CMSC423: Bioinformatic Algorithms, Databases and Tools

Exact string matching:

Computing Z values in linear time

- Recap: Z values capture similarity between beginning of string and internal parts of the string

- Recap: Z values can be used to speed up matching

- Stop and think: Write an algorithm to compute the Z values of a string.

Naïve computation of Z values

AAAGGTACAGTTCCCTCGACACCTACTACCTAAG

Z[1]?

compare T[1] with T[0], T[2] with T[1], etc. until mismatch

in this case Z[1] = 2

Z[2] ?

Same process applies: compare T[2] to T[0], T[3] to T[1], etc. until mismatch

Stop and Think! What is the worst-case run-time of this algorithm?

Can Z values be computed in linear time?

AAAGGTACAGTTCCCTCGACACCTACTACCTAAG

The naïve process is still expensive:

T[2] is compared when computing both Z[1] and Z[2].

Trick to computing Z values in linear time:

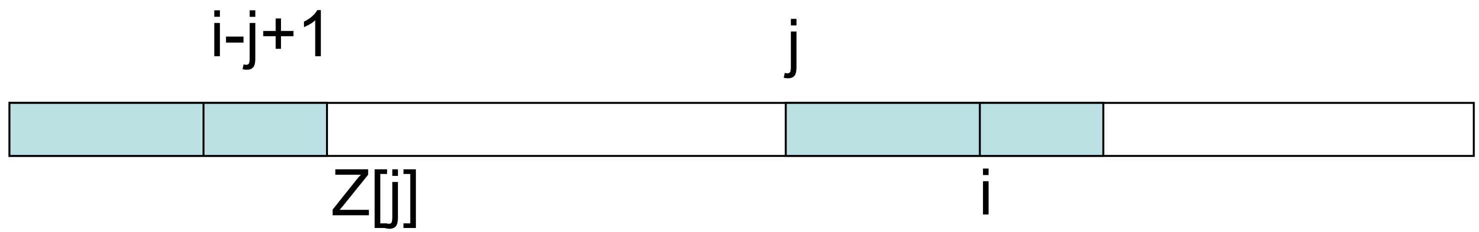
each comparison must involve a character that was not compared before

Intuition: once we match a character we have learned something about it and do not need to look at it again.

Conjecture: Since there are only m characters in the string, the overall # of comparisons will be O(m).

Basic idea: 1-D dynamic programming

Induction: Can Z[i] be computed with the help of Z[j] for j < i?



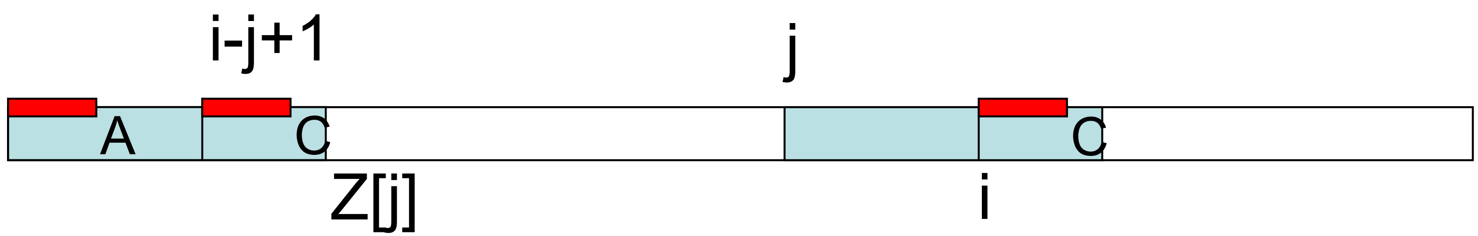
Assume there exists j < i, such that j + Z[j] – 1 > i
then Z[i – j + 1] provides information about Z[i]

If there is no such j, simply compare characters T[i..] to T[0..]
since they have not been seen before.

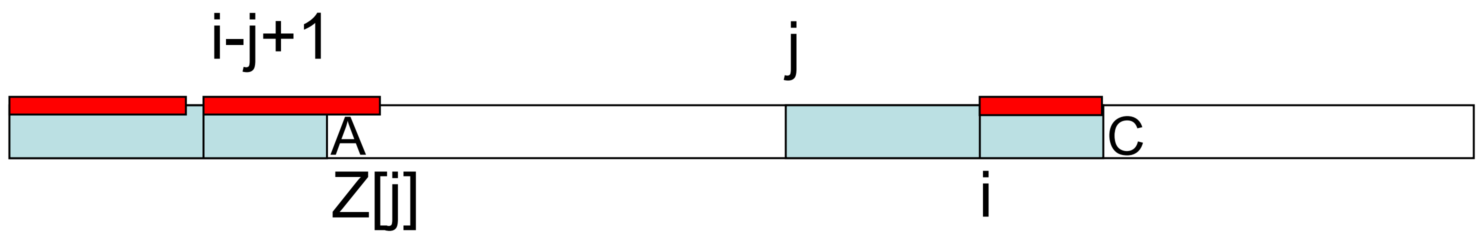
Three cases

Let $j < i$ be the coordinate that maximizes $j + Z[j] – 1$
(ihe $Z[j]$ that extends the furthest)

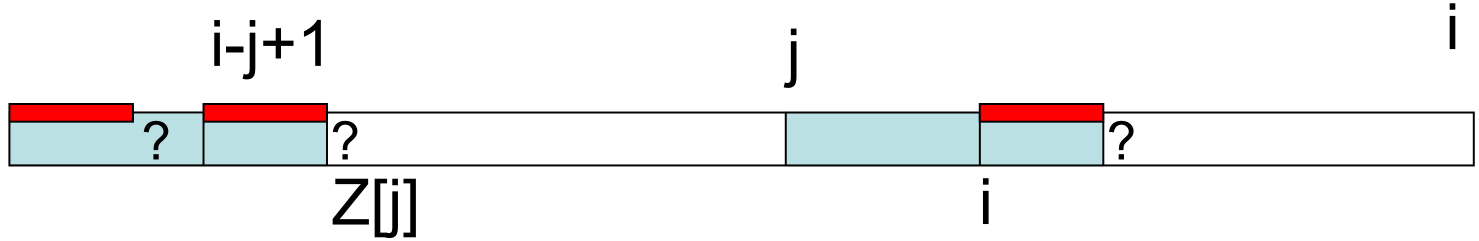
I. $Z[i – j + 1] < Z[j] – i + j – 1 \Rightarrow Z[i] = Z[i – j + 1]$



II. $Z[i – j + 1] > Z[j] – i + j – 1 \Rightarrow Z[i] = Z[j] –i + j - 1$



III. $Z[i – j + 1] = Z[j] – i + j – 1 \Rightarrow Z[i] =$??, compare from
$i + Z[i – j + 1]$



Time complexity analysis

- Why do these tricks save us time?

1. Cases I and II take constant time per Z-value computed – total time spent in these cases is O(n)
2. Case III might involve 1 or more comparisons per Z-value

   however:

   - every successful comparison (match) shifts the rightmost character that has been visited
   - every unsuccessful comparison terminates the “round” and algorithm moves on to the next Z-value

   total time spent in III cannot be more than # of characters in the text

Overall running time is O(n)

NEXT: KMP algorithm