Bilkent University
CS481 | cs583 - Bioinformatics Algorithms

Fall 2022

Homework Assignment #1

GENERAL INSTRUCTIONS

FAILURE TO FULFILL ANY OF THE FOLLOWING ITEMS WILL RESULT IN A GRADE SCORE OF 0 (zero) WITHOUT ANY CHANCE OF REDEMPTION.

- You must **write your code yourself**. Sufficient evidence of plagiarism will be treated the same as for plagiarism or cheating.
- **C / C++, Python or Java** will be used as programming language. STL is allowed. The assignments are created with C / C++ in mind, so using other languages would required minimum tweaking.
- Your code must **compile**.
- Your code must **be complete**.
- Your code must **run on dijkstra.cs.bilkent.edu.tr** server.
- Your code must make use of **argument parsing**.
  Refer to: `https://docs.google.com/presentation/d/1rUODhBg6yVXbfEtNuK73pjc1yWSPG9OYnGNH-b-kklQ`
- Submit your answers **ONLY** through the Moodle page.
- **Zip** your files and send them in only one zipped file.
- File name format `surname_name_hw#.zip`.
- The zip file must contain the following items:
  - All the source files.
  - `Makefile` to compile the source code and produce the binary. Even if you use Python, include this Makefile.
  - A `README.txt` file that briefly describes how your program works.
- All submissions must be made **strictly before the stipulated deadline**.
- A **bonus** will be given for the fastest code that solves the assignment successfully.

1) MULTIPLE PATTERN MATCHING

Aim: In this assignment, you will get familiar with the data structures associated with string searches in bioinformatics. Given a set of patterns and a text, find all occurrences of any of patterns in text.

1.1 Data

- **Input**
  1. A text file containing k patterns p to search. The file will hold one pattern per line. The path of this file will be passed as CLI argument to the program with the `-p` switch.
  2. A text file containing a text t to be searched. The file can have multiple lines. The path of this file will be passed as CLI argument to the program with the `-t` switch.
- **Output**
  1. A text file containing the positions $1 < i < m$ where substring of t starting at i matches p_j for $1 < j < k$. The path of this file will be passed as CLI argument to the program with the `-o` switch. The file is composed of one line per pattern followed by the index of the match in the text. If no match is found, then no index is saved.
  2. A text file containing the definition of your suffix tree in DOT[1] language for visualization.

1.2 Data Structures

Implement the data structures based on the information given in class and other resources you might find, but be sure to document them in your README file.

- Keyword Tree with Naive Threading
- Suffix Tree

1.3 Tasks

Using **each** data structure, perform pattern matching and along with the output file, print on the standard output, the status of the search while it is being carried out.

2) EXAMPLE

The numbers presented in the example are not necessarily correct. They just show how the program should output the information.

```
user@dijkstra$ cat patterns.txt
ba
ana
nananas
bananas
banana
```

```
user@dijkstra$ cat text.txt
banana
```

```
user@dijkstra$ ./hw1 -p patterns.txt -t text.txt -o output.txt
> Keyword Tree
matching "ss"
    /* ... */
    node@0xABCD11: "$". comparisons = 3
    node@0xABCD21: "a". comparisons = 4
    node@0xABCD35: "na$". comparisons = 5
    node@0xABCD4F: "na". comparisons = 6
    /* ... */

> Suffix Tree
matching "radar"
    /* ... */
    node@0xFFCD10: "$". comparisons = 3
    /* ... */
    node@0xFFCD20: "banana$". comparisons = 4
    /* ... */
```

[1]https://graphviz.org/documentation/

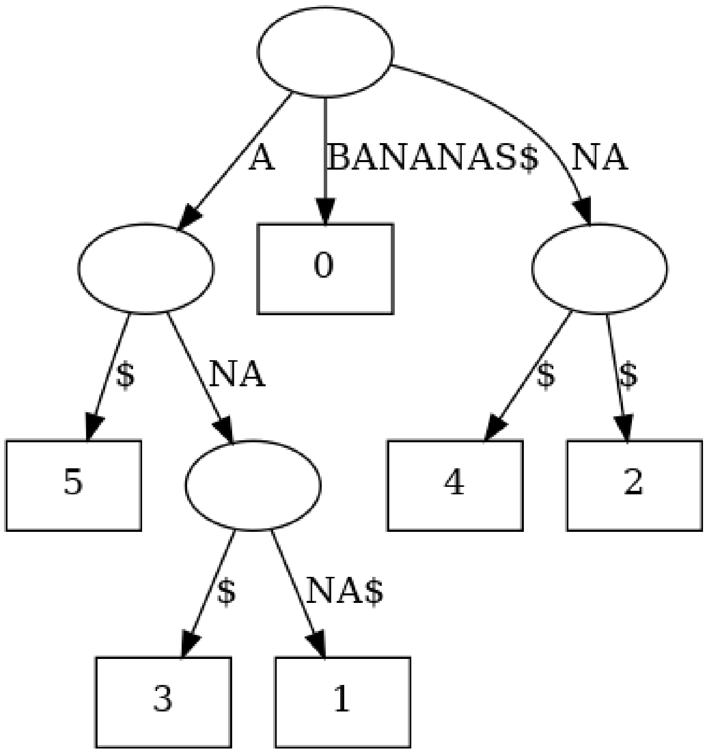
```
user@dijkstra$ cat output.txt
1: 1
2: 2
3:
4:
5: 1
```

```
user@dijkstra$ cat output.dot
digraph BST {

  n1  [label=""]
  n2  [label=""]
  n3  [label="0" shape=box]
  n4  [label=""]
  n5  [label="5" shape=box]
  n6  [label=""]
  n7  [label="4" shape=box]
  n8  [label="2" shape=box]
  n9  [label="3" shape=box]
  n10  [label="1" shape=box]

  n1 -> n2 [label="A"]
  n1 -> n3 [label="BANANAS$"];
  n1 -> n4 [label="NA"];

  n2 -> n5 [label="$"]
  n2 -> n6 [label="NA"]

  n4 -> n7 [label="$"]
  n4 -> n8 [label="$"]

  n6 -> n9 [label="$"]
  n6 -> n10 [label="NA$"]

}
```

After running the **Graphviz** tool (available in Dijkstra), out of the `output.dot` file, an PNG will be generated with your tree. Follow the example to create the layout for the suffix trees you create.

```
[ricardo@dijkstra ~]$ dot output.dot -Tpng > output.png
```

The `output.png` will be contain a tree similar the one shown below.


A
BANANAS$
NA
0
$
NA
$
$
5
4
2
$
NA$
3
1