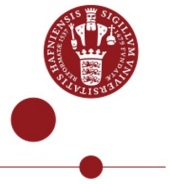
Alignment, mapping, Suffix Arrays, Burrows-Wheeler transform, uncertainty

Advanced topics in bioinformatics
week1: 4-09-2023

Thorfinn Sand Korneliussen

UNIVERSITY OF COPENHAGEN



Student feedback from 2022

In the last year evaluation of the course there was several good suggestions from the students;

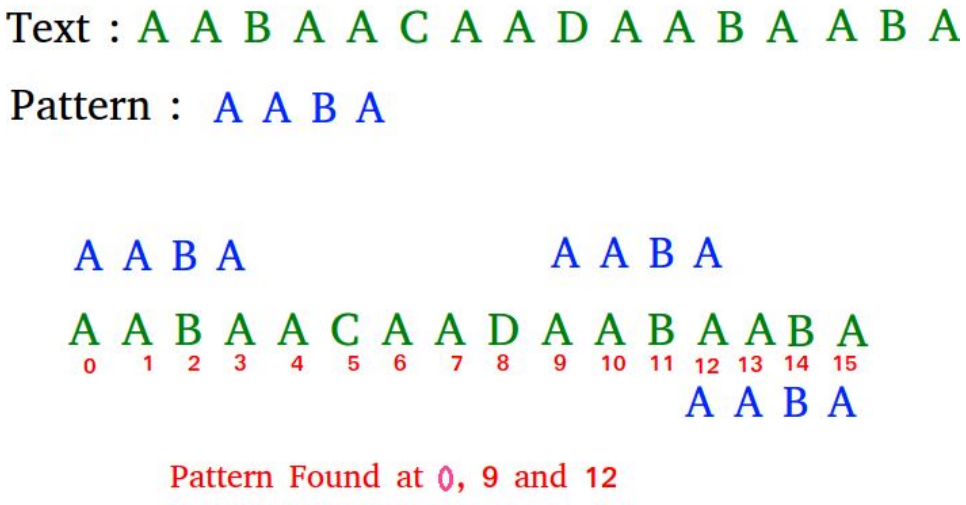
1. More streamlined used of absalon between topics by having a plan for each week
2. Better intro at the start of each topic (important when the topics differ so much)
3. Oral introduction to the assignments.

(Naive Sequence) Alignment

Formal: Text is an array of length n, and the pattern is an array of length m<n. We assume that these array contains elements from a shared alphabet A. We seek the *shifts* of pattern relative to text

Complexity: O((n-m+1)m)~O(nm)

This is ***exact matching***



Algorithm	Preprocessing	Matching time
Naive	0	O(nm)
Rabin-Karp	O(m)	O(nm)
Knuth-Morris-Pratt	O(m)	O(n)

Terminology:

- String: Sequence of characters S=ababana
- Substring: contiguous sequence of characters from the string
- Prefix and Suffix: special substrings occurring in the beginning and end of a string
- Alphabet of S is Σ = {a,b,n}
- DNA alphabet Σ = {A,G,C,T,N} (4,5)
- Protein alphabet Σ = {A,R,N,D,C,…,V} (20)

S = banana, substring = ana, prefix = ban, Suffix = nana

- How many possible suffixes can we have?

Algorithm2 - Rabin Karp

Idea: Define a **rolling** *hash function* h(S,i,j). S is string, i is shift, j is length. This effectively allows for check for matching in O(1) for each shift, instead of O(m).

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

A	T	G	C
0	3	2	1

Hash1: h_1(S,i,j) = \sum_{n=0}^j S_{i+n}
= h_1(S,i-1,j)-S_{i-1}+S_{i+j}

example

h_1(atgc) = 0+3+2+1=6
h_1(text,0,4) = h_1(AATG) = 0+0+3+2 = 5
h_1(text,1,4) = h_1(text,0,4)-S_0+S_{1+4} = 5-0+1 = 6

Problems? h_1(text,4,4)? 6

There might be several overlap e.g. when the length is different but the pattern is the same AACC and CCAA

```
Step 0:  aatgcgtaatc
Step 1:  aatgcgtaatc
Step 2:  aatgcgtaatc
Step 3:  aatgcgtaatc
Step 4:  aatgcgtaatc
Step 5:  aatgcgtaatc
Step 6:  aatgcgtaatc
Step 7:  aatgcgtaatc
```

Hash2: h_2(S,i,j) = \sum_{n=0}^j S_{i+n}*10^n
= (h_2(S,i-1,j)-S_{i-1}*10^j)*10+S_{i+j}

example

h_2(atgc) = 0321
h_2(test,0,4) = 0*10^3+0*10^2+3*10^1+2*10^0 = 30+2=32
h_2(text,1,4) = (32-0*10^3)*10+1=0321

Problems? What if pattern is very long?

If the pattern very long it would be overflow

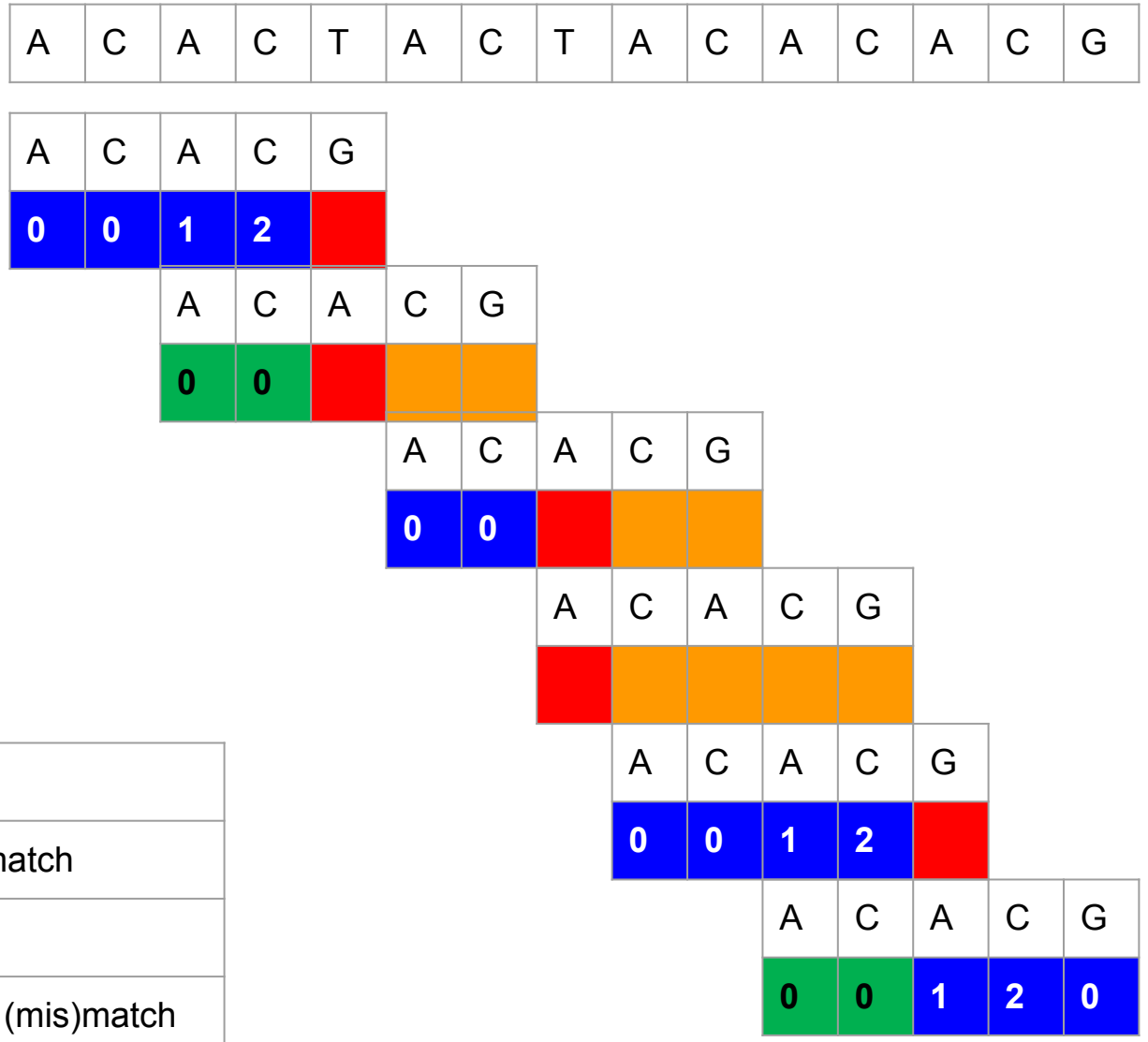
Hash3: h_3(S,i,j)
= \sum_{n=0}^j S_{i+n}*10^n mod q
= (h_3(S,i-1,j)-S_{i-1}*10^j)*10+S_{i+j} mod q

The q could be 123 prime number

Problems?

Algorithm 3 -Knuth Pratt Morrison

Idea: Find longest common prefix that occurs within the pattern. If a mismatch occurs then we do not need recheck if the prefix was already matched



Preprocess pattern so we keep track of subsequences that are also prefixes

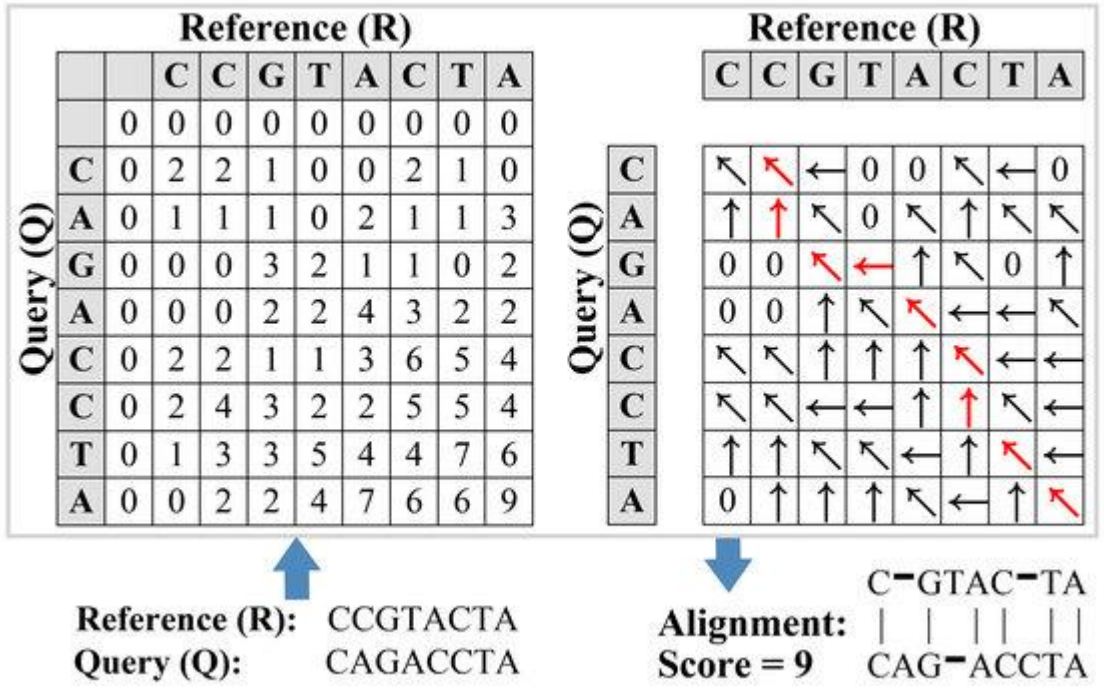
A	C	A	C	G
0	0	1	2	0

	Blue means that we have a match
	Green means that do not need to check for match
	Red means that we have a mismatch
	Orange means that we should not search for (mis)match

Alignment4 - inexact match (Smith Waterman)

Exact and optimal alignment with regards to scoring scheme (substitution model+gap model)

- Left to right alignment
- gaps
- Fill in cells of a matrix.
- Each cell can be:
  1) match/mismatch ↘
  2) gap in query ↓
  3) gap in reference →
  We choose the "best" of the 3, and keep track of what we choose for each cell
- When all cells are filled out, we pick the cell with the highest value. This is our "alignment score". To build the actual alignment we backtrack.
- Scoring system can be either {0,1} or could be using actual substiution matrices BLOSUM or PAM. Gaps can be gap open (high), gap extend(lower)



Adaptively Banded Smith-Waterman Algorithm for Long Reads and Its Hardware Accelerator. (Yi-Lun Yiao 2018,)

Mapping Problem- Reference based alignment

- Let us assume we have a sequencing run with over 100 million reads. After processing, the reads are between 75 - 150 nucleotides long.
- We would like to know if these sequences are in the human genome, and if so where. The human genome has a size of 3Gb.
- A naïve approach would be to simply search for a sequence such as TCTGAGCGGAGGAGAG (n=16), this takes ~ 6 seconds
- But querying 100 million reads will take over 20 years

- Metagenomics studies uses genetic material recovered directly from environmental samples so to identify all organism we need a reference databases of 1.5 Tb.
- Which is why we need faster search algorithms and more efficient data structures to perform various string algorithms

Exercise round 1

1. Login to the ricco server:

   Details for logging in can be found here:

   https://github.com/ANGSD/adv_binf_2023_week1
2. Copy this folder: /TEACHING/BIOINF22/intro into your home directory

   cp -r /TEACHING/BIOINF22/intro .
3. Have a look at the files:
   a) How many files do we have?
   b) What do they contain?