Sequence Database Search

The (Sequence) Database Search Problem

Given a database *D* of sequences (DNA, Protein, Books, Web Pages) and a query string *Q* find the sting(s) *S* in *D* which is/are closest matches to *Q* under a defined scoring function.

The (Sequence) Database Search Problem

Given a database *D* of sequences (DNA, Protein, Books, Web Pages) and a query string *Q* find the sting(s) *S* in *D* which is/are closest matches to *Q* under a defined scoring function.

Scoring functions are typically either

- **Semi-global alignment** -- The best possible alignment score between a substring *A* of *S* and *Q*, or
- **Local alignment** -- The vest possible alignment score between a substring *A* of *S* and a substring *B* of *Q*.

Evaluating Database Search

Sensitivity -- Ratio of true positives (substrings in the database matching the query string) found by the algorithm to the true number of positives.

Efficiency -- Running time of the method.

Types of Algorithms

Exhaustive Search -- Enumerate all possible solutions to find the best one. *very sensitive, very slow*

Heuristic Search -- Reduce the search space by estimating alignments but sometimes overlooks solutions. *less sensitive, fast*

Filter Based -- Select candidate positions in the database where the query is likely to match. *medium sensitivity, moderately fast*

Smith-Waterman's Revenge

For each sequence *S* in *D*, run Smith-Waterman between *S* and *Q*

Return the sequence(s) with the largest alignment score.

Running time is *O(mn)* per sequence, this is very slow, but very accurate.

FastP and FastA

The first attempts at speeding up search.

Both are based on the idea that (in protein sequences) replacements are more common than indels.

Developed in 1983 and 1988 respectively, FastP does not allow for gaps at all while FastA will find gapped alignments, but only in certain circumstances.

FastP

Step 1: Identify "hotspots" -- find *k*-mers that are shared between the query and the database using a lookup table (this table is 4^k for DNA and RNA, 20^k for Proteins)

Query	CAACTTGCC
Database	ACGGTTACGTAGGTCCG
	GCGTAGGCAGAAGTTGCCTGCGT
	ACGAAGTAGCCGTCAGTC
	TAGTCCGTATGAAGTCGTAGTC

FastP

Step 2: locating diagonal runs -- pairs (or larger groups) of hot spots such that the distance between the hot-spots is the same in both the query and the database sequence

```
Query              CAACTTGCC

Database     ACGGTTACGTAGGTCCG

             GCGTAGGCAGAAGTTGCCTGCGT

             ACGAAGTAGCCGTCAGTC

             TAGTCCGTATGAAGTCGTAGTC
```

FastP

Step 2: locating diagonal runs -- pairs (or larger groups) of hot spots such that the distance between the hot-spots is the same in both the query and the database sequence

```
Query               CAACTTGCC

Database   ACGGTTACGTAGGTCCG

           GCGTAGGCAGAAGTTGCCTGCGT

           ACGAAGTAGCCGTCAGTC

           TAGTCCGTATGAAGTCGTAGTC
```

FastP

Step 2: locating diagonal runs -- pairs (or larger groups) of hot spots such that the distance between the hot-spots is the same in both the query and the database sequence

Query CAACTTGCC

Database ACGGTTACGTAGGTCCG

GCGTAGGCAGAAGTTGCCTGCGT

ACGAAGTAGCCGTCAGTC

TAGTCCGTATGAAGTCGTAGTC

FastP

Step 2: locating diagonal runs -- pairs (or larger groups) of hot spots such that the distance between the hot-spots is the same in both the query and the database sequence

Query CAACTTGCC

Database ACGGTTACGTAGGTCCG

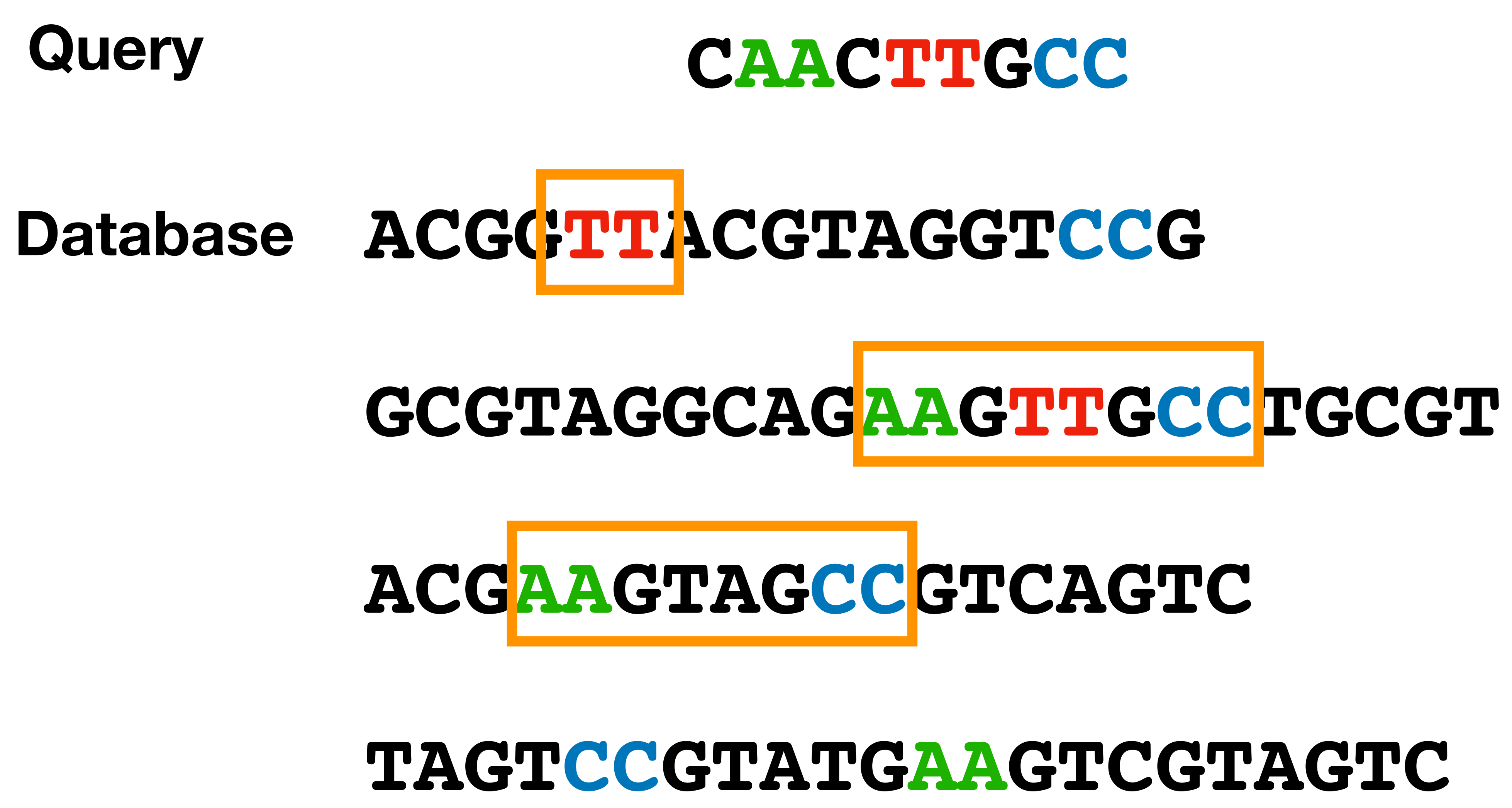
GCGTAGGCAGAAGTTGCCTGCGT

ACGAAGTAGCCGTCAGTC

TAGTCCGTATGAAGTCGTAGTC

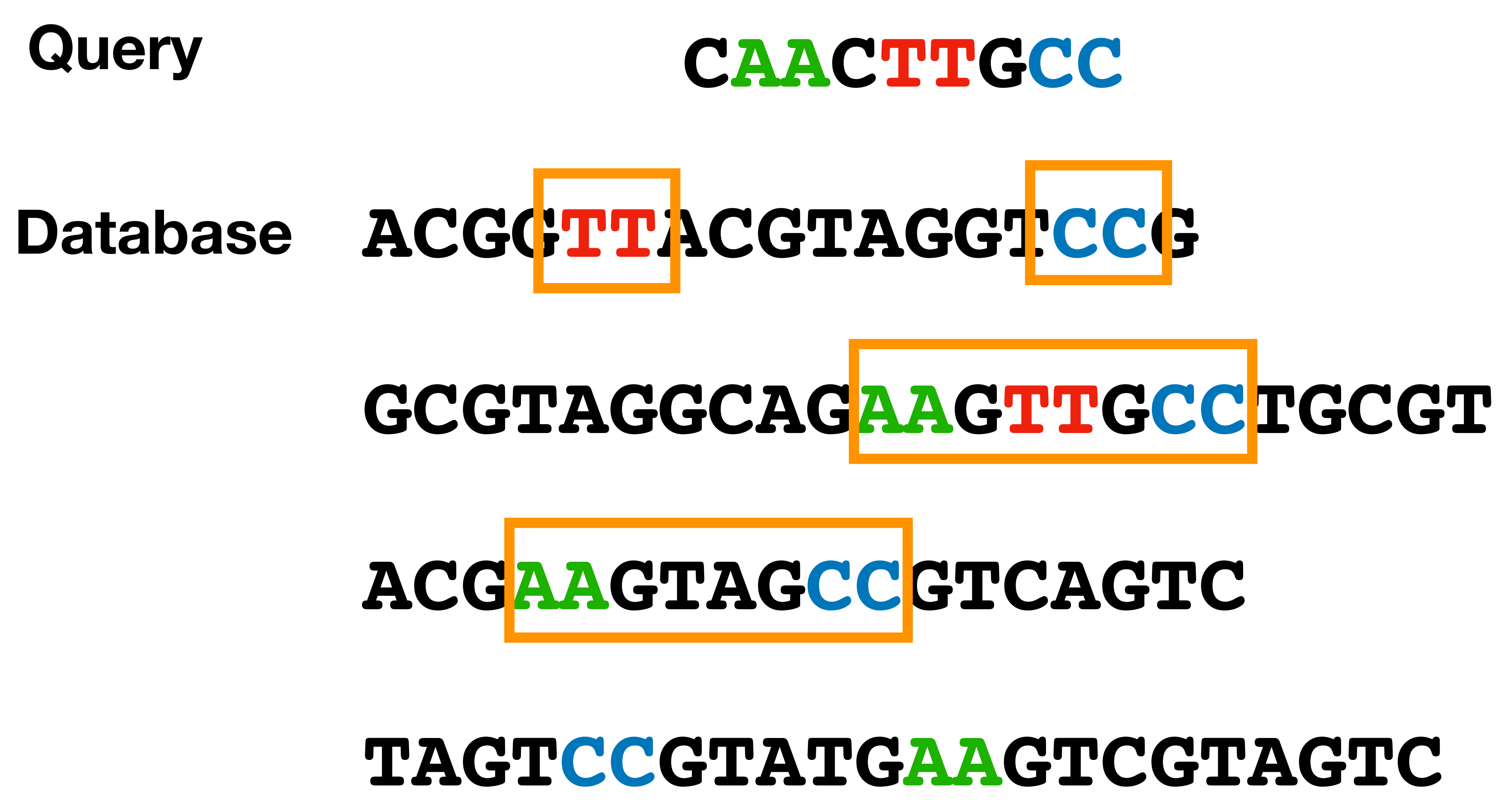
FastP

Step 2: locating diagonal runs -- pairs (or larger groups) of hot spots such that the distance between the hot-spots is the same in both the query and the database sequence



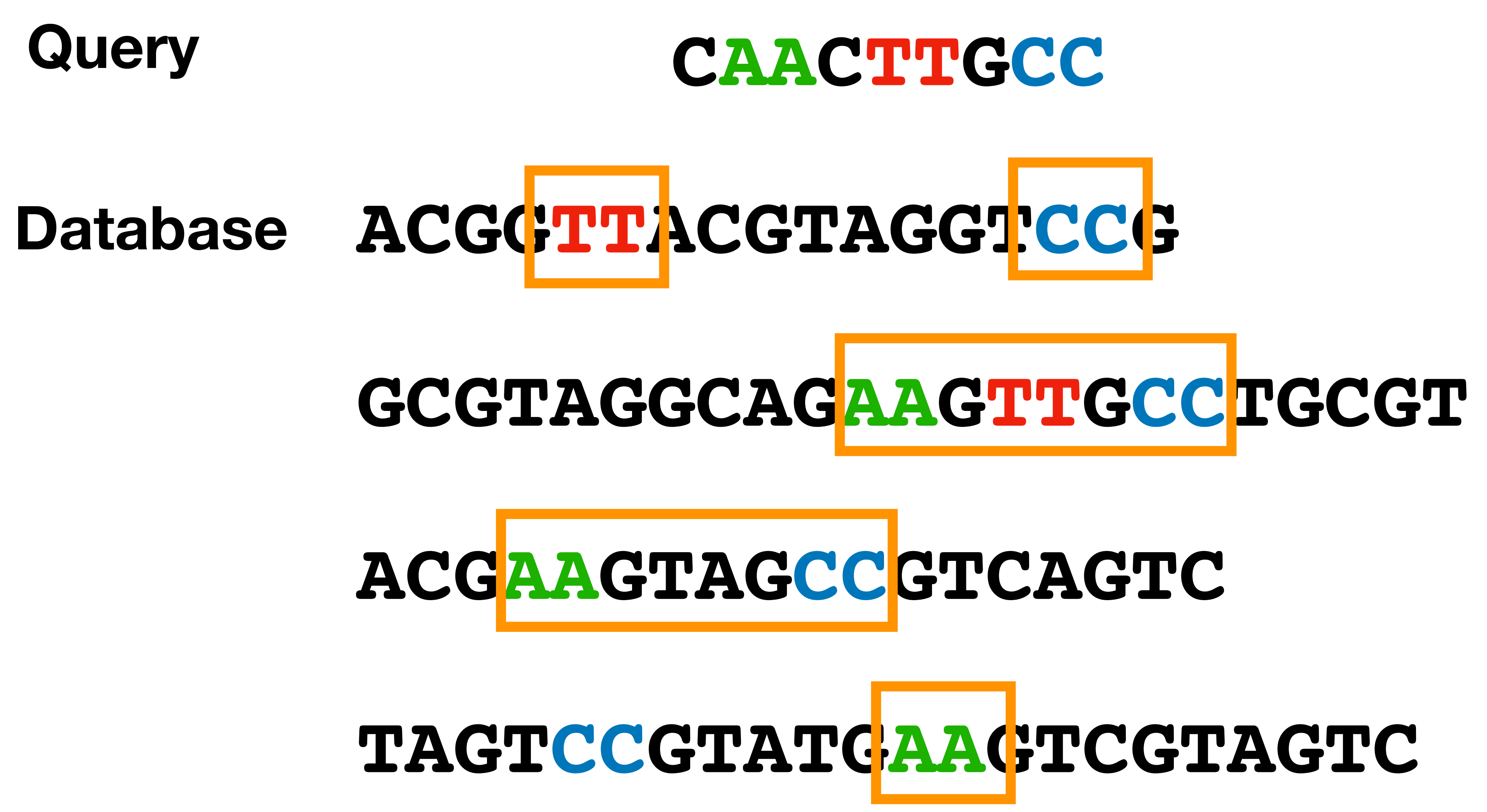
FastP

Step 2: locating diagonal runs -- pairs (or larger groups) of hot spots such that the distance between the hot-spots is the same in both the query and the database sequence



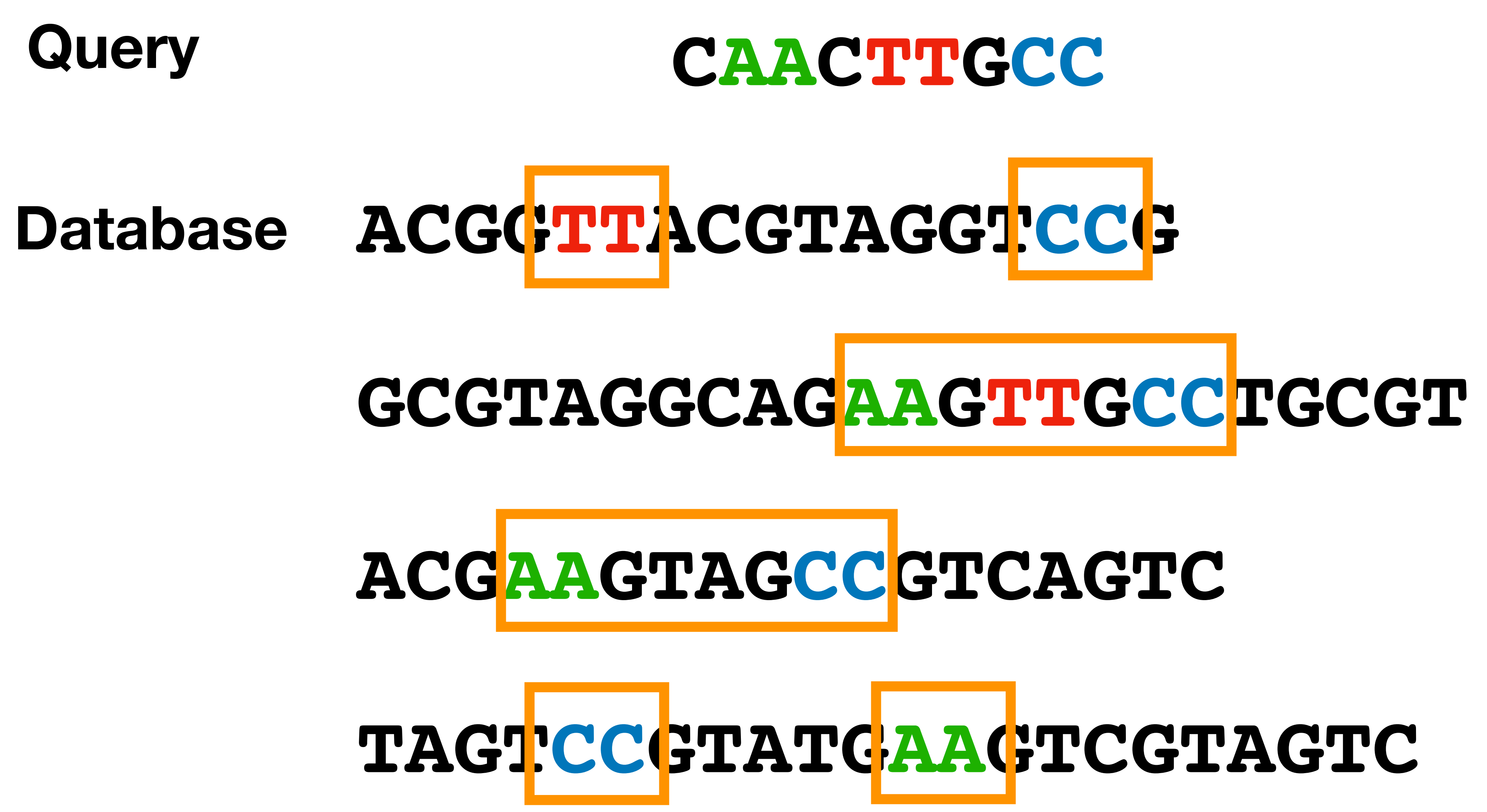
FastP

Step 2: locating diagonal runs -- pairs (or larger groups) of hot spots such that the distance between the hot-spots is the same in both the query and the database sequence



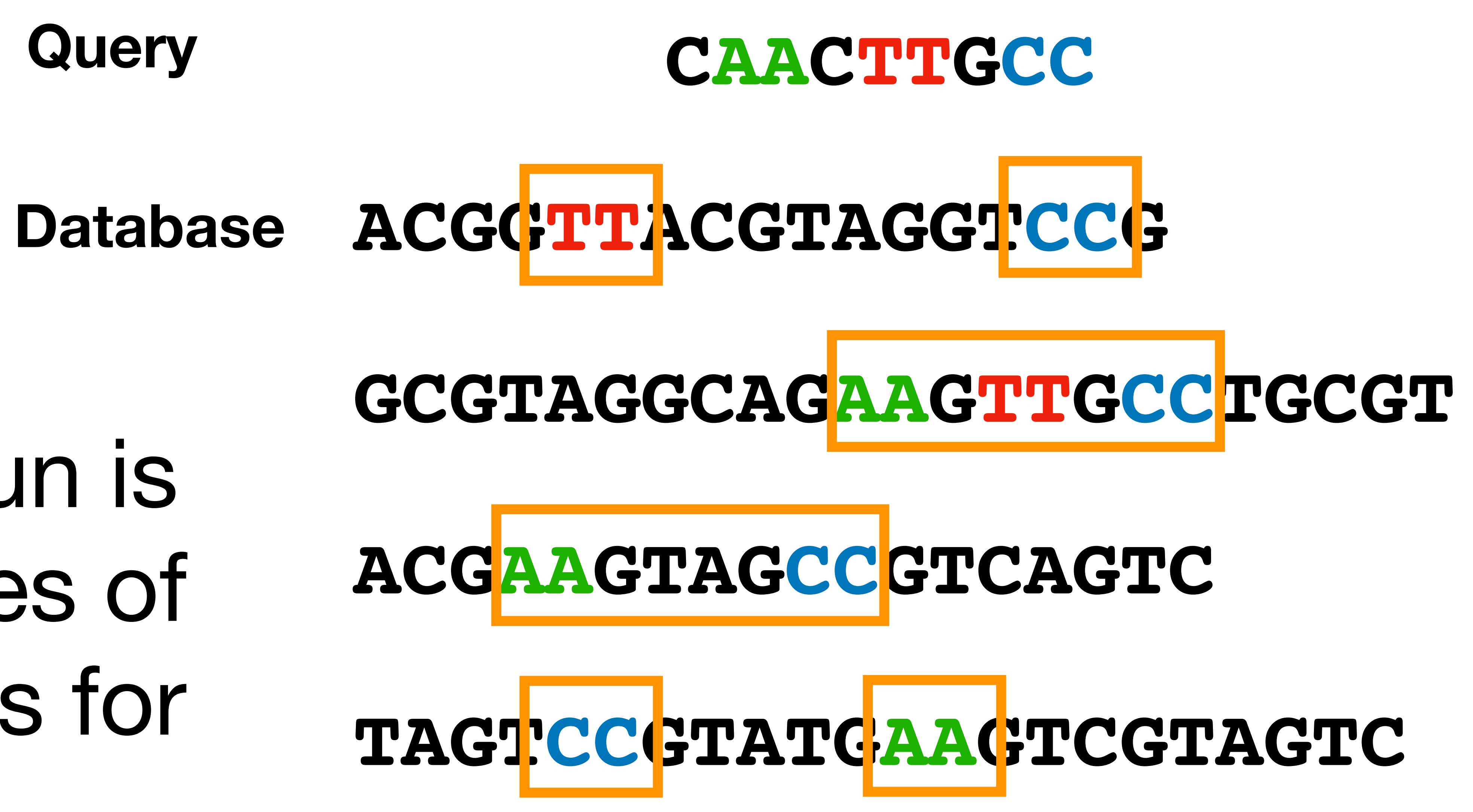
FastP

Step 2: locating diagonal runs -- pairs (or larger groups) of hot spots such that the distance between the hot-spots is the same in both the query and the database sequence



FastP

Step 2: locating diagonal runs -- pairs (or larger groups) of hot spots such that the distance between the hot-spots is the same in both the query and the database sequence



The score of a diagonal run is the sum of the base-scores of the hotspots and penalties for inter-spot characters

FastP

Step 3: re-score the best diagonal runs -- rather than fixed inter-spot scores based on length, rescore the alignments using actual character matches

Query CAACTTGCC

Database ACGGTTACGTAGGTCCG

GCGTAGGCAGAAGTTGCCTGCGT

ACGAAGTAGCCGTCAGTC

TAGTCCGTATGAAGTCGTAGTC

FastP

Step 3: re-score the best diagonal runs -- rather than fixed inter-spot scores based on length, rescore the alignments using actual character matches

Query CAACTTGCC

Database ACGGTTACGTAGGTCCG

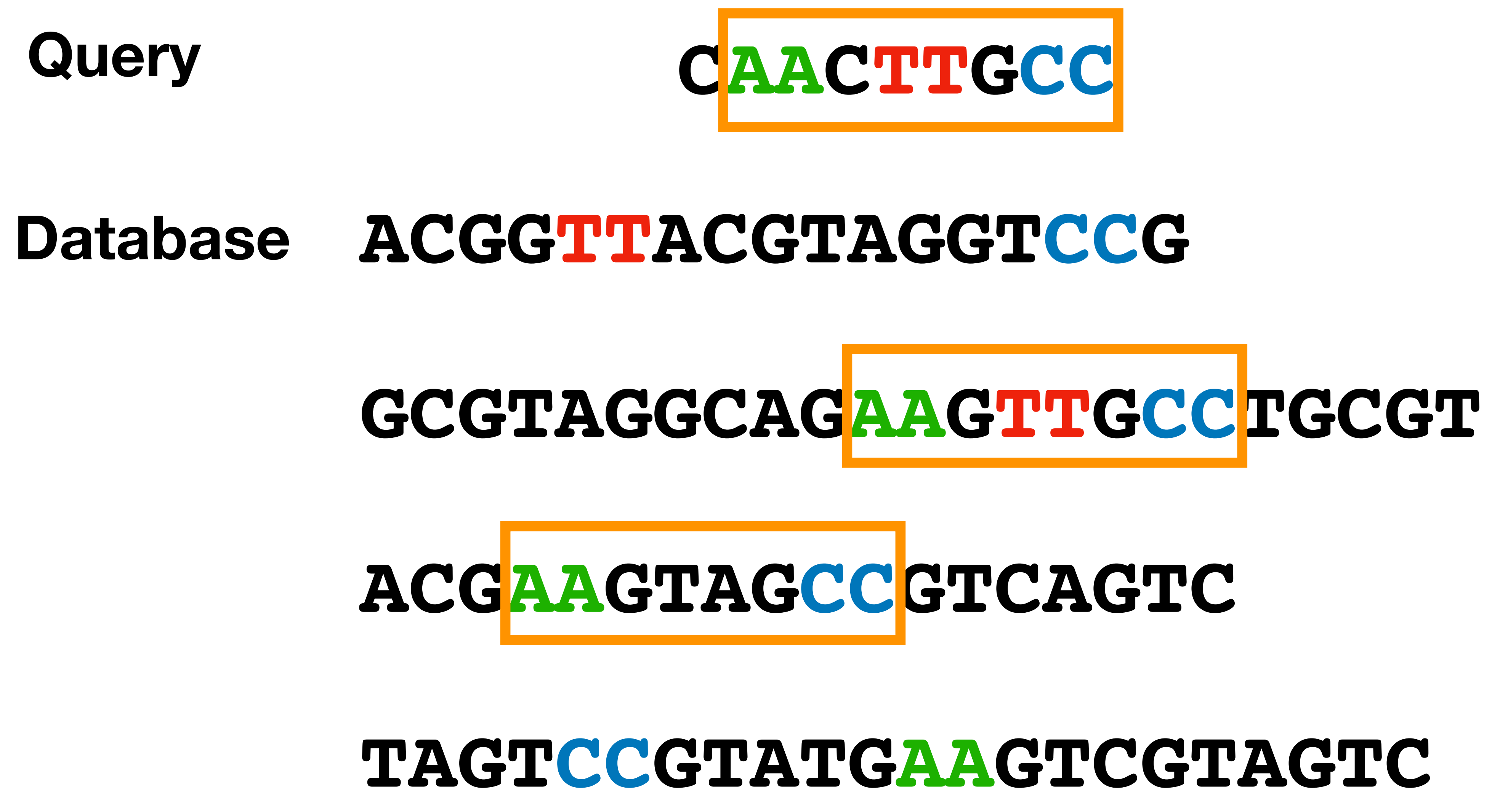
GCGTAGGCAGAAGTTGCCTGCGT

ACGAAGTAGCCGTCAGTC

TAGTCCGTATGAAGTCGTAGTC

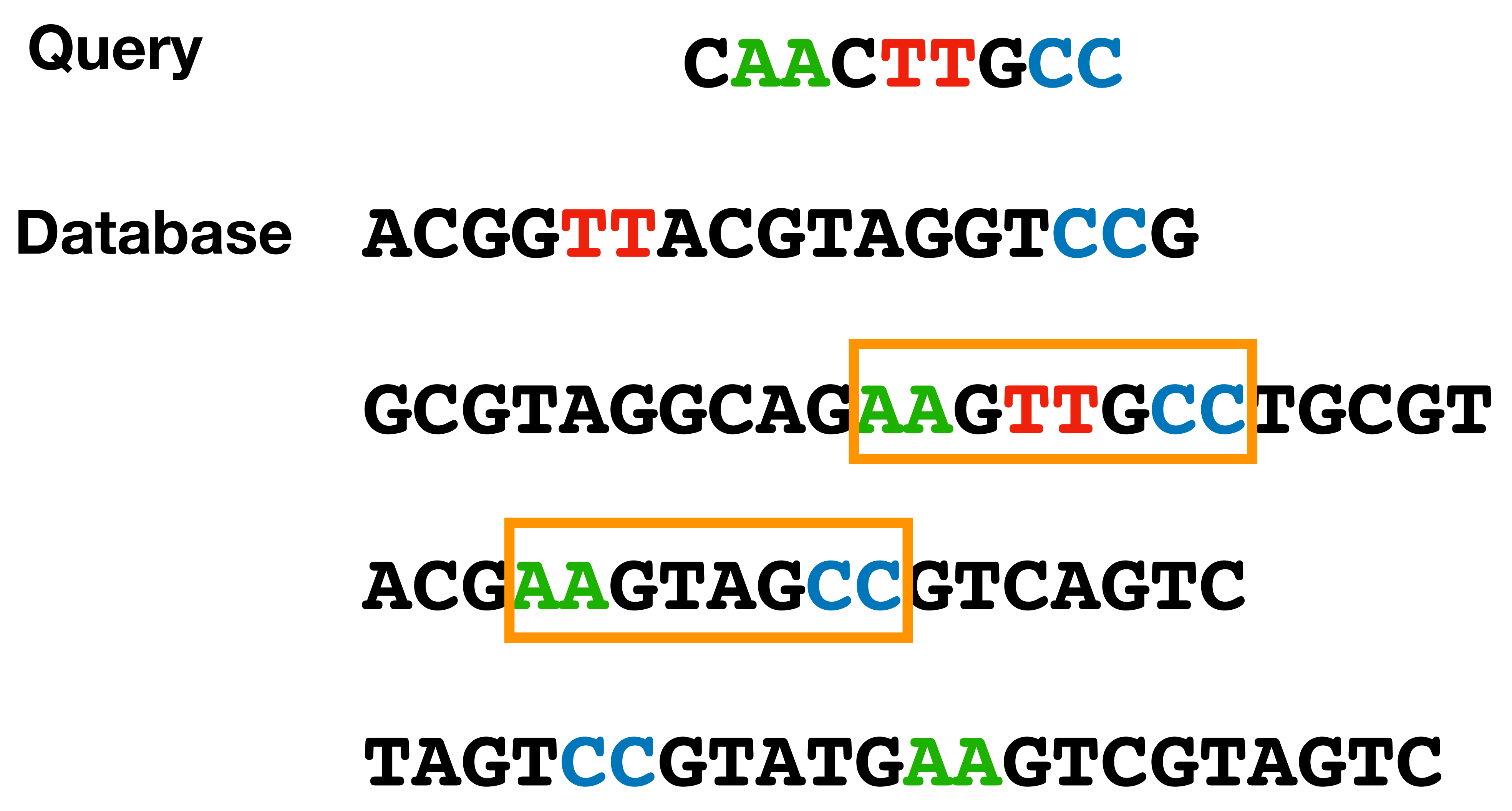
FastP

Step 3: re-score the best diagonal runs -- rather than fixed inter-spot scores based on length, rescore the alignments using actual character matches



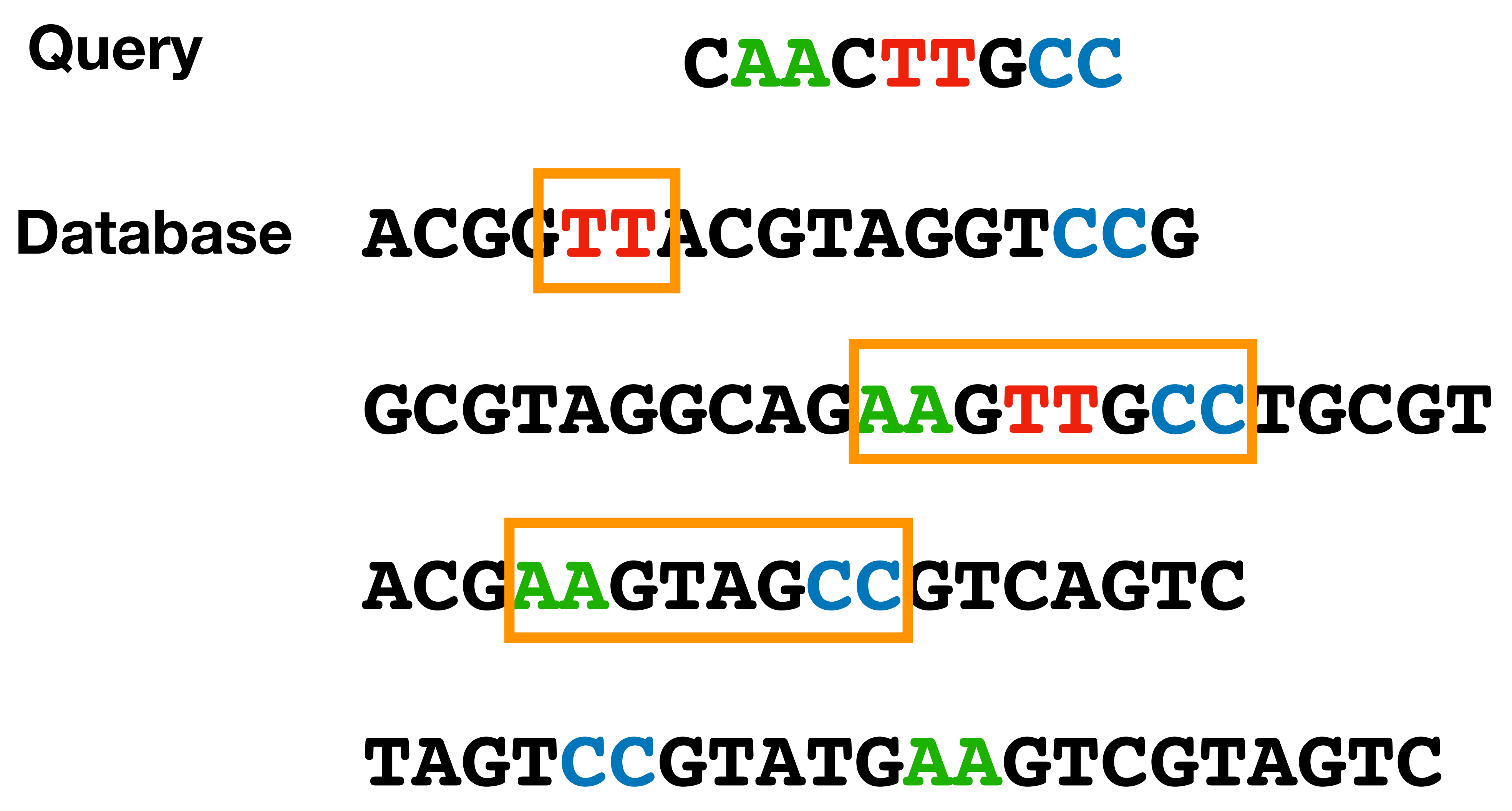
FastP

Step 3: re-score the best diagonal runs -- rather than fixed inter-spot scores based on length, rescore the alignments using actual character matches



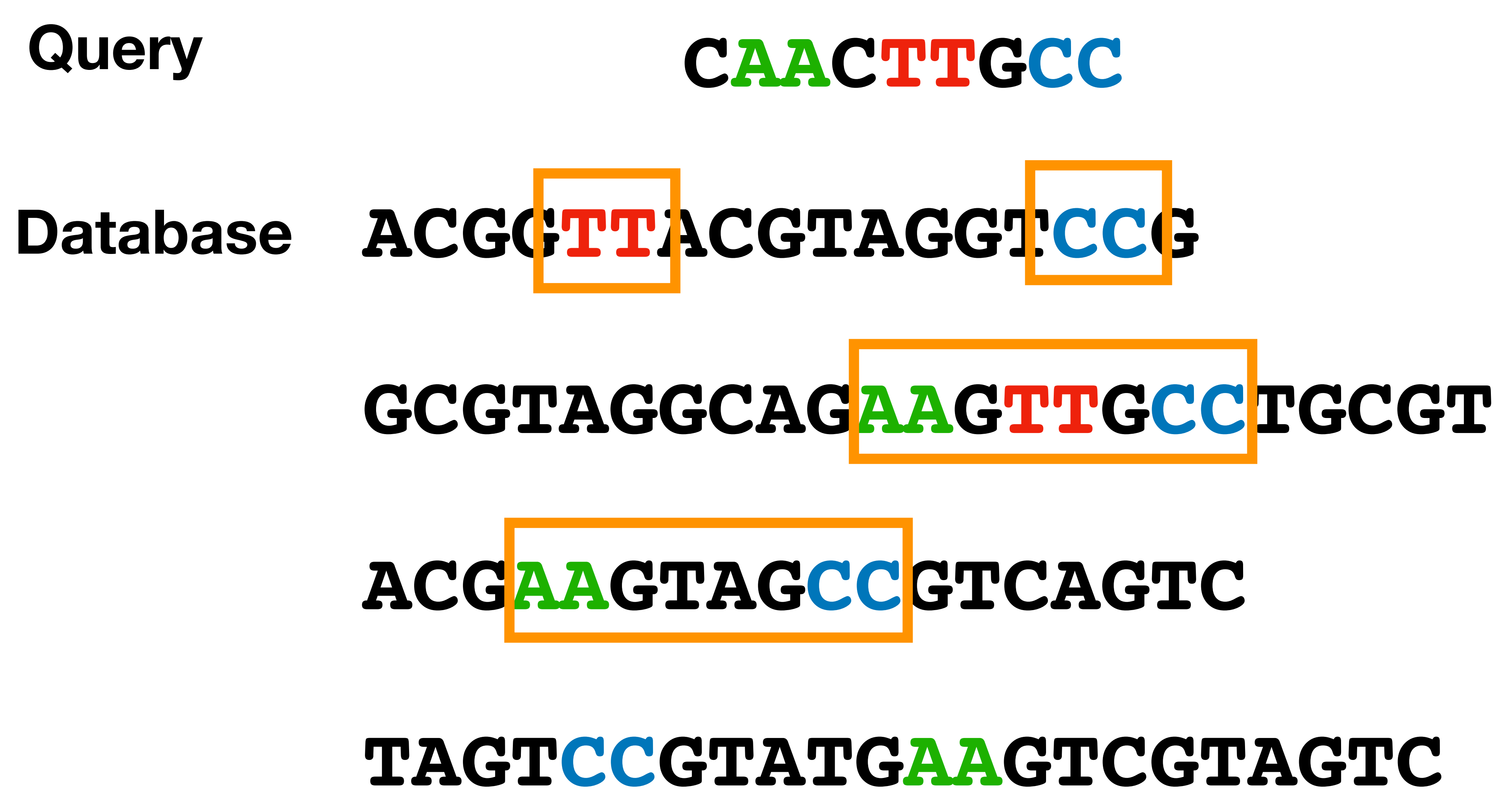
FastP

Step 3: re-score the best diagonal runs -- rather than fixed inter-spot scores based on length, rescore the alignments using actual character matches



FastP

Step 3: re-score the best diagonal runs -- rather than fixed inter-spot scores based on length, rescore the alignments using actual character matches



FastP

Step 3: re-score the best diagonal runs -- rather than fixed inter-spot scores based on length, rescore the alignments using actual character matches

Query CAACTTGCC

Database ACGGTTACGTAGGTCCG

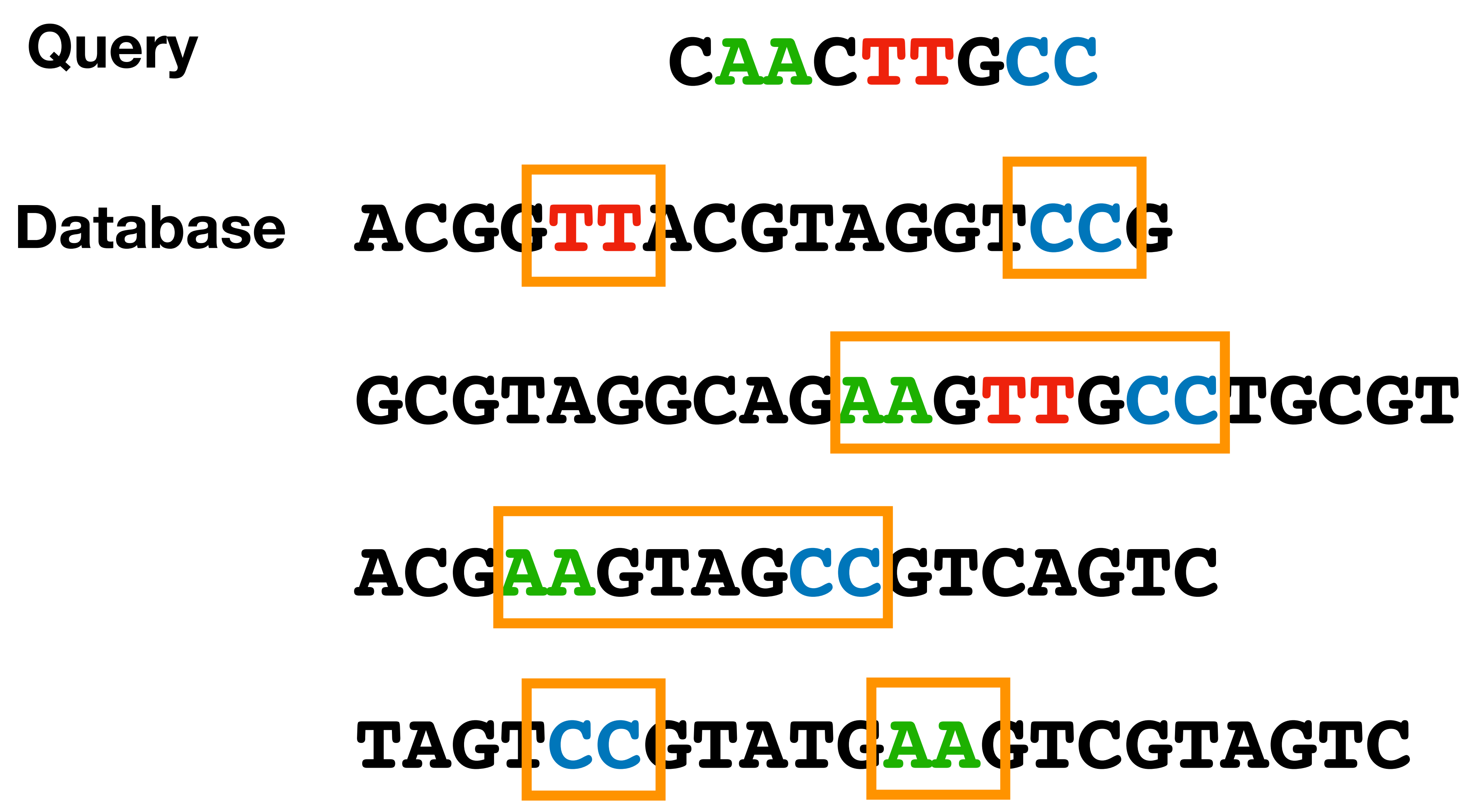
GCGTAGGCAGAAGTTGCCTGCGT

ACGAAGTAGCCGTCAGTC

TAGTCCGTATGAAGTCGTAGTC

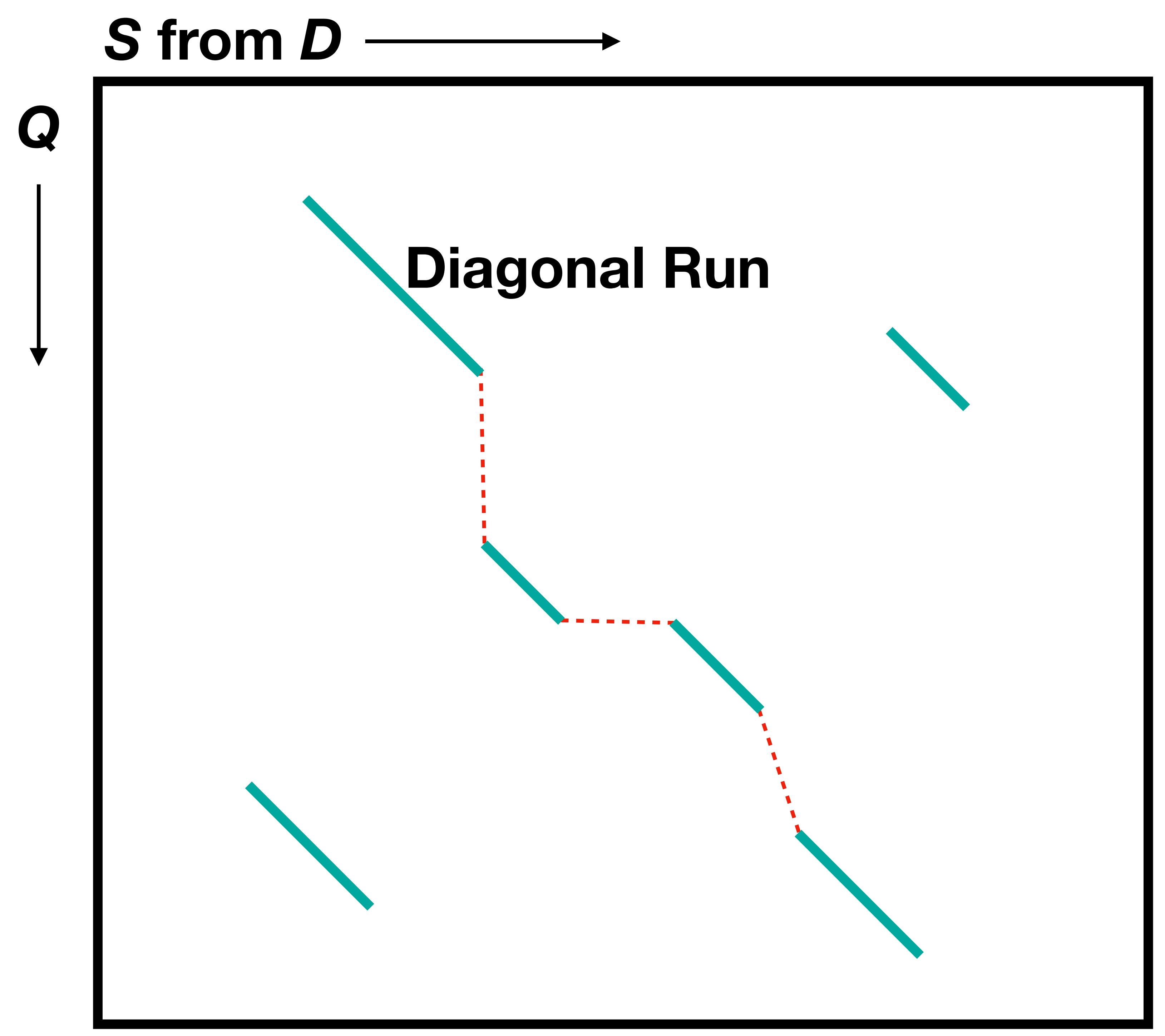
FastP

Step 3: re-score the best diagonal runs -- rather than fixed inter-spot scores based on length, rescore the alignments using actual character matches



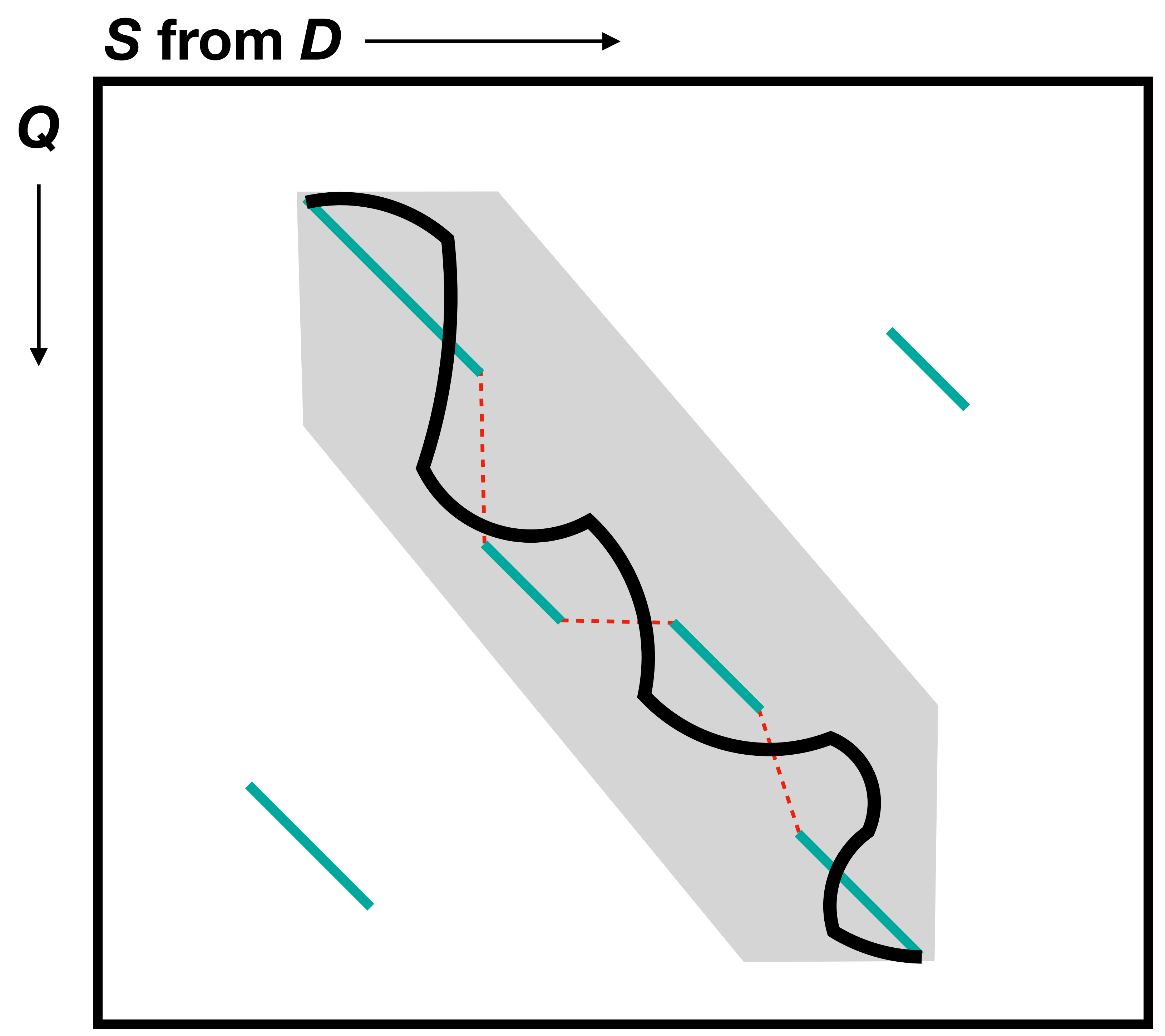
FastA (adding gaps)

Step 4: join diagonal runs -- using a fixed score based on the locations of the regions, join them with a fixed gap-style cost



FastA (adding gaps)

Step 5: (banded) Smith-Waterman -- using a fixed score based on the locations of the regions, join them with a fixed gap-style cost



Basic Local Alignment Search Tool (BLAST)

Most commonly used database search tool in computational biology.

Originally published in 1990 by Altschul, Gish, Myers, Miller and Lipman.

Faster than FastA.

Basic Local Alignment Search Tool (BLAST)

Step 1: Query-preprocessing:

1. split the query into *k*-mers
2. create a set of *neighbors* of each *k*-mer, other *k*-mers such that the replacement scores are not too high (this can be done with a Σ^k lookup table)

```
ACCTAGAT
ACC
 CCT
  CTA
   TAG
    AGA
     GAT
```

Basic Local Alignment Search Tool (BLAST)

Step 1: Query-preprocessing:

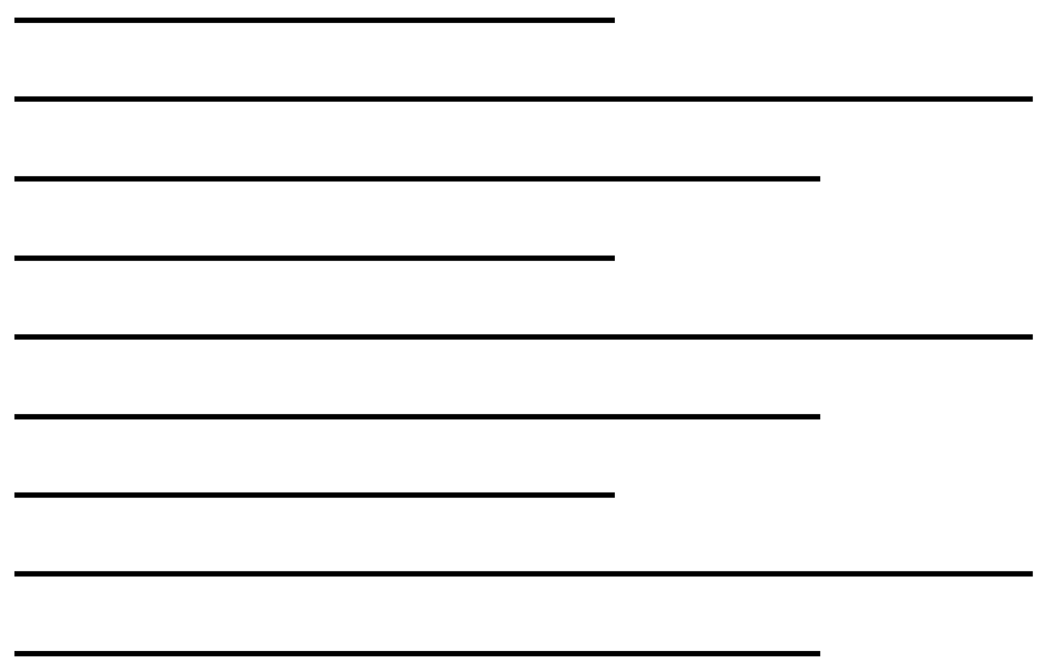
1. split the query into *k*-mers
2. create a set of *neighbors* of each *k*-mer, other *k*-mers such that the replacement scores are not too high (this can be done with a Σ^k lookup table)

```
ACCTAGAT
ACC ──────────→ {ACC,TCC,AGC,ACG}
 CCT
  CTA
   TAG
    AGA
     GAT
```

Basic Local Alignment Search Tool (BLAST)

Step 2: Database scanning -- label any instance of a neighbor of *Q* in any sequence *S* of *D* as a "hit", collect all of these hits

Database

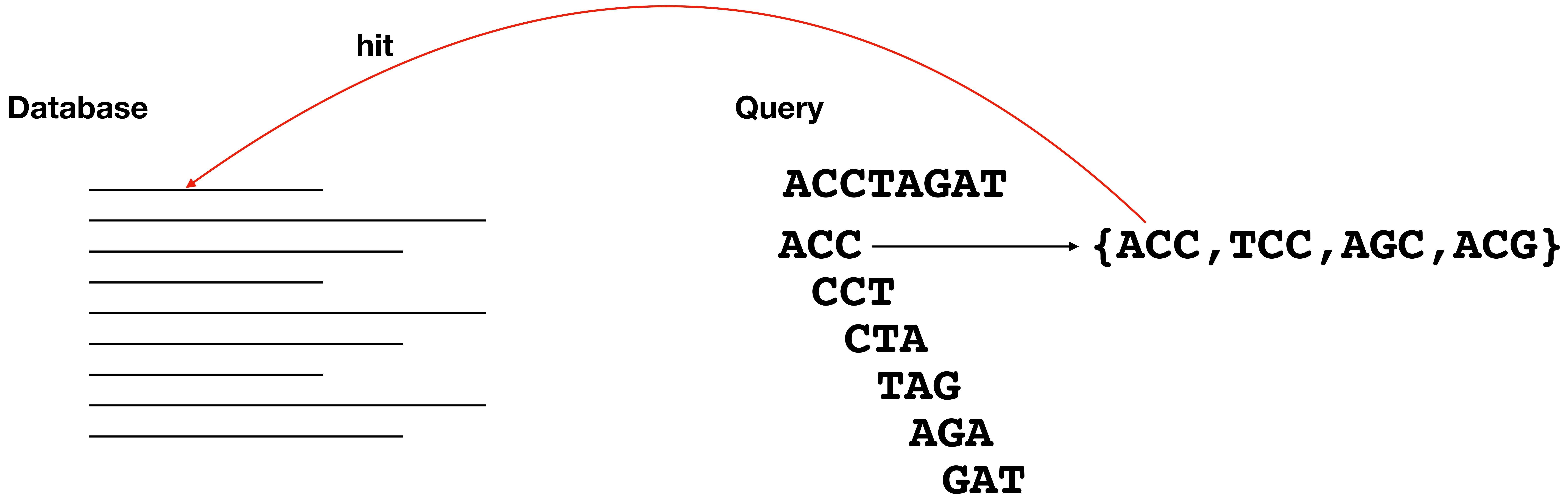


Query

```
ACCTAGAT
ACC ——————→ {ACC,TCC,AGC,ACG}
 CCT
  CTA
   TAG
    AGA
     GAT
```

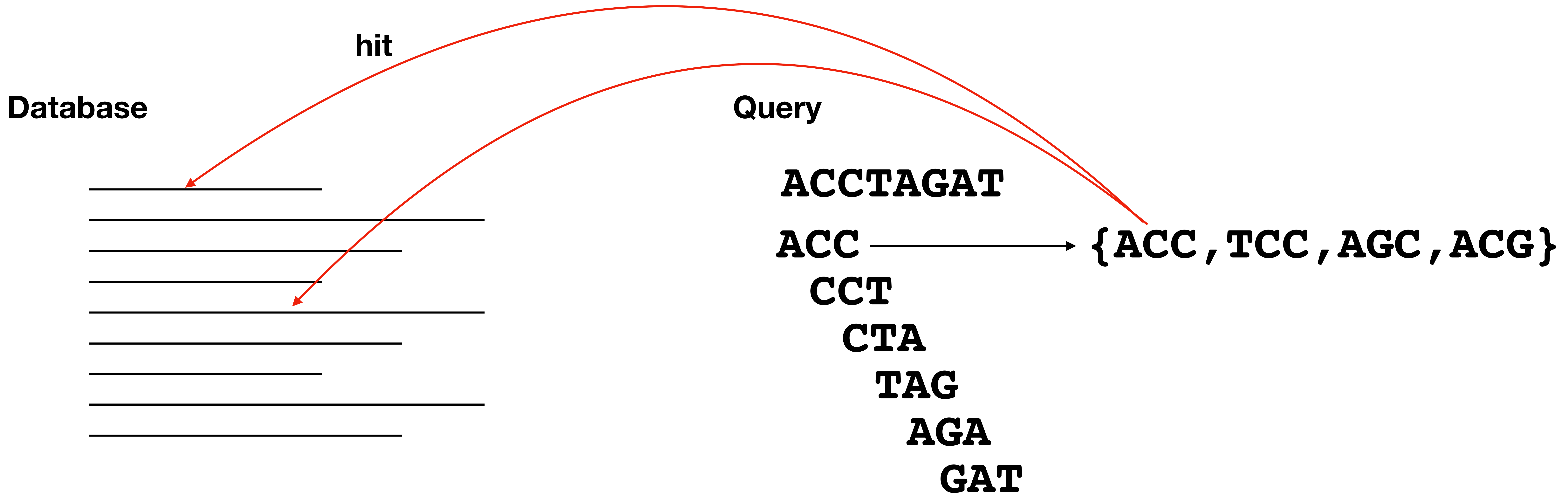
Basic Local Alignment Search Tool (BLAST)

Step 2: Database scanning -- label any instance of a neighbor of *Q* in any sequence *S* of *D* as a "hit", collect all of these hits



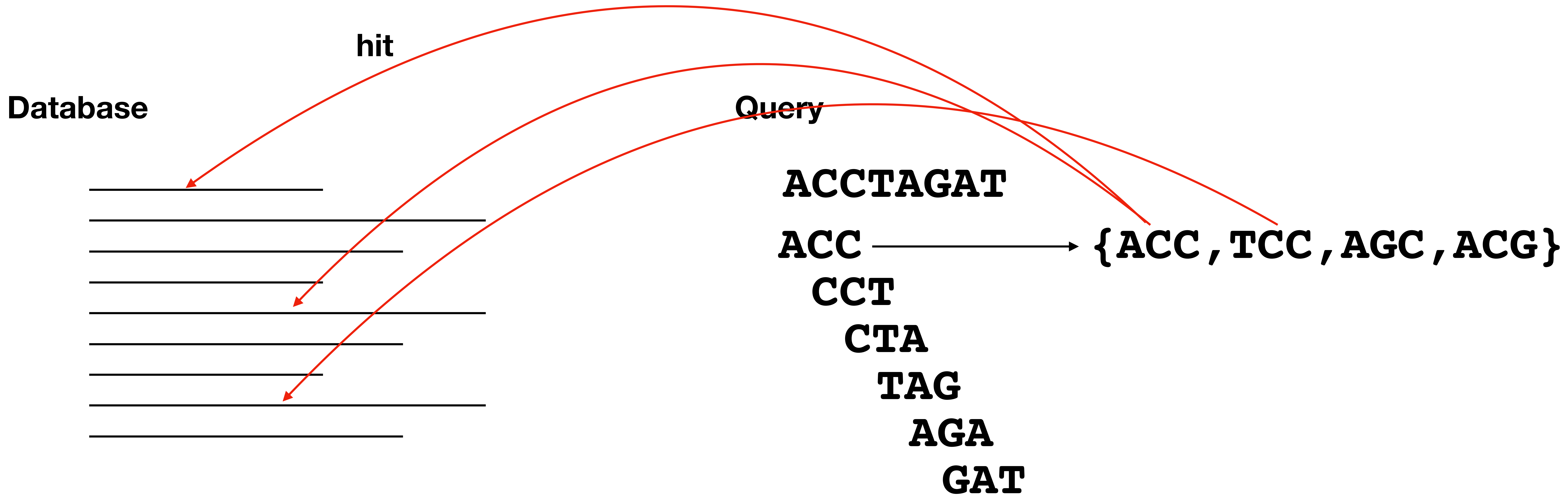
Basic Local Alignment Search Tool (BLAST)

Step 2: Database scanning -- label any instance of a neighbor of *Q* in any sequence *S* of *D* as a "hit", collect all of these hits



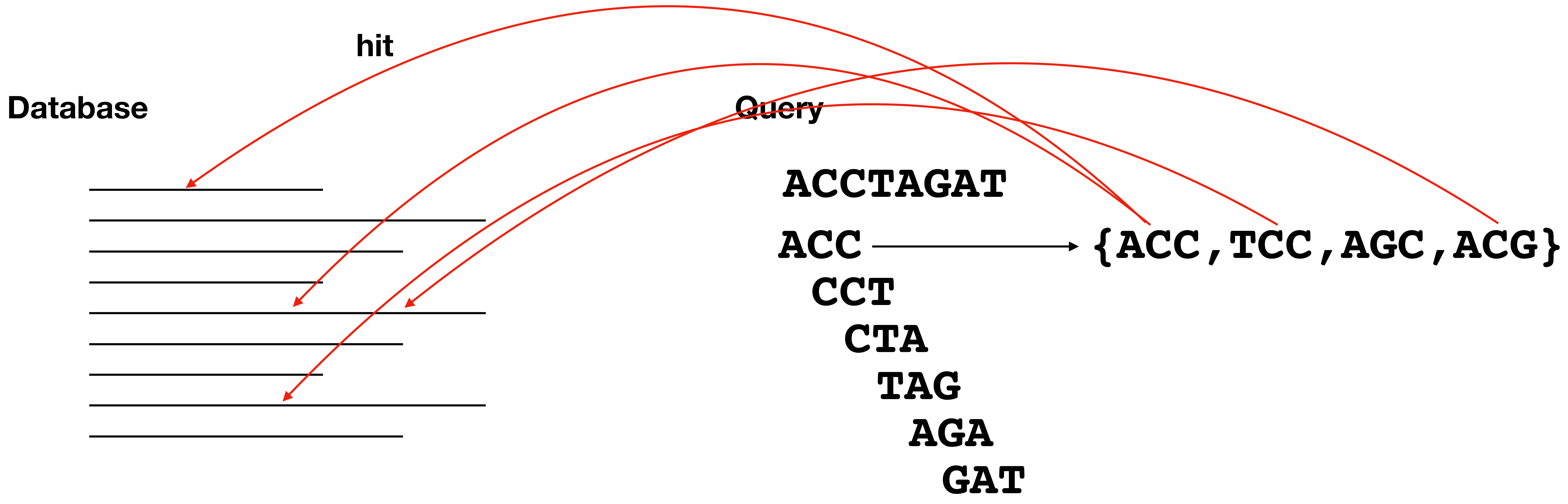
Basic Local Alignment Search Tool (BLAST)

Step 2: Database scanning -- label any instance of a neighbor of *Q* in any sequence *S* of *D* as a "hit", collect all of these hits



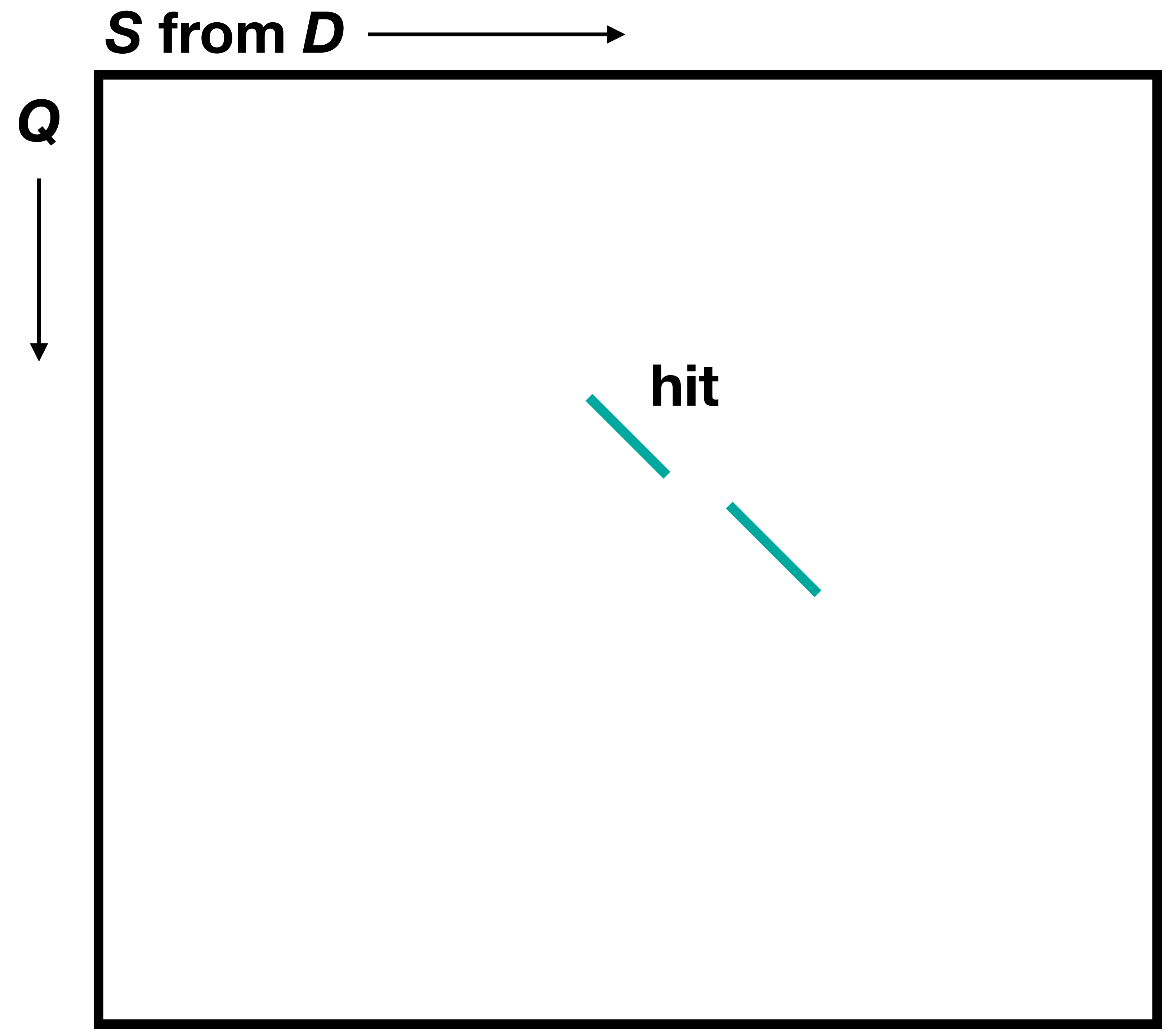
Basic Local Alignment Search Tool (BLAST)

Step 2: Database scanning -- label any instance of a neighbor of *Q* in any sequence *S* of *D* as a "hit", collect all of these hits



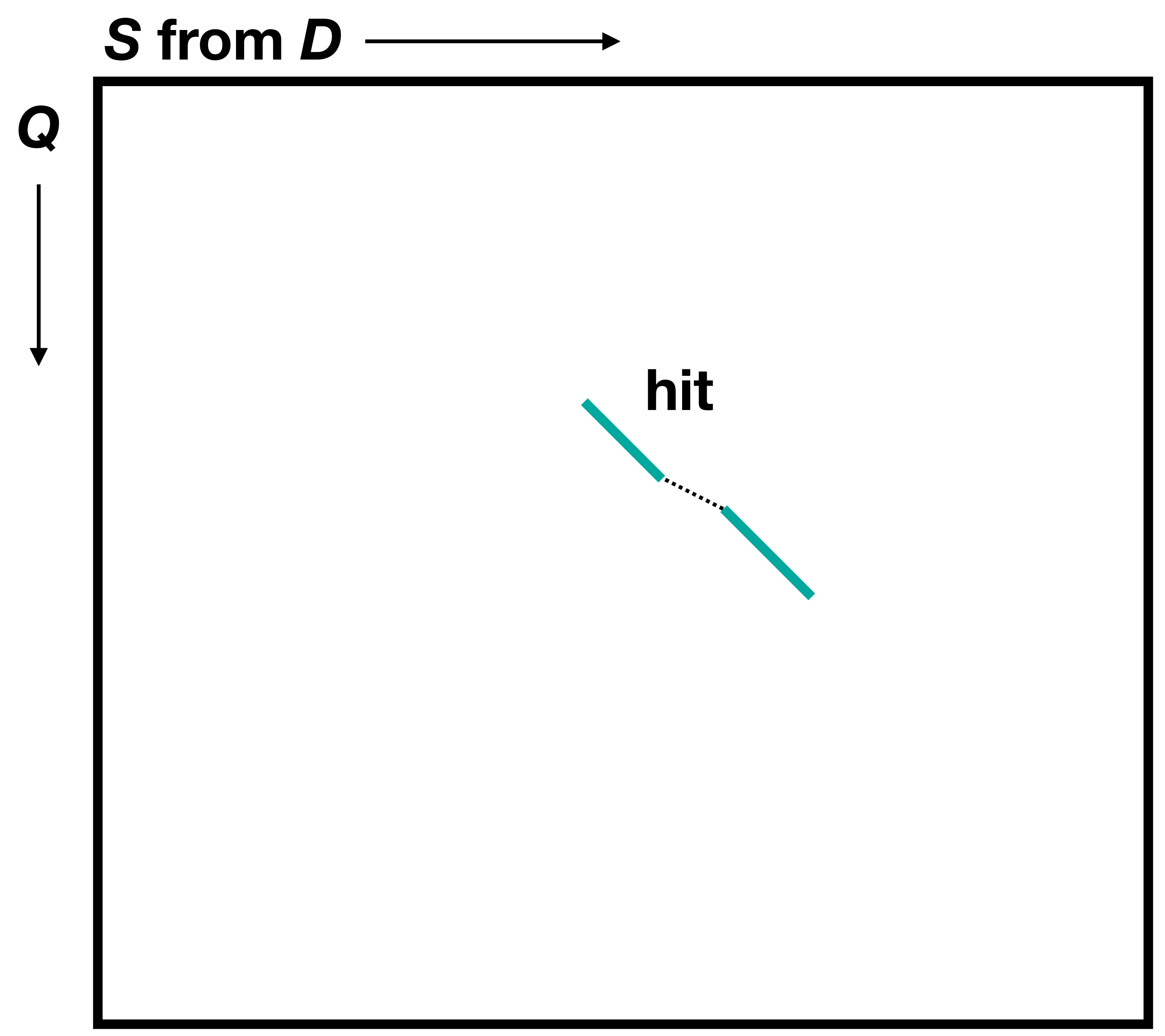
Basic Local Alignment Search Tool (BLAST2)

Step 3: Hit extension -- for any sequence *S* in *D*, with two hits (for protein, one for DNA) extend in either direction without gaps until the score drops too low



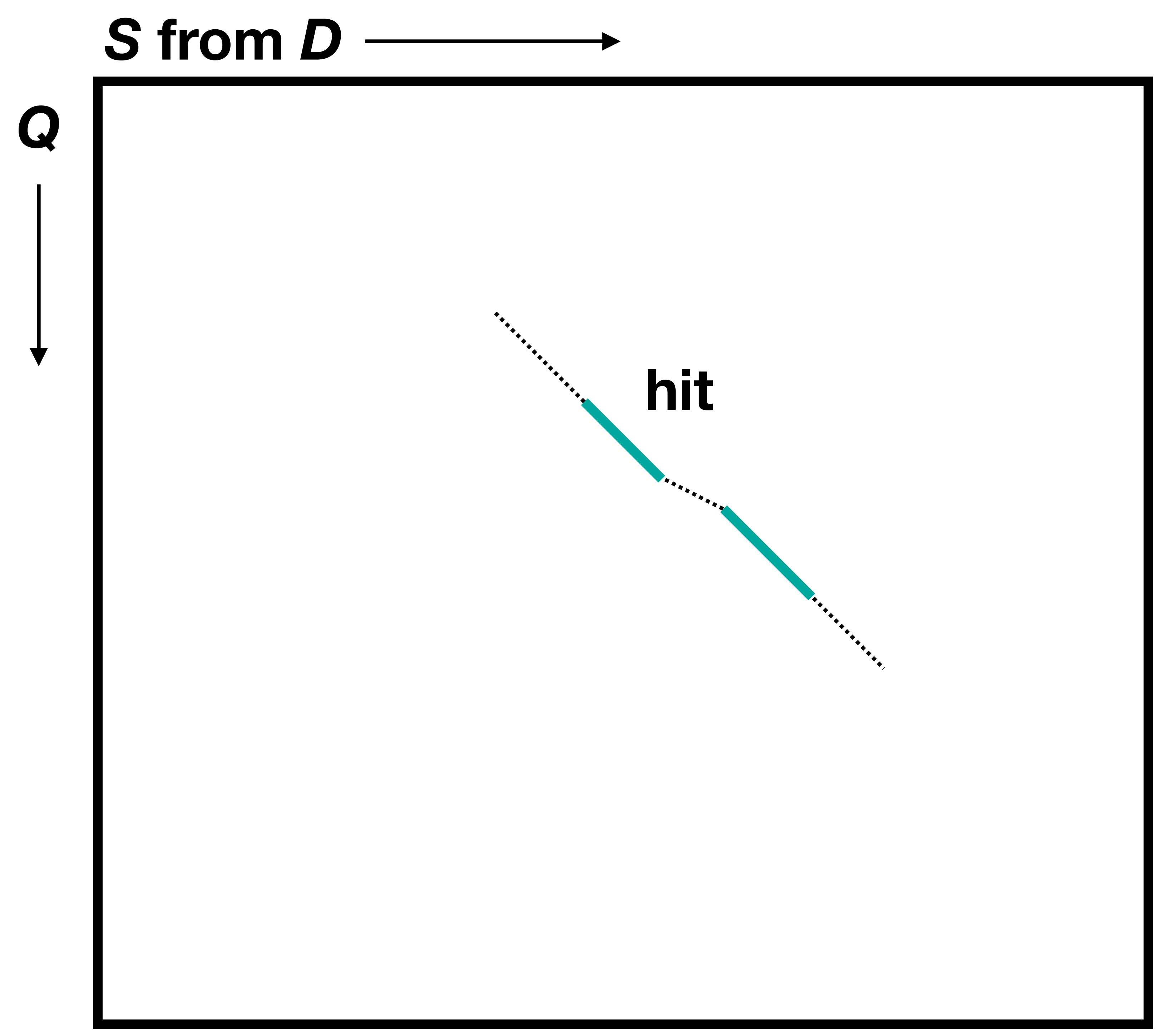
Basic Local Alignment Search Tool (BLAST2)

Step 3: Hit extension -- for any sequence *S* in *D*, with two hits (for protein, one for DNA) extend in either direction without gaps until the score drops too low



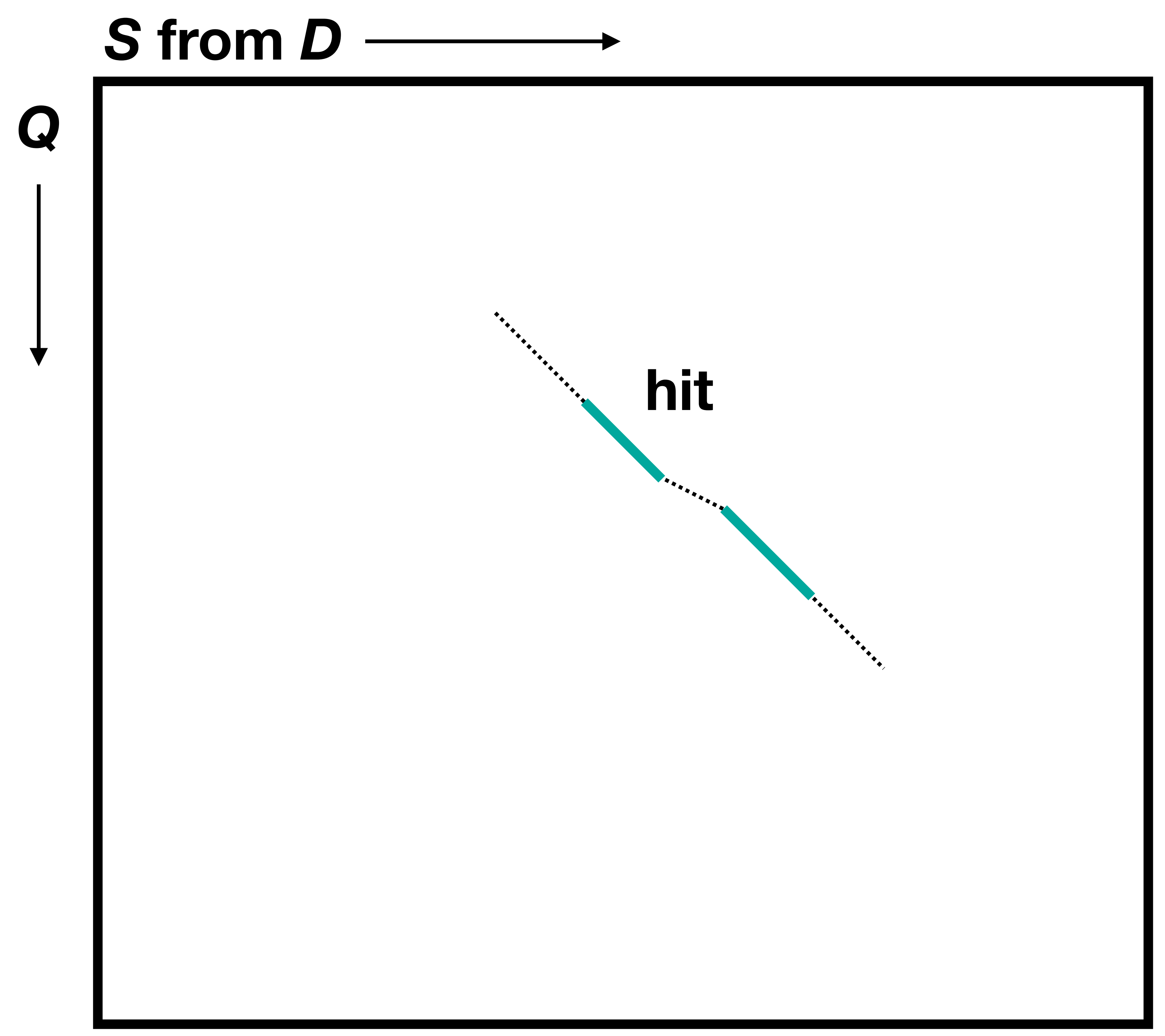
Basic Local Alignment Search Tool (BLAST2)

Step 3: Hit extension -- for any sequence *S* in *D*, with two hits (for protein, one for DNA) extend in either direction without gaps until the score drops too low



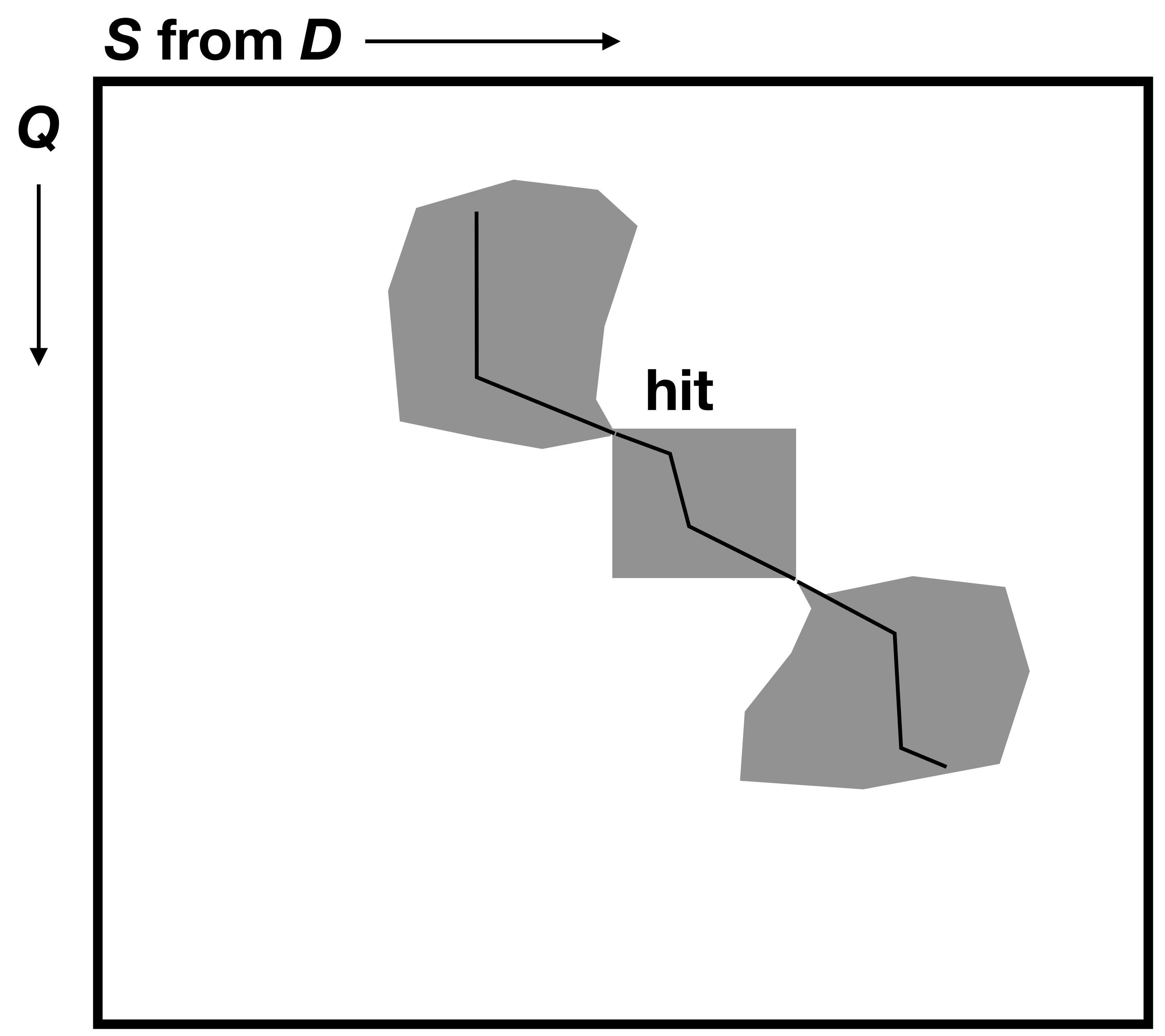
Basic Local Alignment Search Tool (BLAST2)

Step 4: Gapped extension -- run modified Smith-Waterman in each direction from the mid-point of the hits until the alignment score goes too low.



Basic Local Alignment Search Tool (BLAST2)

Step 4: Gapped extension -- run modified Smith-Waterman in each direction from the mid-point of the hits until the alignment score goes too low.



Database Search Statistics

Both BLAST and FastA return a hit quality score called an *E-value* and a *bit score*.

Database Search Statistics

Both BLAST and FastA return a hit quality score called an *E-value* and a *bit score*.

- **E-value** is the expected number of alignments having an alignment score >*S* at random.

$$E = Kmne^{-\lambda S}$$

  - *K* and *λ* are parameters based on the scoring scheme
  - as the lengths double, the number of sequences with that score does
  - as the score doubles, the number of sequences is exp. smaller

Database Search Statistics

Both BLAST and FastA return a hit quality score called an *E-value* and a *bit score*.

- **E-value** is the expected number of alignments having an alignment score >*S* at random.

$$E = Kmne^{-\lambda S}$$

  - *K* and *λ* are parameters based on the scoring scheme
  - as the lengths double, the number of sequences with that score does
  - as the score doubles, the number of sequences is exp. smaller

- **Bit score** is the normalized scoring value

$$S' = \frac{\lambda S - \ln K}{\ln 2}$$

  - Note that now $E=mne^{-S'}$ so when *S'* is big, the alignment is significant

Database Search Statistics

Both BLAST and FastA return a hit quality score called an *E-value* and a *bit score*.

- **E-value** is the expected number of alignments having an alignment score >*S* at random.

$$E = Kmne^{-\lambda S}$$

  - *K* and *λ* are parameters based on the scoring scheme
  - as the lengths double, the number of sequences with that score does
  - as the score doubles, the number of sequences is exp. smaller

- **Bit score** is the normalized scoring value

$$S' = \frac{\lambda S - \ln K}{\ln 2}$$

  - Note that now $E=mne^{-S'}$ so when *S'* is big, the alignment is significant

- You can calculate *p-values* from the *E-value* is $1-e^{-E}$.

MegaBLAST

Greedy adaptation that only works for DNA

Takes in multiple query sequences rather than one

- concatenates the sequences together
- runs the query on this longer sequences
- results are resorted after

Uses linear (affine) gap costs by default

BLAST-Like Alignment Tool (BLAT)

Only works for DNA (not Protein or RNA)

Instead of creating a lookup table for the query, create one for the database

- this takes a lot of memory to store
- only store **non-overlapping** *k*-mers

Still uses a 2-hit requirement

Stitches together local alignments on the same database sequence to create larger alignments (think intron splicing)

PatternHunter

Only works on DNA

Uses a patented concept called **Spaced Seeds**

A spaced seed is a binary sequence *BS* has two parameters:

- weight, *w*, and
- length, *m*.
- It contains *w* 1's, and *(m-w)* 0's

Two sequences sequences of length *m* are a match if the characters at the positions of *BS* that are 1's match

Spaced seeds reduce the number of false matches

PatternHunter

```
11111111111
AGCATTCAGTC
|||||||||||
AGCATTCAGTC
```

```
111010010100110111
ACTCCGATATGCGGTAAC
|||-|--|-|--||-|||
ACTTCACTGTGAGGCAAC
```

```
111010010100110111
ACTCCAATATGCGGTAAC
|||-|--|-|--x|-|||
ACTCCAATATGCAGTAAC
```

```
11111111111
 11111111111
```

```
111010010100110111
 111010010100110111
```

PatternHunter

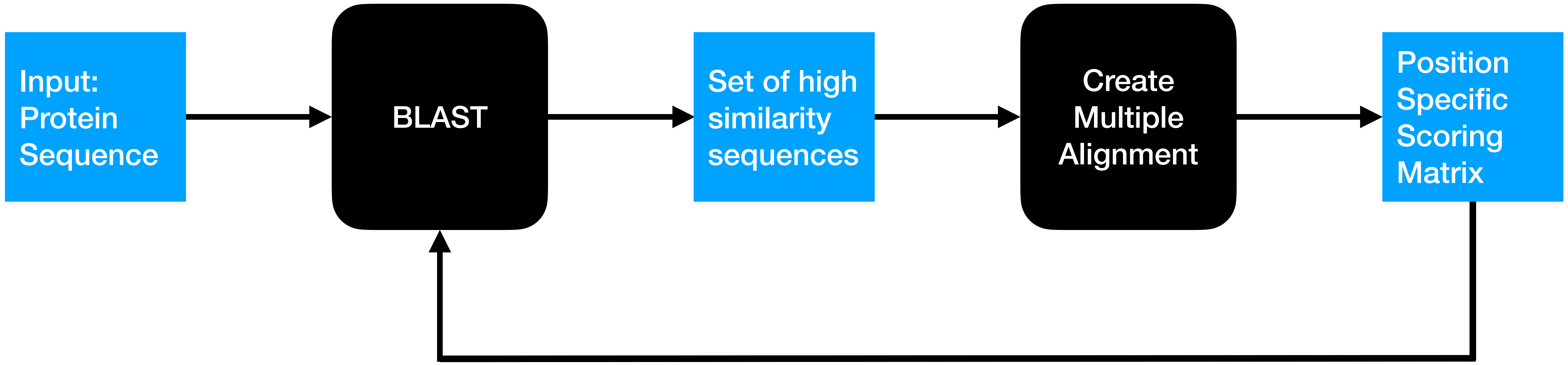
Lemma The expected number of hits of a weight-*w* length-*m* seed model within a length *L* region with similarity *p* ($p \in [0,1]$) is $(L-m+1)p^w$.

Proof For each possible position within the region, the probability of having *w* specific matches is p^w. Since there are *L-m+1* possible positgions within the region, the expected number of hits is $(L-m+1)p^w$.

Example, a region of 64 characters, with 70% similarity. BLAST is expected to have 1.07 hits, and PatternHunter would have 0.93. (w=11, m=11 for BLAST, m=18 for PatternHunter)

Position-Specific Iterated BLAST (PSI-BLAST)

Designed to find distant protein sequences.


Input: Protein Sequence
BLAST
Set of high similarity sequences
Create Multiple Alignment
Position Specific Scoring Matrix

Position-Specific Iterated BLAST (PSI-BLAST)

```
NGL ... M
NEL ... M
-GL ... M
NE- ... M
```

Position	1	2	3		n
A	0	0	0	...	0
R	0	0	0	...	0
N	100	0	0	...	0
D	0	0	0	...	0
C	0	0	0	...	0
W	0	0	0	...	0
E	0	50	0	...	0
G	0	50	0	...	0
H	0	0	0	...	0
I	0	0	0	...	0
L	0	0	100	...	0
K	0	0	0	...	0
M	0	0	0	...	100
F	0	0	0	...	0
P	0	0	0	...	0
S	0	0	0	...	0
T	0	0	0	...	0
W	0	0	0	...	0
Y	0	0	0	...	0
V	0	0	0	...	0

Q-gram Alignment base on Suffix ARrays (QUASAR)

Given

- a database, *D*
- a query, *S*
- a maximum difference, *k*, and
- the window size, *w*

Find:

- a set of *(X,Y)* where *X* and *Y* are length-*w* substrings in *D* and *S* respectively,
- such that the edit distance between *X* and *Y* is at most *k.*

Q-gram Alignment base on Suffix ARrays (QUASAR)

Based on splitting the windows into *q*-grams (*k*-mers)

Lemma Given two length *w* sequences *X* and *Y*, if their edit distance is at most *k*, then they must share at least *w+1-(k+1)q* common *q*-grams.

Q-gram Alignment base on Suffix ARrays (QUASAR)

Based on splitting the windows into *q*-grams (*k*-mers)

Lemma Given two length *w* sequences *X* and *Y*, if their edit distance is at most *k*,
then they must share at least $w+1-(k+1)q$ common *q*-grams.

Proof

Q-gram Alignment base on Suffix ARrays (QUASAR)

Based on splitting the windows into *q*-grams (*k*-mers)

Lemma Given two length *w* sequences *X* and *Y*, if their edit distance is at most *k*, then they must share at least $w+1-(k+1)q$ common *q*-grams.

Proof

- Let
  - *(X',Y')* be an optional alignment of *X* and *Y*

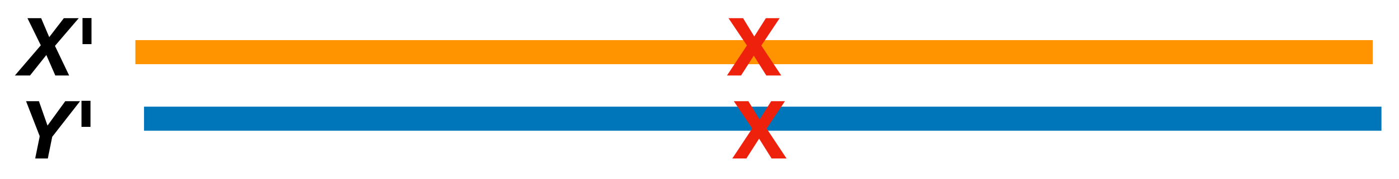
Q-gram Alignment base on Suffix ARrays (QUASAR)

Based on splitting the windows into *q*-grams (*k*-mers)

Lemma Given two length *w* sequences *X* and *Y*, if their edit distance is at most *k*, then they must share at least *w+1-(k+1)q* common *q*-grams.

Proof

- Let
  - *(X',Y')* be an optional alignment of *X* and *Y*
  - *r* be the number of differences between *X'* & *Y'* (*r <= k)*



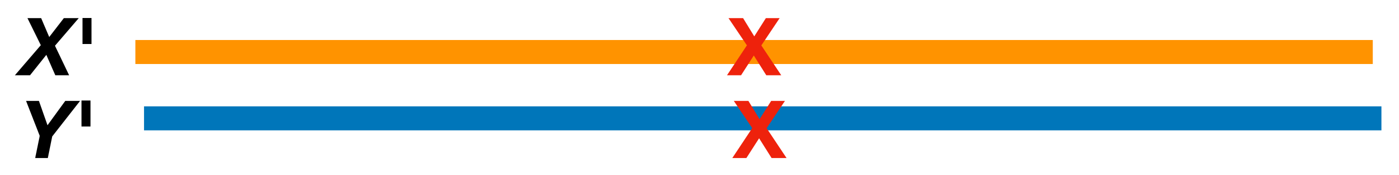
Q-gram Alignment base on Suffix ARrays (QUASAR)

Based on splitting the windows into *q*-grams (*k*-mers)

Lemma Given two length *w* sequences *X* and *Y*, if their edit distance is at most *k*, then they must share at least *w+1-(k+1)q* common *q*-grams.

Proof

- Let
  - *(X',Y')* be an optional alignment of *X* and *Y*
  - *r* be the number of differences between *X'* & *Y'* (*r <= k)*
  - *L* be the length of *X'* and *Y' (L >= w)*.



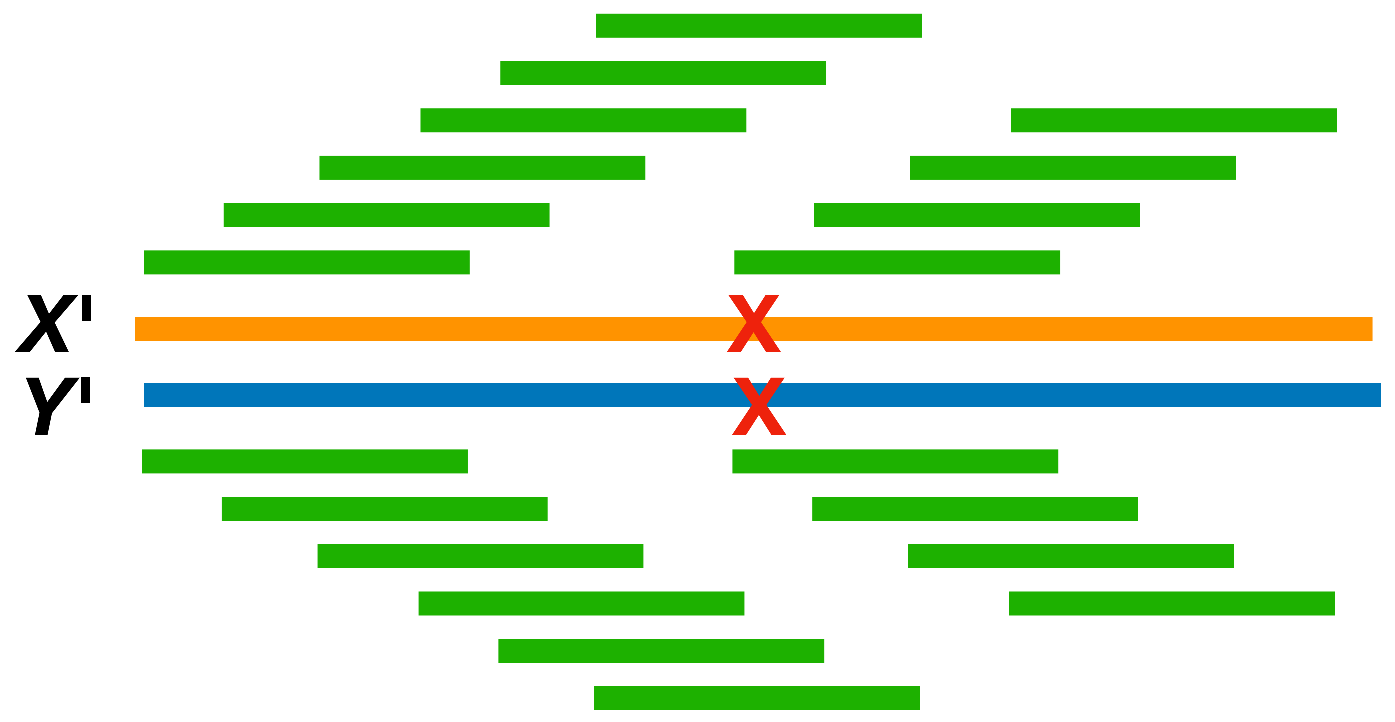
Q-gram Alignment base on Suffix ARrays (QUASAR)

Based on splitting the windows into *q*-grams (*k*-mers)

Lemma Given two length *w* sequences *X* and *Y*, if their edit distance is at most *k*, then they must share at least *w+1-(k+1)q* common *q*-grams.

Proof

- Let
  - *(X',Y')* be an optional alignment of *X* and *Y*
  - *r* be the number of differences between *X'* & *Y'* (*r <= k)*
  - *L* be the length of *X'* and *Y' (L >= w)*.
- Consider the *L+1-q* pairs of *q-grams* of *X'* and *Y'* starting at the same position.



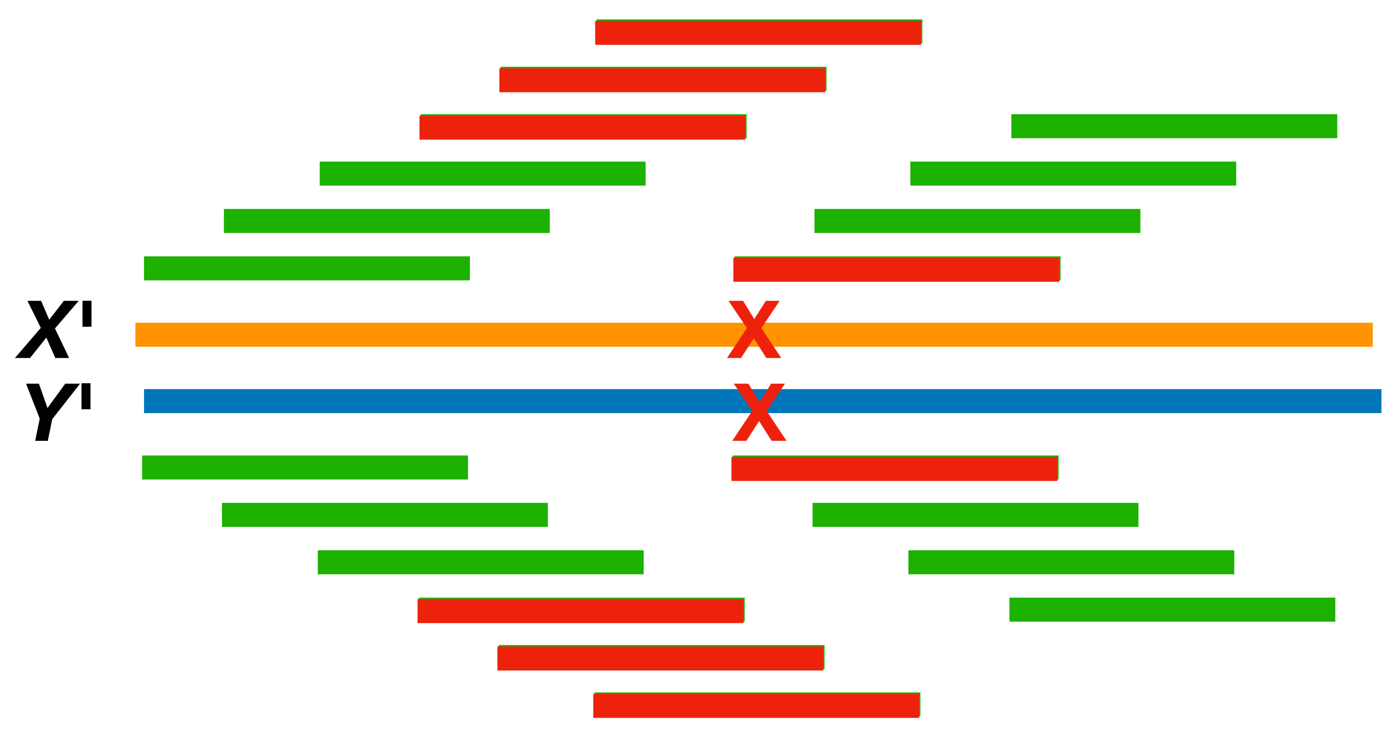
<u>Q</u>-gram <u>A</u>lignment base on <u>S</u>uffix <u>AR</u>rays (QUASAR)

Based on splitting the windows into *q*-grams (*k*-mers)

Lemma Given two length *w* sequences *X* and *Y*, if their edit distance is at most *k*, then they must share at least *w+1-(k+1)q* common *q*-grams.

Proof

- Let
  - *(X',Y')* be an optional alignment of *X* and *Y*
  - *r* be the number of differences between *X'* & *Y'* (*r <= k)*
  - *L* be the length of *X'* and *Y' (L >= w)*.
- Consider the *L+1-q* pairs of *q-grams* of *X'* and *Y'* starting at the same position.
- Each difference between *X'* and *Y'* can create at most *q* such pairs to be different.



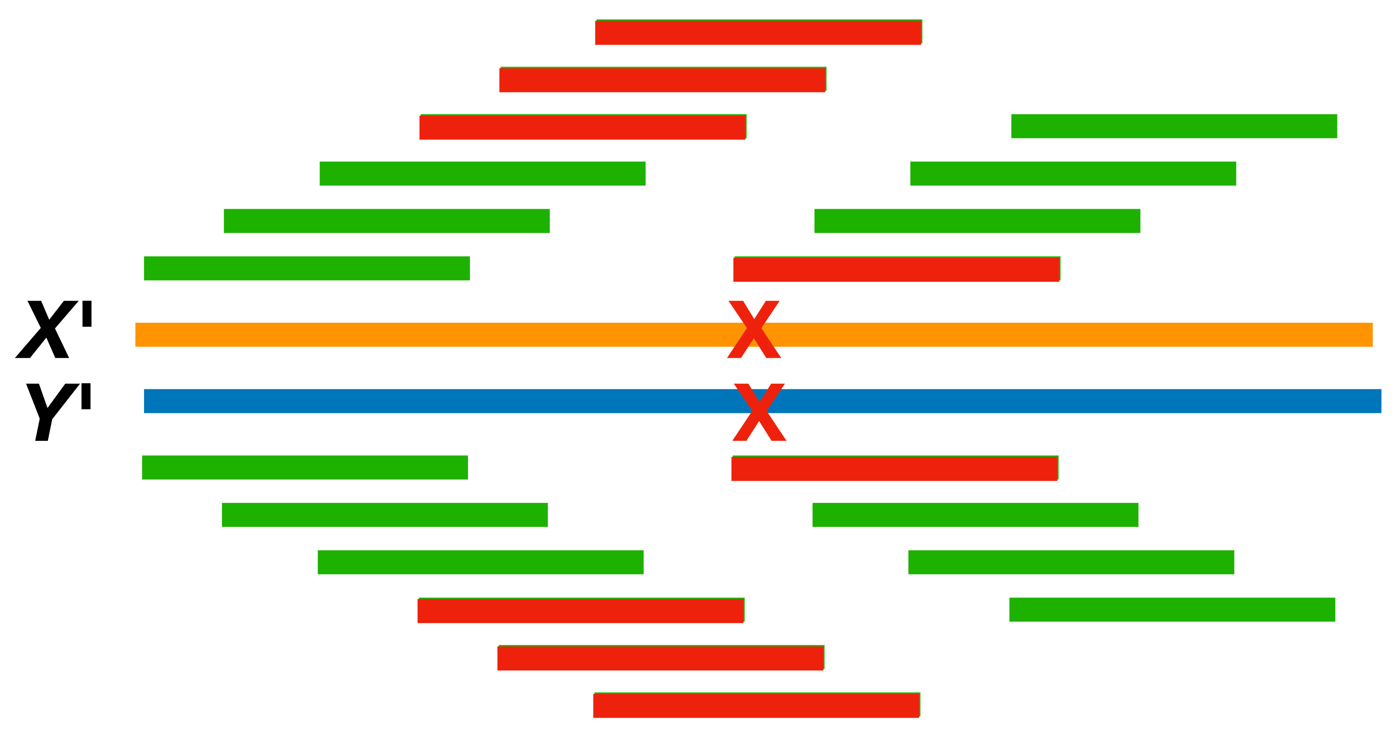
Q-gram Alignment base on Suffix ARrays (QUASAR)

Based on splitting the windows into *q*-grams (*k*-mers)

Lemma Given two length *w* sequences *X* and *Y*, if their edit distance is at most *k*, then they must share at least *w+1-(k+1)q* common *q*-grams.

Proof

- Let
  - *(X',Y')* be an optional alignment of *X* and *Y*
  - *r* be the number of differences between *X'* & *Y'* (*r <= k*)
  - *L* be the length of *X'* and *Y' (L >= w)*.
- Consider the *L+1-q* pairs of *q-grams* of *X'* and *Y'* starting at the same position.
- Each difference between *X'* and *Y'* can create at most *q* such pairs to be different.
- Thus *X'* and *Y'* must have *L+1-q-rq >= w+1-(k+1)q* common *q*-grams.



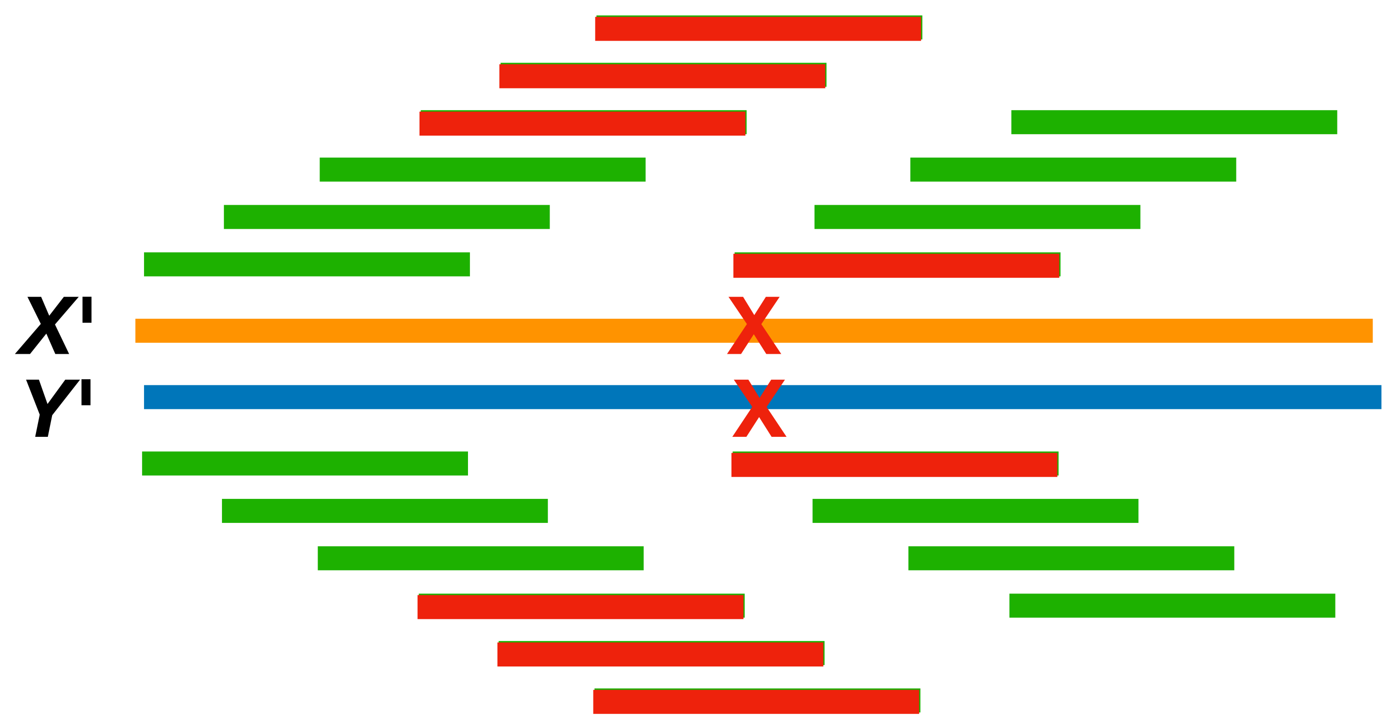
Q-gram Alignment base on Suffix ARrays (QUASAR)

Based on splitting the windows into *q*-grams (*k*-mers)

Lemma Given two length *w* sequences *X* and *Y*, if their edit distance is at most *k*, then they must share at least *w+1-(k+1)q* common *q*-grams.

Proof

- Let
  - *(X',Y')* be an optional alignment of *X* and *Y*
  - *r* be the number of differences between *X'* & *Y'* (*r <= k)*
  - *L* be the length of *X'* and *Y' (L >= w)*.
- Consider the *L+1-q* pairs of *q-grams* of *X'* and *Y'* starting at the same position.
- Each difference between *X'* and *Y'* can create at most *q* such pairs to be different.
- Thus *X'* and *Y'* must have *L+1-q-rq >= w+1-(k+1)q* common *q*-grams.
- Any common *q*-gram for *X'* and *Y'* is also common for *X* and *Y.*



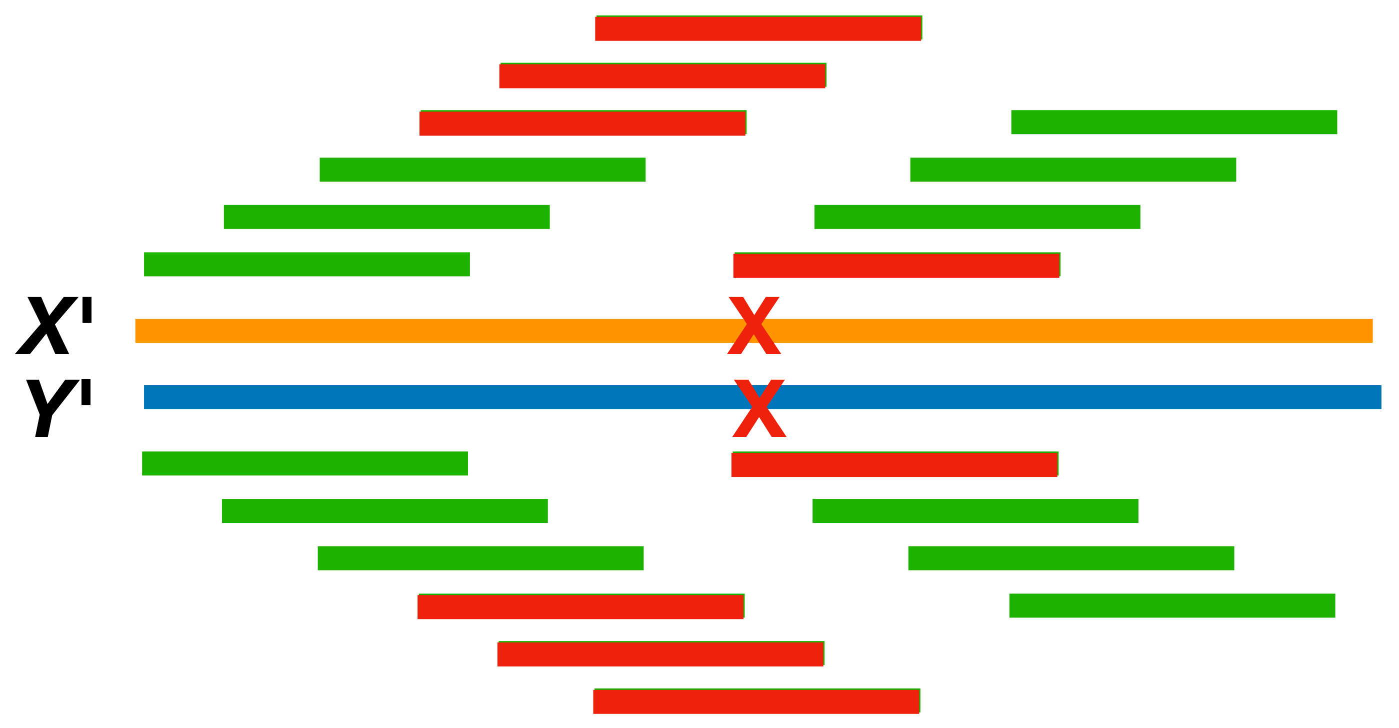
Q-gram Alignment base on Suffix ARrays (QUASAR)

Based on splitting the windows into *q*-grams (*k*-mers)

Lemma Given two length *w* sequences *X* and *Y*, if their edit distance is at most *k*, then they must share at least *w+1-(k+1)q* common *q*-grams.

Proof

- Let
  - *(X',Y')* be an optional alignment of *X* and *Y*
  - *r* be the number of differences between *X'* & *Y'* (*r <= k*)
  - *L* be the length of *X'* and *Y' (L >= w)*.
- Consider the *L+1-q* pairs of *q-grams* of *X'* and *Y'* starting at the same position.
- Each difference between *X'* and *Y'* can create at most *q* such pairs to be different.
- Thus *X'* and *Y'* must have *L+1-q-rq >= w+1-(k+1)q* common *q*-grams.
- Any common *q*-gram for *X'* and *Y'* is also common for *X* and *Y.*



This is just an application of the pigeon-hole principle

Q-gram Alignment base on Suffix ARrays (QUASAR)

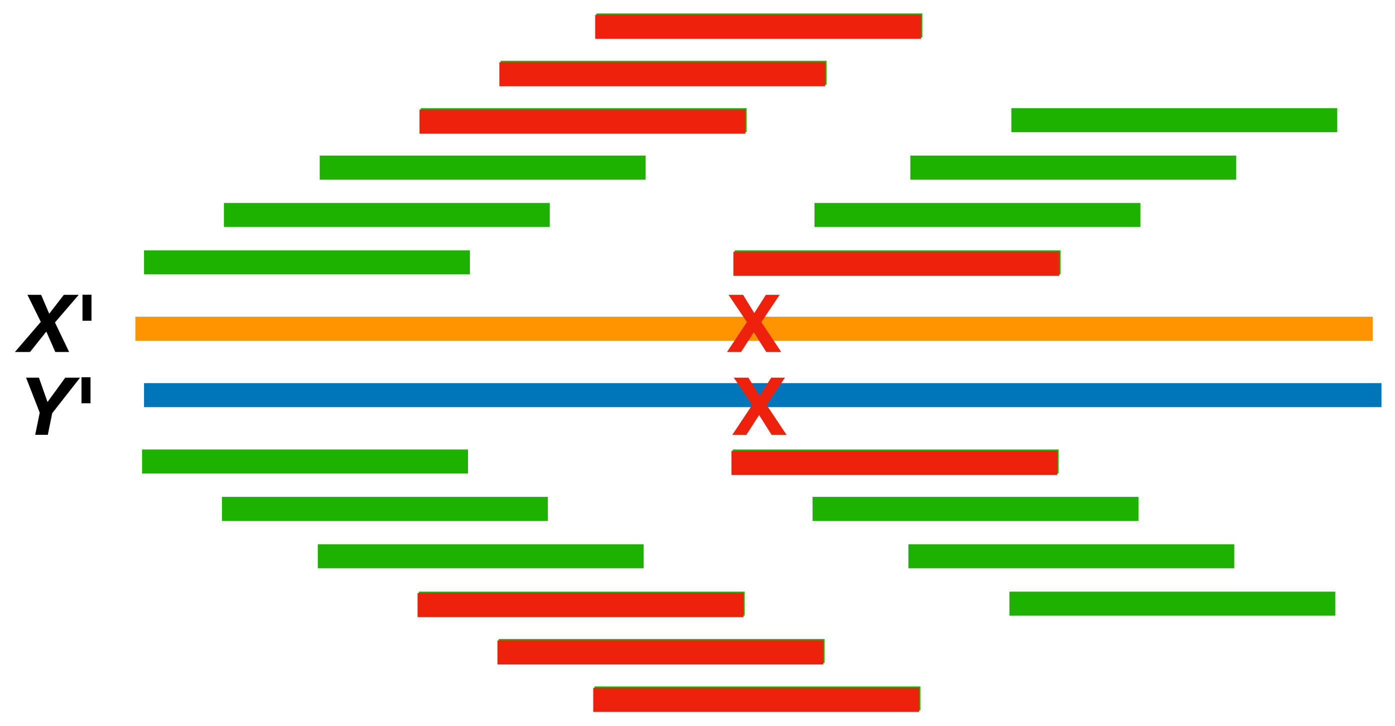
Based on splitting the windows into *q*-grams (*k*-mers)

Lemma Given two length *w* sequences *X* and *Y*, if their edit distance is at most *k*, then they must share at least *w+1-(k+1)q* common *q*-grams.

Proof

- Let
  - *(X',Y')* be an optional alignment of *X* and *Y*
  - *r* be the number of differences between *X'* & *Y'* (*r <= k)*
  - *L* be the length of *X'* and *Y' (L >= w)*.
- Consider the *L+1-q* pairs of *q-grams* of *X'* and *Y'* starting at the same position.
- Each difference between *X'* and *Y'* can create at most *q* such pairs to be different.
- Thus *X'* and *Y'* must have *L+1-q-rq >= w+1-(k+1)q* common *q*-grams.
- Any common *q*-gram for *X'* and *Y'* is also common for *X* and *Y.*

This is just an application of the pigeon-hole principle



If *w+1-(k-1)q* *q*-grams match, can edit distance be higher than *k*?

Q-gram Alignment base on Suffix ARrays (QUASAR)

The actual QUASAR algorithm uses this principle to find potential alignments:

- for each *w* length substring of *S, X* and
    - maintain counters for each *w* length substring of *D, Y*
    - for each *q*-gram in S, increment the counters for the *Y* that contain it
    - for all Y with counter greater than $w+1-(k+1)q$, run a sequence alignment algorithm

Q-gram Alignment base on Suffix ARrays (QUASAR)

The actual QUASAR algorithm uses this principle to find potential alignments:

- for each *w* length substring of *S, X* and
  - maintain counters for each *w* length substring of *D, Y*
  - for each *q*-gram in S, increment the counters for the *Y* that contain it
  - for all Y with counter greater than $w+1-(k+1)q$, run a sequence alignment algorithm

Where do suffix arrays come in?

- When searching for the *q*-grams in *D*
- along with an additional array *idx(Q)* which points to the begining of the locations that start with *Q* in the suffix array.

Q-gram Alignment base on Suffix ARrays (QUASAR)

The actual QUASAR algorithm uses this principle to find potential alignments:

- for each *w* length substring of *S, X* and
  - maintain counters for each *w* length substring of *D, Y*
  - for each *q*-gram in S, increment the counters for the *Y* that contain it
  - for all Y with counter greater than *w+1-(k+1)q,* run a sequence alignment algorithm

Where do suffix arrays come in?

- When searching for the *q*-grams in *D*
- along with an additional array *idx(Q)* which points to the begining of the locations that start with *Q* in the suffix array.

i	SA[i]		
1	5	ACT	*idx*(AC)
2	2	AGCT	*idx*(AG)
3	4	CACT	*idx*(CA)
4	1	CAGCACT	
5	6	CT	*idx*(CT)
6	3	GCACT	*idx*(GC)
7	7	T	

Q-gram Alignment base on Suffix ARrays (QUASAR)

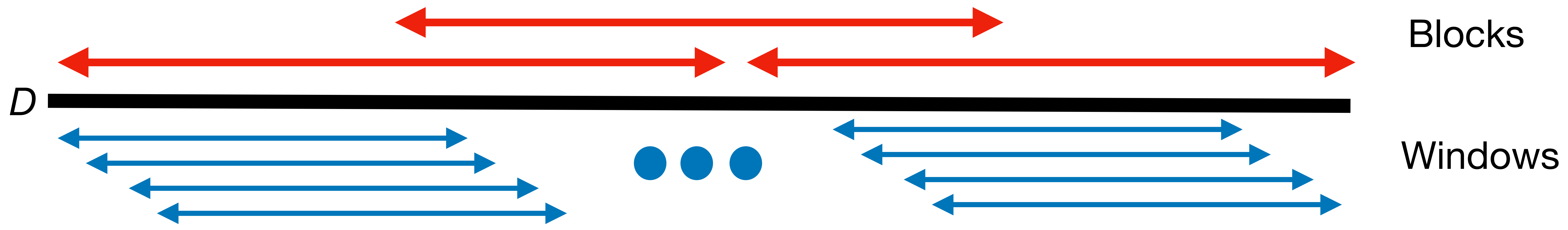
Speedups

Window Shifting

- Similar to the solution to homework 2, each window shared quite a few *q*-grams with the one before it, use that to reduce running time.

Block Addressing

- Rather than counting the occurrences in all *Y*, break *D* into non-overlapping blocks of *b (> 2w)* and keep counters there
- Keep a second offset set of blocks to not miss any spanning windows.
- If any block contains enough matching *q*-grams, run a more detailed pass



Q-gram Alignment base on Suffix ARrays (QUASAR)

Running time

Q-gram Alignment base on Suffix ARrays (QUASAR)

Running time

- Suffix array construction *O(|D| log |D|)*

Q-gram Alignment base on Suffix ARrays (QUASAR)

Running time

- Suffix array construction $O(|D| \log |D|)$
- S has $O(|S|)$ q-grams, which we expect $|D|/4^q$ hits each, therefore the initial hit list is generated in $O(|S||D|/4^q)$ expected time.

Q-gram Alignment base on Suffix ARrays (QUASAR)

Running time

- Suffix array construction $O(|D| \log |D|)$
- S has $O(|S|)$ q-grams, which we expect $|D|/4^q$ hits each, therefore the initial hit list is generated in $O(|S||D|/4^q)$ expected time.
- If c blocks meet the hit requirements, the alignment takes $O(c\ b^2)$ time

Q-gram Alignment base on Suffix ARrays (QUASAR)

Running time

- Suffix array construction *O(|D| log |D|)*
- *S* has *O(|S|)* *q*-grams, which we expect $|D|/4^q$ hits each, therefore the initial hit list is generated in $O(|S||D|/4^q)$ expected time.
- If *c* blocks meet the hit requirements, the alignment takes $O(c\,b^2)$ time
- Total search time is $O\left(\frac{|S||D|}{4^q} + cb^2\right)$

Q-gram Alignment base on Suffix ARrays (QUASAR)

Running time

- Suffix array construction $O(|D| \log |D|)$
- S has $O(|S|)$ q-grams, which we expect $|D|/4^q$ hits each, therefore the initial hit list is generated in $O(|S||D|/4^q)$ expected time.
- If c blocks meet the hit requirements, the alignment takes $O(c\, b^2)$ time
- Total search time is $O\left(\frac{|S||D|}{4^q} + cb^2\right)$

Space

Q-gram Alignment base on Suffix ARrays (QUASAR)

Running time

- Suffix array construction $O(|D| \log |D|)$
- S has $O(|S|)$ q-grams, which we expect $|D|/4^q$ hits each, therefore the initial hit list is generated in $O(|S||D|/4^q)$ expected time.
- If c blocks meet the hit requirements, the alignment takes $O(c\ b^2)$ time
- Total search time is $O\left(\frac{|S||D|}{4^q} + cb^2\right)$

Space

- Suffix array takes $O(|D| \log |D|)$ space, then $O(|D|/b + b^2)$ space for the query.

Locality Sensitive Hashing

The idea of locality sensitive hashes, is that you can use an efficient to compute hash to *estimate* something that is computationally difficult.

Let *s* be the similarity you would like to estimate, and *h* be a hash function on the same types of elements. (*d* would take two arguments and return a distance, *h* takes one argument and returns something).

We say *h* is an LSH for *d* if

- $s(x,y) = pr(h(x)=h(y))$

We say *h* is a **gapped** LSH for *d* if the following holds:

- if $s(x,y) \leq s_1$ then $pr(h(x)=h(y)) \leq p_1$, and
- if $s(x,y) \geq s_2$ then $pr(h(x)=h(y)) \geq p_2$.
- more precisely it's (s_1,s_2,p_1,p_2)-sensitive.

Quick digression to Hamming Distance

We know edit distance is the minimum number of insertions, deletions, and mismatches to convert one string into another.

Hamming distance is the minimum number of only mismatches.

Also used in vectors, the number of dimensions that have different values.

Locality Sensitive Hashing

Let $h_{k,\pi}(s)$ be a function that takes string *s* and return a selected set of *k* characters based on some random ordering of integers π.

If the hamming distance of s_1 and s_2, both of length *w,* is *d,* then

$$Pr\left(h_{k,\pi}(s_1) = h_{k,\pi}(s_2)\right) = \prod_{j=1,\ldots,k} Pr\left(s_1\left[\pi\left[j\right]\right] = s_2\left[\pi\left[j\right]\right]\right) = \left(1 - \frac{d}{w}\right)^k$$

In other words, the more similar the sequences are (the lower *d* is and thus) the higher probability of a hash collision.

LSH-ALL-PAIRS

Using the Locality Sensitive Hash described for hamming distance, locate highly-probable match locations.

The LSH can introduce false discoveries:

- **False positive**: s_1 *and* s_2 are dissimilar, but $h_{k,\pi}(s_1) = h_{k,\pi}(s_2)$
  - can be eliminated by checking the actual hamming distance
- **False negative**: s_1 and s_2 are similar, but $h_{k,\pi}(s_1) \neq h_{k,\pi}(s_2)$
  - can be reduced by repeating search using multiple π

LSH-ALL-PAIRS

Algorithm (given *Q, D, w, d, m*)

- generate *m* random orderings $\pi_1, \pi_2, \ldots, \pi_m$.
- for every *w*-mer *s* in *D*, compute $h_{k,\pi 1}(s), h_{k,\pi m}(s), \ldots, h_{k,\pi m}(s)$.
- for every pair of *w*-mers *s* and *t* from *D* and *Q* such that $h_{k,\pi j}(s) = h_{k,\pi j}(t)$ for some *j*
    - if the hamming distance between *s* and *t* is less than *d*, report *(s,t)*

LSH-ALL-PAIRS

Unlike the previous algorithms, LSH-ALL-PAIRS provides a guarantee that all sequences with hamming distance less than *d* will be found with probability

$$\prod_{1 \leq i \leq m} \left(1 - Pr\left(h_{k,\pi_i}(s_1) = h_{k,\pi_i}(s_2)\right)\right)$$

LSH-ALL-PAIRS

Unlike the previous algorithms, LSH-ALL-PAIRS provides a guarantee that all sequences with hamming distance less than *d* will be found with probability

$$\prod_{1 \le i \le m} \left(1 - Pr\left(h_{k,\pi_i}\left(s_1\right) = h_{k,\pi_i}\left(s_2\right)\right)\right)$$

Remember that $Pr\left(h_{k,\pi}(s_1) = h_{k,\pi}(s_2)\right) = \left(1 - \frac{d}{w}\right)^k$

LSH-ALL-PAIRS

Unlike the previous algorithms, LSH-ALL-PAIRS provides a guarantee that all sequences with hamming distance less than *d* will be found with probability

$$\prod_{1 \le i \le m} \left(1 - Pr\left(h_{k,\pi_i}(s_1) = h_{k,\pi_i}(s_2)\right)\right) = \prod_{1 \le i \le m} \left(1 - \left(1 - \frac{d}{w}\right)^k\right)$$

Remember that $Pr\left(h_{k,\pi}(s_1) = h_{k,\pi}(s_2)\right) = \left(1 - \frac{d}{w}\right)^k$

LSH-ALL-PAIRS

Unlike the previous algorithms, LSH-ALL-PAIRS provides a guarantee that all sequences with hamming distance less than *d* will be found with probability

$$\prod_{1 \le i \le m} \left(1 - Pr\left(h_{k,\pi_i}\left(s_1\right) = h_{k,\pi_i}\left(s_2\right)\right)\right) = \prod_{1 \le i \le m} \left(1 - \left(1 - \frac{d}{w}\right)^k\right) = \left(1 - \left(1 - \frac{d}{w}\right)^k\right)^m$$

Remember that $Pr\left(h_{k,\pi}(s_1) = h_{k,\pi}(s_2)\right) = \left(1 - \frac{d}{w}\right)^k$

BWA-SW

What if we still want the optimal local alignment between the query and the text?

Since we know we can't run Smith-Waterman on the whole sequence, we need something faster.

BWA-SW

What if we still want the optimal local alignment between the query and the text?

Since we know we can't run Smith-Waterman on the whole sequence, we need something faster.

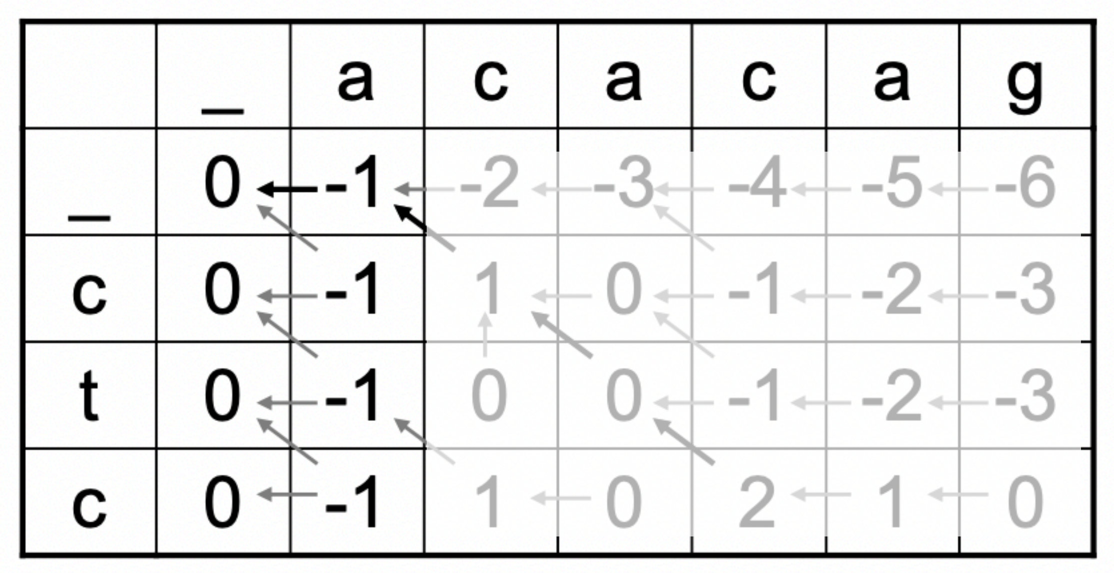
Suffix Trees!

BWT-SW

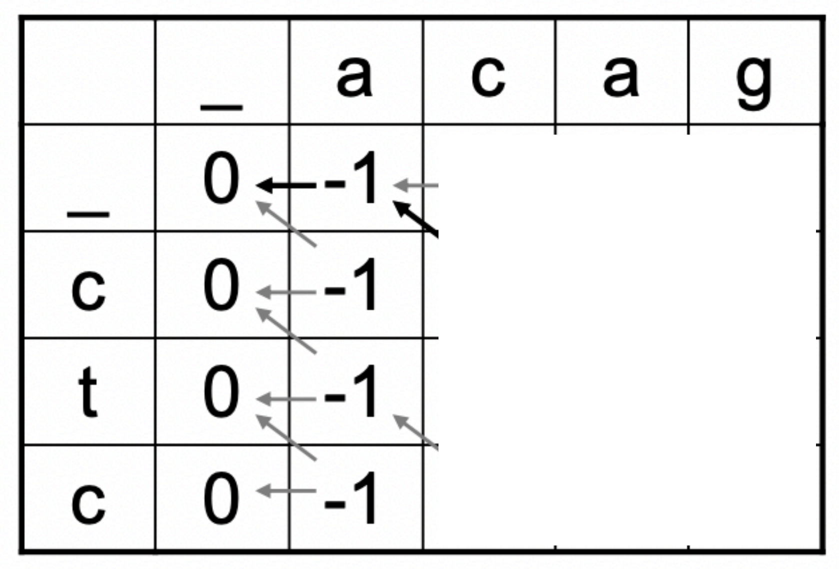
$Q = ctc$
$T = acacag$

each character in the tree has one "column" of the DP table

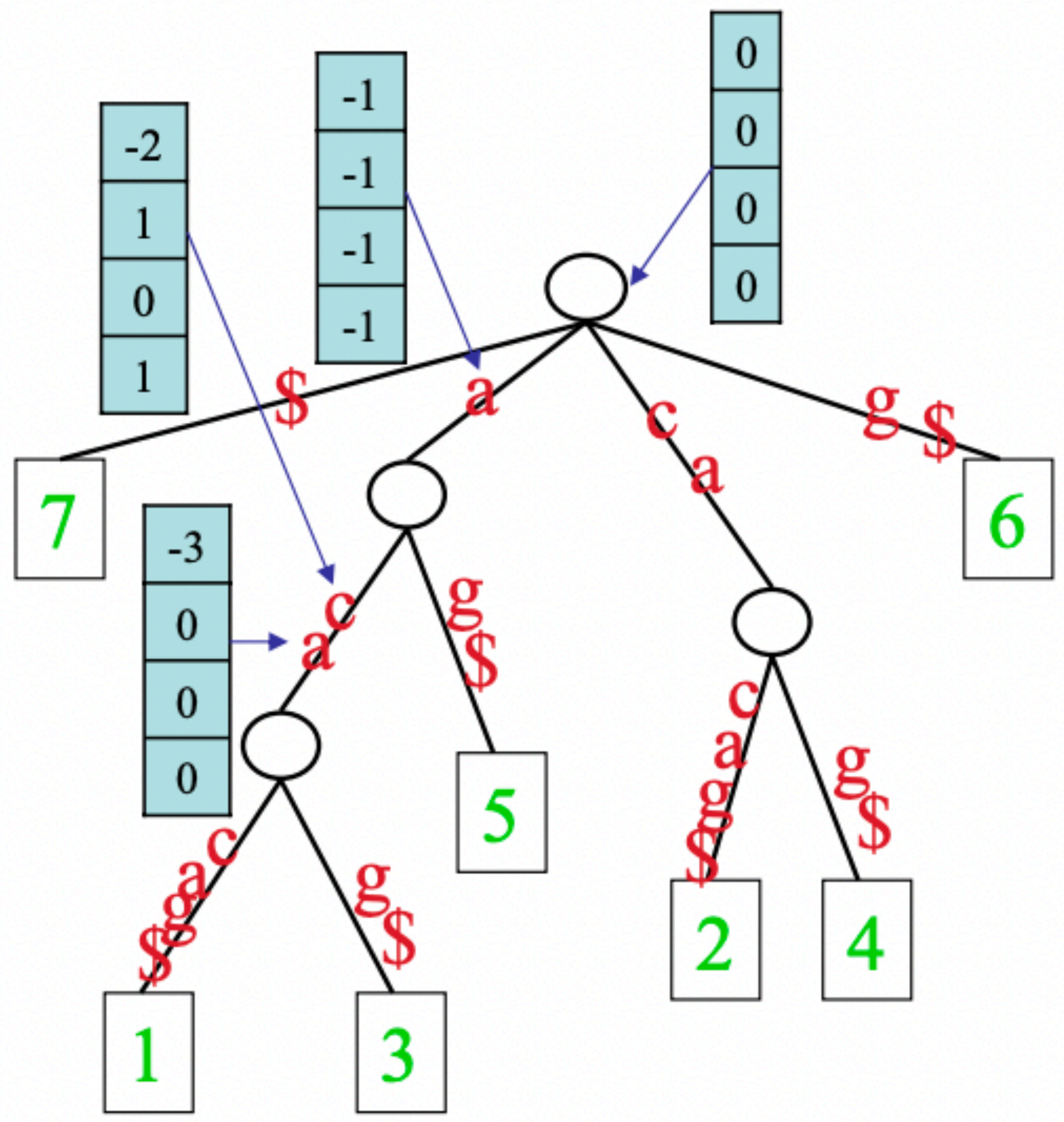
- still use a simple recurrence relation



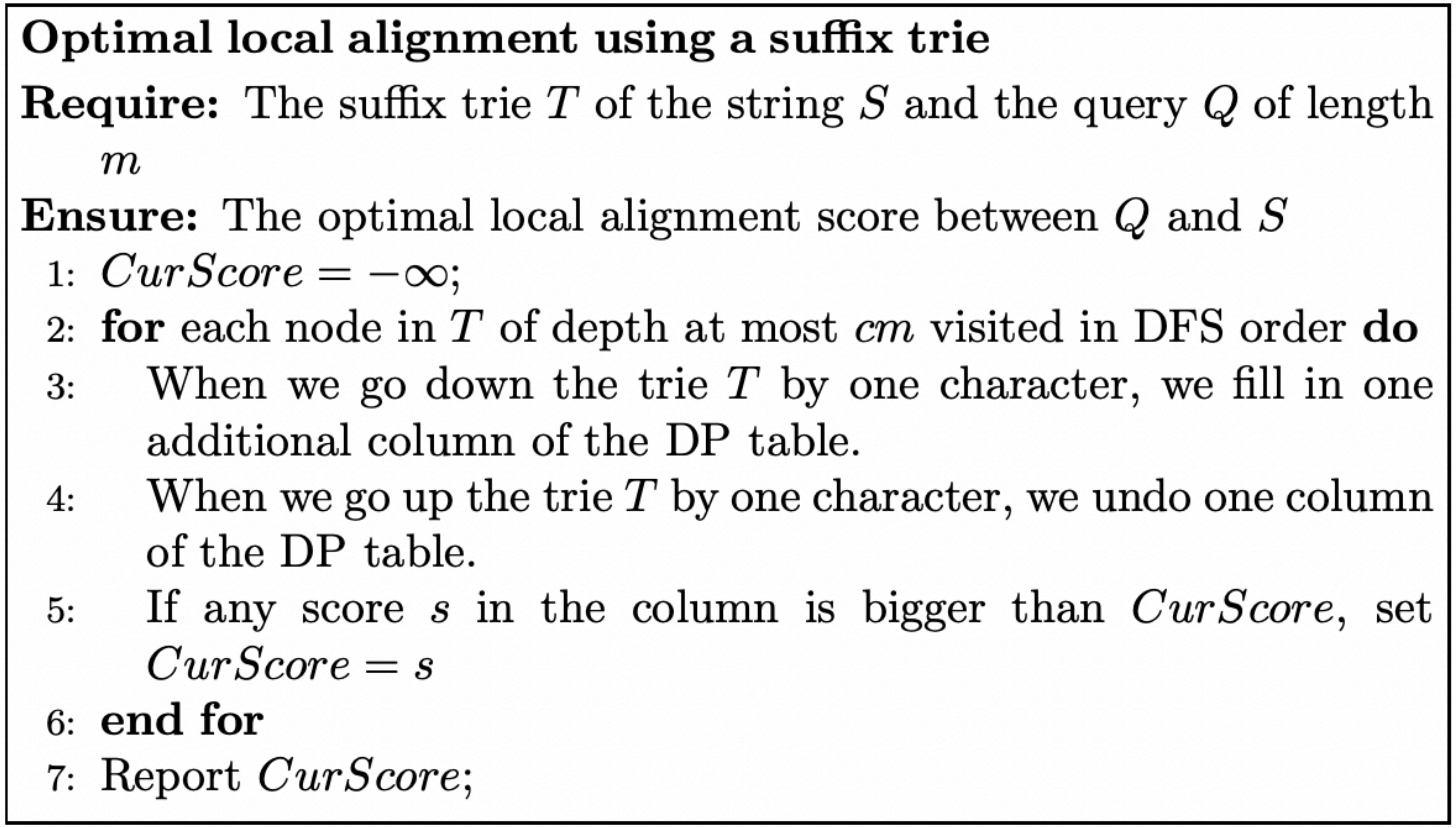
	_	a	c	a	c	a	g
_	0	-1	-2	-3	-4	-5	-6
c	0	-1	1	0	-1	-2	-3
t	0	-1	0	0	-1	-2	-3
c	0	-1	1	0	2	1	0



	_	a	c	a	g
_	0	-1			
c	0	-1			
t	0	-1			
c	0	-1			



BWT-SW



Optimal local alignment using a suffix trie

Require: The suffix trie T of the string S and the query Q of length m

Ensure: The optimal local alignment score between Q and S

```
1: $CurScore = -\infty$;
2: for each node in $T$ of depth at most $cm$ visited in DFS order do
3:     When we go down the trie $T$ by one character, we fill in one additional column of the DP table.
4:     When we go up the trie $T$ by one character, we undo one column of the DP table.
5:     If any score $s$ in the column is bigger than $CurScore$, set $CurScore = s$
6: end for
7: Report $CurScore$;
```

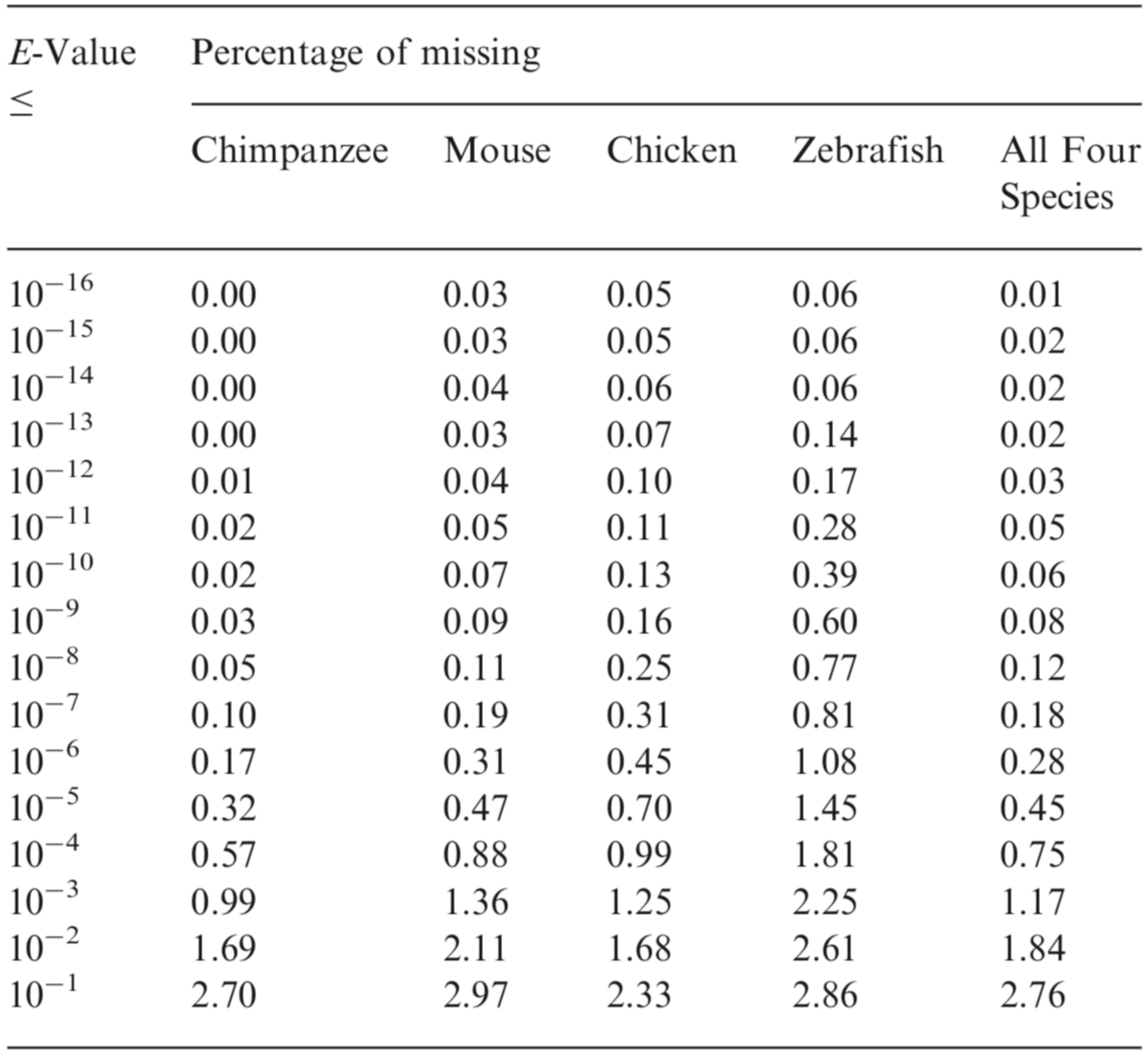
Are the methods presented good enough?

8,000 queries

- 2,000 from each of 4 species: chimpanzee, mouse, chicken, zebrafish
- length ranged from 170-19,000 bases (average of 2,700)

Aligned to the human genome using BLAST

Baseline is an exact search algorithm called BWT-SW



E-Value ≤	Percentage of missing				
	Chimpanzee	Mouse	Chicken	Zebrafish	All Four Species
10^{-16}	0.00	0.03	0.05	0.06	0.01
10^{-15}	0.00	0.03	0.05	0.06	0.02
10^{-14}	0.00	0.04	0.06	0.06	0.02
10^{-13}	0.00	0.03	0.07	0.14	0.02
10^{-12}	0.01	0.04	0.10	0.17	0.03
10^{-11}	0.02	0.05	0.11	0.28	0.05
10^{-10}	0.02	0.07	0.13	0.39	0.06
10^{-9}	0.03	0.09	0.16	0.60	0.08
10^{-8}	0.05	0.11	0.25	0.77	0.12
10^{-7}	0.10	0.19	0.31	0.81	0.18
10^{-6}	0.17	0.31	0.45	1.08	0.28
10^{-5}	0.32	0.47	0.70	1.45	0.45
10^{-4}	0.57	0.88	0.99	1.81	0.75
10^{-3}	0.99	1.36	1.25	2.25	1.17
10^{-2}	1.69	2.11	1.68	2.61	1.84
10^{-1}	2.70	2.97	2.33	2.86	2.76

[T. W. Lam, et al. **Compressed indexing and local alignment of DNA**, *Bioinformatics*, 24(6), March 2008, Pgs. 791–797]

Are the methods presented good enough?

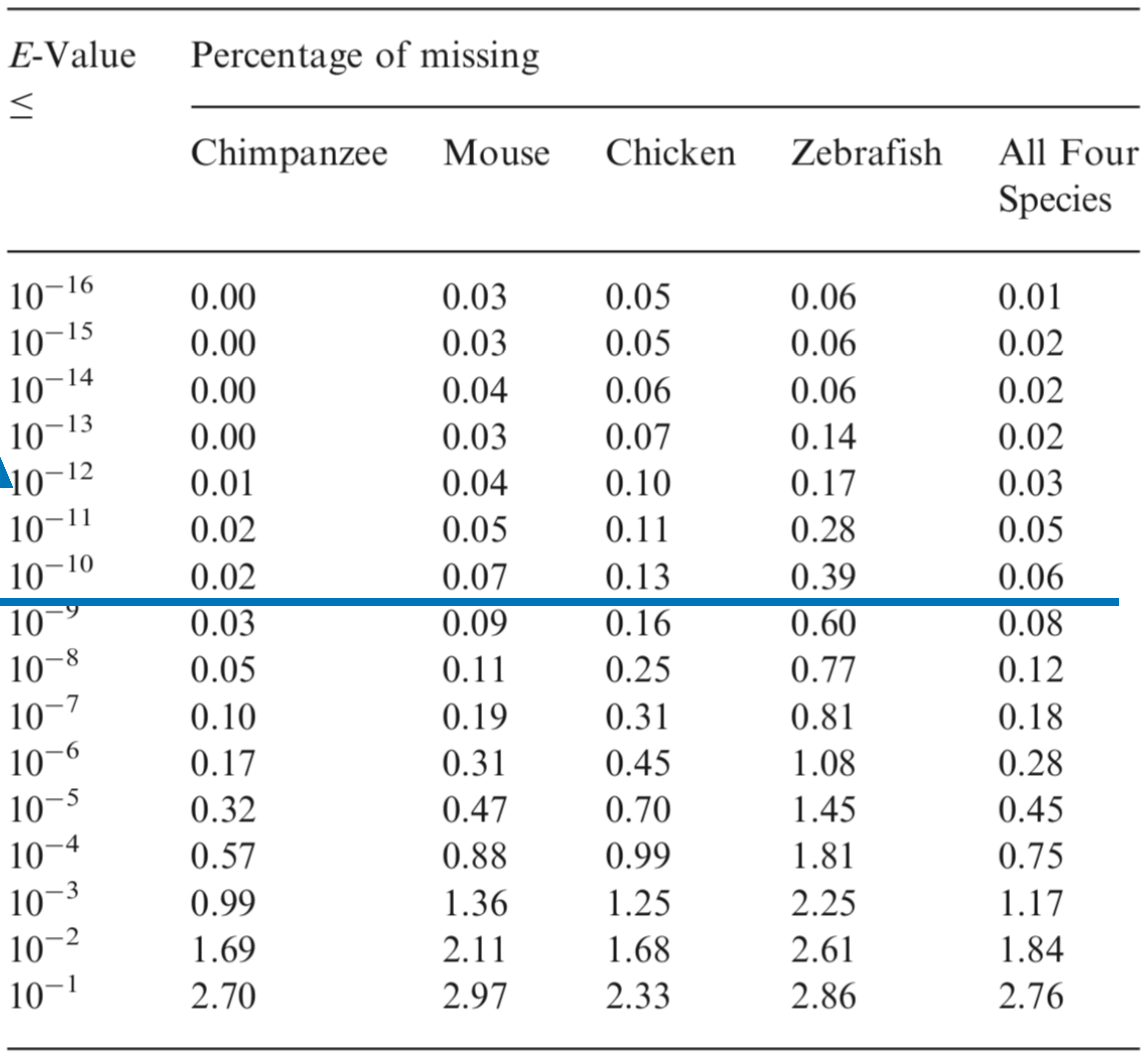
8,000 queries

- 2,000 from each of 4 species: chimpanzee, mouse, chicken, zebrafish
- length ranged from 170-19,000 bases (average of 2,700)

Usually considered a "significant match"

Aligned to the human genome using BLAST

Baseline is an exact search algorithm called BWT-SW



E-Value $\leq$	Percentage of missing				
	Chimpanzee	Mouse	Chicken	Zebrafish	All Four Species
10^{-16}	0.00	0.03	0.05	0.06	0.01
10^{-15}	0.00	0.03	0.05	0.06	0.02
10^{-14}	0.00	0.04	0.06	0.06	0.02
10^{-13}	0.00	0.03	0.07	0.14	0.02
10^{-12}	0.01	0.04	0.10	0.17	0.03
10^{-11}	0.02	0.05	0.11	0.28	0.05
10^{-10}	0.02	0.07	0.13	0.39	0.06
10^{-9}	0.03	0.09	0.16	0.60	0.08
10^{-8}	0.05	0.11	0.25	0.77	0.12
10^{-7}	0.10	0.19	0.31	0.81	0.18
10^{-6}	0.17	0.31	0.45	1.08	0.28
10^{-5}	0.32	0.47	0.70	1.45	0.45
10^{-4}	0.57	0.88	0.99	1.81	0.75
10^{-3}	0.99	1.36	1.25	2.25	1.17
10^{-2}	1.69	2.11	1.68	2.61	1.84
10^{-1}	2.70	2.97	2.33	2.86	2.76

[T. W. Lam, et al. **Compressed indexing and local alignment of DNA**, *Bioinformatics*, 24(6), March 2008, Pgs. 791–797]

Protein Replacement Matricies

To now we have been talking about a "score" between two sequences without gaps with the penalties in the abstract.

Most people will use one of the *PAM* (percent accepted mutations)*,* *BLOSUM* (blocks substitution matrix)*, or VTML* series of replacement (or transition) matrices.

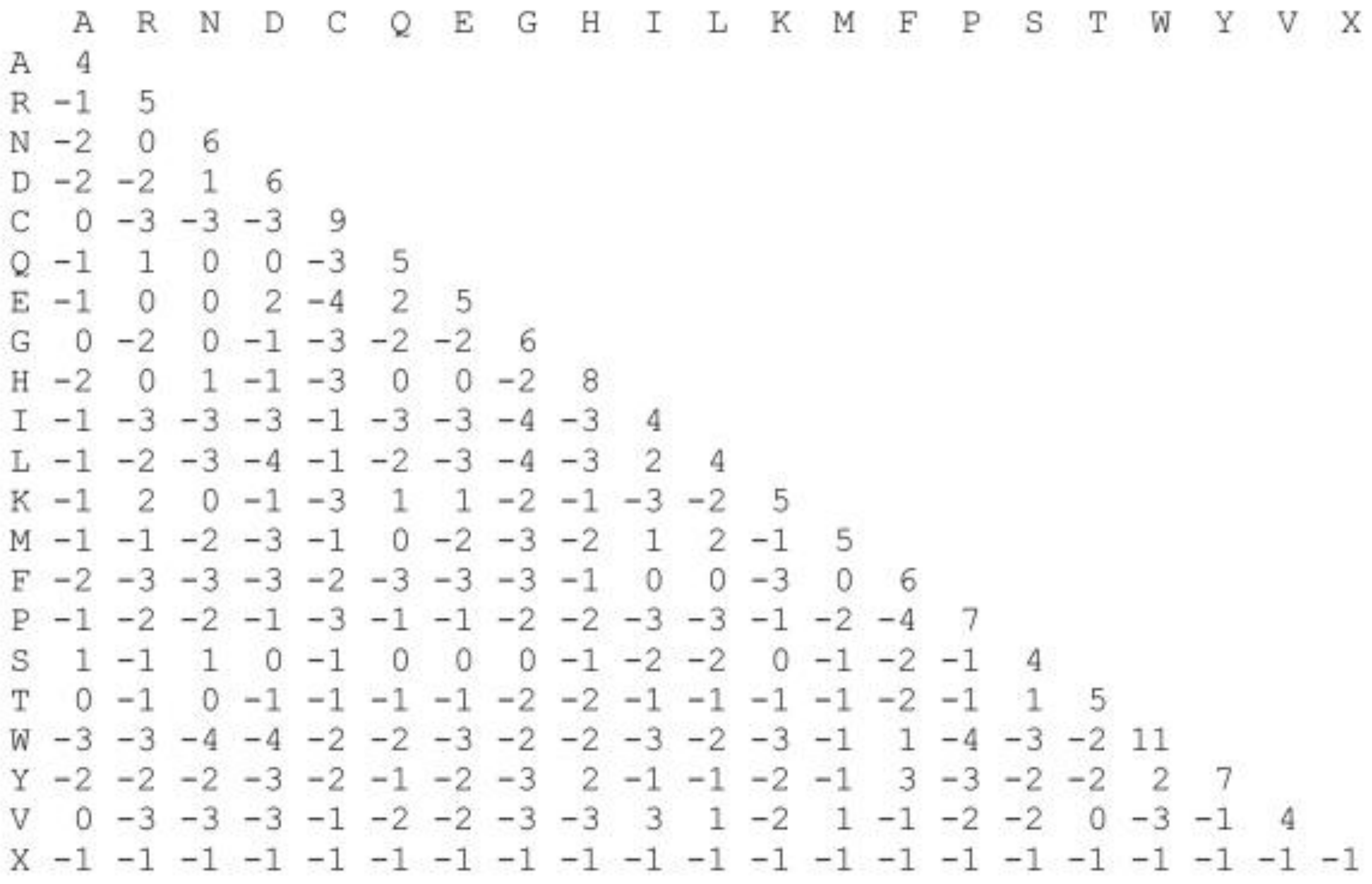
All 3 are based on statistics from databases of proteins labeled in order to match based on function.

Protein Replacement Matrices

BLOSUM (most popular) published by Henikoff & Henikoff in 1992.

Usually accompanied by a number (i.e. BLOSUM62, on the right) which is the percent identity of the pairs of sequences used for training.

The actual value is a log-odds value of the replacements from a large set of examples.



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

Protein Replacement Matrices

PAM and VTML also have numbers associated, but the allowable amount of time between sequences[1], so its inversely correlated with the BLOSUM number.

Somewhat equivalent matrices (by entropy)

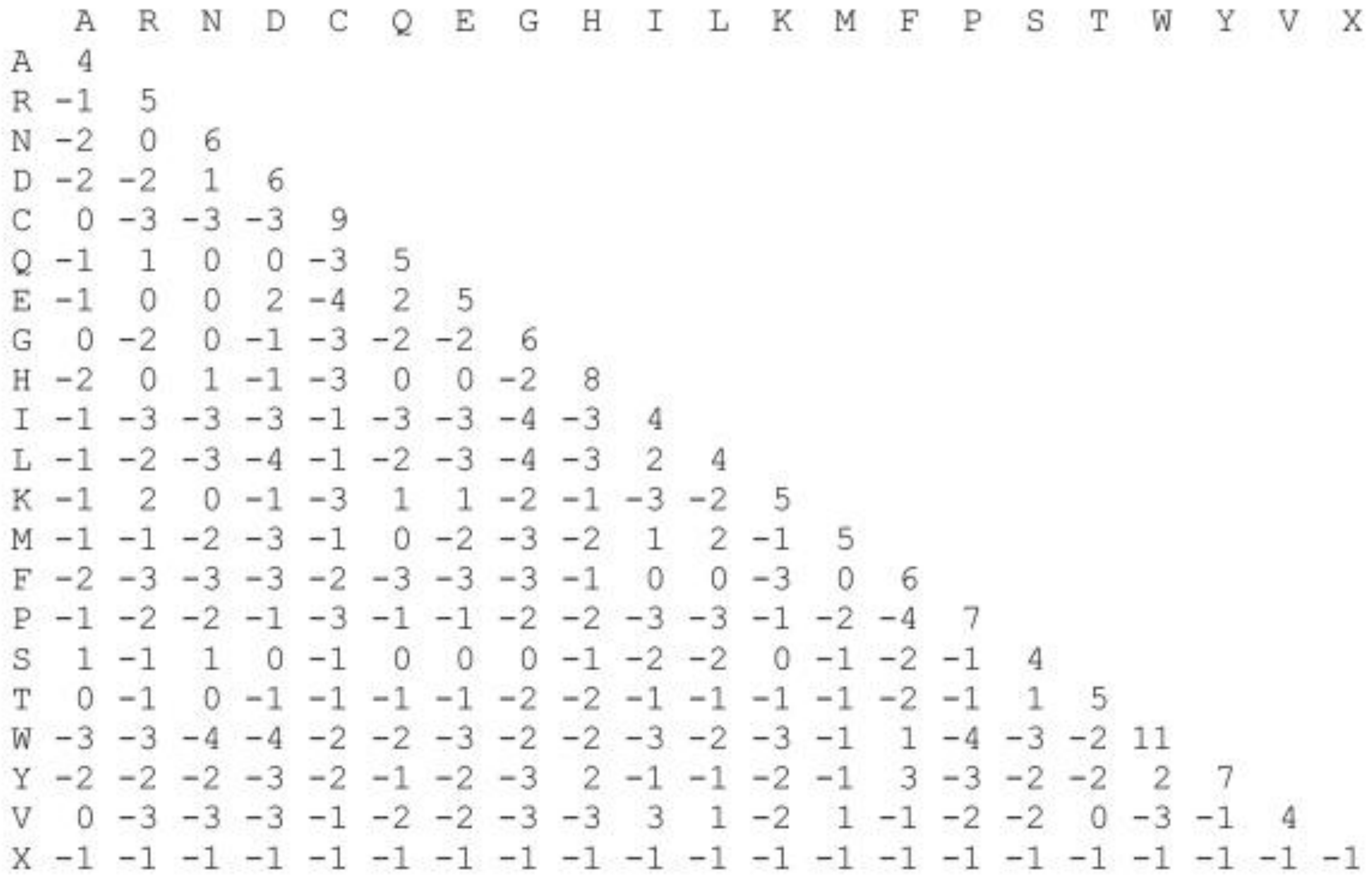
BLOSUM90	PAM100	VTML100
BLOSUM80	PAM120	VTML120
BLOSUM60	PAM160	VTML160
BLOSUM52	PAM200	VTML200
BLOSUM45	PAM250	VTML250

[1]Time is measured relative to the evolutionary time it takes to introduce one change per 100 amino acids.

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.

`V` **`P`** **`N`** **`M`**

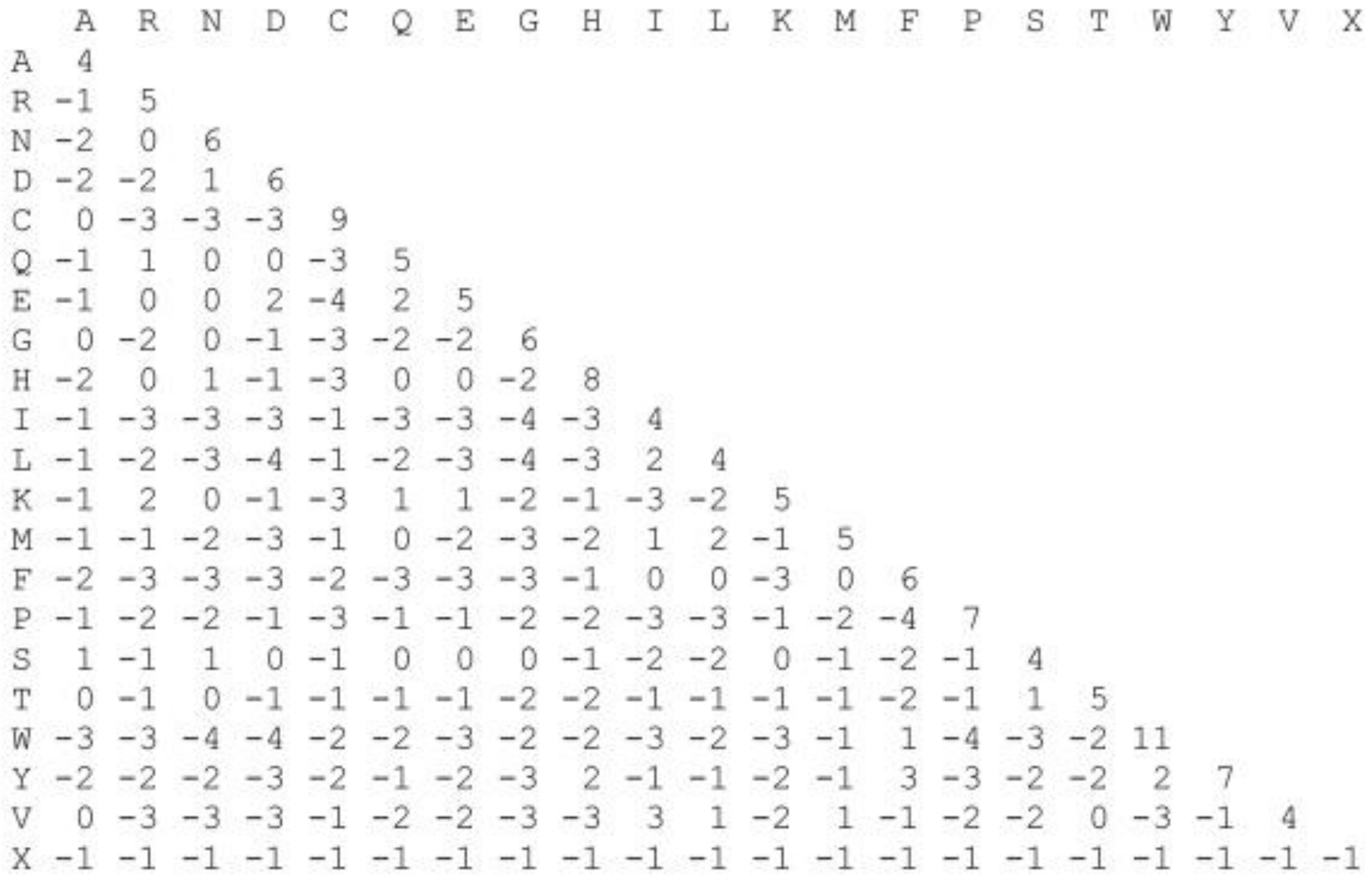


	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	X
A	4																				
R	-1	5																			
N	-2	0	6																		
D	-2	-2	1	6																	
C	0	-3	-3	-3	9																
Q	-1	1	0	0	-3	5															
E	-1	0	0	2	-4	2	5														
G	0	-2	0	-1	-3	-2	-2	6													
H	-2	0	1	-1	-3	0	0	-2	8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

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.

V	**P**	**N**	**M**
V	P		

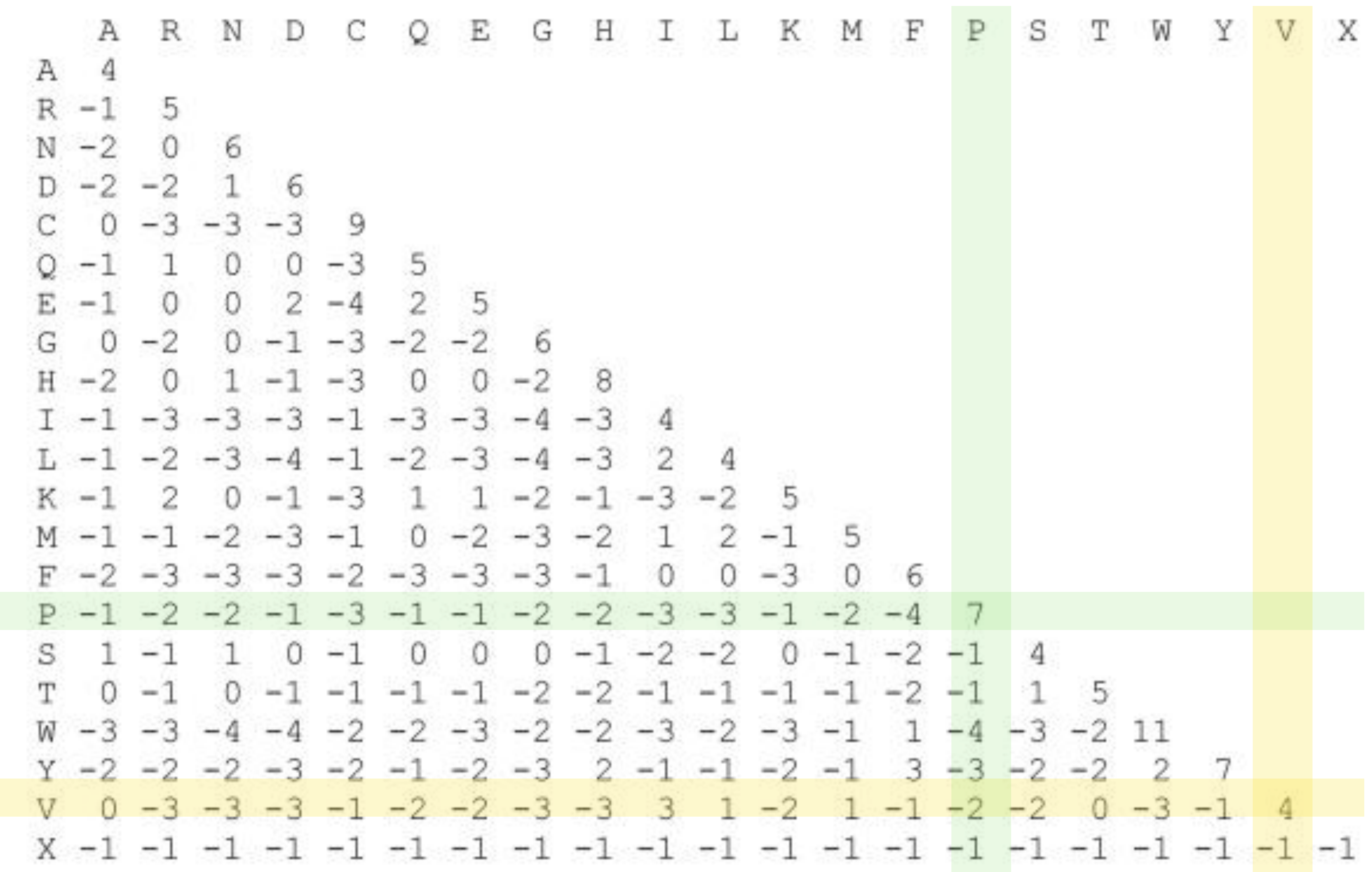


	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	X
A	4																				
R	-1	5																			
N	-2	0	6																		
D	-2	-2	1	6																	
C	0	-3	-3	-3	9																
Q	-1	1	0	0	-3	5															
E	-1	0	0	2	-4	2	5														
G	0	-2	0	-1	-3	-2	-2	6													
H	-2	0	1	-1	-3	0	0	-2	8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

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.

V	**P**	**N**	**M**
V	P		

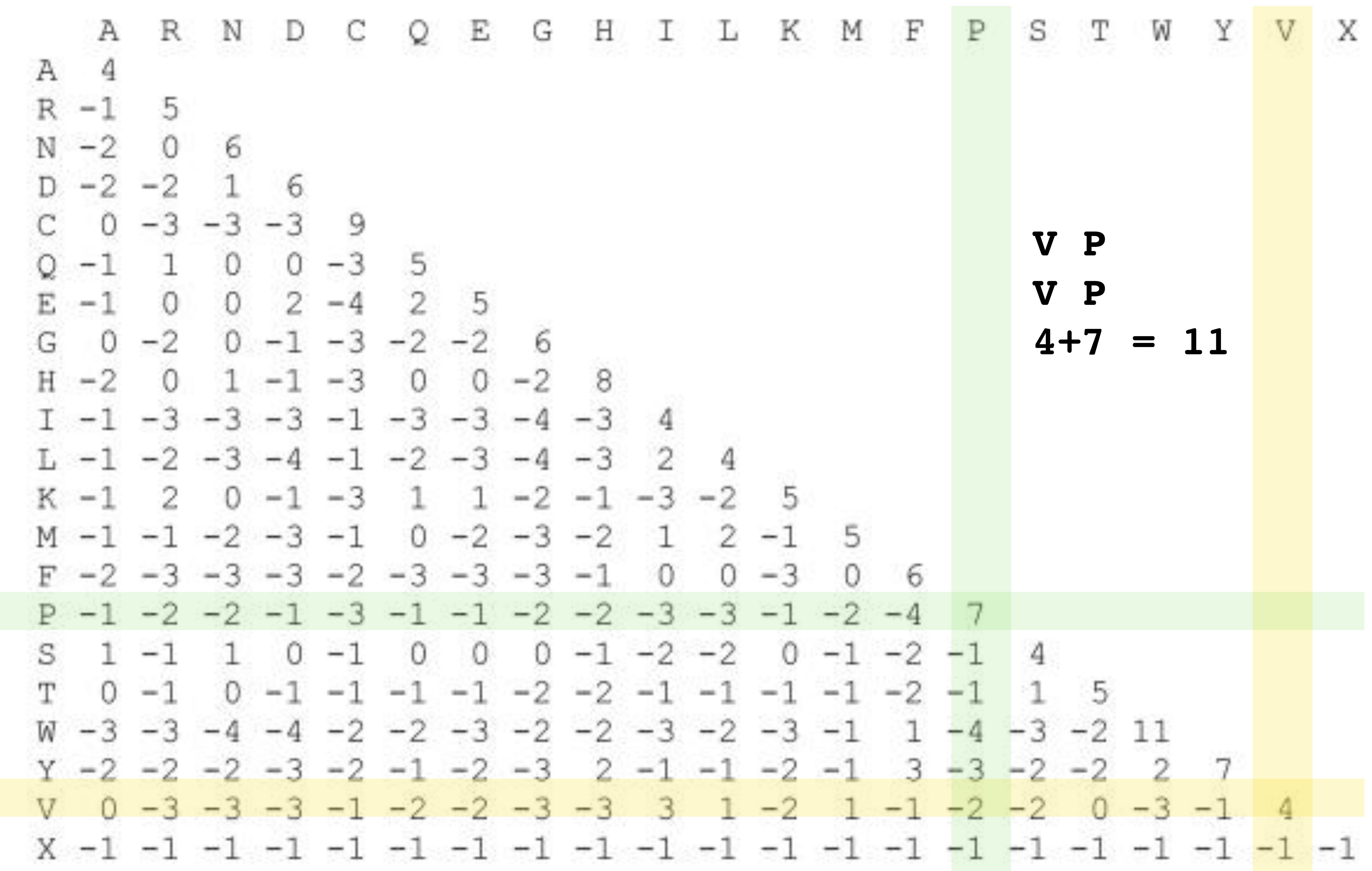


	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	X
A	4																				
R	-1	5																			
N	-2	0	6																		
D	-2	-2	1	6																	
C	0	-3	-3	-3	9																
Q	-1	1	0	0	-3	5															
E	-1	0	0	2	-4	2	5														
G	0	-2	0	-1	-3	-2	-2	6													
H	-2	0	1	-1	-3	0	0	-2	8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

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.

V	**P**	**N**	**M**
V	P		



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

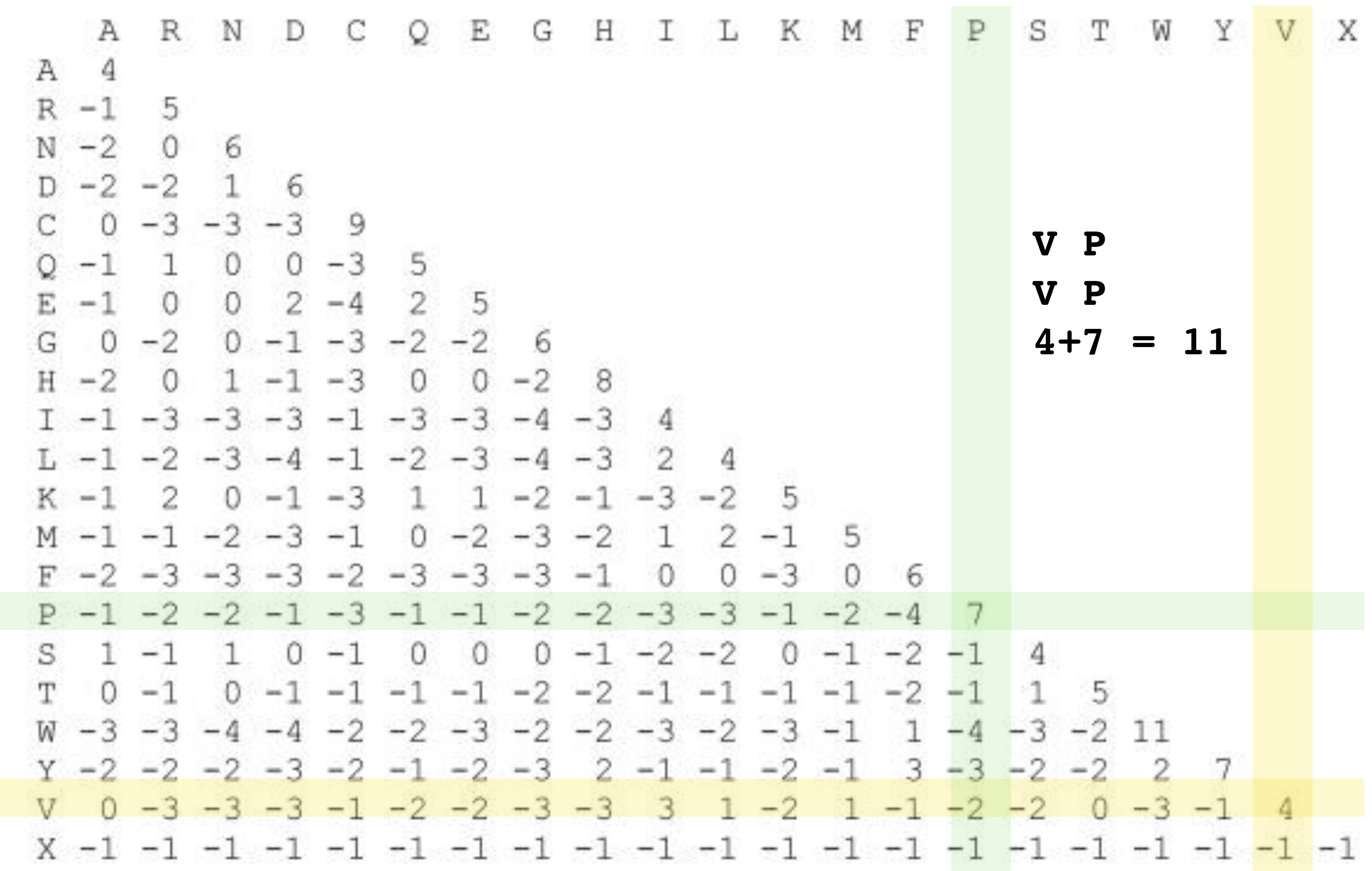
```
V P
V P
4+7 = 11
```

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.

V	**P**	**N**	**M**
V	P		

VP

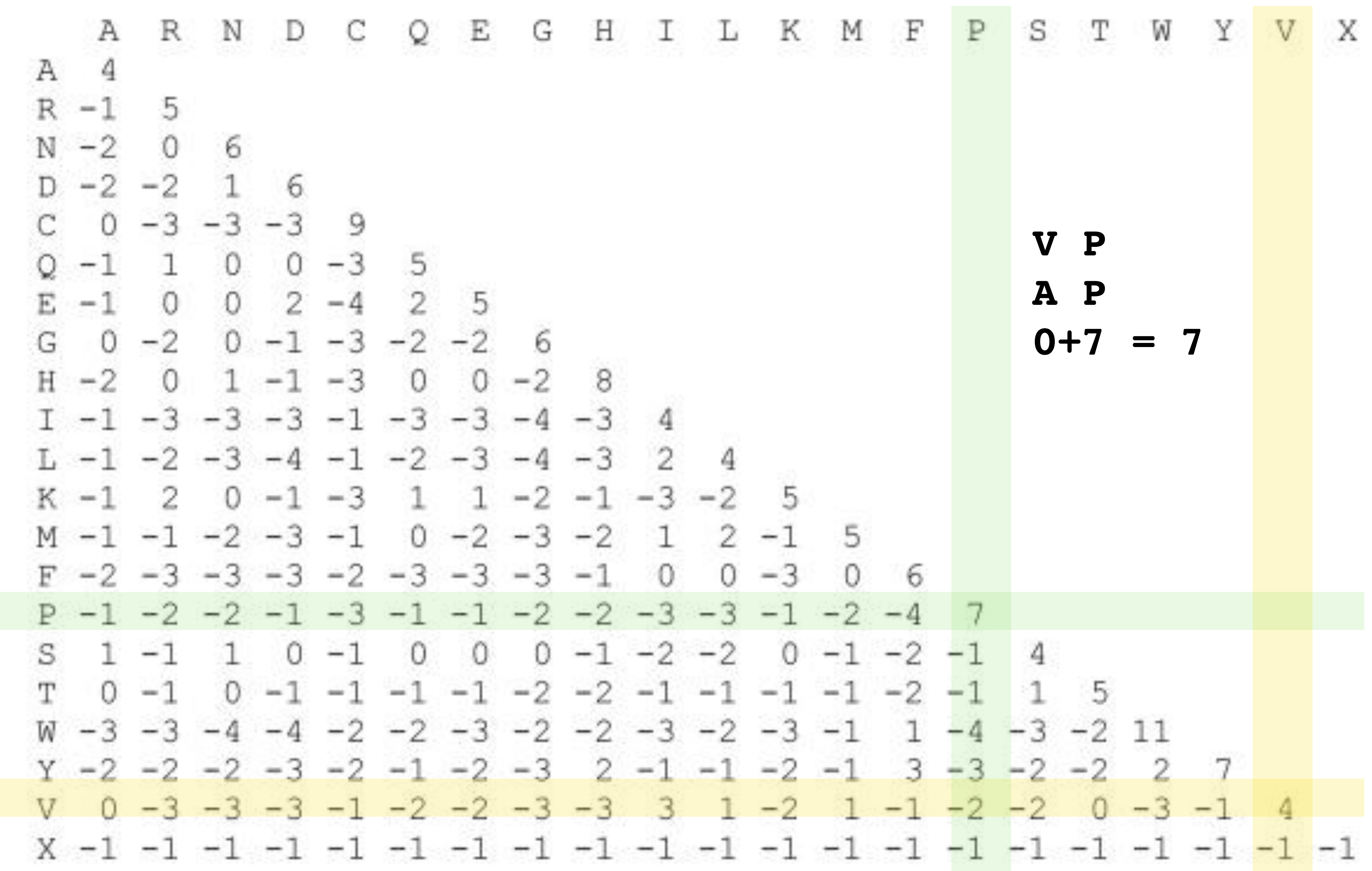


	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	X
A	4																				
R	-1	5																			
N	-2	0	6																		
D	-2	-2	1	6																	
C	0	-3	-3	-3	9																
Q	-1	1	0	0	-3	5															
E	-1	0	0	2	-4	2	5														
G	0	-2	0	-1	-3	-2	-2	6													
H	-2	0	1	-1	-3	0	0	-2	8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

```
V P
V P
4+7 = 11
```

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.



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

```
V P
A P
0+7 = 7
```

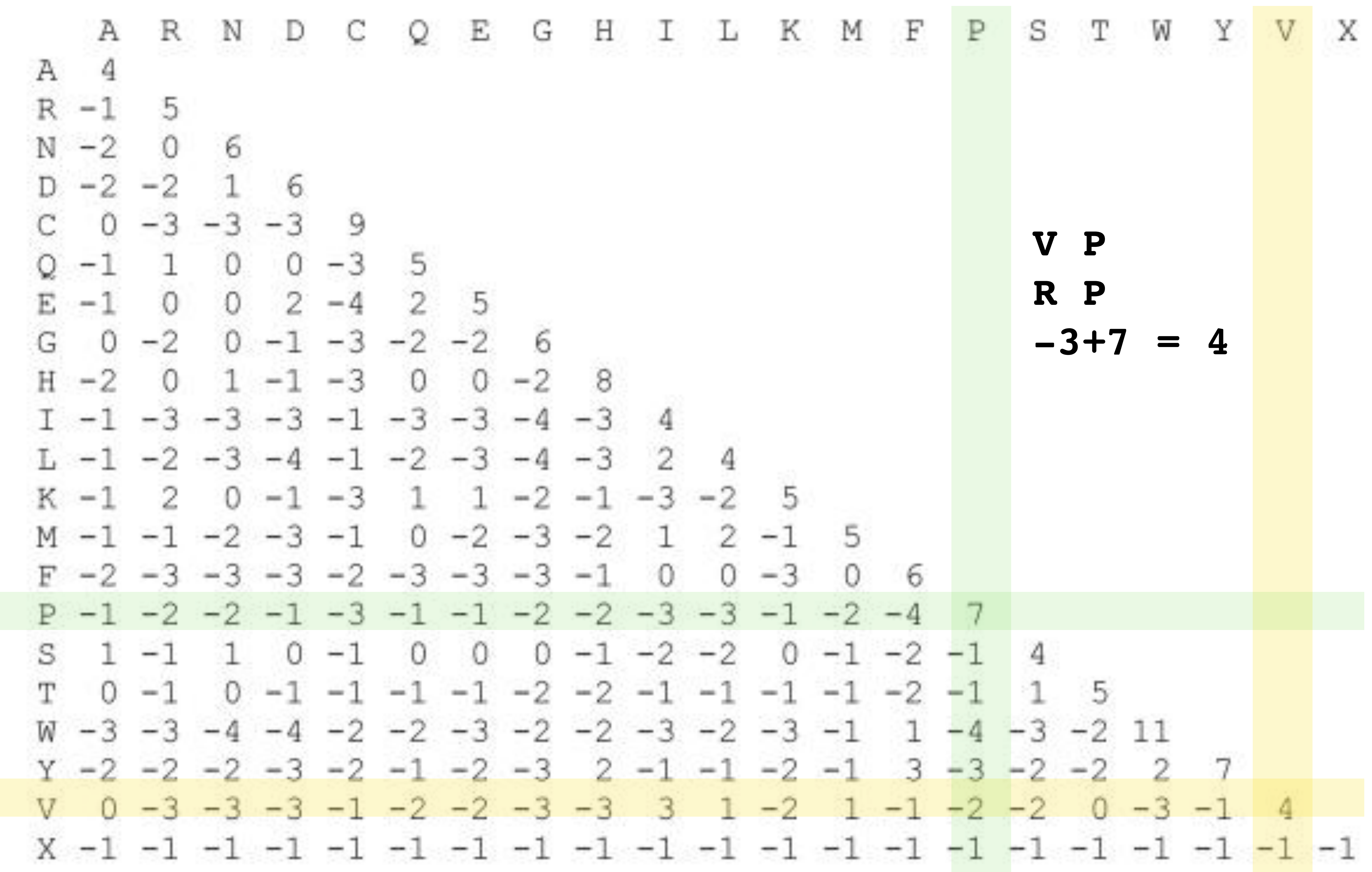
```
V        P        N        M
V        P
```

```
VP
```

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.

V	**P**	**N**	**M**
V	P		



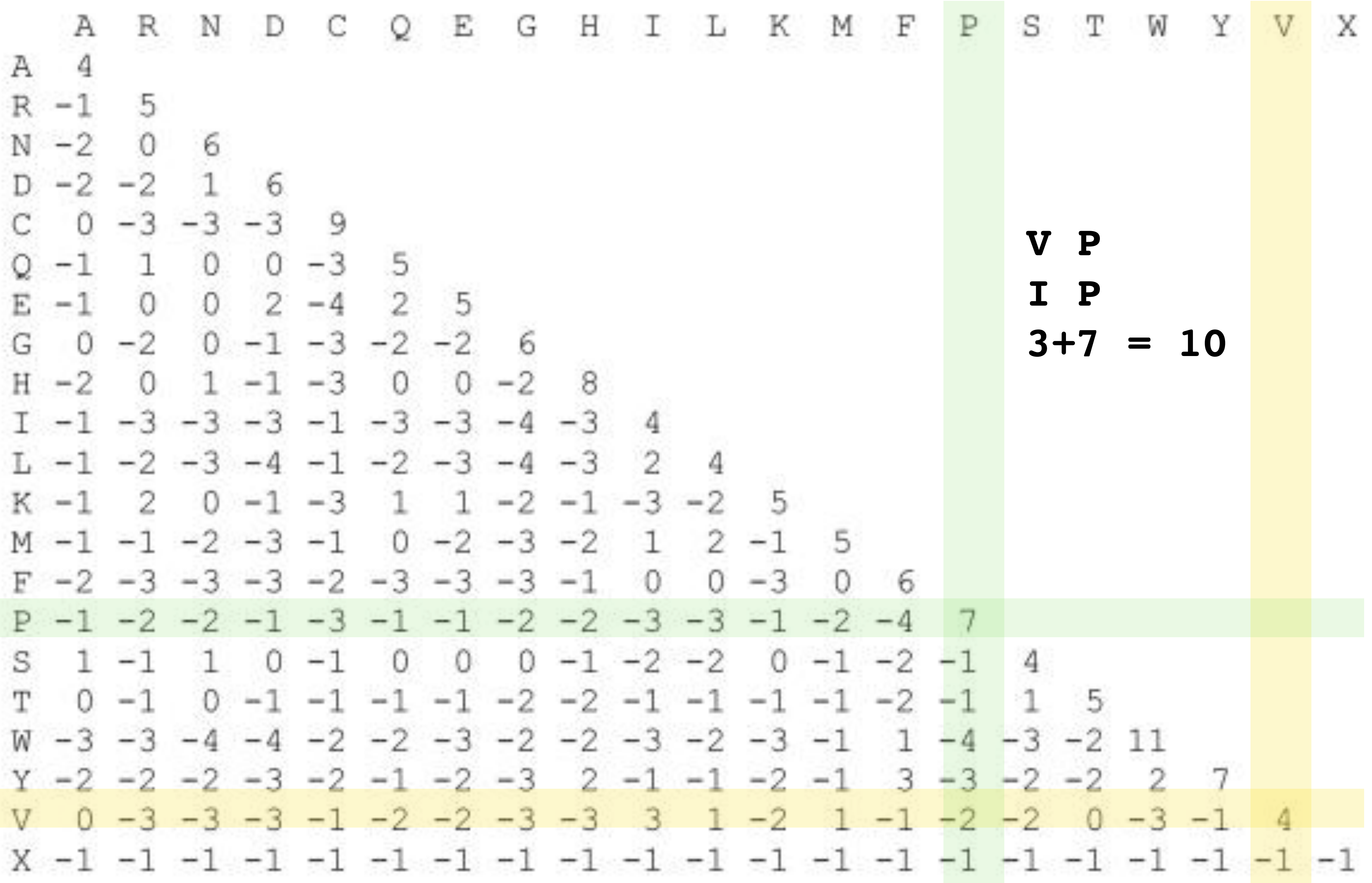
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	X
A	4																				
R	-1	5																			
N	-2	0	6																		
D	-2	-2	1	6																	
C	0	-3	-3	-3	9																
Q	-1	1	0	0	-3	5															
E	-1	0	0	2	-4	2	5														
G	0	-2	0	-1	-3	-2	-2	6													
H	-2	0	1	-1	-3	0	0	-2	8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

```
V P
R P
-3+7 = 4
```

VP

Exercise

Given the sequence **VPNM**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.



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

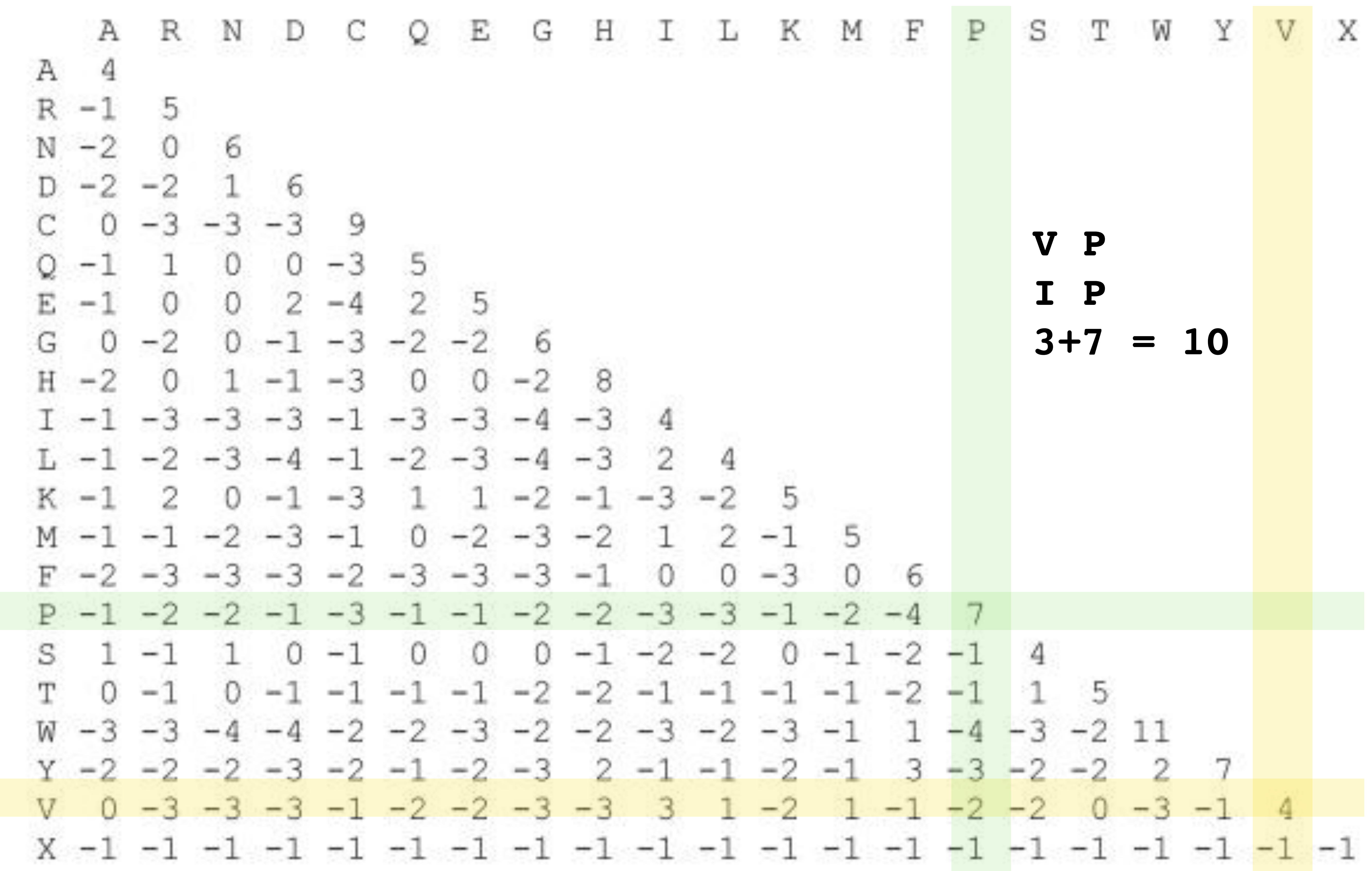
```
V P
I P
3+7 = 10
```

```
V          P          N          M
V          P
```

```
VP
```

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.



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

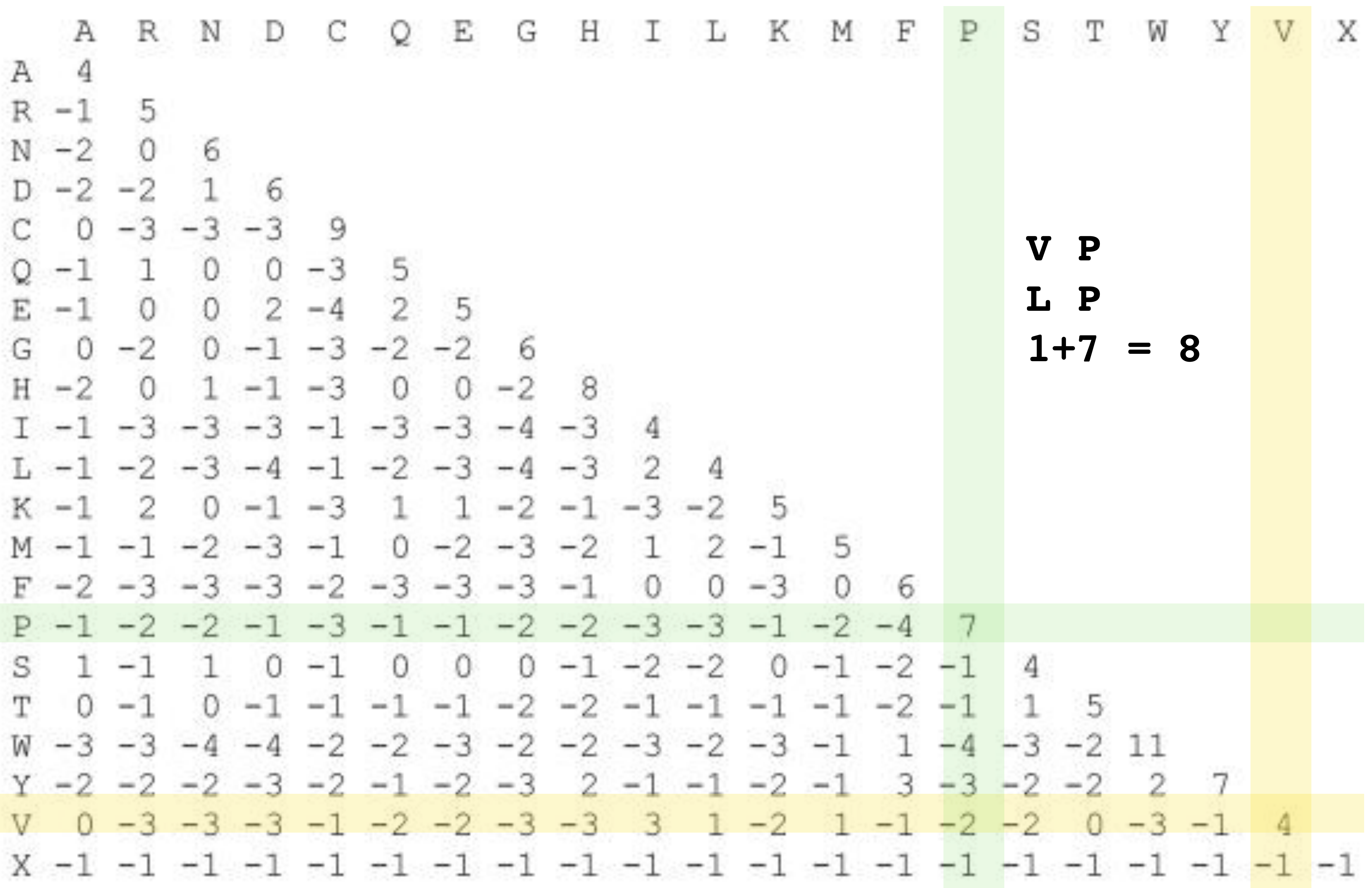
V P
I P
3+7 = 10

V	**P**	**N**	**M**
V	P		

VP IP

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.



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

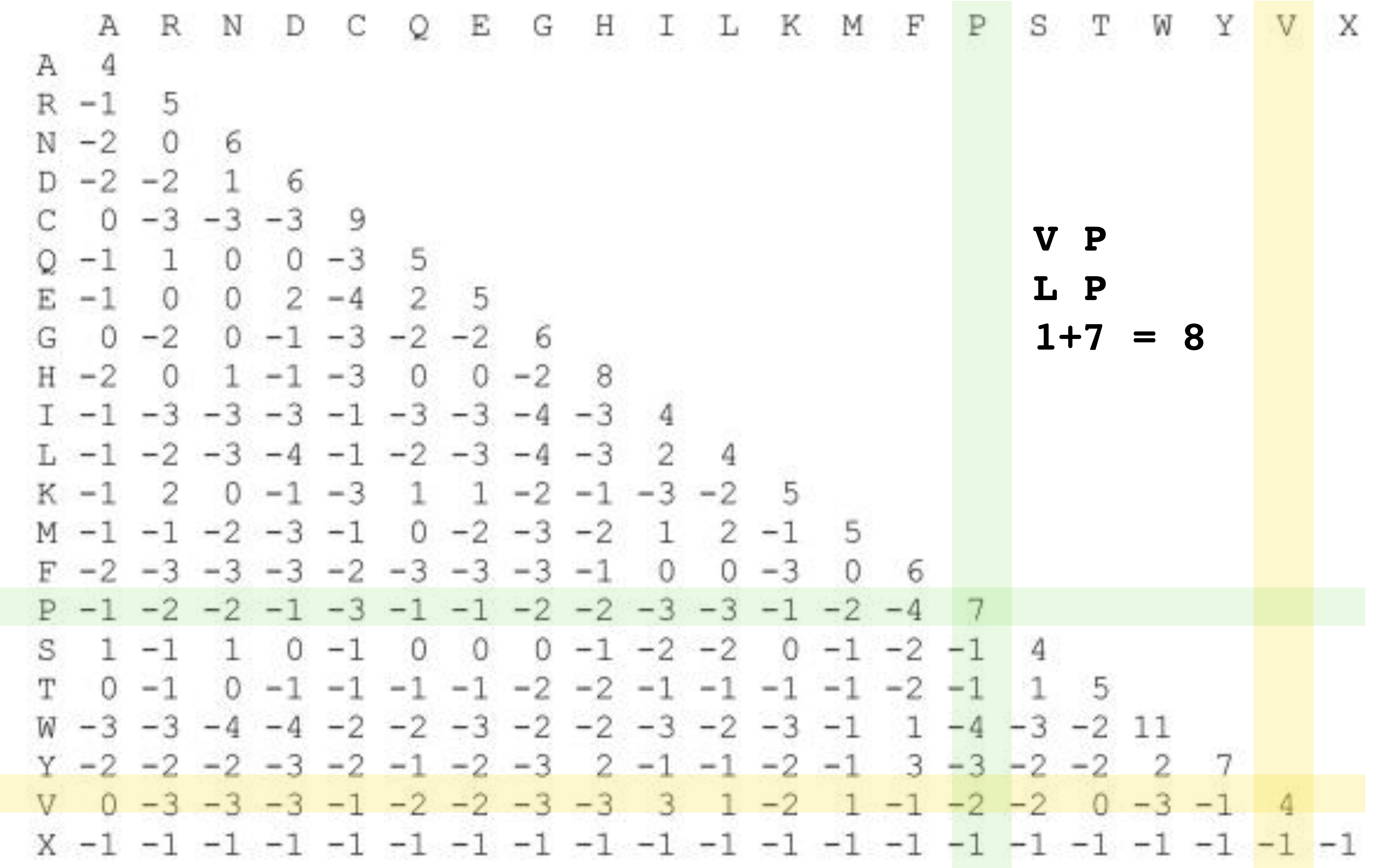
```
V P
L P
1+7 = 8
```

V	**P**	**N**	**M**
`V`	`P`		

```
VP          IP
```

Exercise

Given the sequence **VPNM**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.



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

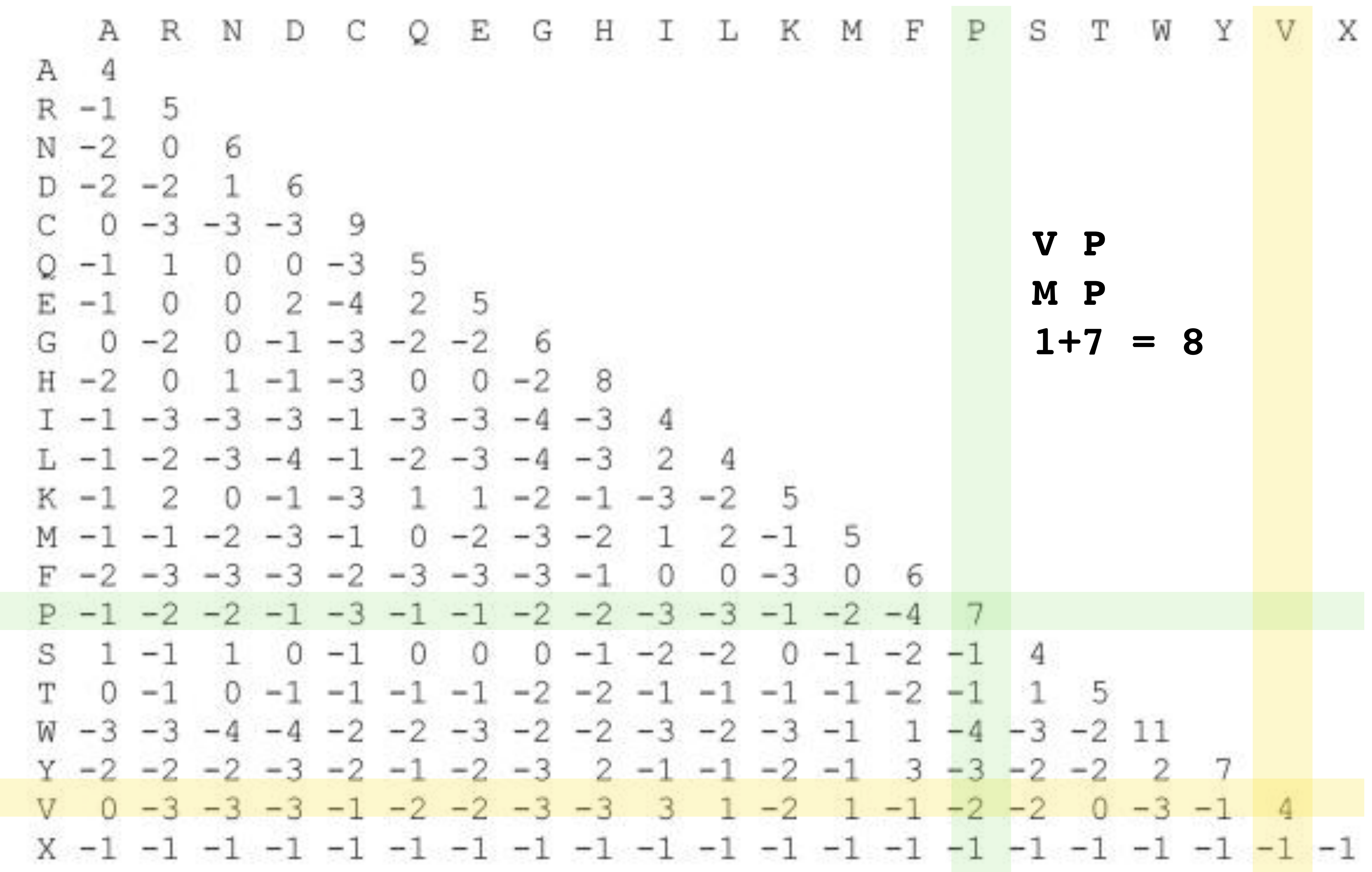
V P
L P
1+7 = 8

V	**P**	**N**	**M**
V	P		

VP IP LP

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.



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

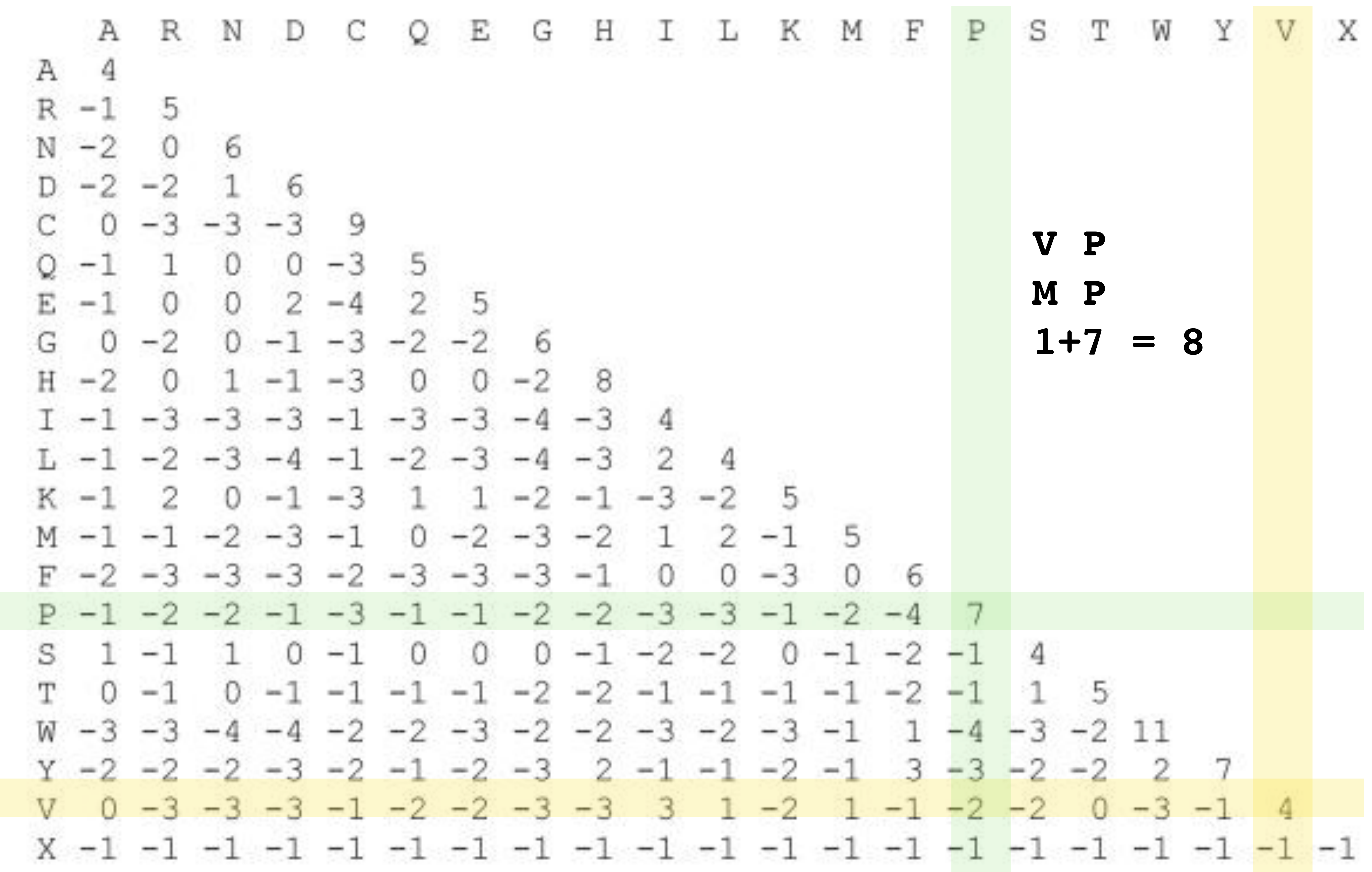
```
V P
M P
1+7 = 8
```

V	**P**	**N**	**M**
V	P		

VP IP LP

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.



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

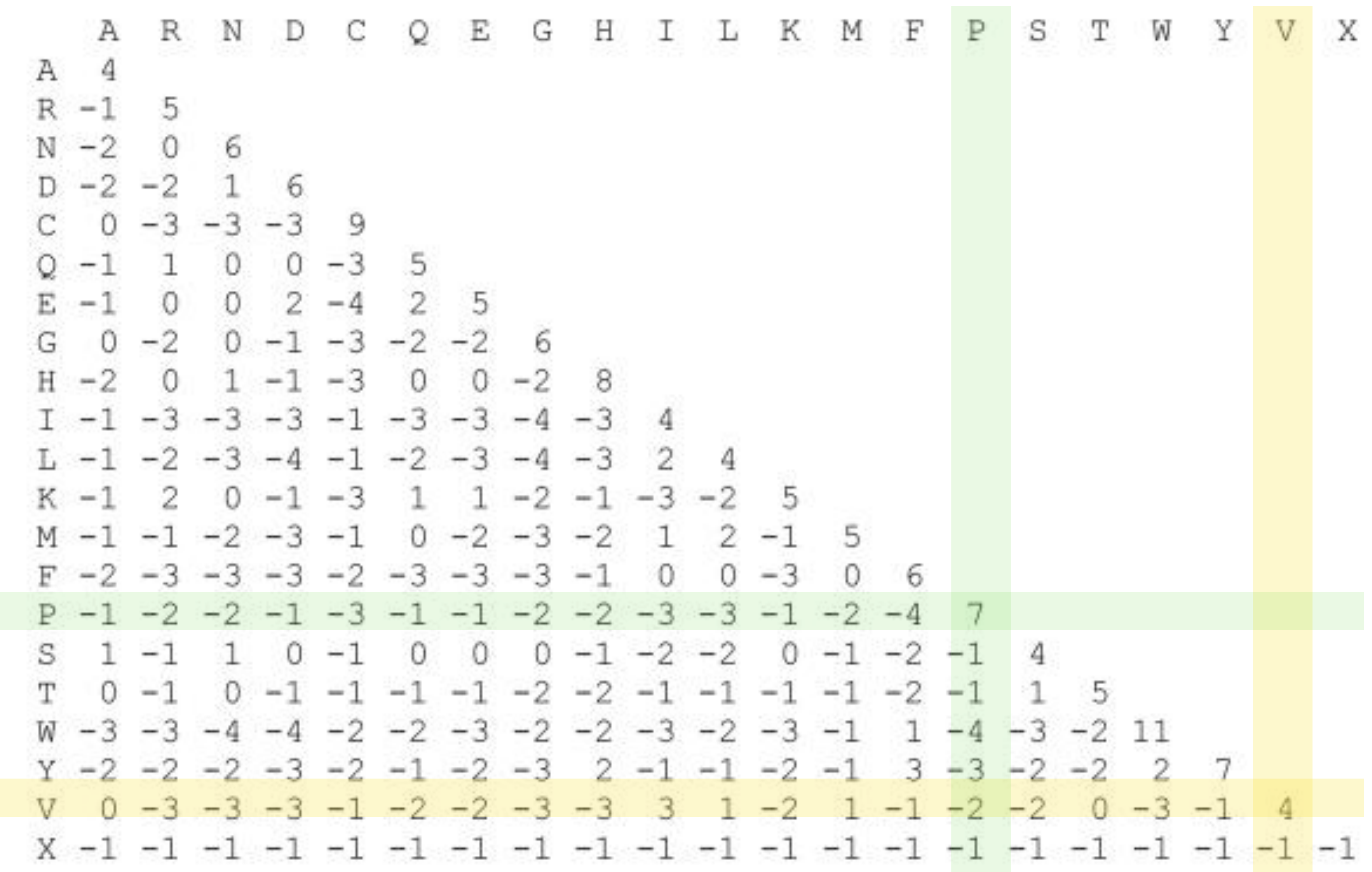
`V P`
`M P`
`1+7 = 8`

V	**P**	**N**	**M**
`V`	`P`		

`VP` `IP` `LP` `MP`

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.



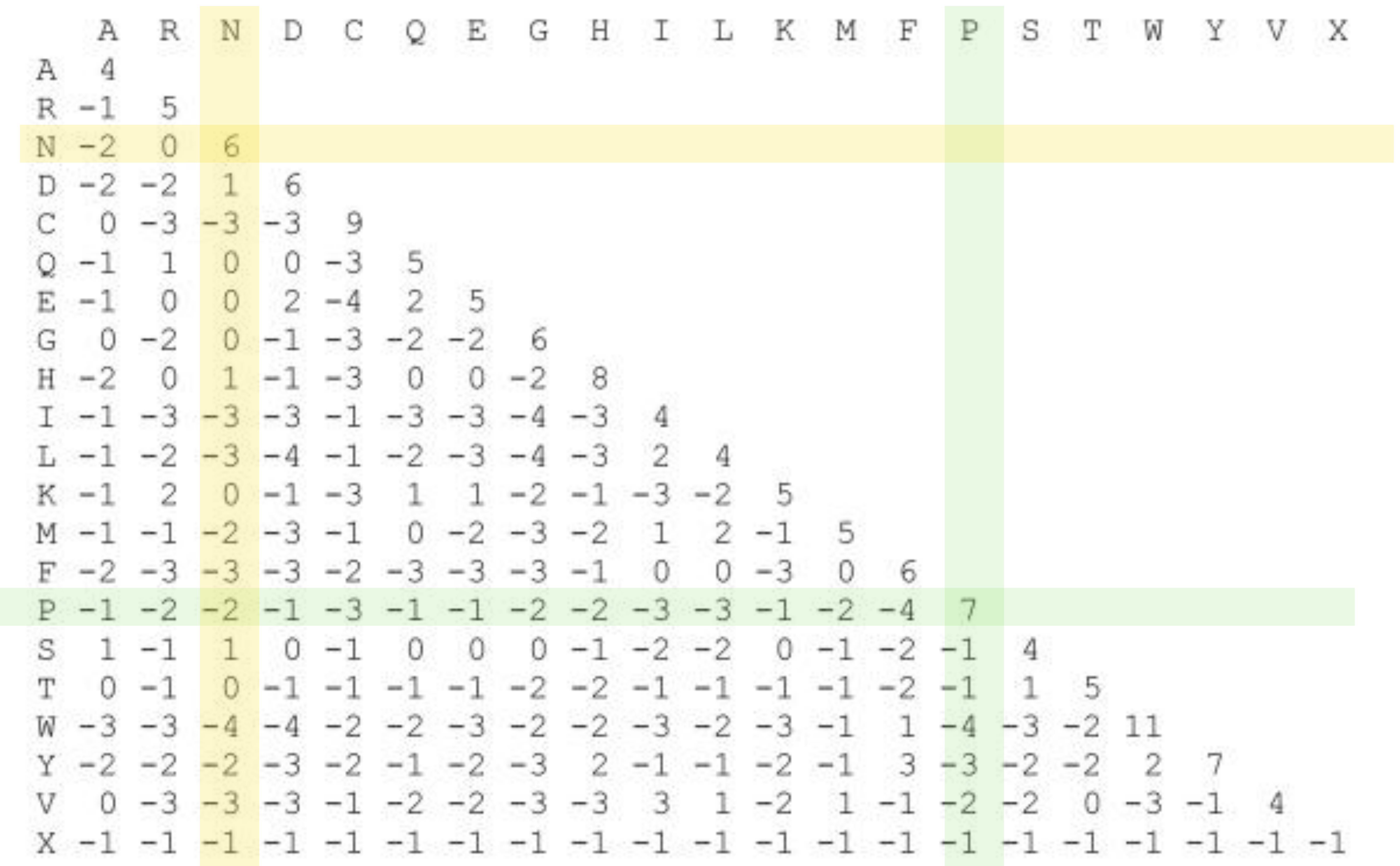
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	X
A	4																				
R	-1	5																			
N	-2	0	6																		
D	-2	-2	1	6																	
C	0	-3	-3	-3	9																
Q	-1	1	0	0	-3	5															
E	-1	0	0	2	-4	2	5														
G	0	-2	0	-1	-3	-2	-2	6													
H	-2	0	1	-1	-3	0	0	-2	8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

V	**P**	**N**	**M**
V	P		

VP IP LP MP

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.



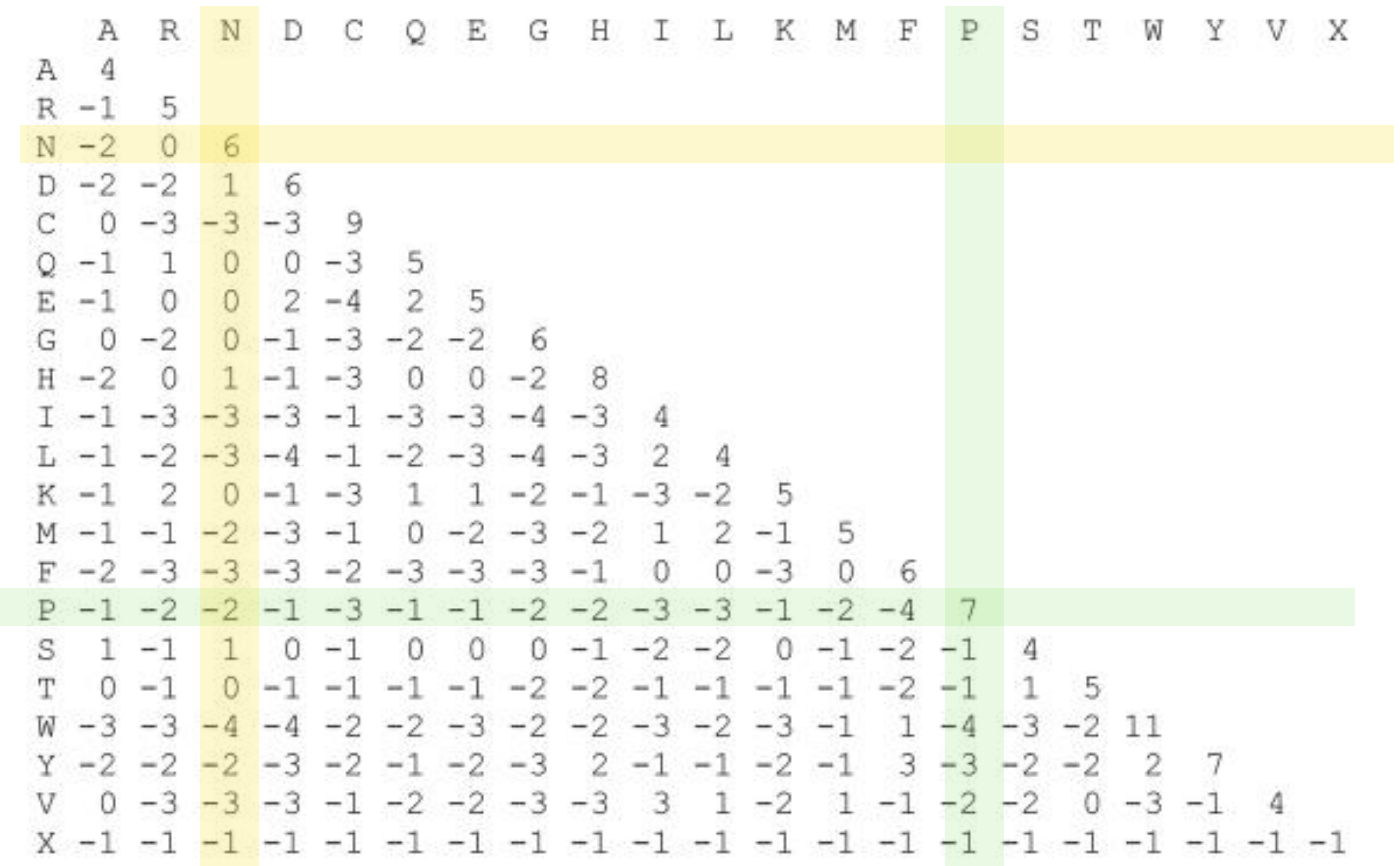
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	X
A	4																				
R	-1	5																			
N	-2	0	6																		
D	-2	-2	1	6																	
C	0	-3	-3	-3	9																
Q	-1	1	0	0	-3	5															
E	-1	0	0	2	-4	2	5														
G	0	-2	0	-1	-3	-2	-2	6													
H	-2	0	1	-1	-3	0	0	-2	8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

V	**P**	**N**	**M**
V	P		
	P	N	

```
VP        IP        LP        MP
```

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.



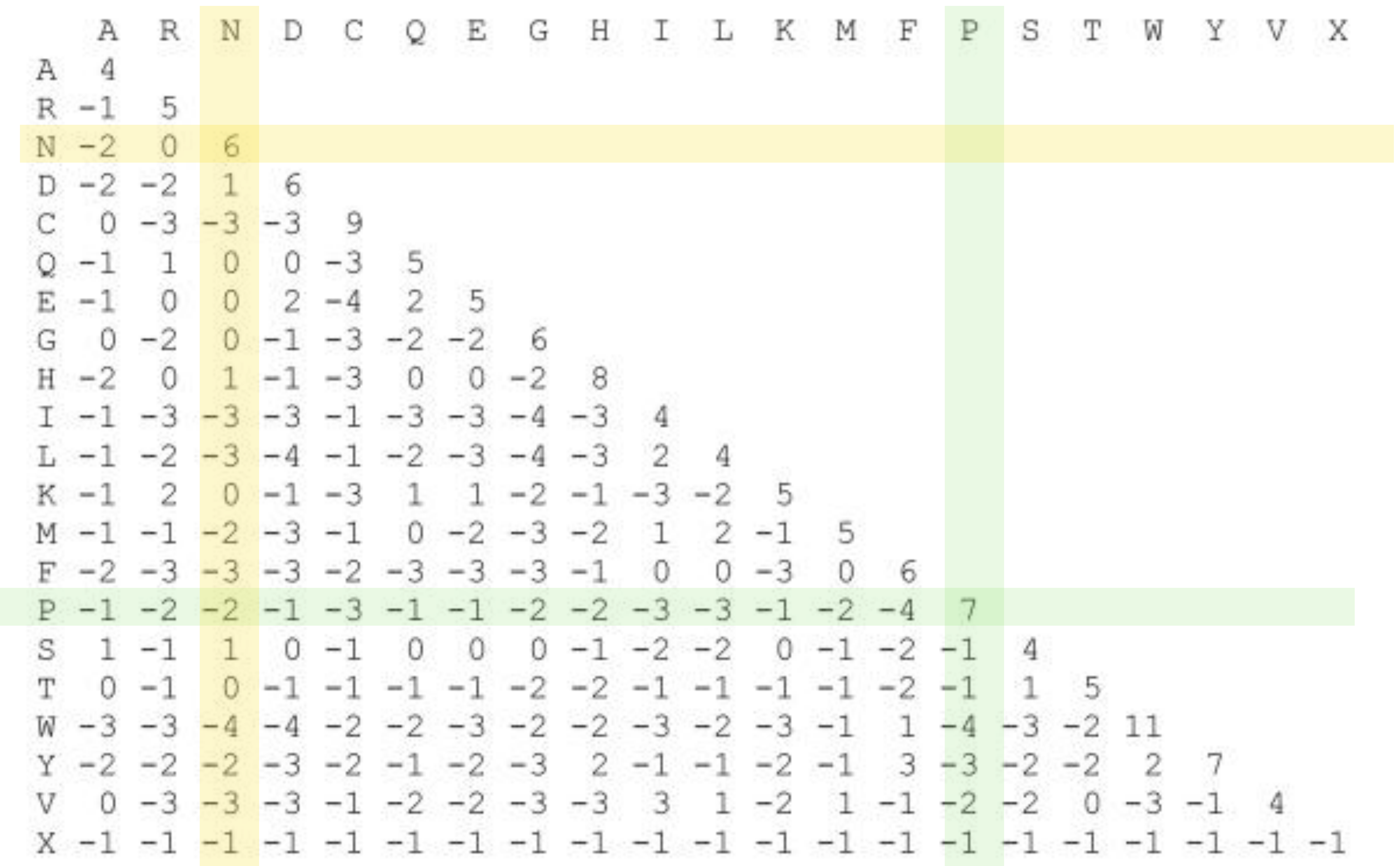
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	X
A	4																				
R	-1	5																			
N	-2	0	6																		
D	-2	-2	1	6																	
C	0	-3	-3	-3	9																
Q	-1	1	0	0	-3	5															
E	-1	0	0	2	-4	2	5														
G	0	-2	0	-1	-3	-2	-2	6													
H	-2	0	1	-1	-3	0	0	-2	8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

V	**P**	**N**	**M**
V	P		
	P	N	

VP	IP	LP	MP
PN			

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.



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

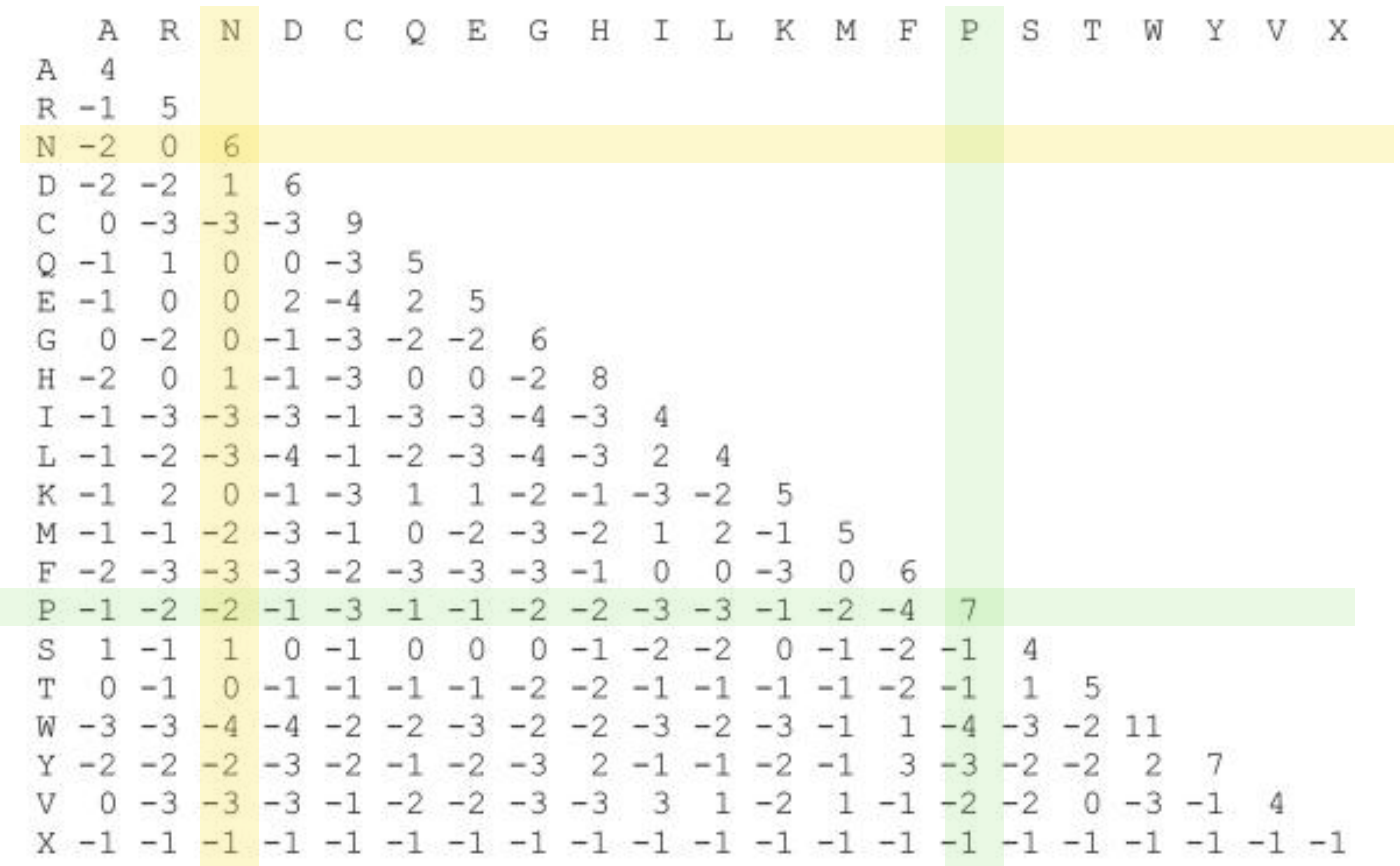
V	**P**	**N**	**M**
V	P		
	P	N	

VP	IP	LP	MP
PN	PD		

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.

V	**P**	**N**	**M**
V	P		
	P	N	

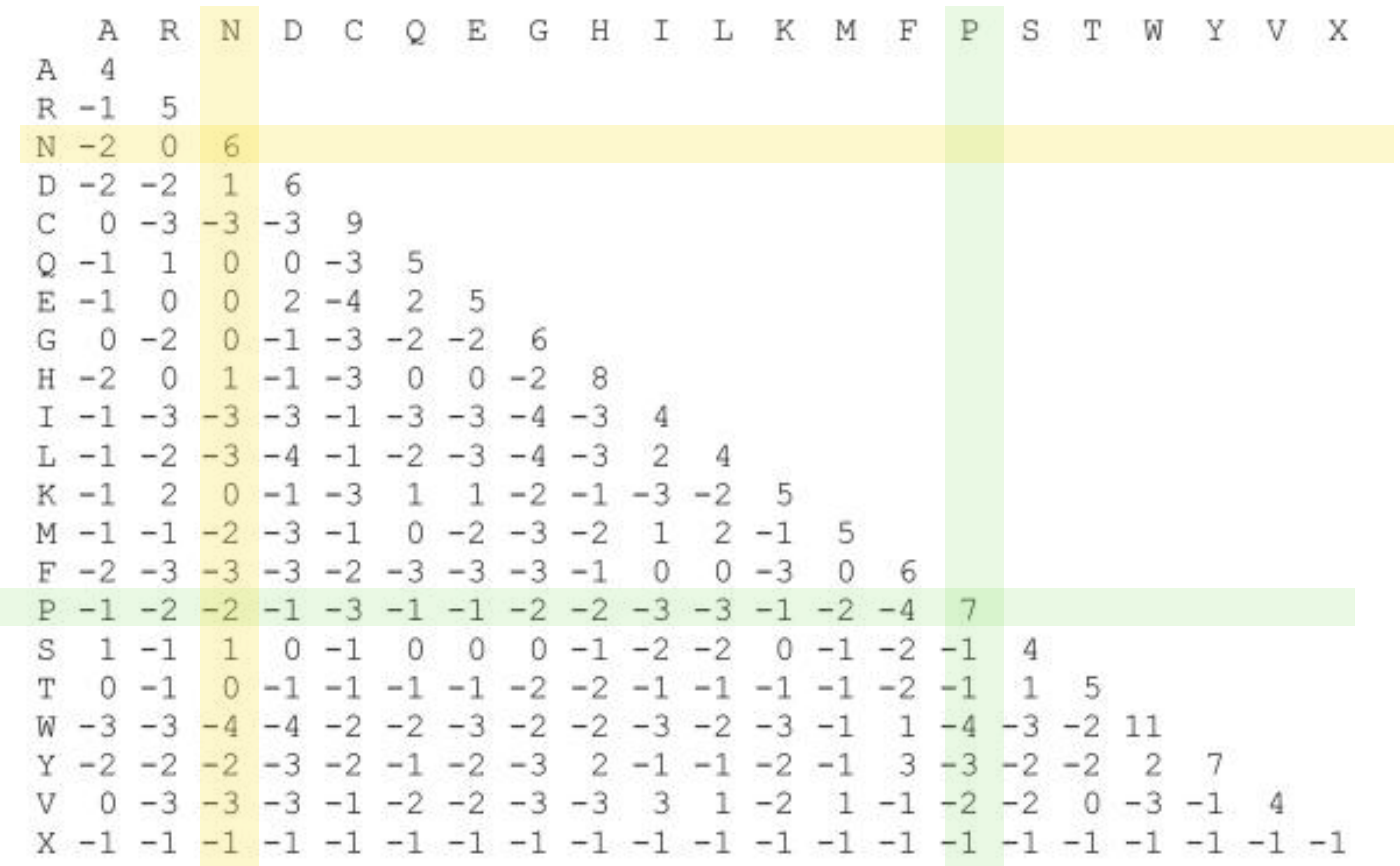


	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	X
A	4																				
R	-1	5																			
N	-2	0	6																		
D	-2	-2	1	6																	
C	0	-3	-3	-3	9																
Q	-1	1	0	0	-3	5															
E	-1	0	0	2	-4	2	5														
G	0	-2	0	-1	-3	-2	-2	6													
H	-2	0	1	-1	-3	0	0	-2	8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

VP	IP	LP	MP
PN	PD	PH	

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.



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

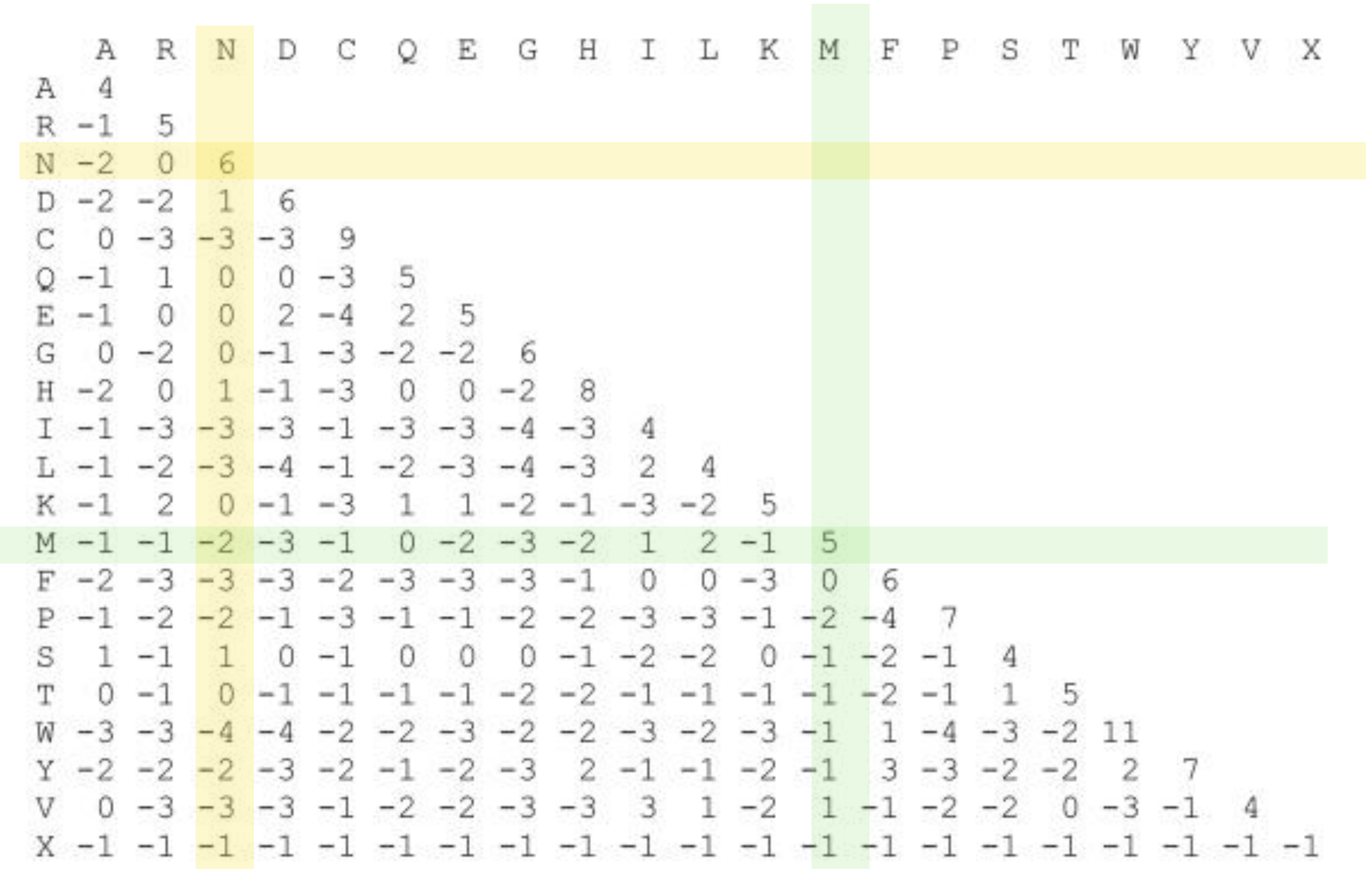
V	**P**	**N**	**M**
V	P		
	P	N	

VP	IP	LP	MP
PN	PD	PH	PS

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.

V	**P**	**N**	**M**
V	P		
	P	N	
		N	M

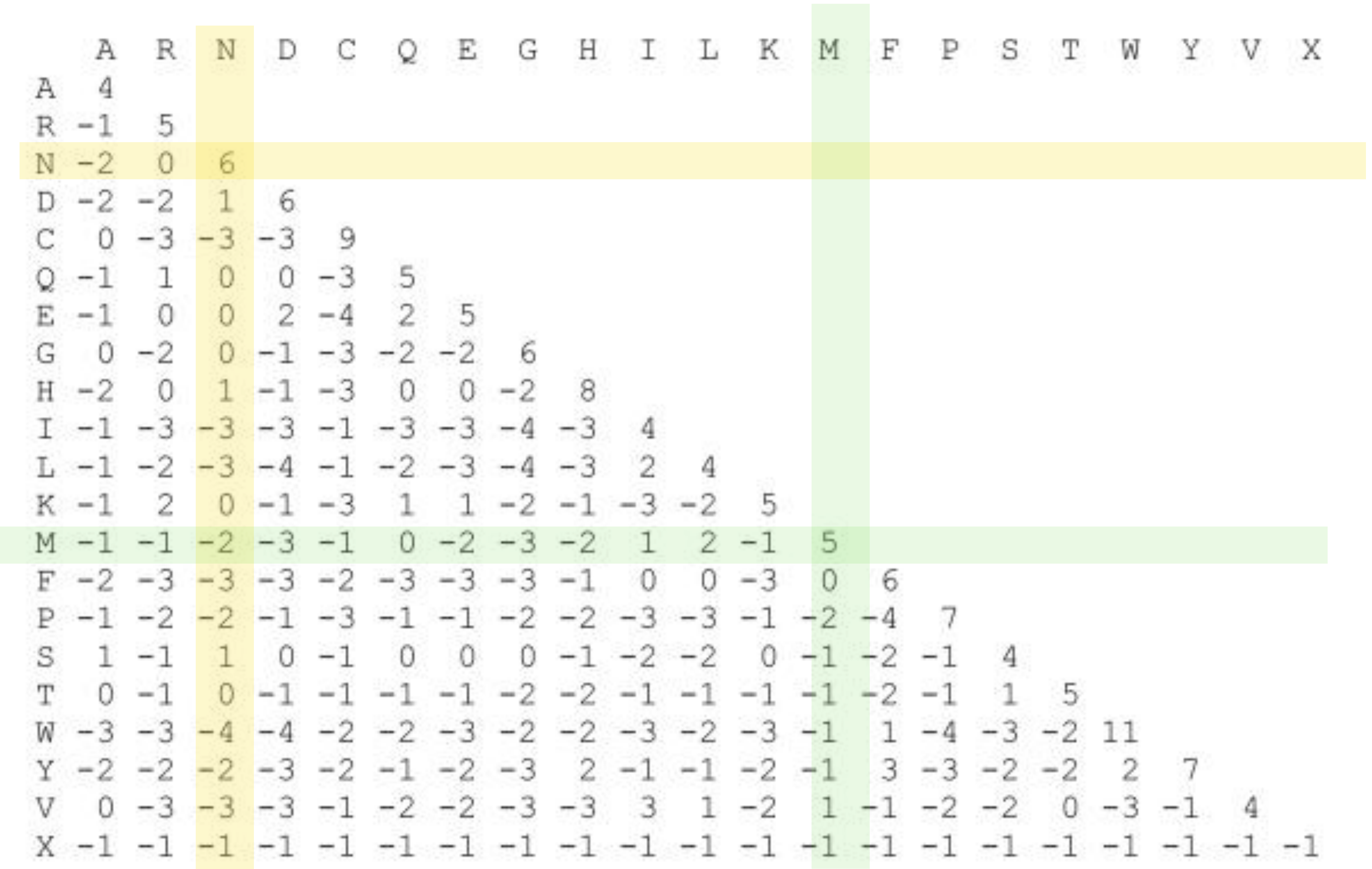


	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	X
A	4																				
R	-1	5																			
N	-2	0	6																		
D	-2	-2	1	6																	
C	0	-3	-3	-3	9																
Q	-1	1	0	0	-3	5															
E	-1	0	0	2	-4	2	5														
G	0	-2	0	-1	-3	-2	-2	6													
H	-2	0	1	-1	-3	0	0	-2	8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

VP	IP	LP	MP
PN	PD	PH	PS

Exercise

Given the sequence **VPNM**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.



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

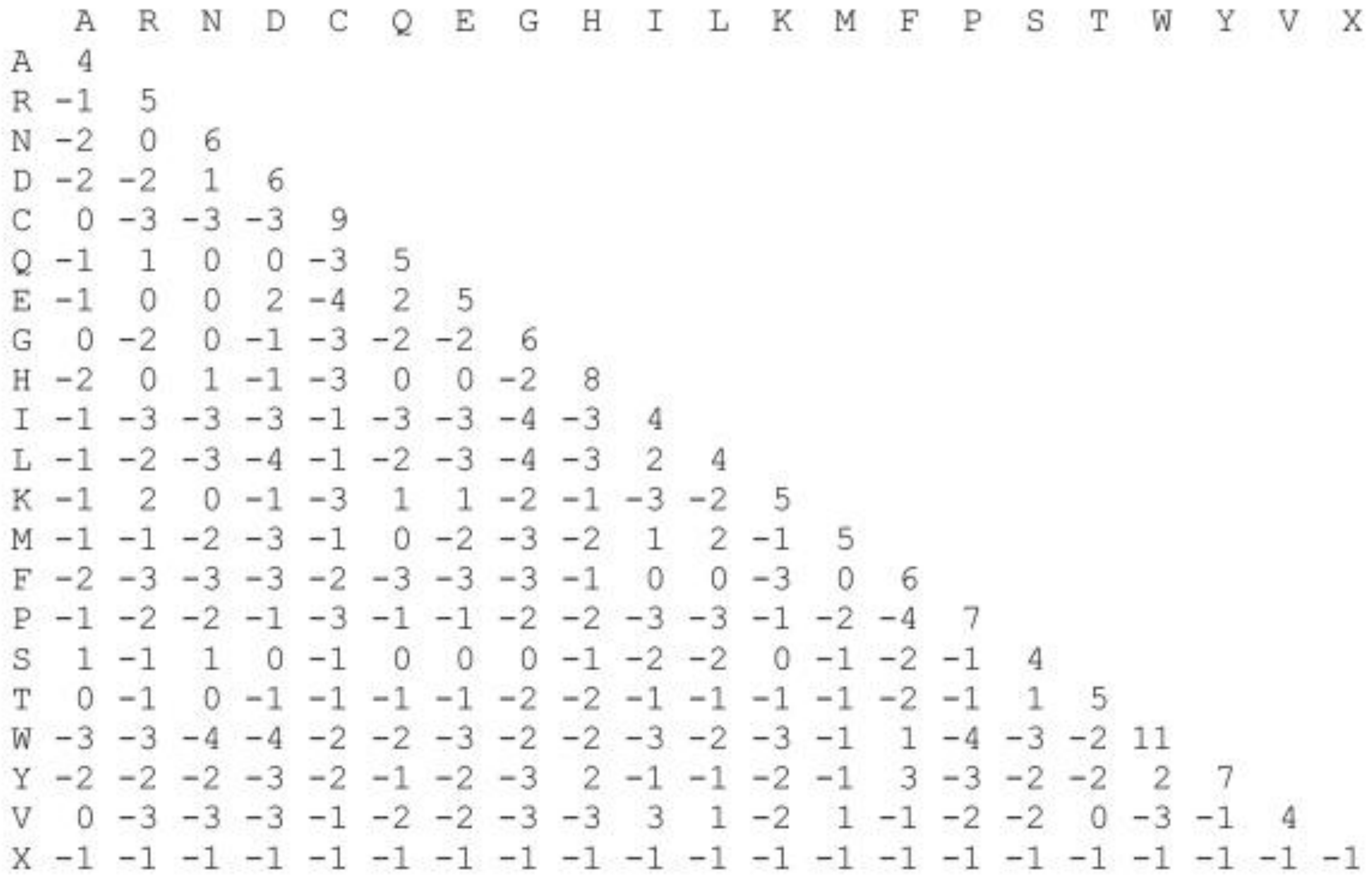
V	**P**	**N**	**M**
V	P		
	P	N	
		N	M

VP	IP	LP	MP
PN	PD	PH	PS
NM			

Exercise

Given the sequence **`VPNM`**, a threshold of 8, and *k*-mer size 2 perform BLAST preprocessing to find the set of *k*-mers to search for.

V	**P**	**N**	**M**
V	P		
	P	N	
		N	M



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

VP	IP	LP	MP
PN	PD	PH	PS
NM	NL		

Lets BLAST some stuff!

https://blast.ncbi.nlm.nih.gov/Blast.cgi