Algorithmics II Assessed Exercise

- Accounts for 20% of final marks
- Marked out of 30 and converted to a band
- 20hrs is intended upper time limit (roughly)
- Exercise is to be done individually
- Deadline is 4.30pm, Friday 7 November
- Concerned with Applications of Suffix Trees
    1. Searching for a substring
    2. Determining all occurrences of a given substring
    3. Finding a longest repeated substring
    4. Finding a longest common substring
- The language of implementation is Java

Use of AI tools

- They are readily available, but
- All external sources, including the use of AI tools, must be acknowledged in the status report
- There will be a question related to the assessed exercise in the final exam

Support

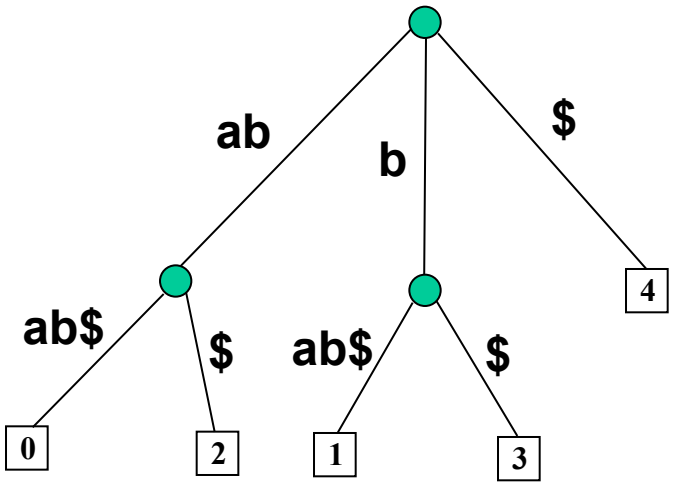
- There will be a lab session devoted to the exercise
- Monday 27 October from 1300-1400 in Boyd Orr 720
- Replacing that day's lecture

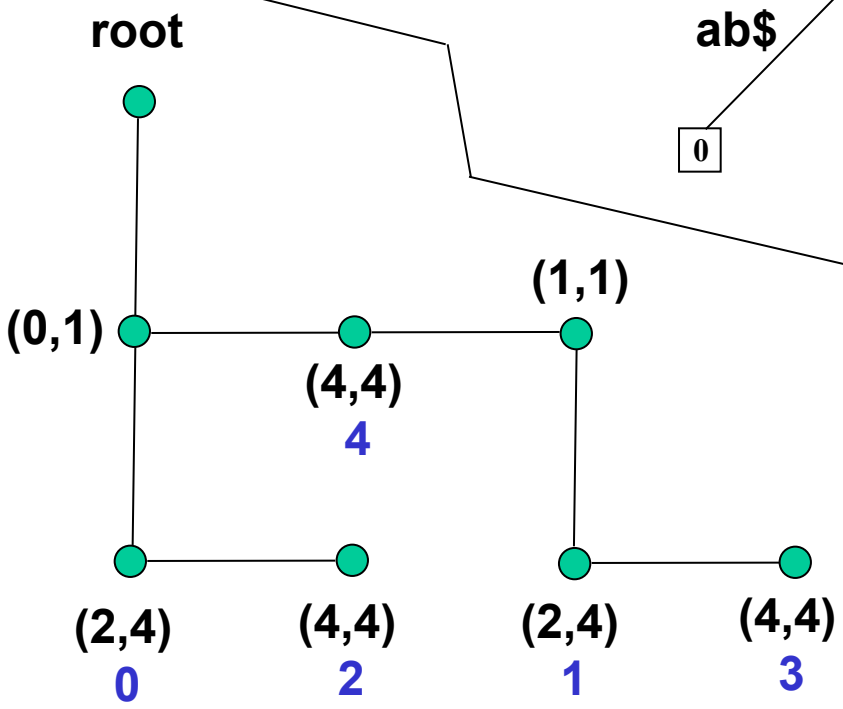
Representation of a Suffix Tree

```
public class suffixTreeNode {
    private suffixTree child;
    private suffixTree sibling;
    private int leftLabel;
    private int rightLabel;
    private int suffix; }

public class SuffixTree {
    private SuffixTreeNode root;
    private byte [] s; }
```

Example: suffix tree for abab$





a b a b $
0 1 2 3 4

What has been given

- Code for reading a text file F into an array of bytes S

- Suffix tree construction algorithm (inside `SuffixTree` class)

```
public void buildSuffixTree () {
 // builds a suffix tree T for a given
 // string S
}
```

- Simple classes for passing back required information from methods corresponding to Tasks 1-4:

```
Task1Info.java
Task2Info.java
Task3Info.java
Task4Info.java
```

- Some code that should be extended to form the command-line interface

- Skeleton files are provided via Moodle

Your task

- **Bodies of methods corresponding to Tasks (1)-(4) in `suffixTreeAppl.java` have been left blank**

```
public Task1Info
          searchSuffixTree(byte[] x) { }

public Task2Info
          allOccurrences(byte[] x) { }

public Task3Info traverseForLrs() { }

public Task4Info traverseForLcs
          (int s1Length) { }
```

- **Also one of the SuffixTree constructors is incomplete:**

```
public SuffixTree (byte[] sInput1,
                   byte[] sInput2)
```

- **Also the `main` method is incomplete in `Main.java`**

- **Task is to complete these**

The required output

- **Along the following lines:**

```
> java Main SearchOne text1.txt "there"
Search string "there" occurs at position
1132 of text1.txt

> java Main SearchOne text1.txt "clutch"
Search string "clutch" not found in
text1.txt

> java Main SearchAll text2.txt "clutch"
The string "clutch" occurs in text2.txt
at positions:
178007
77871
479220
The total number of occurrences is 3

> java Main SearchAll text1.txt "clutch"
The string "clutch" does not occur in
text1.txt
```

The required output (cont)

```
> java Main LRS text2.txt
An LRS in text2.txt is ".
  "Mamma! What sweets are we going to
have?" "
Its length is 47
Starting position of one occurrence is
155839
Starting position of another occurrence
is 156193

> java Main LCS text1.txt text2.txt
An LCS of text1.txt and text2.txt is "
it is absolutely necessary t"
Its length is 29
Starting position in text1.txt is 212557
Starting position in text2.txt is 120985
```

What to submit

- **All `.java` files**
- **Code listing of `Main.java`, `SuffixTree.java` and `SuffixTreeAppl.java`**
- **Status report**
- **Detailed implementation report justifying the method of execution of your solutions to tasks 1-4**

All files to be uploaded via course web page on Moodle

An acceptance test will be carried out after submission – sample long text files for your own testing purposes are provided in setup zip file

Marking scheme

- **Implementation of Tasks 1-4**
  3,2,3,4 marks for tasks 1-4 (awarded on the basis of correctness and efficiency)

  (12)

- **Implementation of following constructor:**

```
public SuffixTree (byte[] sInput1,
                   byte[] sInput2)
```

  (1)

- **Completion of `main` method**

  (3)

- **Implementation report: 3,2,3,4 marks corresponding to each of Tasks 1-4**

  (12)

- **Quality of code (i.e. layout, comments etc.) and general presentation**

  (2)

Total (30)