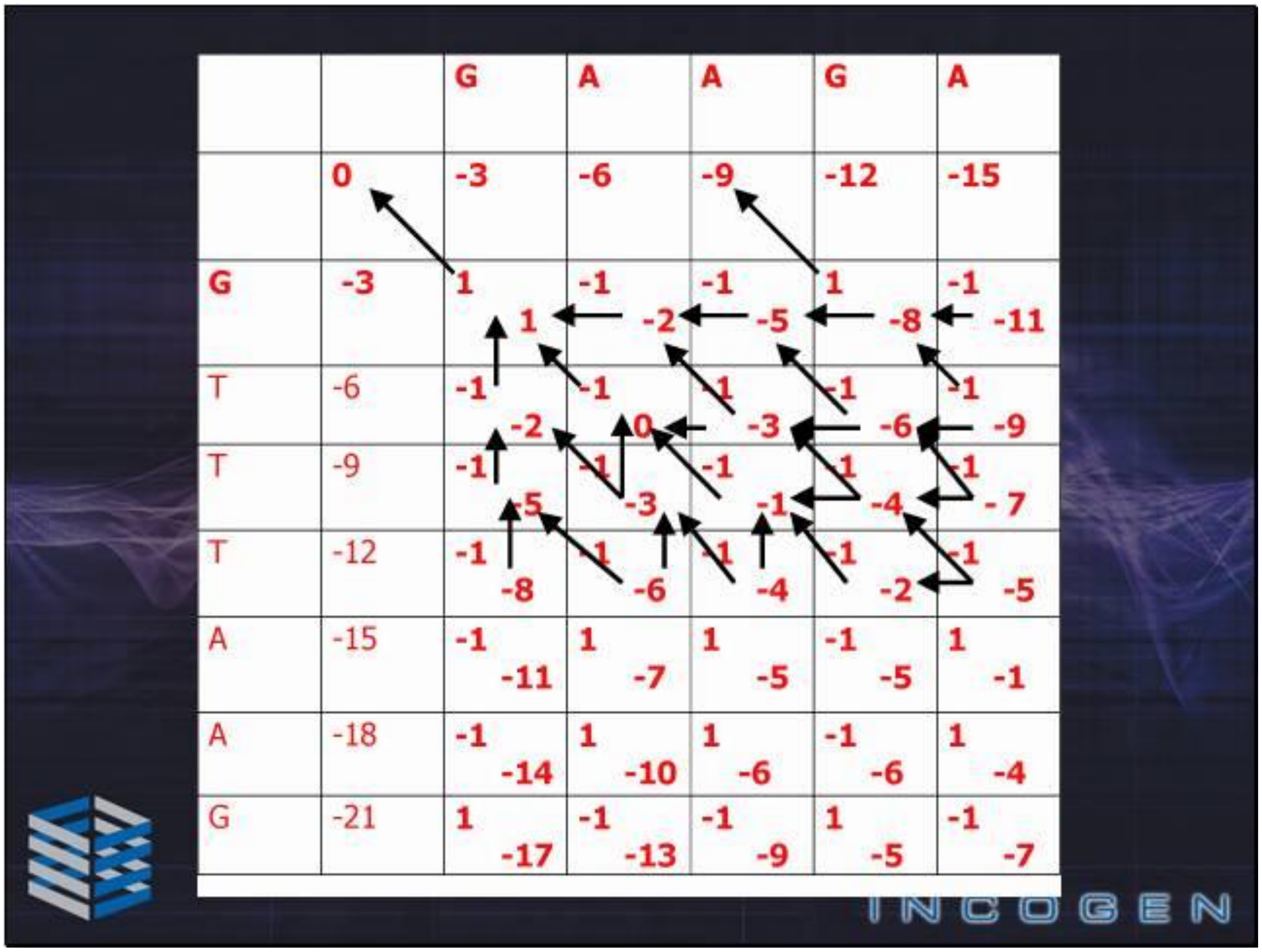


		G	A	A	G	A
	0	-3	-6	-9	-12	-15
G	-3	1 1	-1 -2	-1 -5	1 -8	-1 -11
T	-6	-1 -2	-1 0	-1 -3	-1 -6	-1 -9
T	-9	-1 -5	-1 -3	-1 -1	-1 -4	-1 - 7
T	-12	-1	-1	-1	-1	-1
A	-15	-1	1	1	-1	1
A	-18	-1	1	1	-1	1
G	-21	1	-1	-1	1	-1

Slide notes

Now, we’ve filled out the third row,

Slide 16 - Slide 16

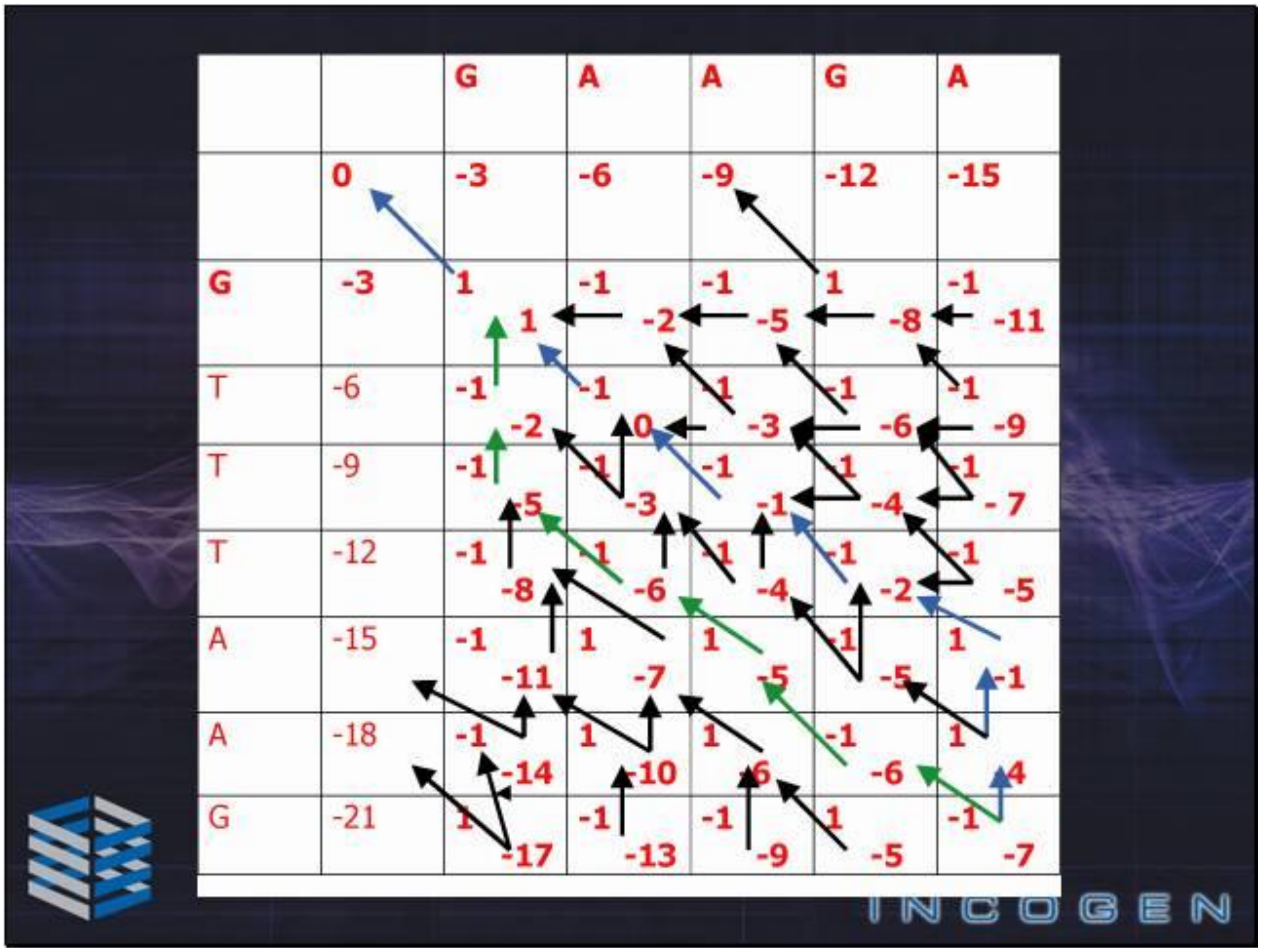


		G	A	A	G	A
	0	-3	-6	-9	-12	-15
G	-3	1 1	-1 -2	-1 -5	1 -8	-1 -11
T	-6	-1 -2	-1 0	-1 -3	-1 -6	-1 -9
T	-9	-1 -5	-1 -3	-1 -1	-1 -4	-1 - 7
T	-12	-1 -8	-1 -6	-1 -4	-1 -2	-1 -5
A	-15	-1 -11	1 -7	1 -5	-1 -5	1 -1
A	-18	-1 -14	1 -10	1 -6	-1 -6	1 -4
G	-21	1 -17	-1 -13	-1 -9	1 -5	-1 -7

Slide notes

and so on until the table is complete.

Slide 17 - Slide 17

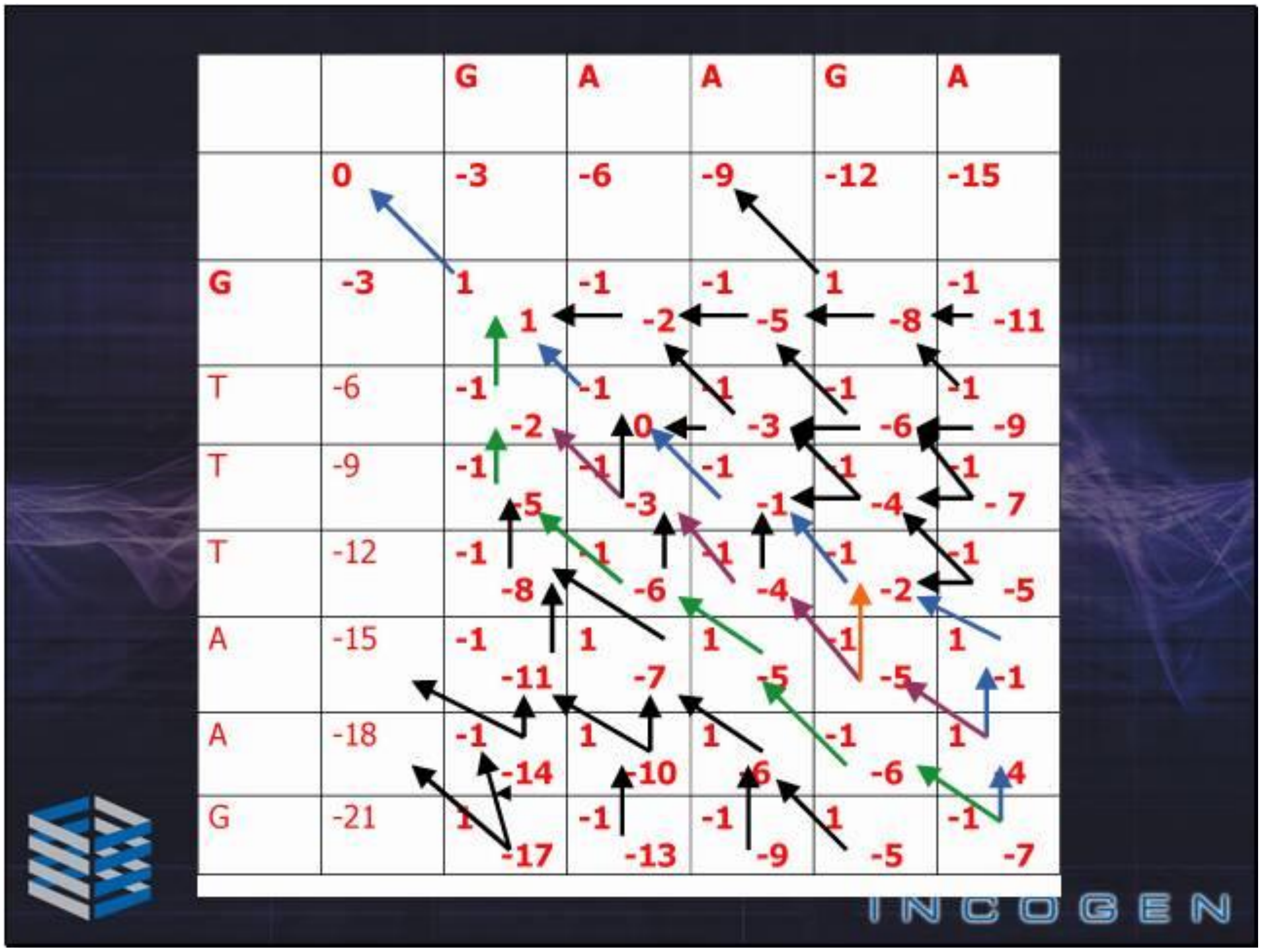


		G	A	A	G	A
	0	-3	-6	-9	-12	-15
G	-3	1 / 1	-1 / -2	-1 / -5	1 / -8	-1 / -11
T	-6	-1 / -2	-1 / 0	-1 / -3	-1 / -6	-1 / -9
T	-9	-1 / -5	-1 / -3	-1 / -1	-1 / -4	-1 / -7
T	-12	-1 / -8	-1 / -6	-1 / -4	-1 / -2	-1 / -5
A	-15	-1 / -11	1 / -7	1 / -5	-1 / -5	1 / -1
A	-18	-1 / -14	1 / -10	1 / -6	-1 / -6	1 / -4
G	-21	1 / -17	-1 / -13	-1 / -9	1 / -5	-1 / -7

Slide notes

Now that we have filled out the table completely, we can use the arrows we drew to trace the optimum alignment. Since we are calculating a global alignment, we must start tracing the arrows from the lower right-hand corner and end in the upper left-hand corner. In this case, there are six alignments that give us the same score. Two are shown here in blue and green.

Slide 18 - Slide 18

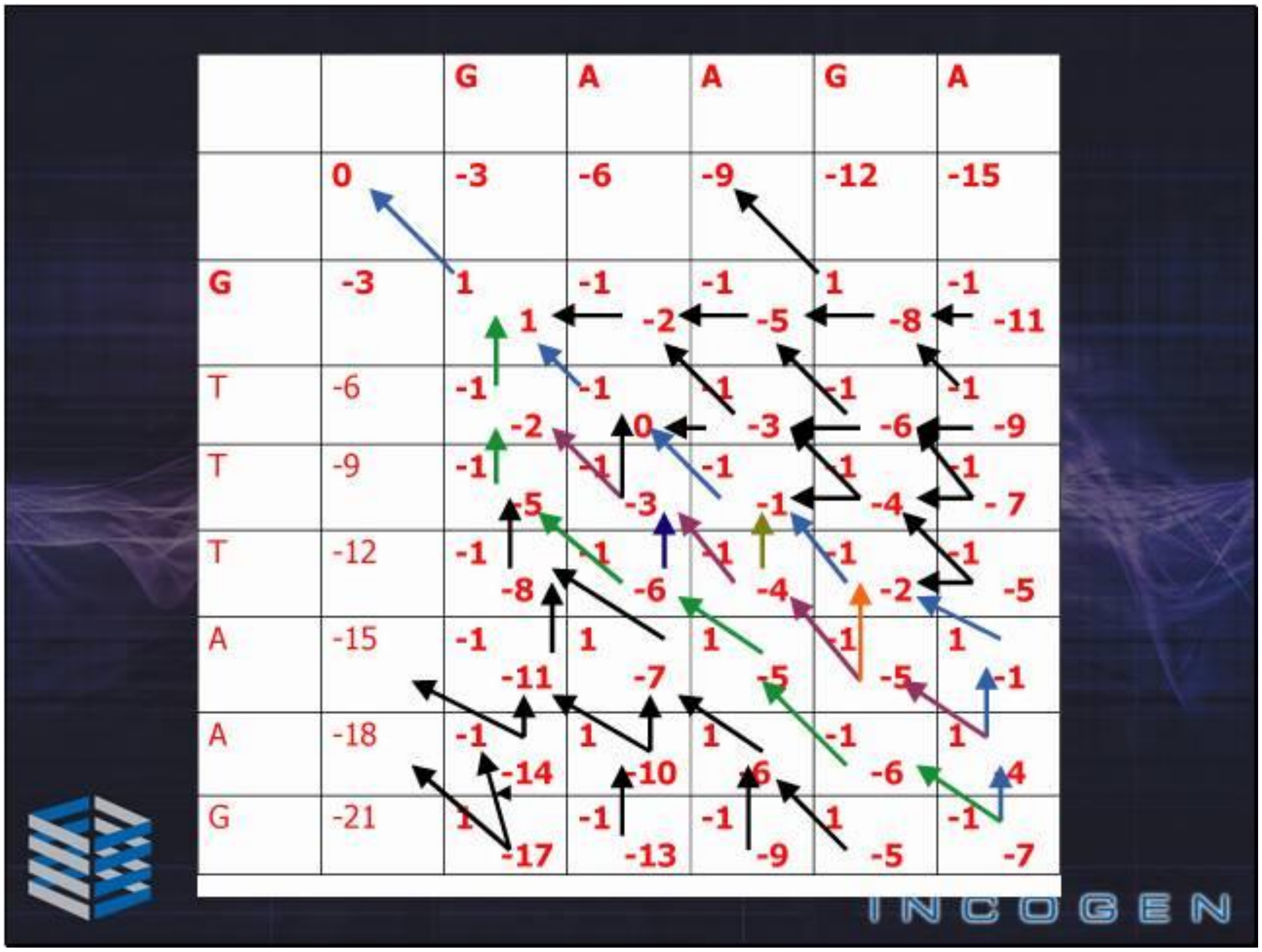


		G	A	A	G	A
	0	-3	-6	-9	-12	-15
G	-3	1 1	-1 -2	-1 -5	1 -8	-1 -11
T	-6	-1 -2	-1 0	-1 -3	-1 -6	-1 -9
T	-9	-1 -5	-1 -3	-1 -1	-1 -4	-1 -7
T	-12	-1 -8	-1 -6	-1 -4	-1 -2	-1 -5
A	-15	-1 -11	1 -7	1 -5	-1 -5	1 -1
A	-18	-1 -14	1 -10	1 -6	-1 -6	1 -4
G	-21	1 -17	-1 -13	-1 -9	1 -5	-1 -7

Slide notes

A third and fourth alignment are shown here. One is mostly in red; the other includes a single gap, shown in orange, that shift us between the red and the blue alignments.

Slide 19 - Slide 19

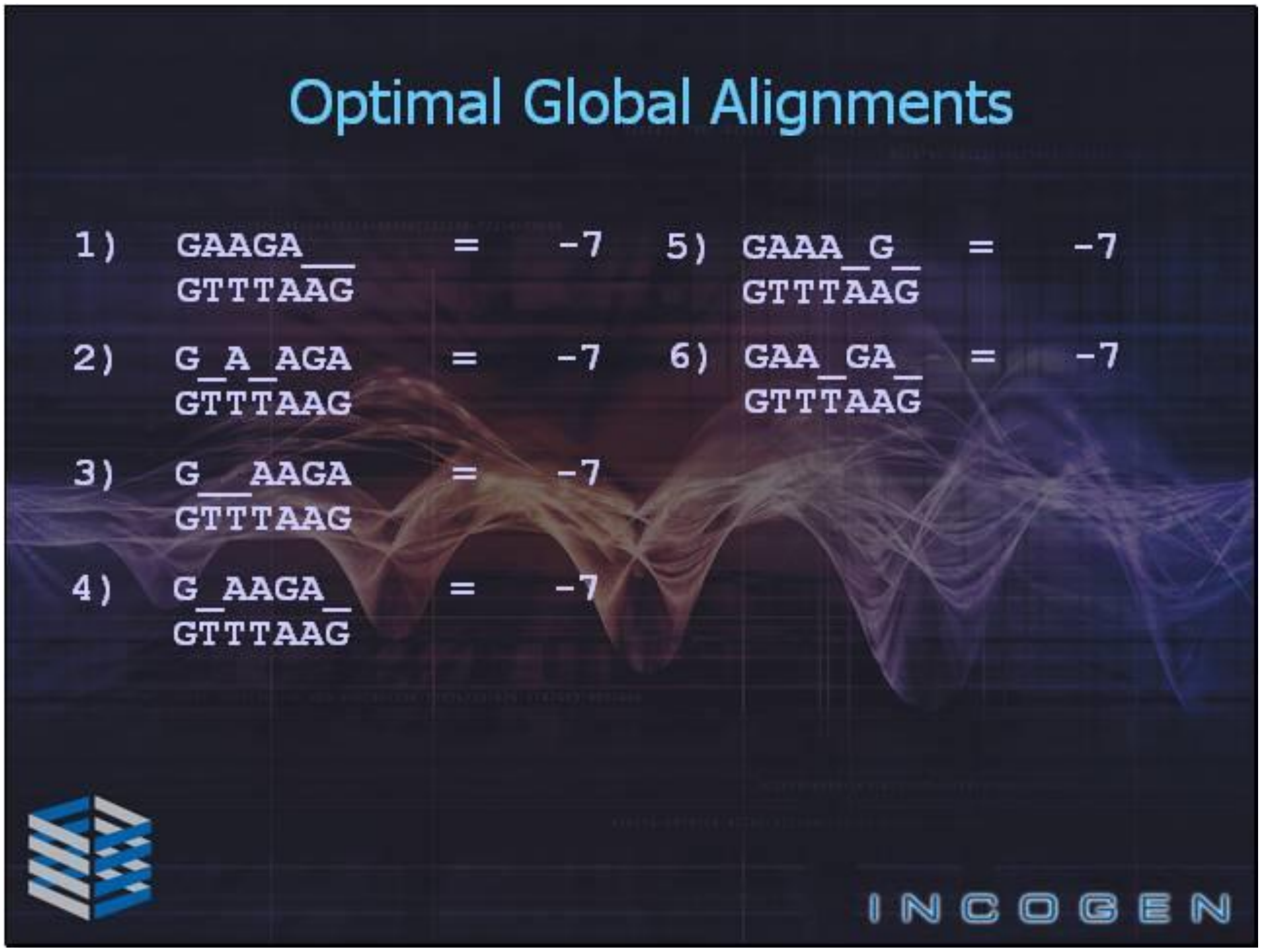


		G	A	A	G	A
	0	-3	-6	-9	-12	-15
G	-3	1 1	-1 -2	-1 -5	1 -8	-1 -11
T	-6	-1 -2	-1 0	-1 -3	-1 -6	-1 -9
T	-9	-1 -5	-1 -3	-1 -1	-1 -4	-1 -7
T	-12	-1 -8	-1 -6	-1 -4	-1 -2	-1 -5
A	-15	-1 -11	1 -7	1 -5	-1 -5	1 -1
A	-18	-1 -14	1 -10	1 -6	-1 -6	1 -4
G	-21	1 -17	-1 -13	-1 -9	1 -5	-1 -7

Slide notes

The final two are shown here. Again, they simply shift a gap to a different position in alignments that are already indicated. The shifted gap is marked in yellow for one alignment and in dark blue for the other.

Slide 20 - Slide 20



Optimal Global Alignments

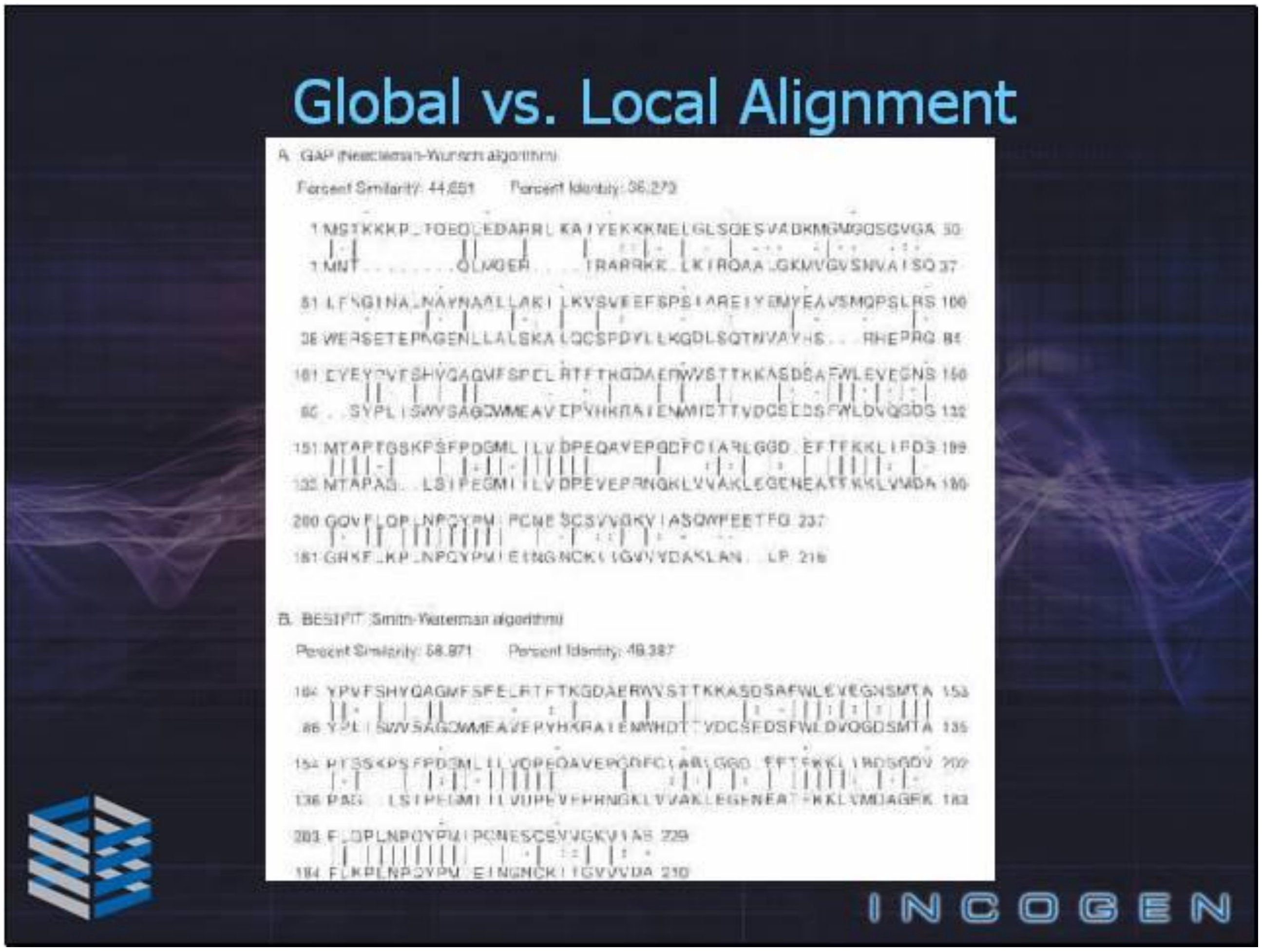
```
1)  GAAGA__    =   -7    5)  GAAA_G_   =   -7
    GTTTAAG                  GTTTAAG

2)  G_A_AGA    =   -7    6)  GAA_GA_   =   -7
    GTTTAAG                  GTTTAAG

3)  G__AAGA    =   -7
    GTTTAAG

4)  G_AAGA_    =   -7
    GTTTAAG
```

INCOGEN

Slide notes

These are the 6 optimal global alignments.

Slide 21 - Slide 21



Global vs. Local Alignment

A. GAP (Needleman-Wunsch algorithm)

Percent Similarity: 44.651 Percent Identity: 36.279

```
  1 MSTKKKPLTQEQLEDARRLKAIYEKKKNELGLSQESVADKMGMGQSGVGA 50
  1 MNT.........QLMGER....IRARRKKLKIRQAALGKMVGVSNVAISQ 37

 51 LFNGINALNAYNAALLAKILKVSVEEFSPSIAREIYEMYEAVSMQPSLRS 100
 38 WERSETEPNGENLLALSKALQCSPDYLLKGDLSQTNVAYHS...RHEPRG 84

101 EYEYPVFSHVQAGMFSPELRTFTKGDAERWVSTTKKASDSAFWLEVEGNS 150
 85 ..SYPLISWVSAGQWMEAVEPYHKRAIENWHDTTVDCSEDSFWLDVQGDS 132

151 MTAPTGSKPSFPDGMLILVDPEQAVEPGDFCIARLGGD.EFTFKKLIRDS 199
133 MTAPAG...LSIPEGMIILVDPEVEPRNGKLVVAKLEGENEATFKKLVMDA 180

200 GQVFLQPLNPQYPMIPCNESCSVVGKVIASQWPEETFG 237
181 GRKFLKPLNPQYPMIEINGNCKIIGVVVDAKLAN...LP 216
```

B. BESTFIT (Smith-Waterman algorithm)

Percent Similarity: 58.871 Percent Identity: 48.387

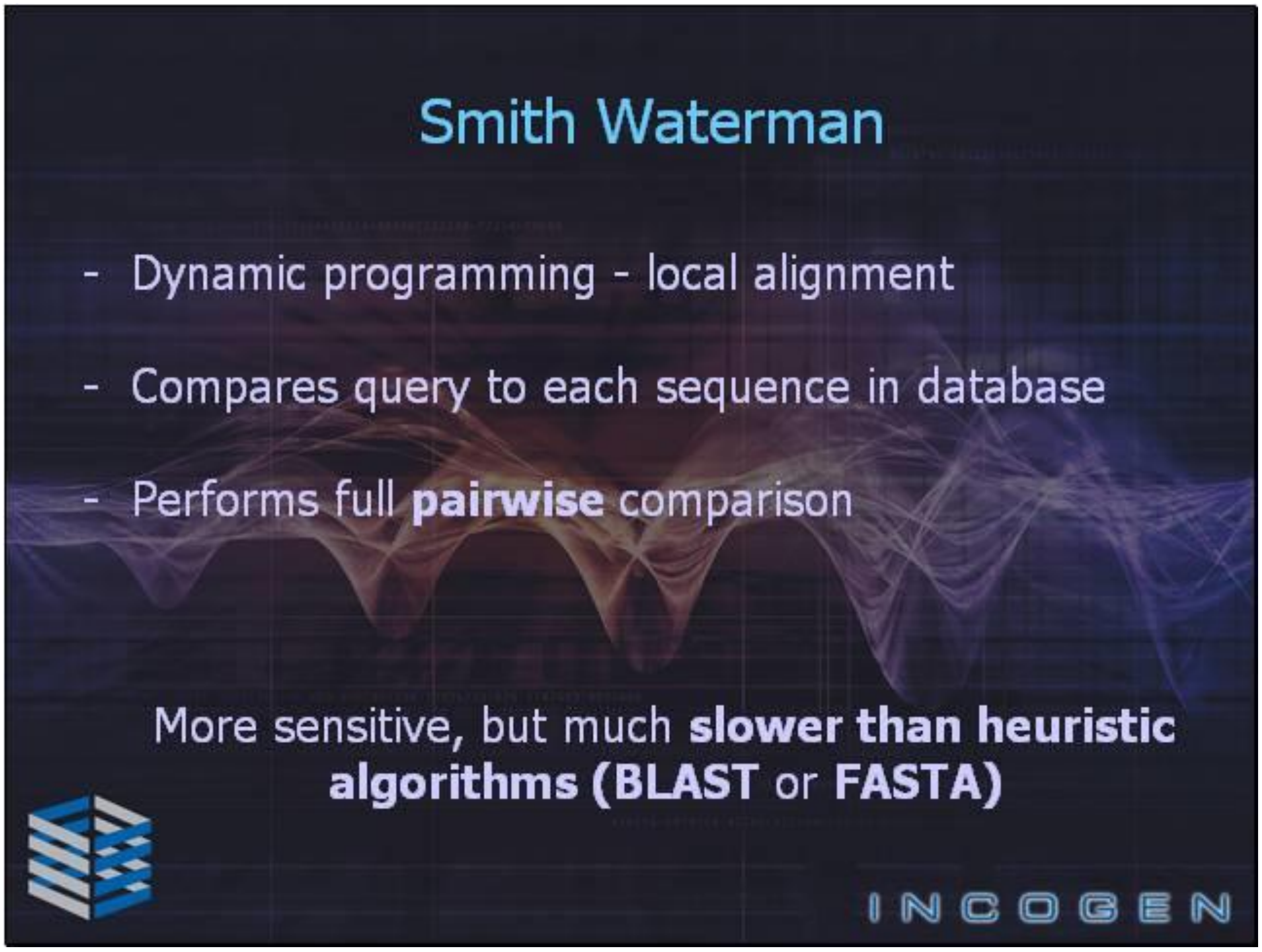
```
104 YPVFSHVQAGMFSPELRTFTKGDAERWVSTTKKASDSAFWLEVEGNSMTA 153
 86 YPLISWVSAGQWMEAVEPYHKRAIENWHDTTVDCSEDSFWLDVQGDSMTA 135

154 PTGSKPSFPDGMLILVDPEQAVEPGDFCIARLGGD.EFTFKKLIRDSGQV 202
136 PAG...LSIPEGMIILVDPEVEPRNGKLVVAKLEGENEATFKKLVMDAGRK 183

203 FLQPLNPQYPMIPCNESCSVVGKVIAS 229
184 FLKPLNPQYPMIEINGNCKIIGVVVDA 210
```

INCOGEN

Slide notes

Sometimes, we are not interested in the best global alignment but in the best local alignment. Local alignments simply align portions of two sequences instead of aligning the entire length of both sequences.

Slide 22 - Smith Waterman



Smith Waterman

- Dynamic programming - local alignment
- Compares query to each sequence in database
- Performs full **pairwise** comparison

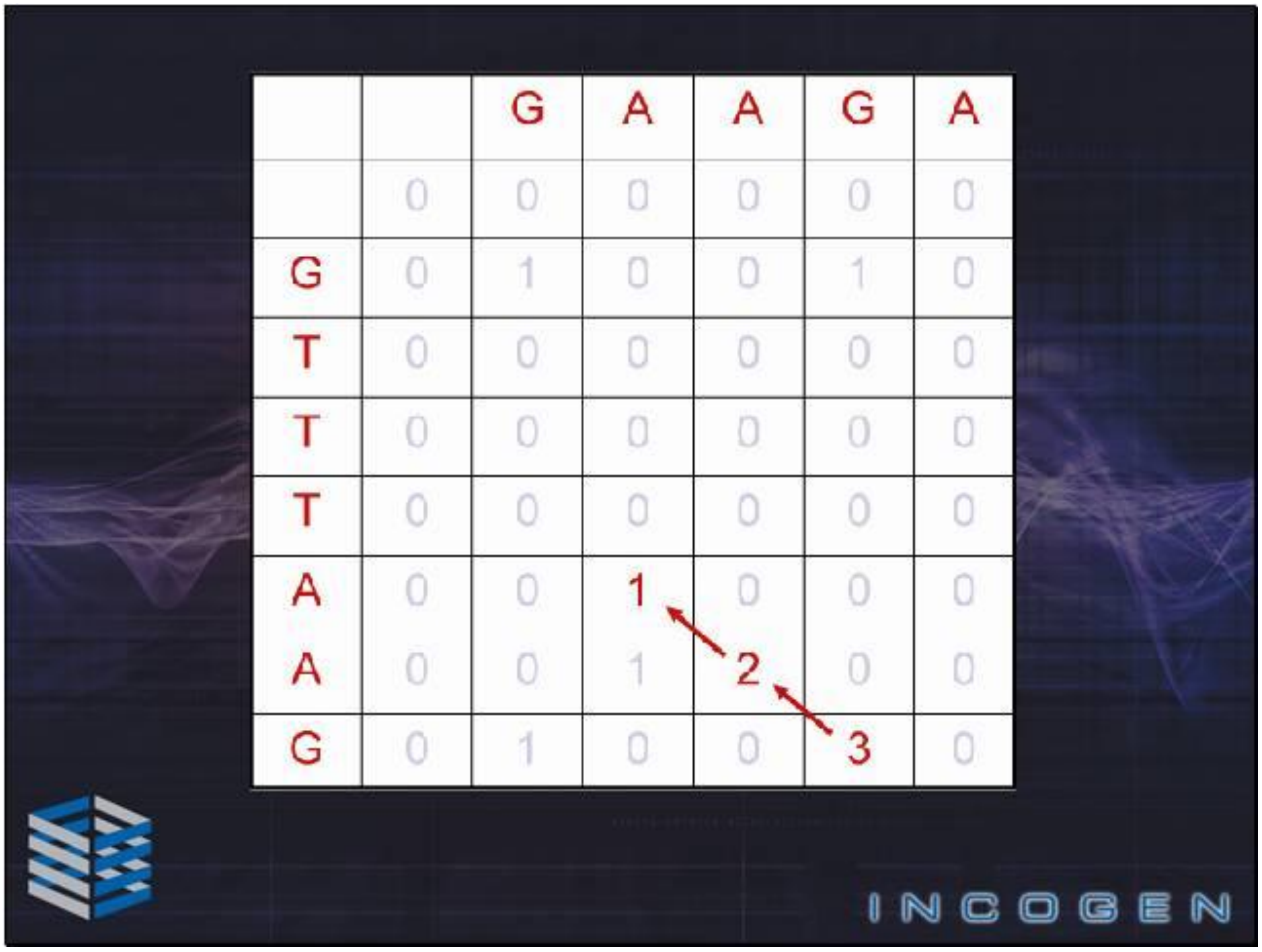
More sensitive, but much **slower than heuristic algorithms (BLAST** or **FASTA)**

INCOGEN

Slide notes

Smith-Waterman is one of the best known examples of a dynamic programming algorithm that finds local alignments. Smith-Waterman does a pairwise comparison between the query and every sequence in a database. The algorithm is very sensitive and is guaranteed to find the highest scoring match. However, it is much slower than either BLAST or FASTA.

Slide 23 - Slide 23

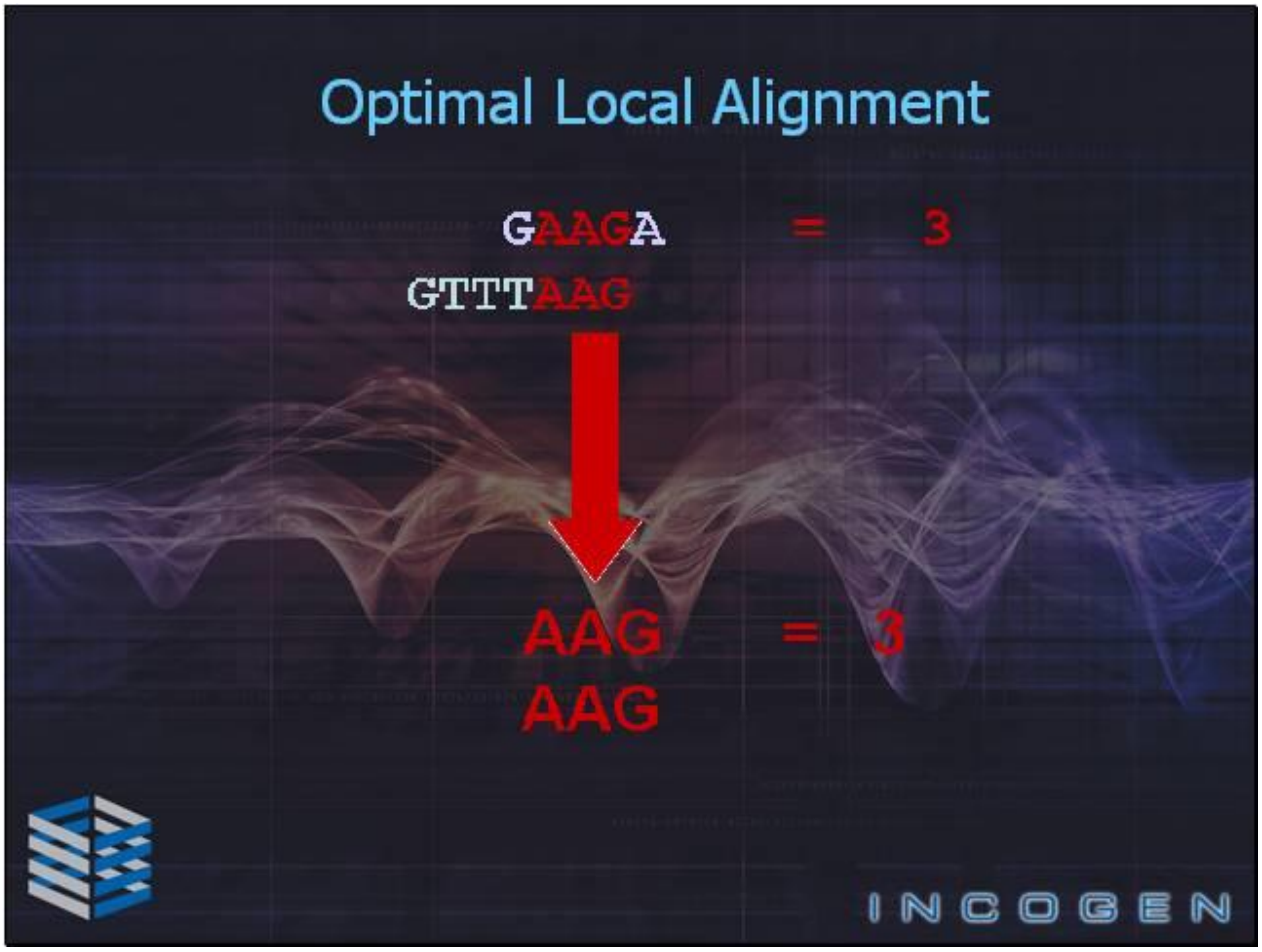


		G	A	A	G	A
	0	0	0	0	0	0
G	0	1	0	0	1	0
T	0	0	0	0	0	0
T	0	0	0	0	0	0
T	0	0	0	0	0	0
A	0	0	1	0	0	0
A	0	0	1	2	0	0
G	0	1	0	0	3	0

Slide notes

We set up a table very similar to the one we set up before. Because this is a local alignment, mismatches and gaps at the beginning and end of the sequences score 0. In addition, we never let the score for any cell drop below zero. Using these modifications, we can find the best local alignment between the same two sequences we used earlier. We simply find the highest score in the table and trace the path back until we come to a cell with a score of zero; this cell is not included in the alignment.

Slide 24 - Slide 24



Optimal Local Alignment

GAAGA = 3

GTTTAAG

AAG = 3

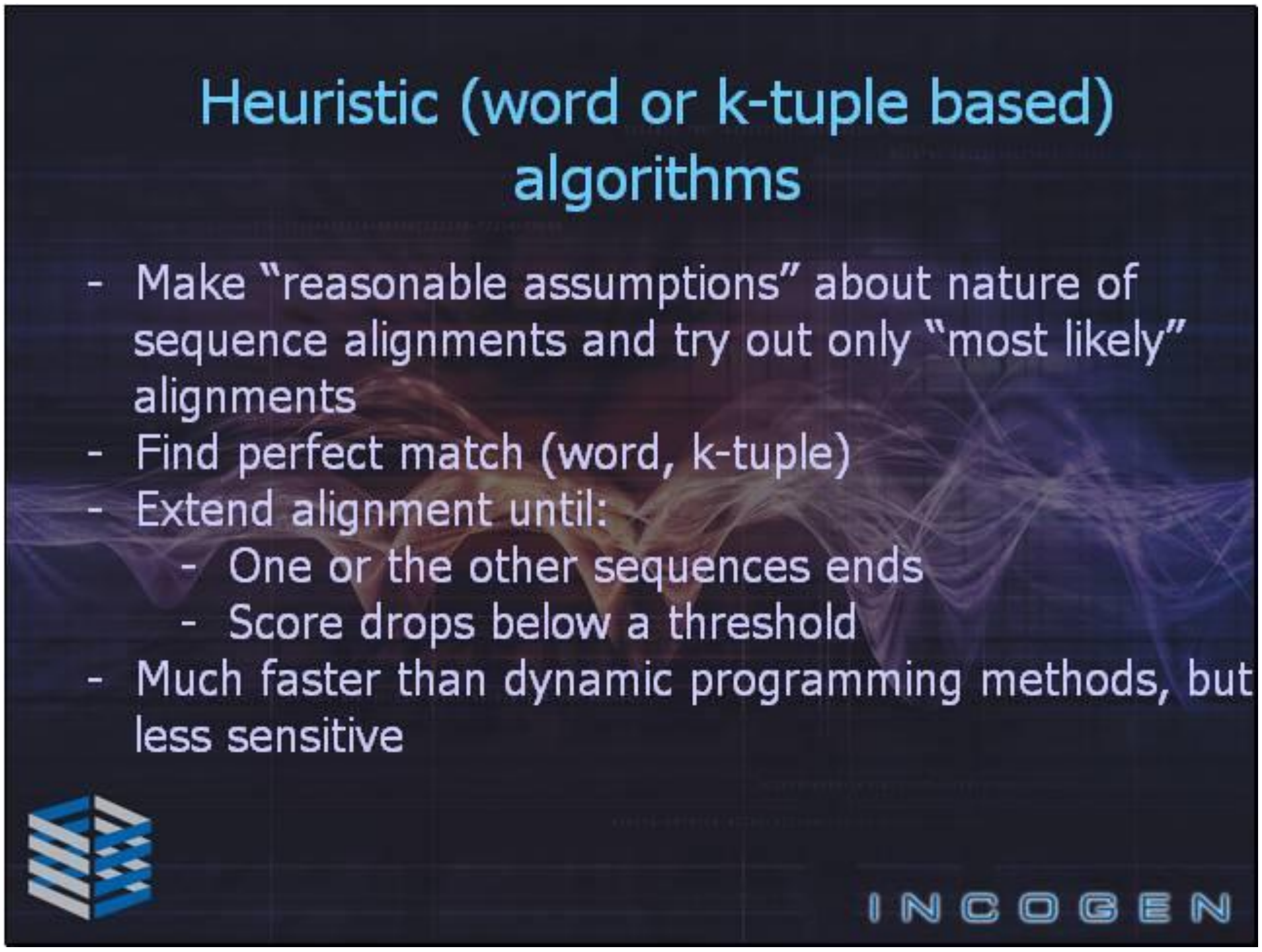
AAG

INCOGEN

Slide notes

Here is the optimal local alignment.

Slide 25 - Heuristic (word or k-tuple based) algorithms



Heuristic (word or k-tuple based) algorithms

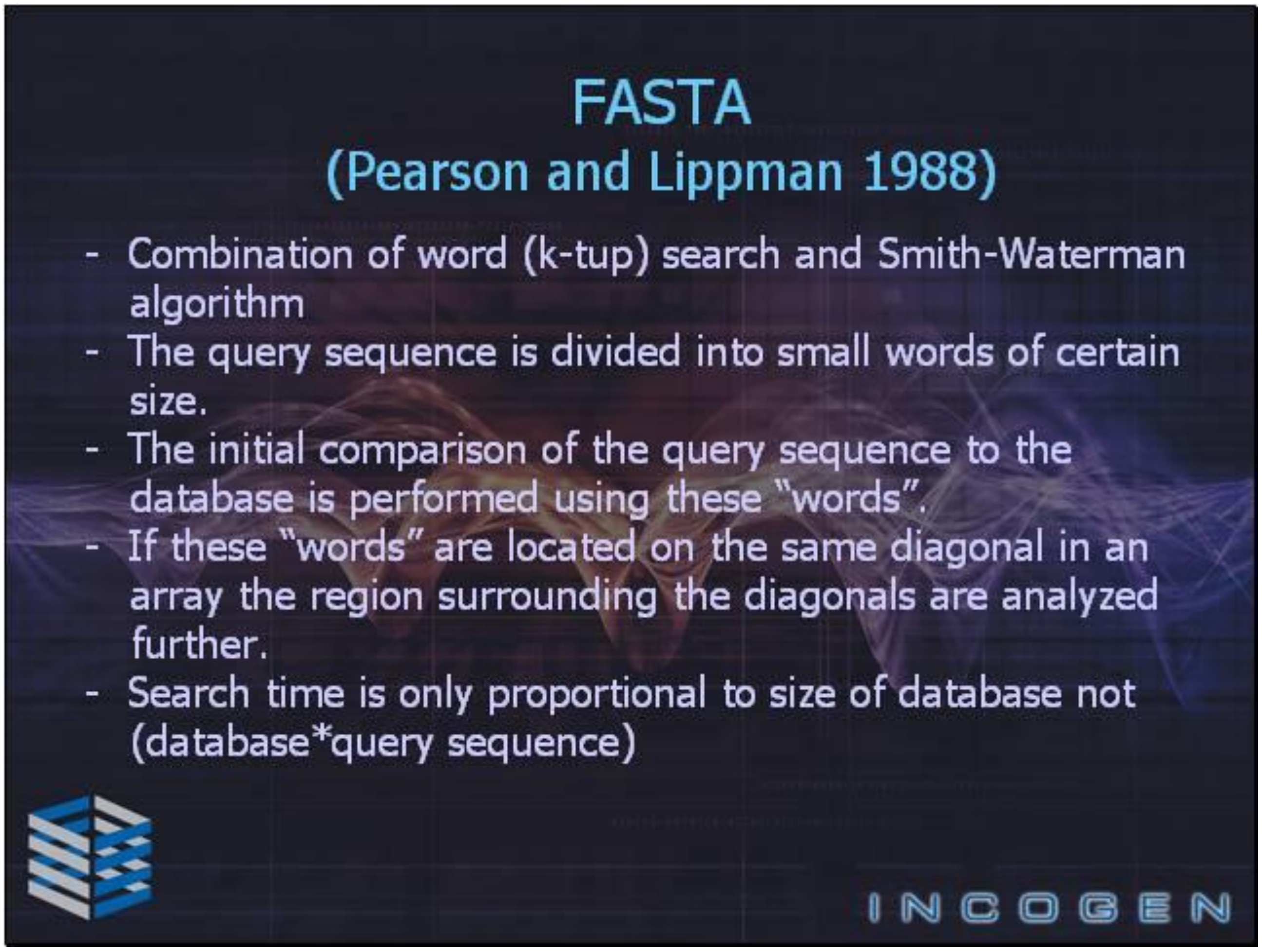
- Make “reasonable assumptions” about nature of sequence alignments and try out only “most likely” alignments
- Find perfect match (word, k-tuple)
- Extend alignment until:
    - One or the other sequences ends
    - Score drops below a threshold
- Much faster than dynamic programming methods, but less sensitive

INCOGEN

Slide notes

The second algorithm type used for finding alignments is the heuristic algorithm. A heuristic algorithm tries to find a perfect match at least as large as a user-entered word size or k-tuple. Once this perfect match is found, the algorithm tries to extend the alignment in both directions until either one of the sequences ends or the score drops below some threshold. Because only the “most likely” alignments are tried, the algorithms run more quickly than Smith-Waterman, but they are less sensitive since they weed out potential good alignments if they don’t contain a large enough perfect match.

Slide 26 - FASTA (Pearson and Lippman 1988)



FASTA
(Pearson and Lippman 1988)

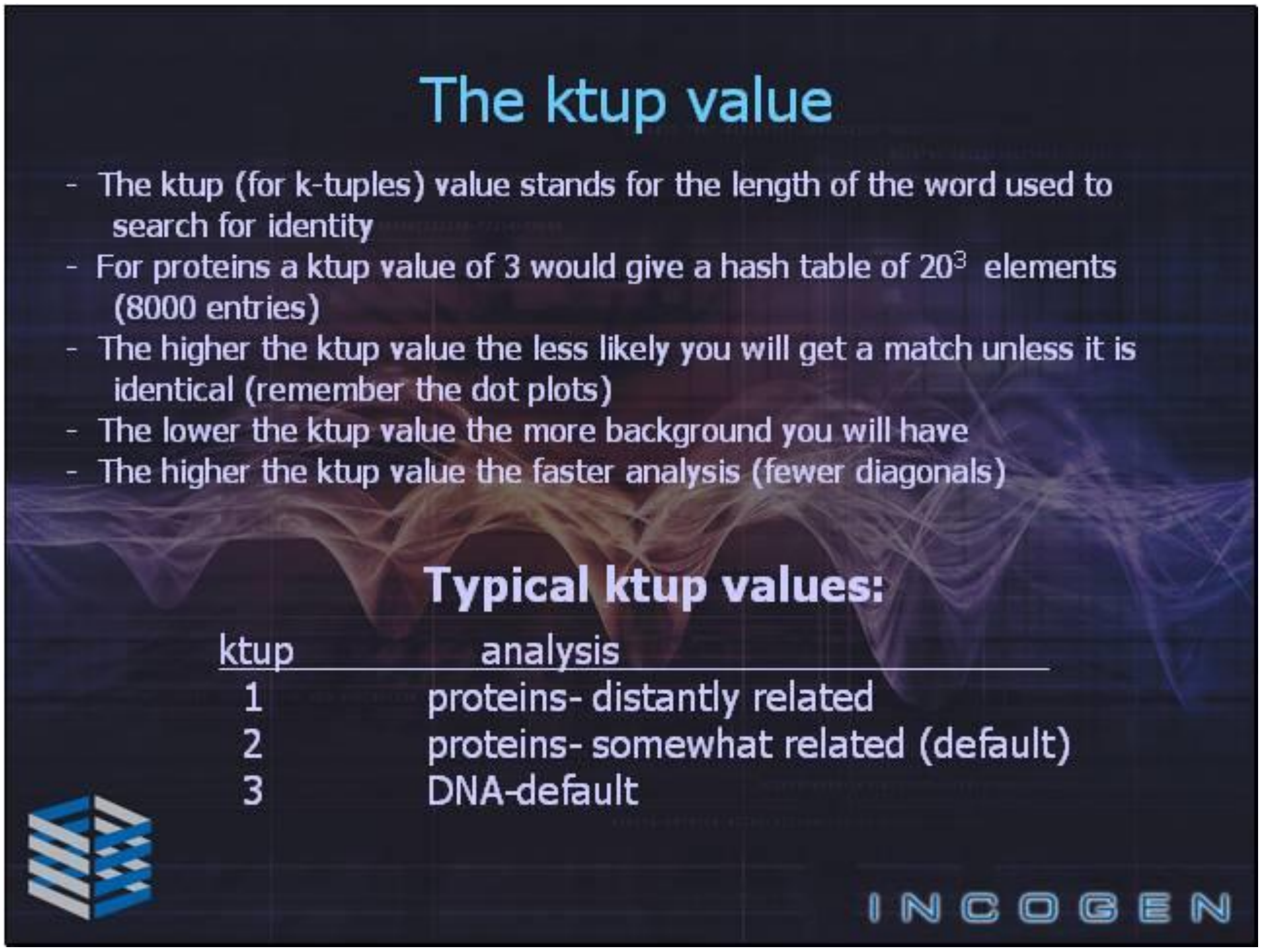
- Combination of word (k-tup) search and Smith-Waterman algorithm
- The query sequence is divided into small words of certain size.
- The initial comparison of the query sequence to the database is performed using these “words”.
- If these “words” are located on the same diagonal in an array the region surrounding the diagonals are analyzed further.
- Search time is only proportional to size of database not (database*query sequence)

INCOGEN

Slide notes

One of the first heuristic algorithms developed was FASTA. FASTA uses a combination of a word search and the Smith-Waterman algorithm. The query sequence is divided into small “words” that are compared with a database. If the “words” are found, the regions surrounding them are then compared and extended as long as they are identical.

Slide 27 - The ktup value



The ktup value

- The ktup (for k-tuples) value stands for the length of the word used to search for identity
- For proteins a ktup value of 3 would give a hash table of 20^3 elements (8000 entries)
- The higher the ktup value the less likely you will get a match unless it is identical (remember the dot plots)
- The lower the ktup value the more background you will have
- The higher the ktup value the faster analysis (fewer diagonals)

Typical ktup values:

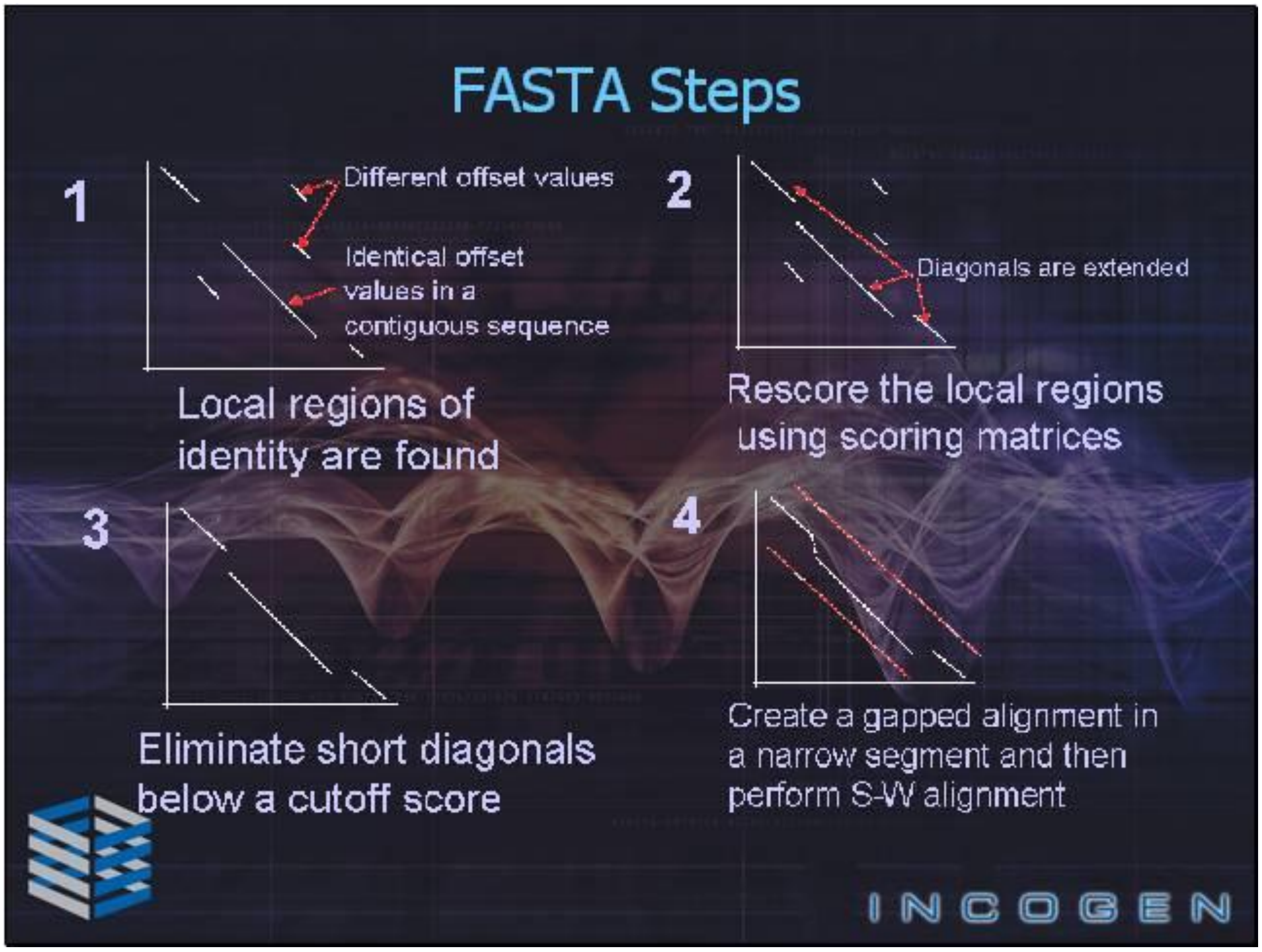
ktup	analysis
1	proteins- distantly related
2	proteins- somewhat related (default)
3	DNA-default

INCOGEN

Slide notes

The length of the “word” used for the initial comparison is called the k-tup value. As the k-tup value increases, the algorithm returns fewer good, but non-identical, alignments, and it runs faster. As the k-tup value decreases, it becomes more likely that you will get superfluous matches. Typical values of k-tup are 1 or 2 for proteins and 3 for DNA.

Slide 28 - FASTA Steps



FASTA Steps

1 Local regions of identity are found

2 Rescore the local regions using scoring matrices

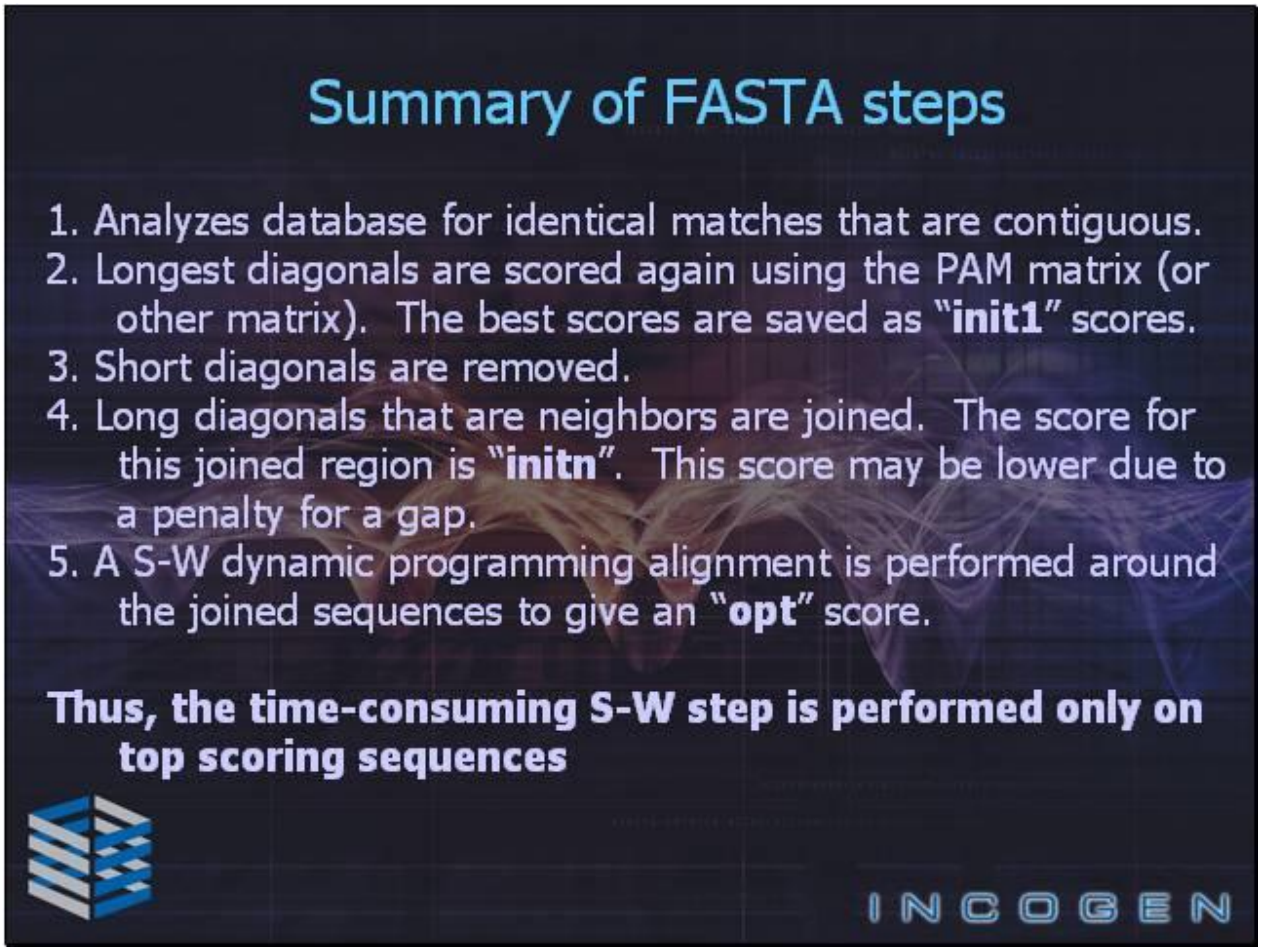
3 Eliminate short diagonals below a cutoff score

4 Create a gapped alignment in a narrow segment and then perform S-W alignment

Slide notes

The first step in the FASTA algorithm is to find all regions in the two sequences that are identical; these show up as a diagonal in the analysis table we looked at earlier. The regions are then scored using a scoring matrix chosen by the user. Matches, or diagonals, with a score below a user-defined cutoff are removed, and the remaining identical regions are aligned using Smith-Waterman.

Slide 29 - Summary of FASTA steps



Summary of FASTA steps

1. Analyzes database for identical matches that are contiguous.
2. Longest diagonals are scored again using the PAM matrix (or other matrix). The best scores are saved as "**init1**" scores.
3. Short diagonals are removed.
4. Long diagonals that are neighbors are joined. The score for this joined region is "**initn**". This score may be lower due to a penalty for a gap.
5. A S-W dynamic programming alignment is performed around the joined sequences to give an "**opt**" score.

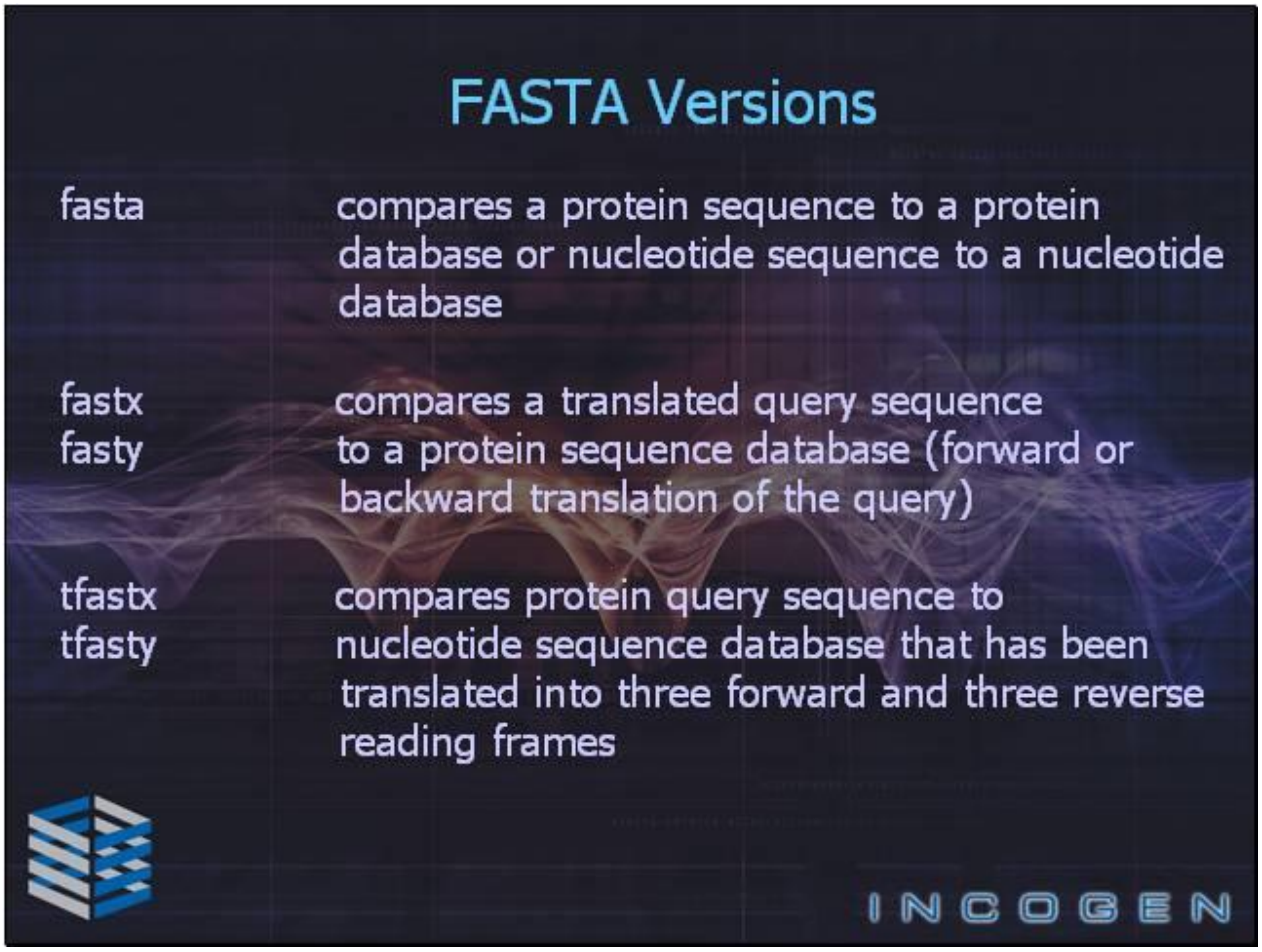
Thus, the time-consuming S-W step is performed only on top scoring sequences

INCOGEN

Slide notes

In summary, FASTA first analyses a database for identical matches, or diagonals, that have a length that is greater than or equal to a user-defined minimum. The diagonals are scored using a scoring matrix, and the shortest ones are removed. The remaining diagonals are joined, and an alignment is performed using Smith-Waterman. A lot of time is saved because Smith-Waterman is only run on the top-scoring matches.

Slide 30 - FASTA Versions



FASTA Versions

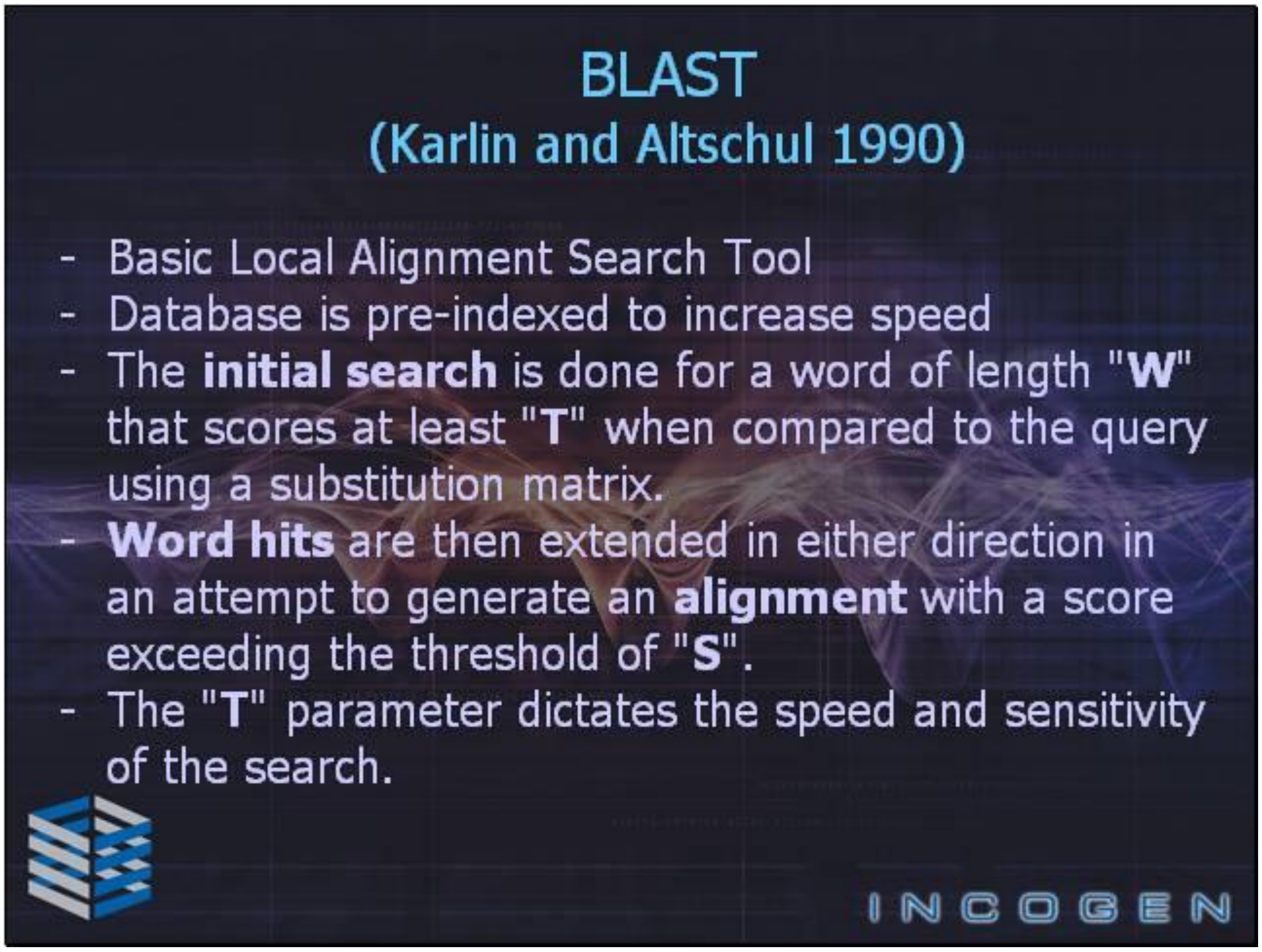
fasta	compares a protein sequence to a protein database or nucleotide sequence to a nucleotide database
fastx fasty	compares a translated query sequence to a protein sequence database (forward or backward translation of the query)
tfastx tfasty	compares protein query sequence to nucleotide sequence database that has been translated into three forward and three reverse reading frames

INCOGEN

Slide notes

The are 5 available versions of FASTA. The fastest, fasta, compares protein sequences to a protein database or DNA sequences to a DNA database. The next two, fastx and fasty, are slower than fasta and compare translated DNA sequences against a protein database. The final two, tfastx and tfasty, are the slowest; they compare a protein sequence to a translated DNA database.

Slide 31 - BLAST (Karlin and Altschul 1990)



BLAST
(Karlin and Altschul 1990)

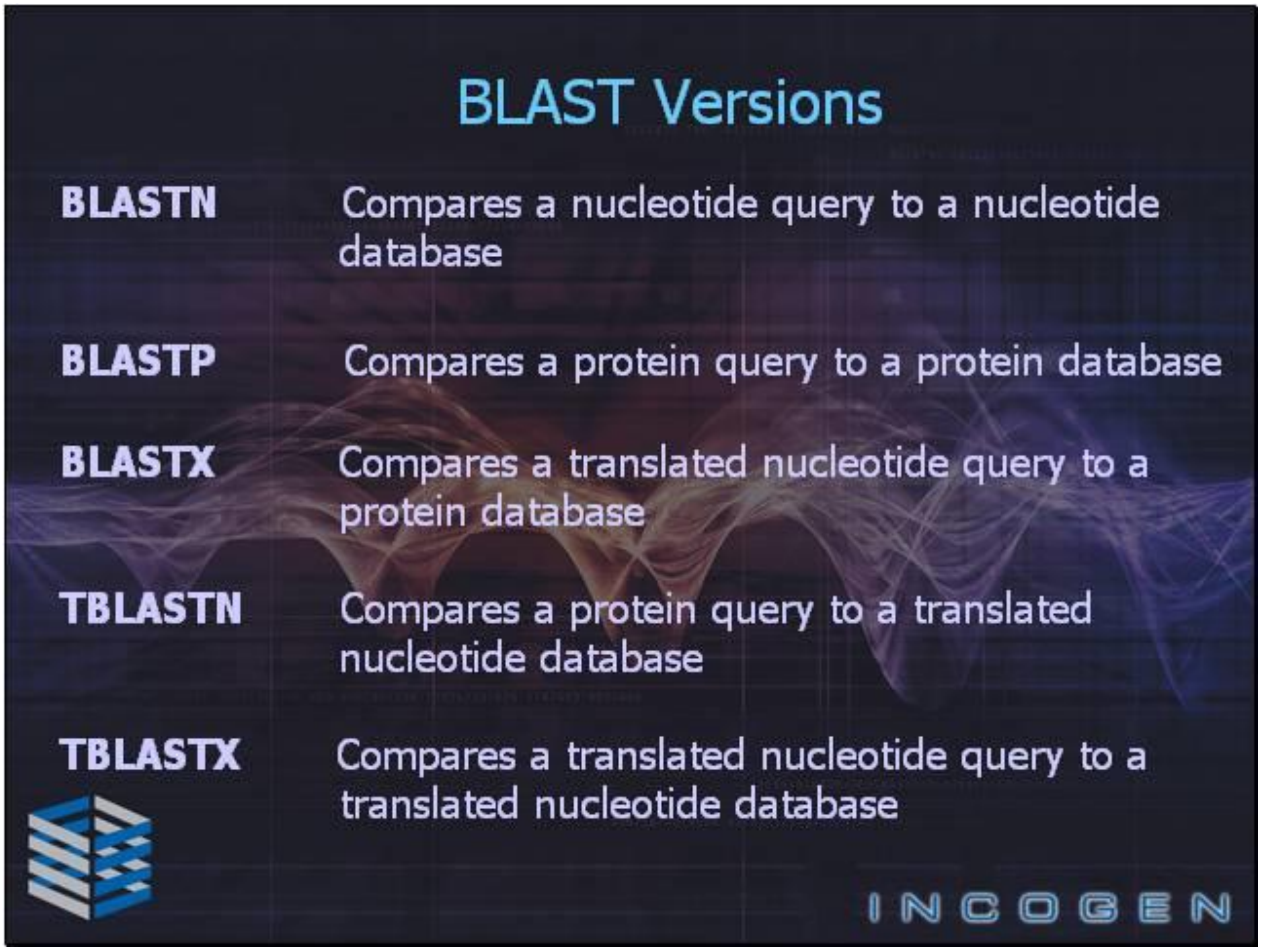
- Basic Local Alignment Search Tool
- Database is pre-indexed to increase speed
- The **initial search** is done for a word of length "**W**" that scores at least "**T**" when compared to the query using a substitution matrix.
- **Word hits** are then extended in either direction in an attempt to generate an **alignment** with a score exceeding the threshold of "**S**".
- The "**T**" parameter dictates the speed and sensitivity of the search.

INCOGEN

Slide notes

Another heuristic algorithm is BLAST. In BLAST, the database is indexed based on “word” and the “words” are compared to the query sequence. The BLAST algorithm does not require identity in the initial matches; it just requires that the score of the alignment between the “word” and the query be higher than a user-defined value. Once a “word” is found that meets this criterion, the aligned region is extended in both directions. If the final alignment has a score that is less than some threshold set by the user, the match is discarded; otherwise it is kept as a potential match.

Slide 32 - BLAST Versions



BLAST Versions

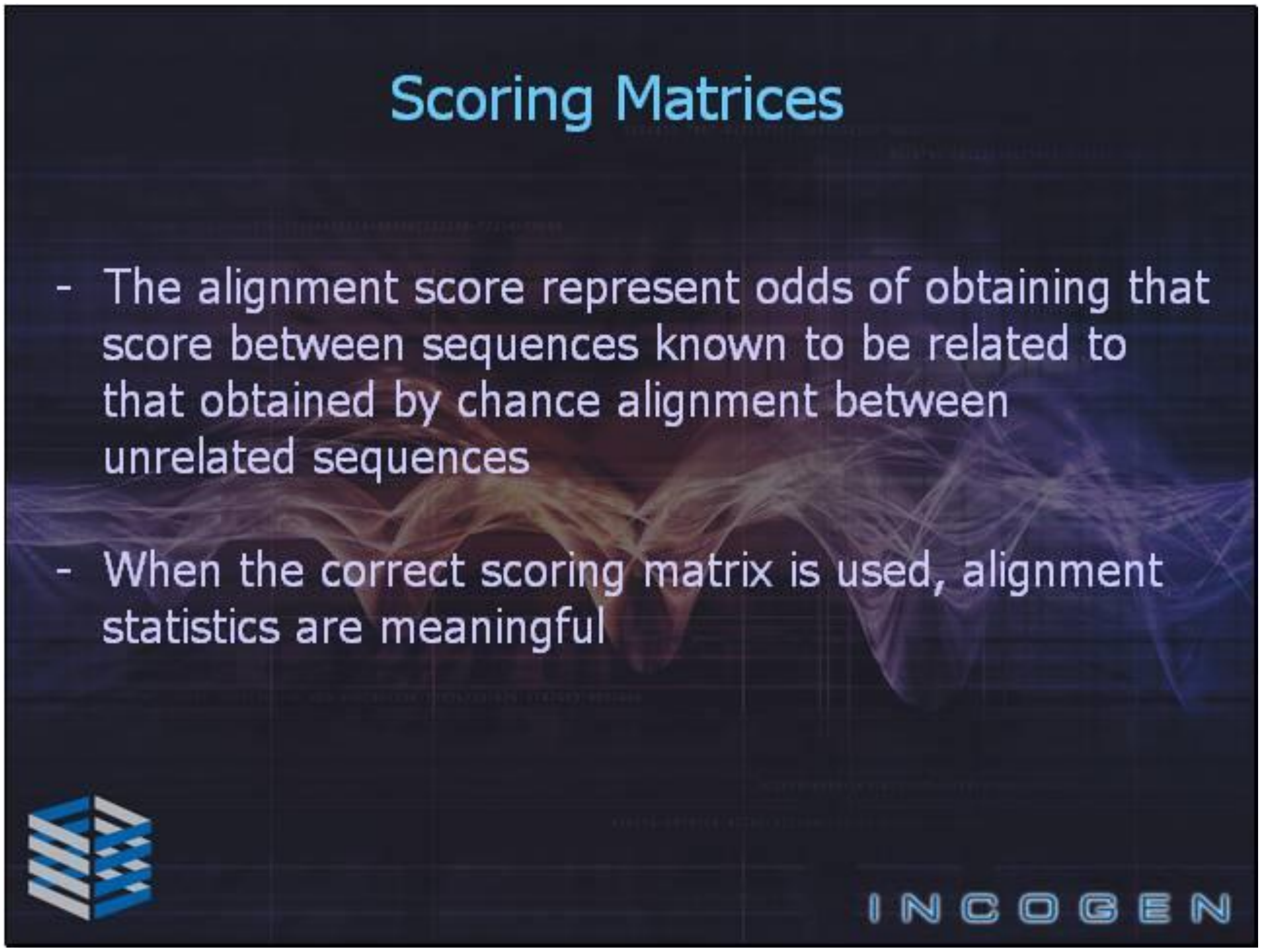
BLASTN	Compares a nucleotide query to a nucleotide database
BLASTP	Compares a protein query to a protein database
BLASTX	Compares a translated nucleotide query to a protein database
TBLASTN	Compares a protein query to a translated nucleotide database
TBLASTX	Compares a translated nucleotide query to a translated nucleotide database

INCOGEN

Slide notes

There are also 5 versions of BLAST available. BLASTN compares a DNA sequence to a DNA database. BLASTP compares a protein sequence to a protein database. BLASTX compares a DNA sequence to a protein database. TBLASTN compares a protein sequence to a translated DNA database. TBLASTX, which typically takes the longest of the five to run, compares a translated DNA sequence to a translated DNA database.

Slide 33 - Scoring Matrices



Scoring Matrices

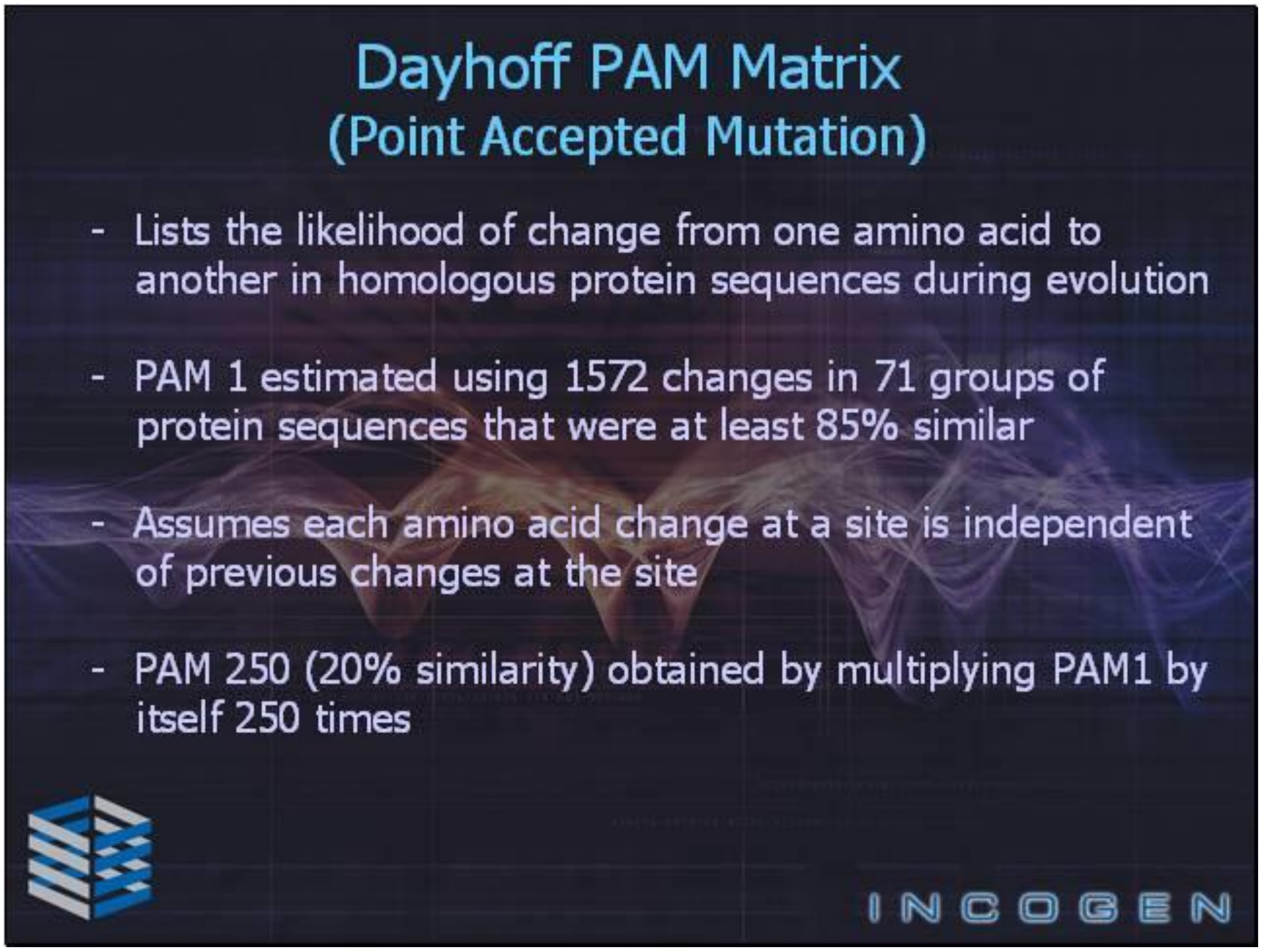
- The alignment score represent odds of obtaining that score between sequences known to be related to that obtained by chance alignment between unrelated sequences
- When the correct scoring matrix is used, alignment statistics are meaningful

INCOGEN

Slide notes

We mentioned the concept of scoring matrices several times in the last few slides. A scoring matrix represents the odds of obtaining a particular match between sequences that are known to be related as opposed to obtaining the match between unrelated sequences. For alignment statistics to be meaningful, the correct scoring matrix must be used.

Slide 34 - Dayhoff PAM Matrix (Point Accepted Mutation)



Dayhoff PAM Matrix (Point Accepted Mutation)

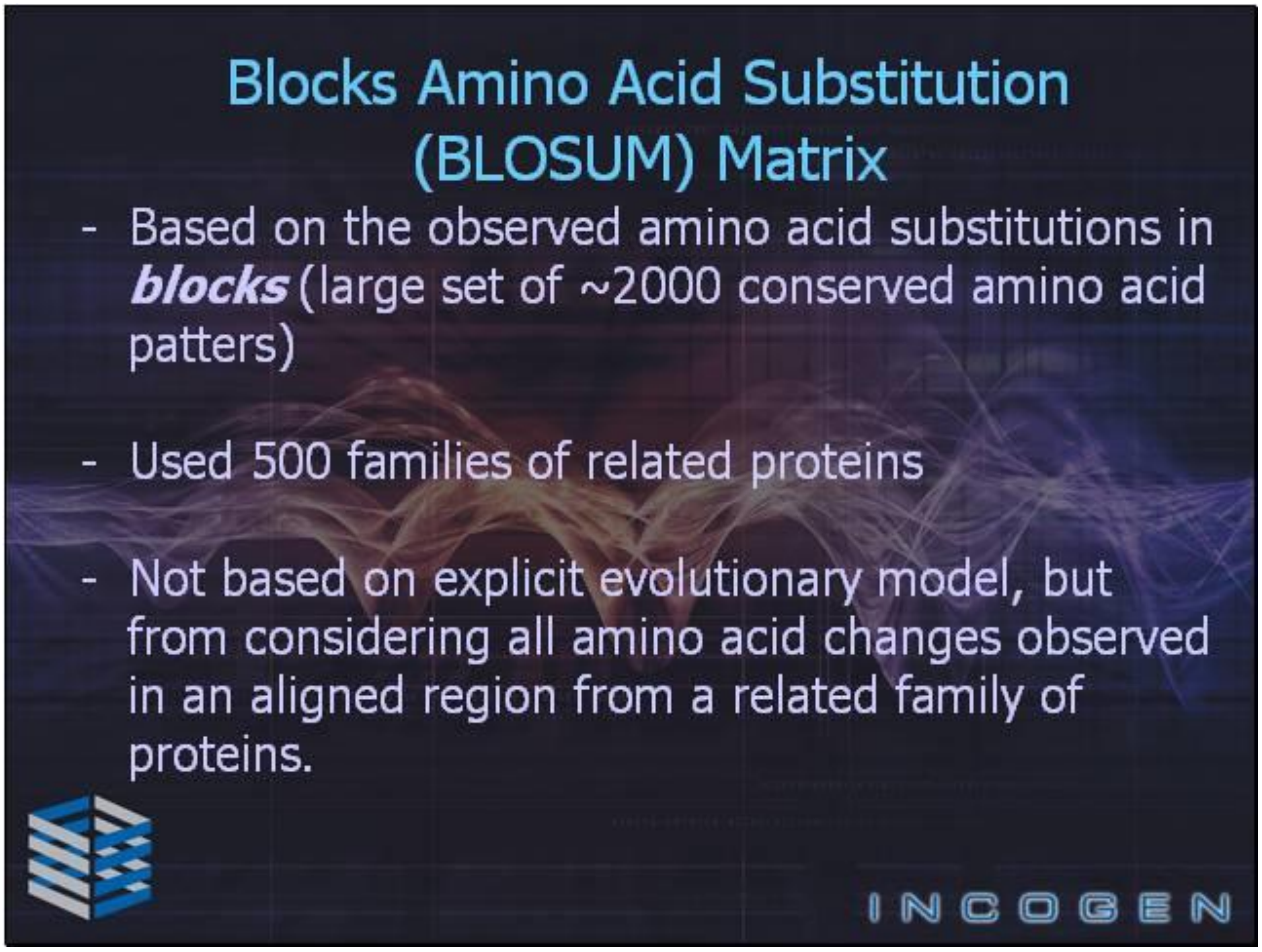
- Lists the likelihood of change from one amino acid to another in homologous protein sequences during evolution
- PAM 1 estimated using 1572 changes in 71 groups of protein sequences that were at least 85% similar
- Assumes each amino acid change at a site is independent of previous changes at the site
- PAM 250 (20% similarity) obtained by multiplying PAM1 by itself 250 times

INCOGEN

Slide notes

The first family of scoring matrices we will look at is the Dayhoff PAM matrix. The first PAM matrix, PAM 1, was initially calculated by looking at the differences between sequences that were 85% similar. To calculate matrices for other similarity levels, it was assumed that each amino acid change was independent of previous changes at the same site. The matrices list the likelihood that one amino acid could change to another in homologous sequences during evolution.

Slide 35 - Blocks Amino Acid Substitution (BLOSUM) Matrix



Blocks Amino Acid Substitution (BLOSUM) Matrix

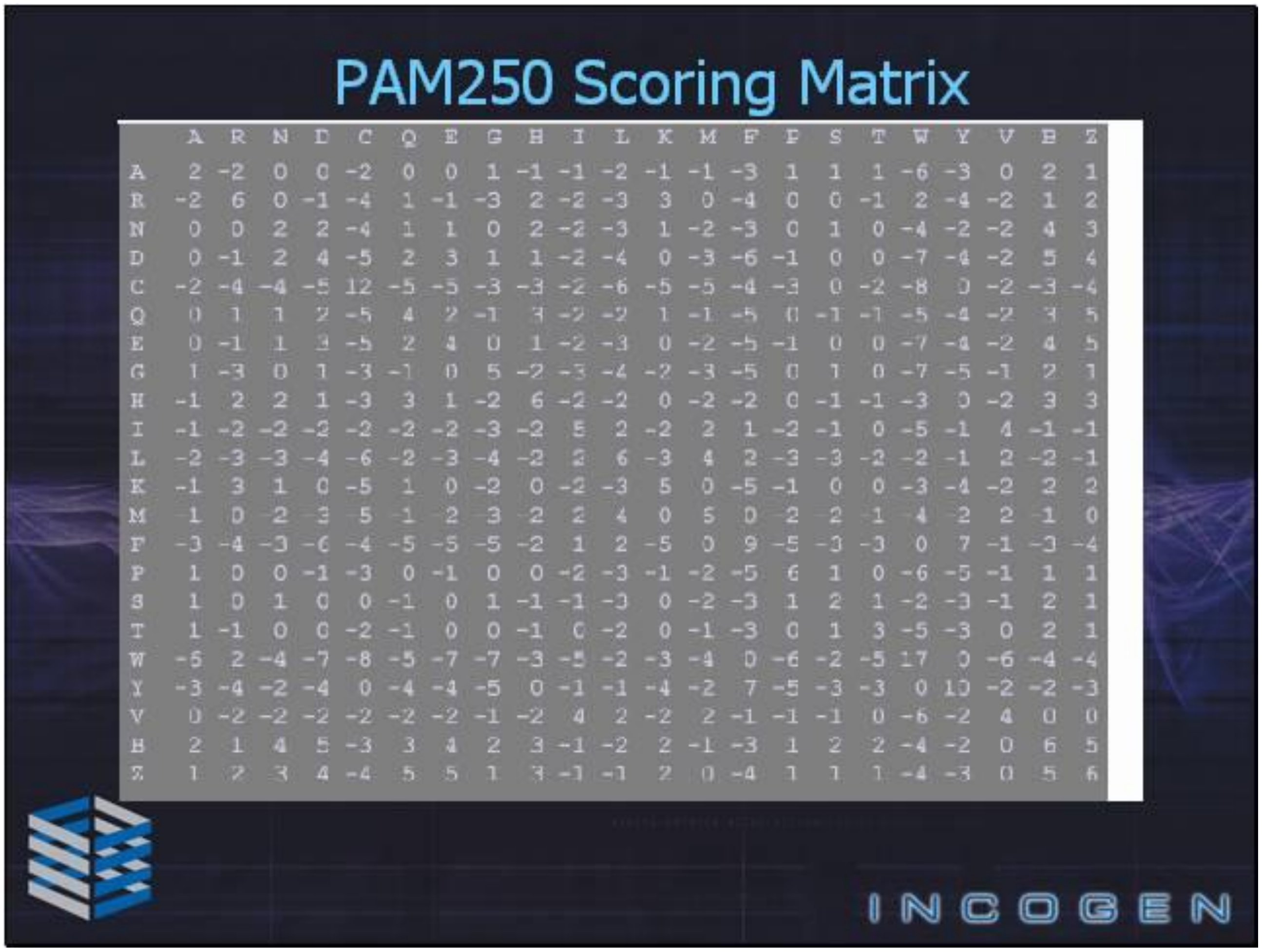
- Based on the observed amino acid substitutions in ***blocks*** (large set of ~2000 conserved amino acid patters)
- Used 500 families of related proteins
- Not based on explicit evolutionary model, but from considering all amino acid changes observed in an aligned region from a related family of proteins.

INCOGEN

Slide notes

The second family of scoring matrices is the BLOSUM matrix. The BLOSUM matrices were based on the amino acid substitution rate in highly conserved blocks. The BLOSUM matrices are not explicitly based on an evolutionary model, as the PAM matrices are, but are based instead on related families of proteins.

Slide 36 - Slide 36



PAM250 Scoring Matrix

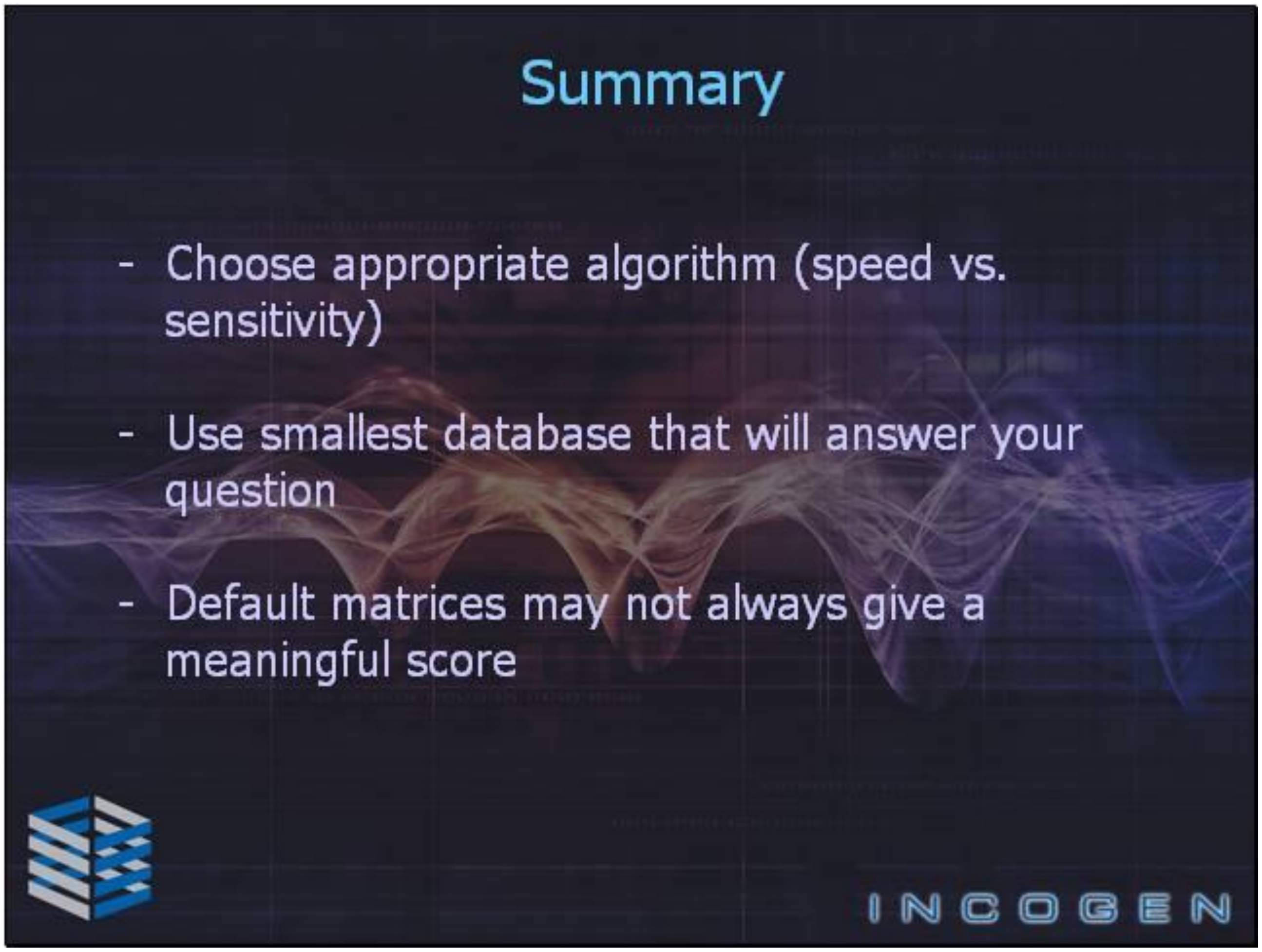
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z
A	2	-2	0	0	-2	0	0	1	-1	-1	-2	-1	-1	-3	1	1	1	-6	-3	0	2	1
R	-2	6	0	-1	-4	1	-1	-3	2	-2	-3	3	0	-4	0	0	-1	2	-4	-2	1	2
N	0	0	2	2	-4	1	1	0	2	-2	-3	1	-2	-3	0	1	0	-4	-2	-2	4	3
D	0	-1	2	4	-5	2	3	1	1	-2	-4	0	-3	-6	-1	0	0	-7	-4	-2	5	4
C	-2	-4	-4	-5	12	-5	-5	-3	-3	-2	-6	-5	-5	-4	-3	0	-2	-8	0	-2	-3	-4
Q	0	1	1	2	-5	4	2	-1	3	-2	-2	1	-1	-5	0	-1	-1	-5	-4	-2	3	5
E	0	-1	1	3	-5	2	4	0	1	-2	-3	0	-2	-5	-1	0	0	-7	-4	-2	4	5
G	1	-3	0	1	-3	-1	0	5	-2	-3	-4	-2	-3	-5	0	1	0	-7	-5	-1	2	1
H	-1	2	2	1	-3	3	1	-2	6	-2	-2	0	-2	-2	0	-1	-1	-3	0	-2	3	3
I	-1	-2	-2	-2	-2	-2	-2	-3	-2	5	2	-2	2	1	-2	-1	0	-5	-1	4	-1	-1
L	-2	-3	-3	-4	-6	-2	-3	-4	-2	2	6	-3	4	2	-3	-3	-2	-2	-1	2	-2	-1
K	-1	3	1	0	-5	1	0	-2	0	-2	-3	5	0	-5	-1	0	0	-3	-4	-2	2	2
M	-1	0	-2	-3	-5	-1	-2	-3	-2	2	4	0	5	0	-2	-2	-1	-4	-2	2	-1	0
F	-3	-4	-3	-6	-4	-5	-5	-5	-2	1	2	-5	0	9	-5	-3	-3	0	7	-1	-3	-4
P	1	0	0	-1	-3	0	-1	0	0	-2	-3	-1	-2	-5	6	1	0	-6	-5	-1	1	1
S	1	0	1	0	0	-1	0	1	-1	-1	-3	0	-2	-3	1	2	1	-2	-3	-1	2	1
T	1	-1	0	0	-2	-1	0	0	-1	0	-2	0	-1	-3	0	1	3	-5	-3	0	2	1
W	-6	2	-4	-7	-8	-5	-7	-7	-3	-5	-2	-3	-4	0	-6	-2	-5	17	0	-6	-4	-4
Y	-3	-4	-2	-4	0	-4	-4	-5	0	-1	-1	-4	-2	7	-5	-3	-3	0	10	-2	-2	-3
V	0	-2	-2	-2	-2	-2	-2	-1	-2	4	2	-2	2	-1	-1	-1	0	-6	-2	4	0	0
B	2	1	4	5	-3	3	4	2	3	-1	-2	2	-1	-3	1	2	2	-4	-2	0	6	5
Z	1	2	3	4	-4	5	5	1	3	-1	-1	2	0	-4	1	1	1	-4	-3	0	5	6

INCOGEN

Slide notes

This is an example of what the scoring matrices look like. In particular, this is the log-odds form of the PAM 250 matrix, which represents about 20% similarity. In general, similar amino acids have a greater chance of substituting for each other.

Slide 37 - Summary



Summary

- Choose appropriate algorithm (speed vs. sensitivity)
- Use smallest database that will answer your question
- Default matrices may not always give a meaningful score

INCOGEN

Slide notes

In summary, take speed and sensitivity into account when choosing the appropriate algorithm for determining alignments. To further speed up the algorithms, use the smallest database that will answer your question. Last, take some thought when picking which scoring matrix to use; the default matrix may not provide you with meaningful results.