# CS 274

Representation and algorithms for computation molecular biology

# Life on Earth

universe --> 13 billion years old
life on earth --> 4.3 billion years ago
life --> orderly system of molecules that evolve and reproduce

Entropy (S) is the main challenge for living organisms

- The tendency towards disorder
- 2nd law of thermodynamics
  - Entropy always increases
  - You face the challenge of increasing entropy
  - Sum of the entire system has to be such that entropy goes up
  - \Sum p_i (log(p_i)) where i is is a potential state of the system
  - High entropy is associated with populating many different states, low entropy is associated with few states


Life = constantly struggle against entropy

- Exists stragtegies
  - In/out distinction --> if continuous, you will increase
    - Cell
  - Gather energy allows you to maintain order
    - ID energy source
    - get access to source
    - bring the energy inside
  - Tranduce all energy into chemical energy (the common currency)
    - ATP is the universal high energy molecule
  - Optimize the above systems (evolution)
  - Transmit the optimized systems to offspring (other living systems)

**CS274 : Computing about the basic life on earth technlogies**

**Central Dogma of Molecular Biology :**

DNA --> Encodes RNA --> protiens --> protein network --> observable traits of life

4 DNA bases --> ACTG
Human Genome : 3E^9
Double helix, one from each parent
5' end and 3' end
Information in one strand is sufficient to make replica cell

Gene : a sub sequence of DNA that is associated with some function

Sum of all genes and functions is life

RNA --> Working copy of the DNA (ACUG)
5' end and 3' end. Differs by just one oygen
Forms complex structures --> can make helices with itself
RNA's main role is to be a messenger or mRNA

DNA --> RNA --> protein --> network of interacting molecules
polymers (4 letters)

Protien --> Polymer, 20 letters, Amio Acids / Residues
Protien string has an N terminus and a C terminus
Ribsome --> this is where the cell translates the messanger RNA code

- Takes sets of 3 from the MRNA, figures out what the matching protein is, and then links the amino acid residues together until we have a protein
- Each amino acid / residue has a central structure and then an additional R group attached on

DNA Databases :
Genbank : 39 * 10 ^ 12 sequences
Gene Expression Omnibus (GEO) : messenger RNA data, 10 million samples

unuiprot —> 253 * 10^6 protein sequences

Protein - Protein interactions : PPIs

Different Pathways

- Metabolic Pathways --> Start with a small molecule, series of enzymic reactions that catalyze some molecule and usually ends with ATP
- Physical Complexes
  - Ribosome : bunch of RNA + bunch of proteins
- Signalling
  - Rhodopsin and photons to see
- Transcirptional Control Network
  - how do protiens / transcription factors turn on / off the transcription of other protiens

StringDB : 59 million proteins, 20 billion PPIs
Kegg : Reactome

Lipids

- Hydrophilic heads and hydrophobic tails, create membrane

How do we take these sequences and analyze them?

Biology is really about the common ancestry of life and really the common ancestry of two proteins.

# Lecture 2

## Agenda

Evolution

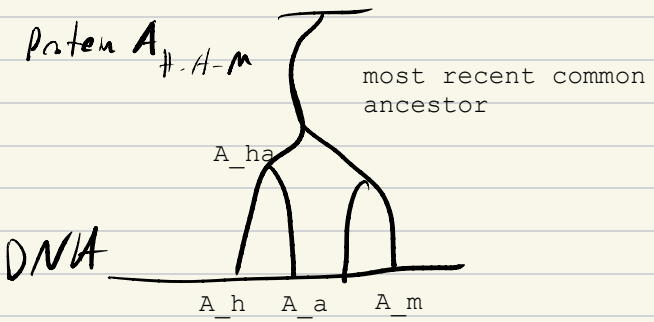
Pairwise Alsignment

Dynamic Programming

Example using DNA

Bad News

Hope



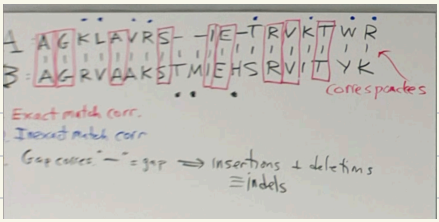

## Evolution

- Common ancestor for all life on earth —> All descended from this ancestor
- Human-ape-mouse —> common node for all three
- Genomes and individual proteins for al of these common ancesstor
- Why does protien A change over time
  - larger amount of different given how far apart in the tree of life
  - Benign mutated drift
    - Small random changes that cause no harm so these paths are not pruned
  - Species level fine tuning
    - Protien may need to be different from specifes to species in order to be optimal for a given animal
- In evolution, structure is rarely rarely changed, while sequence changes often

## Pairwise Sequence Alignment

- Input:
  - 2 strings of letters in a fixed order
  - socring scheme for aligned letters
  - gap penalty scheme
- Output:
  - Optimal set of correspondences between letters in the two sequences
  - Optimal in the sense that it maximizes a scoring function

Question : how to do determine the score for a match




2 sequences

A : AGK LAVRS- - IE - TRVKTWR

B : AGRVAAKSTMIEHSRVITYK

Alignment algorithm produces a set of correspondances. each black line is a correspondance, an exact correspondance.

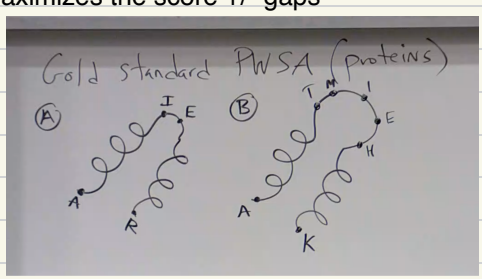
*Exact Match Corr* - A few exact matches, we can say that there is some potential for these to be common ancestors --> High positive

*Inexact Match Corr* - Amino Acids have an additional R group added on, for inexact matches, the R groups are chemically verry similar. We can usually attribute this to begnin drift --> Low positive, if very different, negative

Everything but K & I are quite similar, size, oxygen, electronegativity etc.

*Gap Correspondances* - algorithm has decided that either an insertion or deletion happened --> Leads to a penalty. These gaps in the match are the product of the algorithm. The algorithm tries to maximizes the score +/- gaps

Gold Standard PWSA (proteins)



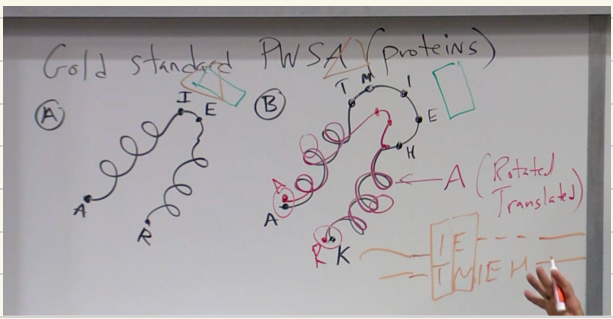

Graphically, the goal of PWSA is to bring A and B together in order to find the best possible alignment.

Sequence alignments are hypotheses

A correspondence fom an alignment is a hypothesis that the the aligned letters have the same structural and functional role

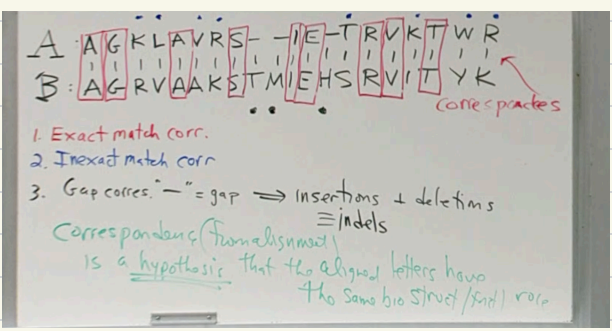
An alginment is a set of hyptoheses

if theres is a gap, we are hypothesizing that the associated protien has no use for a given residue. Thus, we can hypothesize that T for example has no real role in protien B.




There may be a dissonance between the biology and the math. If two residues do something different between the matching proteins, then we have the wrong alignment

In the example above, IE bind to the green triange, and TM bind to the green triangle, than that is the correct match.



## Dynamic Programming Algorithm

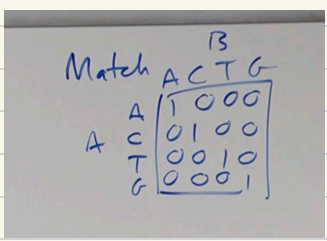
If you allow a gap of any length allowed in both sequences makes this an exponential run time —> dynamic programming gets us around this

Optimal solution is constructed a non-exponential number of sub problems and assembling them correctly

Specific subproblem that we are going to be solving is the score of all alignments of A and B ending at A[i] and B[j]

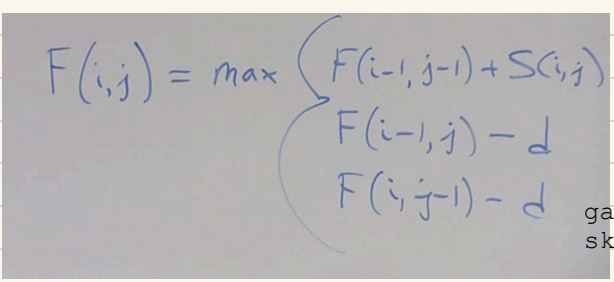
Scoring scheme

For this example we will do a DNA sequence
g = length of the gap
d = penalty per -
gap penalty function = - g * d




A : AGGC
B : AATGB

Score matrix F_(mxn) --> F(i,j) = score of best alignment that includes A[i] and B[j]




S(i,j) =
Score(A[i], B[j])

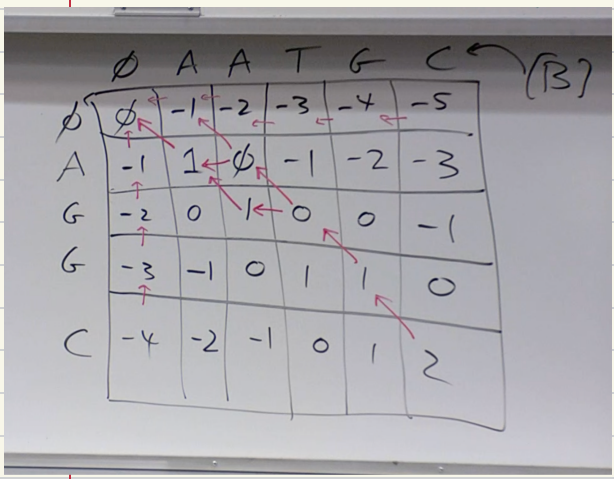
gap penalty is considering skip

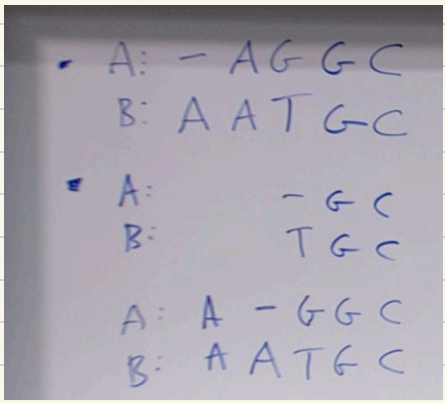
F(0,0) = 0
F(0,j), F(i,0) --> Depends on the gap policy

1) you can console everythign and penalize end gaps
F(0,j) = -dj
F(i, 0) = -di
2) alternatively, you can ingore end gaps and not include them in the penality

Procedure —>

1) Fill in F(0,0) to F(M,N) while tracking the match rule that you used
2) Find score at F(M,N)
3) Trace back to recover optimal alignment(s) depending on the policy you used




traceback rule

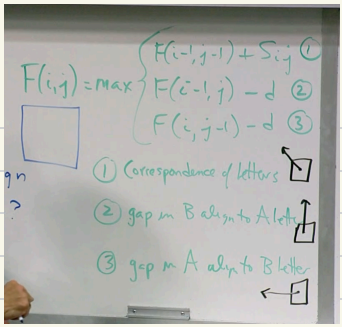
- look at arrow first
- if diagonal, emit 2 alligned letters from A,B
- if horizontal : emit B(j) align "-"
- if vertial : emit A(j) align "-"

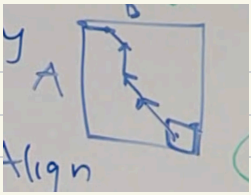
When we move horizontally or vertically we have to take the pentalty, not the score, we used a gap in this context. Make sure the dimensions are going to be the lenghts of the strings that we are trying to align

```
O(mn) - Quadratic in time
```

```
Gap penalty is too simple for most biologist, they add an
additional score matrix for when you have gaps in sequence A and
B
```

# Lecture 3




For project 1, will need to come up with a strategy for back computing the arrows Depending on the backwards arrows, will emit the letters. At any point, evaluating match or a gap in one of the sequences. Path through the matrix tells the least costly path.

why is the gap not dependent on the residue that it is gapped with? - 'great q, algorithm is general'

## Bad News

gap penalty used to be quite trivial : - g * d

Biologist don't like this gap penalty

- The baddness or a gap is not linearly realted to its length
- biggest bad news is finding out that there is one (means there was an insertion or deletion)
- Then longer gap having more penality is true, but not necessarily linearly bad

## New Gap Penalty

Biolgist perferred gap penalty

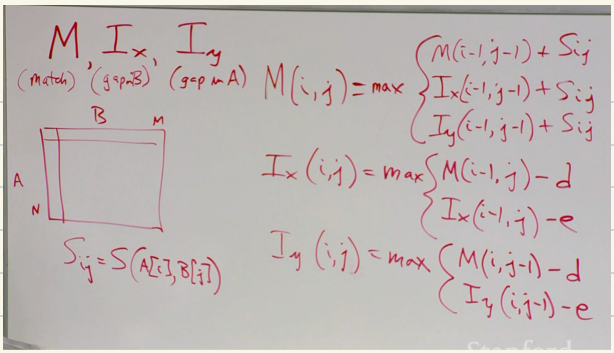
- Penality : - d - (g - 1)e
- d : gap opening (cover charge for the gap)
- e : gap extension penality (like buying drinks), how much you pay for each additional gap
- g : length of the gap

**New Recurence**

M : Match

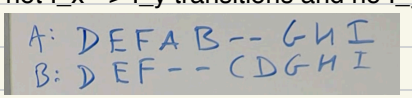
I_x : Gap B

I_y : Gap A




New recurrence using the new matrices, M - we have two letters that are matching. There is no ned to pay the additional gap penalty if you are already in a gap. If you are in one of the Is, you can either open a gap from a M, match, or you can continue an open gap and pay the gap penality e

There is not I_x --> I_y transitions and no I_y --> I_x transitions

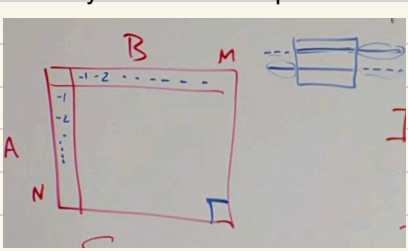



biologically undefined

Biologically, we hate this. The whole point of a gap is that there is a single indel, that is not compatible, an indelible would nevel leave this series of evolutionary events

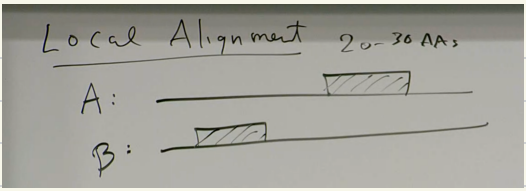
Highest score for each matrix. Pointers between the different matrices. Depending on the recursions, it will point conceptually to a different matrix

Boundary conditions : depends on if we are penalizign these endgaps




Global Alignment - All AAs in both protients have correspondences with other AAs or gaps

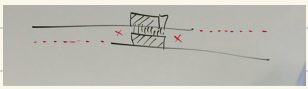
## Local v Global Alignment




Needleman — Wunsch

**Local**

Answers the questions : do sequece A and B have a shared subsequence that is extremely simliar



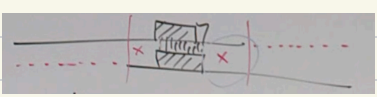
Motifs - re-used subseuqneces of 20-30 amino acids that have the same functionality throughout history. The local alignment we are trying to create will never match the two sequences that we want up

**Smith — Waterman rules**

1. M, I_x, I_y can never have entries less than zero. If there should be a negative, put a zero in there instead
2. Matrix matrix S must have negative scores
3. Start the traceback --> find the best score anywhere in M

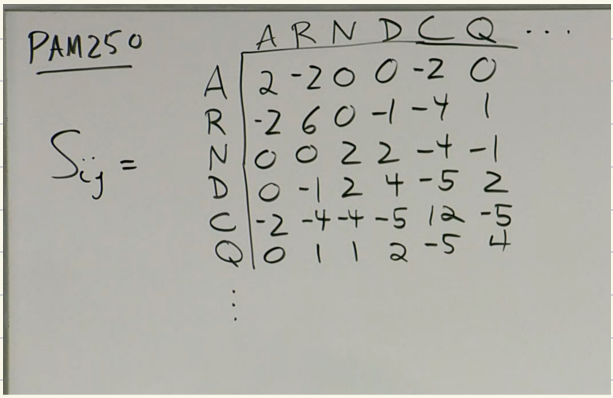
Why these rules?

1. Going to have to ignore a lot of bad, a lot of gaps, that we don't want to penalize while we try to get the match. Don't penalize the global mismatch.
2. Alignment must go down in score at some point. At some point the score must go down so we can actually find an optimum.
3. Start / End Alignment anywhere,



we don not know where these two very similiar subseuqnces may align. Don't want to penalize the bad alignments

## Whence Match Matrix



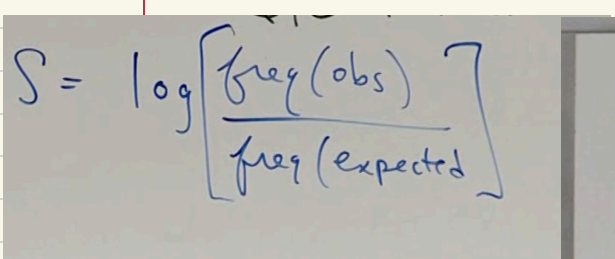

High values along the diagonal, exact matrches

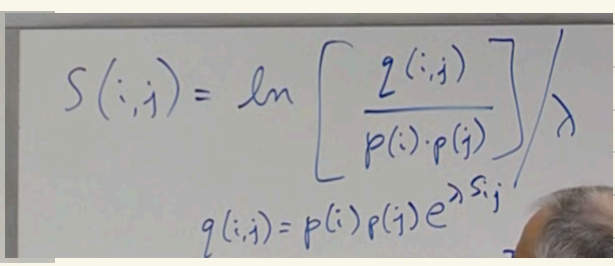
C loves itself and hates matching with pretty much anything else

D-Q is just as good as an extact A match.

D-A is evolutionarily nuertral. Agnostic biologically

These scores in general are equal to the log of the frequency of observing this correspondence in a big data base divided by the expected frequency.




```
AA_x --> 'Amino Acid x'
```

the values are determined by evolutionarily how often these two values are selected out.

p(i) prob oof AA_i in the database of all goldstandard aligments

q(i,j) is the frequnecy of observation AA_i + AA_j in a corrrespondance in a gold standard alignment

p(i) * p(j) is the expected frequency

```
lambda is 'fudge factor' to make
everything look nice
```

d & e

- we essentially look the database of all gold standard alignments and find value of d and e that best match. Search / Gradient descent around until you get a robust d and e
- d and e have to be on the same order of magnitude of the things in the matrix S_ij. Or else, we would never have match.

# Lecture 4

## Agenda

Similarity

BLOSUM62

BLAST overview

BLAST details

Intro : Transcriptomics

## Similarity

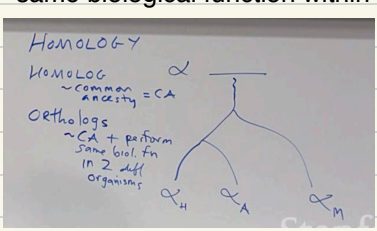
Pairwise sequence alignment begets a score

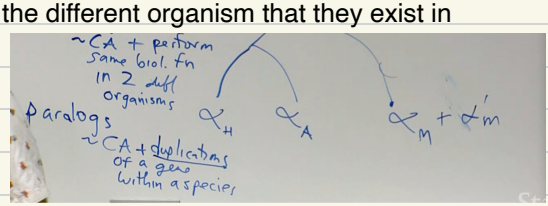
Good evidence of similiary implies good evidence of common ancestry

**Homologous** - sequences that are similar, sure, but more important they have common ancestry

Some DNA strand alpha, goes through different mutations throughout evolutionary history. The protiens with common ancestors are called homologous

**Orthologs** - subtrype of homologs. Have common ancestory in addition to the same biological function within the different organism that they exist in




heavy survival bias in paralogs

If alpha added a glucose to a phosphate and the alpha_h,a,m are all orthologs, that all of these different alphas will indeed add a glucose to a phosphate

Paralogs - common ancestor + they are duplications of a gene within a species. i.e. alpha_m and alpha_m'. alpha_m still does it job, so alpha_m' can go off and keep being mutated. It is free to evolve unconstrained. in favor to organism.

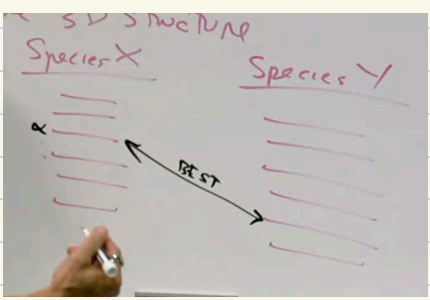
These algorithms do not measure homology, they measure similiarity, which is evidence of homology

Similarity = Score
% identical of amino acids in alignment [M : M] —> will have highest scores
% similarity of AAs [D : E] —> may have higher precentage but less score per

Protein %ID > ~25% —> strong evidence of homology. Structures will almost always look exactly the same

Protien ID% /in [0 - 25]% —> no gaurentee of homology. could look completely different. [15 - 25]% is a twilight zone where this is just evidence but does not really imply anything
there exists a 9% percent similar example where they look exaclty the same

Finding orthologs is quite difficult. Hugely important neverthless. If you study something in mice, and you want to imply something in humans, you must know the orthologs in order to say with any confidence that whatever you are study in model organisms will also be present in human.




***Mutual Best Hit Orthology Algorithm :***
Essentially, if you find the most likely ortholog in each, and then they match, you can say with some confidence that we have a ortholog

algorithm assumes that the paralogs are not more similar to the aa than the true homolog

# BLOSUM62

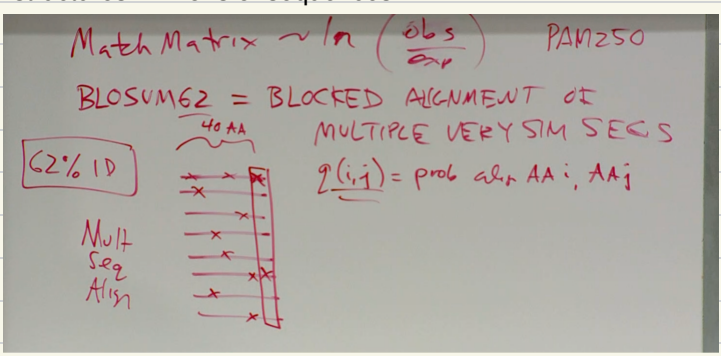
Matrch Matrix ~ ln(obs / exp) —> this is how PAM250 was made

BLO —> blocked alignment of multiple very similary sequences

Fragments of sequence, maybe forty amino acids. All of these protiens are ~62 percent identical. Have super high confidence that the structure of the sequences will look the same

Look at columns, assume that they all have the same strucuture, do multi-sequeence align. q(i,j) = prob align AA_i, AA_j.

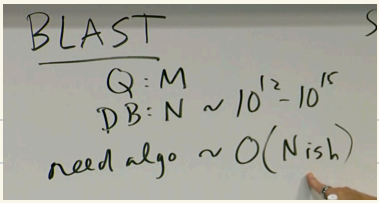
Advantage is that you have way more protein sequences than you have structures. Millions of sequences




Too low --> worried that they might not be the same structure
Too high --> you wouldn't get enough that did not match such that you could properly compute the probabilities

# BLAST

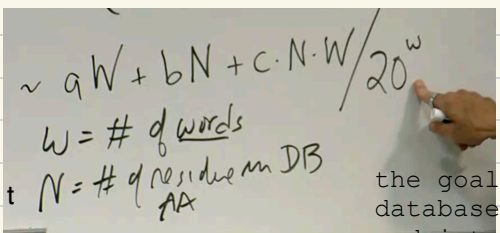
SW / NM --> O(N*M)




Large sequences are going to be a huge problem for SW / NM algorithms

**BLAST** - Basic Local Alignment of Sequence Tool

Computation complexity :




the goal of blast is to query a database of everything you seen before and introduce a new M and see what the best matches are

Most commonly used algorithm

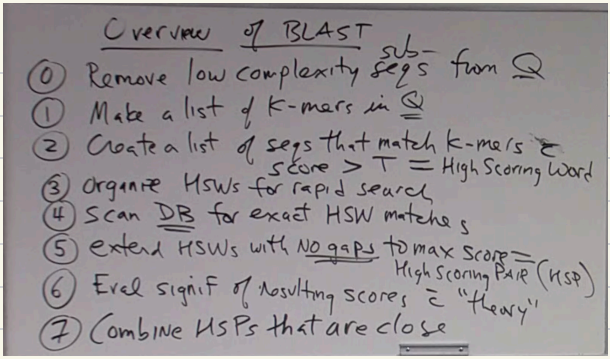
Tries to approx smith waterman, approximate local alignment

**3 Core Ideas of BLAST**

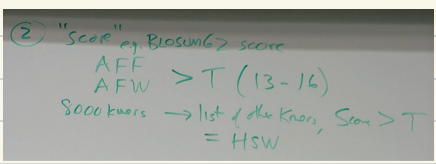
1. Find small areas of exact match in the query and in the database. BLAST assume that there is some bit of sequence that is still shared
2. Important to have method to evaluate statistical signficance of a score
3. Extend these small areas to the left and right greedily

BLAST Pseudocode

1. Remove low complexity subsequences from Q
2. Make a list of k-mers in Q
3. Create a list of sequences that match k-mers <= score > T = High scoring word
4. Organizes HSWw for rapid search
5. Scan DB for Exact HSW matches
6. Extend HSWs with no gaps to max score = high scoring pair
7. Evaluate significant cant




0) Exists not intersteing sequences [KKK....]_n
Triplet repeasts [KLM]_n --> we simply mask all of this stuff out of the DB

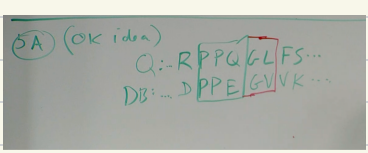



1) K-mer : string of length k
protein k = 3 or 4 i.e. AFF
k = 3 --> 8000 k-mers, k = 4 --> 16k kmers
at postiions 1, 2, and 3, you have 20 options (20 different amino acids)

2) "Score" could be Blossom 62 for example. i.e. [AFF:AFW] > T \in (score bound)
High scoring word if it gets larger than t for a given kmers, get a list of these kmers

3) Organize in some tree structure (i.e. suffix trees)

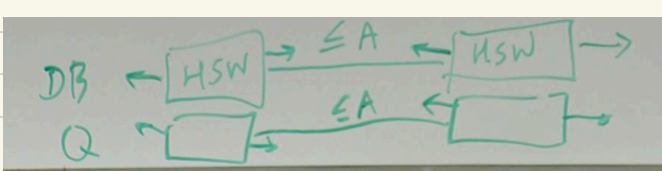
4) Use the tree that you just built to scan for HSWs, scanning database. These can be seeds for the eventual base alignment. We will call the hits 'high scoring pairs'



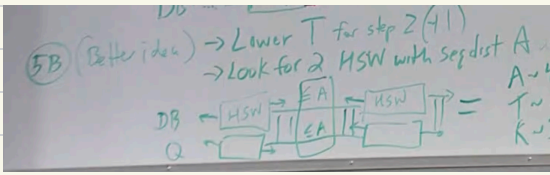

5a) extend high scoring pairs until score decreases

5b) (better idea) pick a lower T threshold for step 2. Next, look for 2 HSW( high scoring words) withing a sequence distance *A* from each other

Doing this on the database side

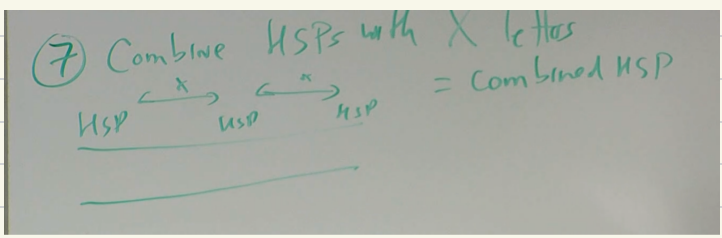


Assumption is that : there must be two high scoring alignments within A : int of each other in a maximum value alignment




6) Repeat all HSPs > that have score = significant

7) combine HSPs within X letters --> combined HSP




8) Take the query sequence and do the smtigh waterman (going to be an approx)