Nate Evans

Christina Zheng & Shannon McWeeney

BMI650

11/5/18

Midterm, Fall 2018

Reminder: This is a take-home exam. Students are expected to develop, write up, and hand in their own individual solutions and, in doing so, develop a sufficient understanding of the problem and solution so as to be able to explain it adequately to the instructor. Under no circumstances should a student discuss the exam with others while taking it, copy or consult the solution of another student, or copy a solution from any other source, including the Internet. You must show all work.

1. Provide the pseudo-code for an algorithm that will determine the **length** of the shortest substring that occurs exactly once in a genomic sequence of length N. You must provide all assumptions made for your algorithm.

Assumptions

* Case sensitive
* Spaces (gaps) are treated as a distinct character
* Input string needs no parsing, what is provided is used to find shortest substring
* If there is no shortest substrings found, the length of the full string will be returned
* The given string is interpreted as is. t does not represent DNA/RNA/AAs and therefore the only substring match is the substring. Complements, reverse-complements and translation sequences do not count as a match. e.g. ‘ATG’ != ‘TAC’

**For code prototyping this algorithm, SEE “evans\_midterm.py”**

Given a string S, to find the shortest substring that occurs exactly once, the following algorithm can be employed.

1. Generate a BWT from S
2. Using the F,L backtracking method, build out each substring (row)
   1. At each additional character, compare each substring (row) with all other substrings (rows).
      1. If there are any unique substrings (a substring that has no other matches do not include substrings with terminal character),
         1. Return the number of characters in substring (iteration +1)
3. If the entire BW matrix is reconstructed, return the length of S (OR iteration +1)

Example: S = “ACTGTC”

Algorithm search visualized (red=unknown, black=known)

<iteration = 0>

[['$' 'A' 'C' 'A' 'C' 'T' 'G' 'G' 'T' 'C']

['A' 'C' 'A' 'C' 'T' 'G' 'G' 'T' 'C' '$']

['A' 'C' 'T' 'G' 'G' 'T' 'C' '$' 'A' 'C']

['C' '$' 'A' 'C' 'A' 'C' 'T' 'G' 'G' 'T']

['C' 'A' 'C' 'T' 'G' 'G' 'T' 'C' '$' 'A']

['C' 'T' 'G' 'G' 'T' 'C' '$' 'A' 'C' 'A']

['G' 'G' 'T' 'C' '$' 'A' 'C' 'A' 'C' 'T']

['G' 'T' 'C' '$' 'A' 'C' 'A' 'C' 'T' 'G']

['T' 'C' '$' 'A' 'C' 'A' 'C' 'T' 'G' 'G']

['T' 'G' 'G' 'T' 'C' '$' 'A' 'C' 'A' 'C']]

<iteration = 1>

[['$' 'A' 'C' 'A' 'C' 'T' 'G' 'G' 'T' 'C']

['A' 'C' 'A' 'C' 'T' 'G' 'G' 'T' 'C' '$']<non-unique

['A' 'C' 'T' 'G' 'G' 'T' 'C' '$' 'A' 'C']<non-unique

['C' '$' 'A' 'C' 'A' 'C' 'T' 'G' 'G' 'T']<terminal

['C' 'A' 'C' 'T' 'G' 'G' 'T' 'C' '$' 'A']<unique! <<

['C' 'T' 'G' 'G' 'T' 'C' '$' 'A' 'C' 'A']<NA

['G' 'G' 'T' 'C' '$' 'A' 'C' 'A' 'C' 'T']<NA

['G' 'T' 'C' '$' 'A' 'C' 'A' 'C' 'T' 'G']<NA

['T' 'C' '$' 'A' 'C' 'A' 'C' 'T' 'G' 'G']<NA

['T' 'G' 'G' 'T' 'C' '$' 'A' 'C' 'A' 'C']]<NA

Compare each substring (first column) with all other substrings (other first column, rows)

<<< return substring length (iteration + 1)

2. As reported in the HG38 version of the human genome, please answer the following questions about the gene, CD8B.

a. What chromosome is this gene on?

Chromosome 2

b. What is the start and end coordinate of this gene?

5’ 3’

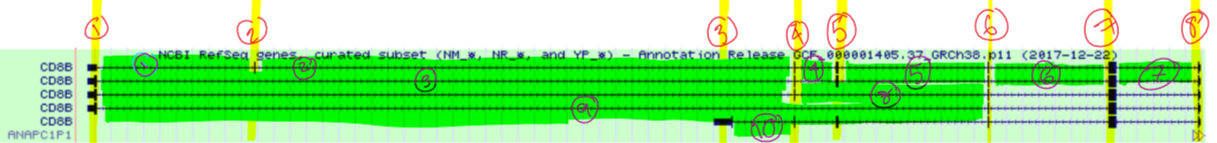
[86815557, 86861915]

Please note that while the start and end is reported as it’s displayed on UCSC, our gene is on the negative strand, and therefore the gene is transcribed in the opposite direction and hence the 3’ starts at the end coordinate.

c. What strand of the DNA contains this gene?

Minus (-) strand.

It is important to remember that the bases displayed on UCSC represent the bases of the positive strand, so if our gene is on the negative strand, we must take the reverse complement to get an accurate representation of the gene.



d. How many transcripts are in this gene as reported by the RefSeq Consortium?

5.

e. How many distinct exons are reported in the RefSeq transcripts?

8.

f. How many distinct introns are reported in the RefSeq transcripts?

10.

g. List the genomic coordinates of 3 introns (start-end) from above which contains a canonical splice sites.

Canonical Splice Sites are characterized by GU at the 5’ end of the intron, AG at the 3’ end and are spliced by the major spliceosome. This can be found by looking at the ends of each intron. At the end of each intron [start, end] displayed in UCSC we can find our gene intron splice site bases by taking the reverse complement and transcribing, because our gene is on the negative strand, and this will give us the beginning and end of our intron:

[86815717, 86844921] 🡪 5’ end: AC (+) 🡪 GU (-, RNA) [86844921]

🡪 3’ end: CT (+) 🡪 AG (-, RNA) [86815717]

🡪 canonical splice site

[86815717, 86852996] 🡪 5’ end: AC (+) 🡪 GU (-, RNA) [86852996]

🡪 3’ end: CT (+) 🡪 AG (-, RNA) [86815717]

🡪 canonical splice site

[86844958, 86852996] 🡪 5’ end: AC (+) 🡪 GU (-, RNA) [86852996]

🡪 3’ end: CT (+) 🡪 AG (-, RNA) [86844958]

🡪 canonical splice site

There seems to be some disagreement on what makes a canonical splice site, for this assignment we will assume, for simplicity, that only the GU-AG rule pertains to canonical splicing. I’d like to clarify this at some point though.

“non-canonical splice sites (that is, with sequences other than GT–AG, GC–AG or AT–AC at the intron/exon boundaries)” [1]

“…when the intronic flanking sequences do not follow the GU-AG rule, noncanonical splicing is said to occur.” [2]

h. List the genomic coordinates of each intron (start-end) from above which contains a noncanonical splice site.

Non-canonical splice sites are characterized by any splice sites (start, end of intron) that does not follow the GU-AG rule, and is commonly spliced by the minor spliceosome. I believe that on UCSC this can be identified by looking for introns whose directionality (shown by arrows) is opposite to the rest of the gene. However, there are no non-canonical splice sites which can be identified this way on CD8B, and, with additional investigation of each intron’s head and tail, it can be confirmed that there are no non-canonical splice sites in the ref-seq reported introns of CD8B at this time.

i. What is the official gene symbol of the nearest gene to CD8B? What strand of the DNA contains this nearest neighboring gene? How would the location of these genes be described in relation to one another?

ANAPC1P1

This is on the positive (+) strand and starts at starts at 8686191. This gene is therefore on the complementary DNA strand to CD8B and starts at CD8B’s end coordinate (CD8B’s 5’ end) and is transcribed in the opposite direction as CD8B; ANAPC1P1 is on chr2 on the positive strand of DNA and the 5’ head begins at 86861915.

1. Describe the minimum information needed to unambiguously define the location of a gene? Provide an example using a gene of your choice.

Assuming that this must apply to genes whose symbols are not yet in the UCSC browser, and thus cannot be accessed via gene symbol. e.g. how to define the location of a previously unknown gene

1. A Reference Genome
2. Chromosome ID/number
3. Start and end coordinates
4. DNA strand (+/-)

To unambiguously define the location of CD8B as if it was an unknown gene, we must define which reference genome we will use, for this gene, we specify the human genome **HG38 version.** Next, we must define that it’s on **chromosome 2.** To define where on the chromosome it’s located, we must specify the start and end coordinates as **[86815557, 86861915]** meaning that the gene starts at the 86815557th base pair from the 5’ end of the positive strand of chromosome 2. Lastly, we have to define which strand of DNA the gene is located on, positive (5’ end is at 0th base) or negative (3’ end is at the 0th complementary base). Negative and positive are complementary strands and the displayed bases are representative of the positive strand, so if the gene is on the negative strand, the complement of the gene shown should be considered. In this example, we will define the gene as being on the **negative strand.**

4. Align the protein sequences S1 and S2 with a **gap opening penalty of 2** and a **gap extension penalty of 1** and the **“H2O” substitution cost defined as**:

For amino acids x and y,

H(x,y) =

+5, if x=y

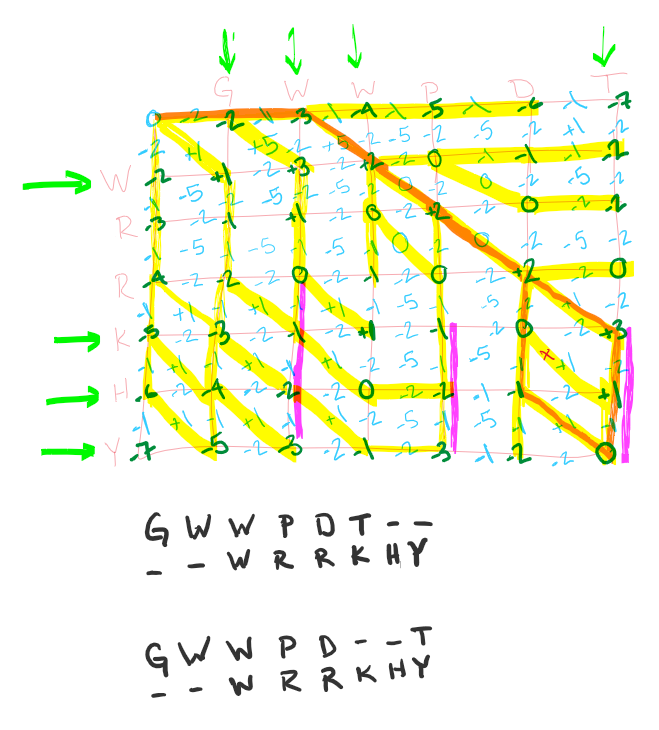
+1, if x≠y, but x,y are both hydrophobic

‐5, if x≠y, and only one is hydrophobic

0, if x≠y, neither are hydrophobic

S1 = GWWPDT

S2 = WRRKHY



In the table above:

Blue : score for specific transition

Dark Green : score at that point

Yellow: Transition resulting in node score

Purple: Gap transition dependent on extension scoring, think of it as an overpass; If traveling along purple route, you can only get off at the end.

Orange: Highest scoring alignment route

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| G | W | W | P | D | T | - | - |
| G | W | W | P | D | - | - | T |
| - | - | W | R | R | K | H | Y |

5. Describe a scenario where a researcher would be interested in investigating both the local and global alignment of two sequences.

When comparing the genomes of two distinct organisms, local alignment of a gene may show highly conserved regions pertaining to functional regions of RNA or Protein. However, global alignment can be useful in measuring the genetic separation of the two organisms; The global alignment can be used to measure how long ago the two species diverged.

6. Bacterial genomes are often circular. To transform to a linear form, some genome assembly programs will pick a random location in the genome to break the circle. Thus, it is possible that running the same program multiple times we would get different answers, corresponding to different circular rotations of the same string. Provide the psuedo-code that will determine if two DNA strings are circular rotations of each other. For example TTGATC is a circular rotation of ATCTTG. You must state all assumptions.

Given strings S1, S2

Assume that S1 and S2 are the same length

Assume that S1 and S2 only contains characters {ATCG}

Assumes that the genome assembly program always outputs the same strand and directionality, eg. S1 and S2 will never be complements or reverse complements of each other

Assumes S1 and S2 represent only DNA, no mRNA or Residue sequences

**Please see evans\_midterm.py for code**

7. We can define a set of distinct substrings of a string S that includes all substrings. However, each repeat is only represented once. For example, for the string S = AATATT, this set is:

{A, T, AA, AT, TA, TT, AAT, ATA, TAT, ATT, AATA, ATAT, TATT, AATAT, ATATT, AATATT}

You are given a suffix tree of S. Provide the pseudo-code for an algorithm that counts the number of distinct substrings of S. For full credit, this should run in O(n) time.

The number of distinct substrings in S can be found by counting the number of nodes in the suffix tree. Programmatically, this can be done in networkx by len(DiGraph.nodes) but algorithmically, a recursive traversal of the graph is an effective way to count the connected nodes in O(n).

Assumes:

* The suffix tree is in the form that, edges hold characters, and nodes represent branching points. The last edge in each branch will include a ‘$’ character.
* The suffix tree is provided in a network data structure with the methods:
  + Node.has\_no\_children() – which tests if there are any additional nodes connecting to the given node
  + Node.get\_children\_nodes() – returns the IDs representing nodes on the next level down that it is connected to. This is similar to networkx’s .neighbors() but omits the node traversed from.

Pseudocode : recursive algorithm to count the number of tips of the tree

-------------------------------------------------------------------------------

def rec\_search(node, tree):

count = 0

If (node.has\_no\_children()):

return 1

For child in node.get\_children\_nodes():

count += rec\_search (child, tree)

return count

num\_distinct\_substrings = rec\_search(root, tree)

-------------------------------------------------------------------------------

# References

[1] Parada, Guillermo E et al. “A comprehensive survey of non-canonical splice sites in the human transcriptome” Nucleic acids research vol. 42,16 (2014): 10564-78.

[2] Ng, Bernard; Yang, Fan; Huston, David P.; Yan, Yan; Yang, Yu; Xiong, Zeyu; Peterson, Leif E.; Wang, Hong; Yang, Xiao-Feng (2004). ["Increased noncanonical splicing of autoantigen transcripts provides the structural basis for expression of untolerized epitopes"](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3902068). Journal of Allergy and Clinical Immunology. 114 (6): 1463–70. [doi](https://en.wikipedia.org/wiki/Digital_object_identifier):[10.1016/j.jaci.2004.09.006](https://doi.org/10.1016%2Fj.jaci.2004.09.006). [PMC](https://en.wikipedia.org/wiki/PubMed_Central) [3902068](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3902068). [PMID](https://en.wikipedia.org/wiki/PubMed_Identifier) [15577853](https://www.ncbi.nlm.nih.gov/pubmed/15577853).