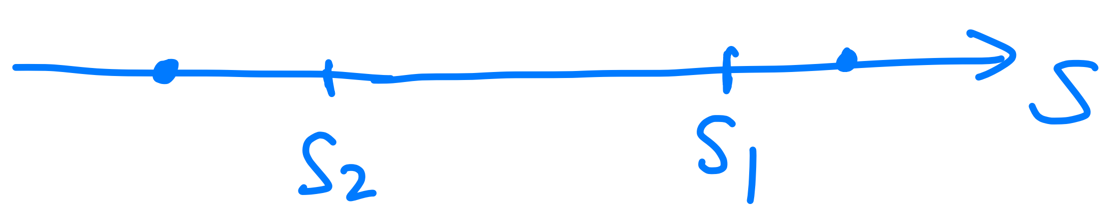
CSE 566 Spring 2023

Minimizers

Instructor: Mingfu Shao

Gapped LSH Family

- A set of hash functions $\mathscr{F}$ is said to be a (s_1, s_2, p_1, p_2)-*gapped LSH family* for similarity measure $s(\cdot, \cdot)$, where $s_1 \geq s_2$ and $p_1 \geq p_2$, if for any two x and y we have
  - If $s(x, y) \geq s_1$, then $\Pr_{f \in \mathscr{F}}(f(x) = f(y)) \geq p_1$;
  - If $s(x, y) \leq s_2$, then $\Pr_{f \in \mathscr{F}}(f(x) = f(y)) \leq p_2$.
- If $\mathscr{F}$ is a (r, r, r, r)-*gapped LSH family for measure* $s(\cdot, \cdot)$ for every $0 < r < 1$, then $\mathscr{F}$ is a LSH family for measure $s(\cdot, \cdot)$



x= ACGTGTAC
y= ACTGTAC

Results for Edit Distance

- Define $es(x, y) := 1 - d(x, y)/n$ as the edit-similarity between strings x and y of length n, where $d(x, y)$ is the edit distance.
- Edit distance/similarity is fundamentally different from the distance/similarity on normed vector space.

$x = 0110$
$y = 1011$

$x = \{1, 0, \cdots 1\}$
$y = \{0, 1, \cdots 1\}$
$U = \{x_1, \cdots x_n\}$

$x = (\quad) \in \mathbb{R}^d.$
$y = (\quad) \in \mathbb{R}^d$

- It remains open whether a LSH family exists for edit-similarity.
- A gapped LSH family for the edit distance exists!
  - OMH: https://doi.org/10.1093/bioinformatics/btz354 (2019)

order min hash

Minimizers

k=4

w=3

L=k+w-1=6

- Given a string *S*, the Minimizers select the smallest kmer (according to order *π*) in each sliding window of *w* kmers.
  - *k*: the length of kmer
  - *w*: the window size
  - *π*: an order over all kmers

AAAA, AAAC, ..., TTTT

```
    12345678901234
S = TGTCAACTACGGCT
    TGTC
     GTCA √
      TCAA
       CAAC √
        AACT √
         ACTA √
          CTAC
           TACG
            ACGG √
             CGGC
              GGCT
```

minimizers

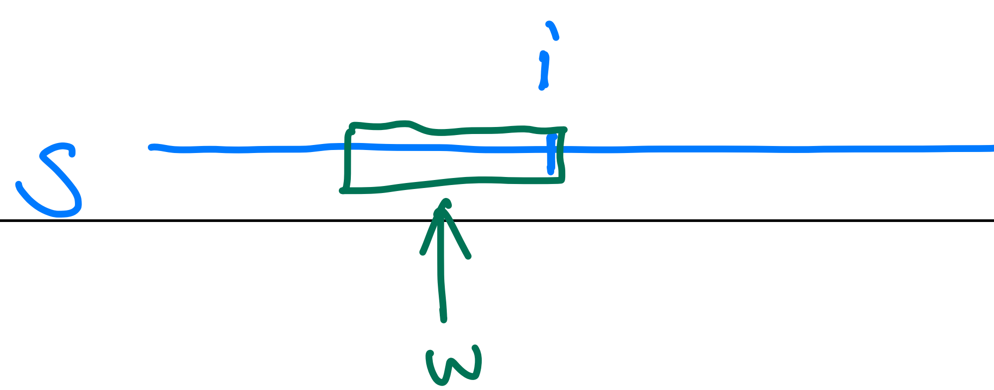
{2, 4, 5, 6, 9}

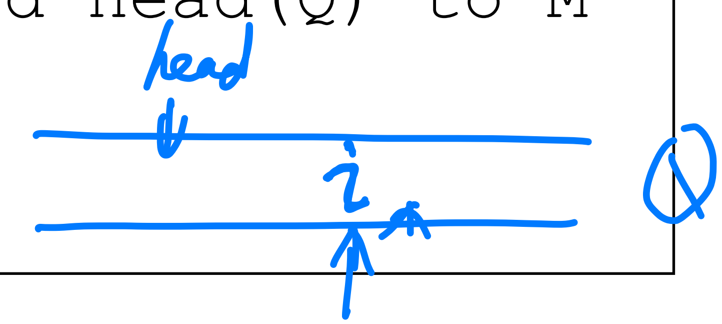
naive algo: $O(|S| \cdot w)$ comparisons

Calculating Minimizers

```
Input: string S, k, w, order pi
Output: array M to store the positions of minimizers
init an empty queue Q
for i = 1 to (|S| - k + 1)
    let m = S[i..i+k-1] be the current kmer
    while m <(pi) S[tail(Q)..tail(Q)-k+1]: pop-tail(Q)
    append i to the tail of Q
    while head(Q) <= i - w: pop-head(Q)
    if i >= w && head(Q)!= tail(M): append head(Q) to M
end for
return M;
```

kmer at the tail of Q



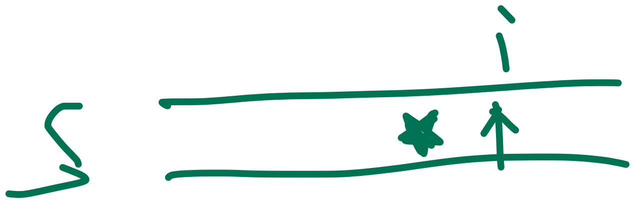


An Example

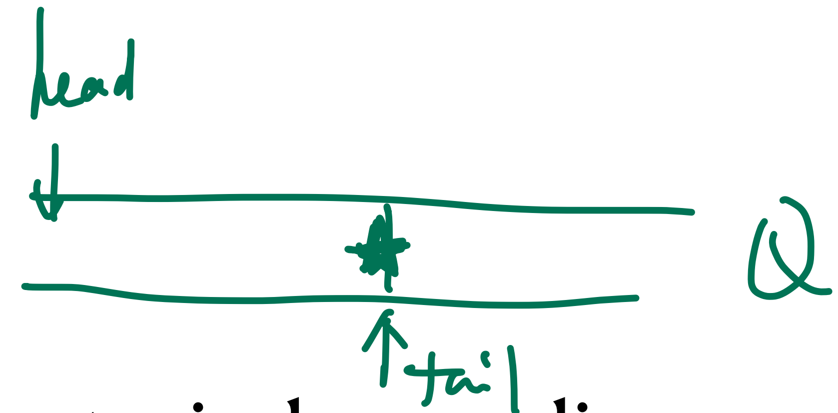
```
12345678901234
TGTCAACTACGGCT
TGTC
 GTCA
  TCAA
   CAAC
    AACT
     ACTA
      CTAC
       TACG
        ACGG
         CGGC
          GGCT
```

Q: #4

M [2, 4 . .



Correctness



- Claim: head(Q) is the minimizer of the current window ending with the i-th kmer
  - Fact 1: it is safe to pop-tail(Q) when m is smaller than the kmer at tail(Q)
  - Fact 2: it is safe to pop-head(Q) when it is out of the current window
  - Fact 3: the kmers in Q are in increasing order
  - Fact 4: head(Q) is the smallest kmer in Q

Running Time

- The entire algorithm takes $O(|S|)$ comparisons
  - Each kmer gets added into Q once
  - Hence #pops is also bounded by $|S|$

#pops ≤ #added-kmers

$L = w+k-1$ S_1

S_2 L

Properties of Minimizers

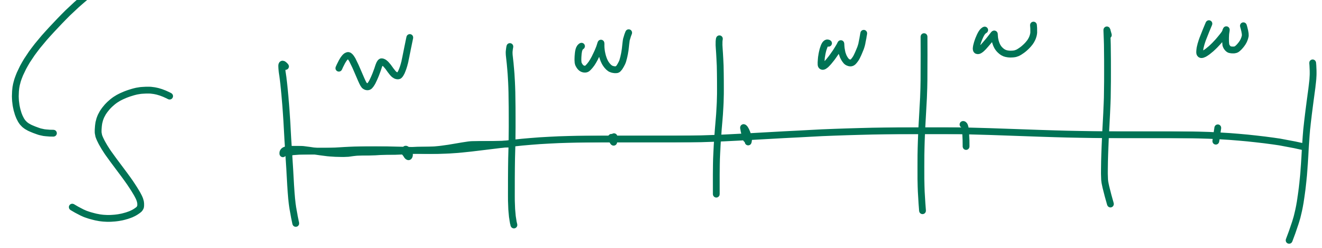
$w+k-1$

- Small gap: the distance between two selected positions is at most w, and hence provides a good "coverage".
- Consistency: if two sequences share a substring of length $w + k - 1$, then they share a minimizer.
- Locality-sensitive: if two sequences share a "similar" substring of length $w + k - 1$, then with high probability that they also share a minimizer.

Density of Minimizers

- Given w, k, and and order π, the particular density of a sequence S is defined as: $|M(S)|/(|S| - k + 1)$.
- Given w, k, and and order π, the expected density is defined as the particular density over an infinity-length, random sequence (i.e., each base is sampled uniformly at random).
- Trivial bounds for expected density: $1/w \leq d \leq 1$.
- Proved lower bound for expected density: $\cancel{1.5/(w+1) \leq} d$

$d \geq \frac{1.5}{1+w}$



Density with Random Order

- Order π is picked uniformed at random from all orderings.
- **Theorem**: the expected density wrt a random order π is $2/(1+w) + o(1/w)$.
- **Proof**: consider the probability that the next window uses a new minimizer.

$$Pr = \frac{2}{1+w}$$