Excercise

- For query RRR, threshold 11, what are the other 3-mers that can generate hits?

RRR: 15

RRQ: 11

RQR: 11

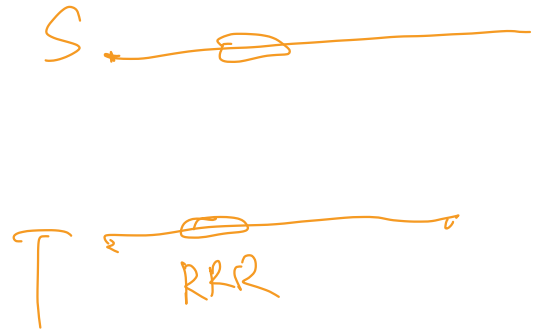
QRR: 11

RRK: 12

RKR: 12

KRR: 12

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4
B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3



How to implement that?

- With BLASTP:
  - Build an automaton that reflects all string close to short strings in T (the short sequence)
  - Scan S (the longer sequence), looking for matches.
- We do not study the classic ways to match multiple patterns efficiently. If interested, you can read at https://en.wikipedia.org/wiki/Aho%E2%80%93Corasick_algorithm

A Simpler Way

- There is another way:

1) For every 3-mer, find all "neighboring" 3-mers that, score at least +11 (or whatever). Build these into a data structure NeighborList.
2) Build a hash table H for S of its 3-mers, just like for the nucleotide case
3) For every 3-mer x in T, retrieve all neighbors from NeighborList. For each neighbor, query H to find hits in S.

NeighborList is a small structure: there are only 8000 3-mers.

Which sequence to index?

- That's actually a tough question.

- Here's a typical scenario:
- S is the human genome (length n)
- P_1 is a short protein sequence (length m_1)
- P_2 is another short sequence (length m_2)

- If we're smart, build an index for S, *once*, and then look up the short sequences in it.
- Added time for P_2 is more like $O(m_2)$, not $O(n+m_2)$.

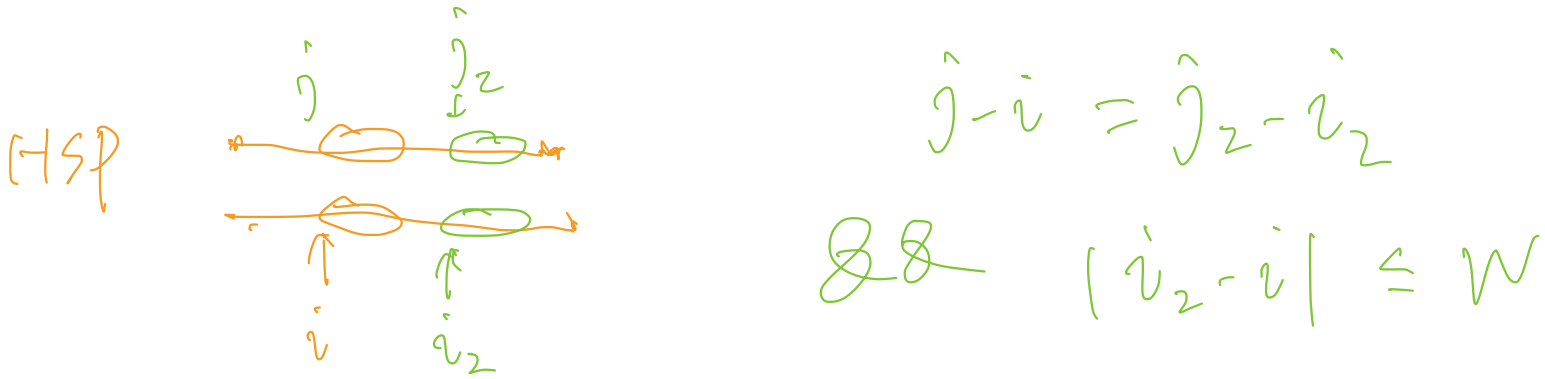
More on indexing

- But memory is a concern:
- Indexing the human genome is expensive!
- Oh, wait. No, it isn't, not anymore… you probably should index the longer sequence.
- BLASTN (1990) indexes the query, not the database.
- BLAT (2000) indexes the database, not the query.

- BLASTP also indexes the query.

Extensions to this idea

- Two-hit BLAST:
- Require two seeds (probably shorter) that are nearer than k from each other, and base the alignment on their enclosing box.
- Potentially even fewer false positives, but one has to use shorter seeds. There's quite a tradeoff here.



Wrap-up

- Local alignment slow when sequences are large
- Use 11 consecutive matches as hits
  - How these hits are found efficiently
  - What to do after hits are found
- Spaced seeds better
  - How sensitivity is computed and how optimal seed is found
  - How hits are found for spaced seed
- Multiple spaced seed.
- Protein seeds.
- Two hits.