Algorithms & Complexity 2/27/17 – 3/22/17

0145-344-001

Note Taker: Jai Punjwani

ANNOUNCEMENTS

Midterm Monday 3/6 (study from midterm review sheet)

* induction question will be on it
* determining Big-OH
* know all complexity cases for sorts
* expect polyphase merge
* Minimum spanning trees
* prim’s & kruskal’s algorithm

Topic: Pattern Matching

PowerPoint: <http://home.adelphi.edu/~siegfried/cs344/344l7.pdf>

* Start by looking for first character of pattern in the query string, and looking for each subsequent character. If no match, start by looking for first character of pattern again.
* Very useful in computational biology (DNA sequence matching)
* Pattern matching can work for any type of pattern (i.e. bits)
* GREP used in Unix machines for pattern matching

**Analysis**

Worst-Case is NOT NECESSARILY when we don’t find any matches. In fact, if we find partial matches which eventually fail our pattern, we have to backtrack our index, which turns out to be worse.

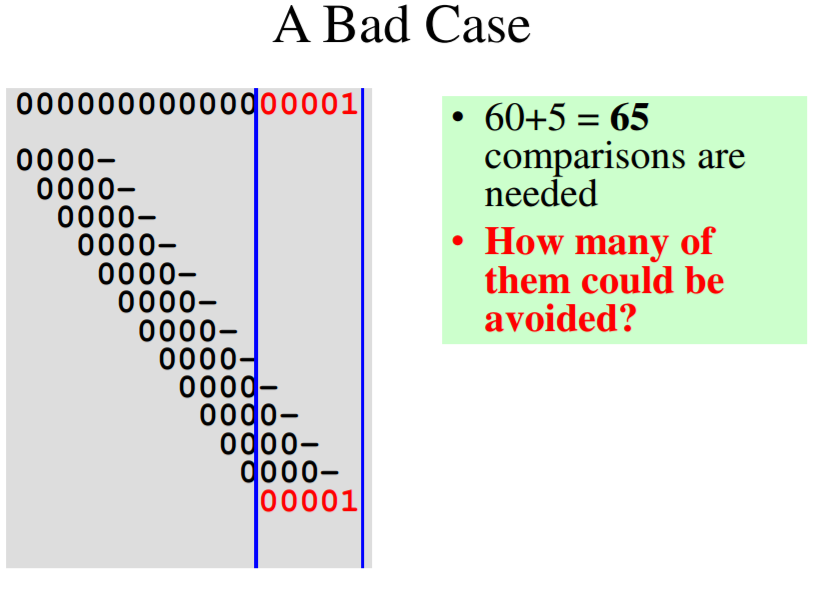
ex:

String: aaaafj

pattern:aaaay

Initially we have an index at the first letter of our String. Our pattern will match the first four letters, but will fail the fifth. In that case, we will have to move our index to the **second** index. This is known as **backtracking** (perhaps not officially, but in terms of this algorithm). At the end of our pattern match search, we will have made **more than n** comparisons.

**Complexity:** O(m x n), where m = string length & n = pattern length

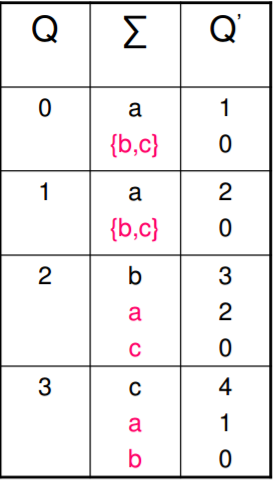
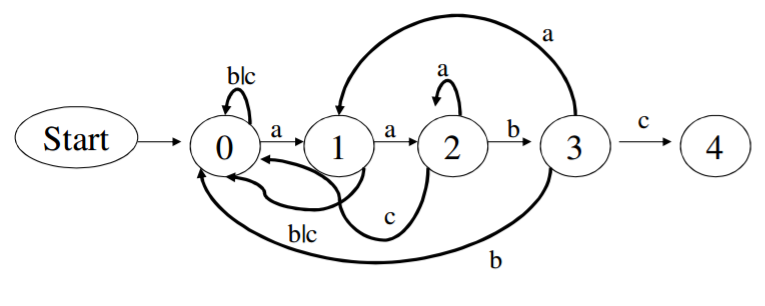


**Finite State Machines (FSM)**

* input String
* output Yes/No

Finite State Machines (FSM) – a (theoretical) computing machine that takes a string as its input and outputs yes/no. **Meant to keep state to solve backtracking issue** and keep track of letters that have already matched without having to recheck ALL of them.

Example of translating state table into diagrams:

Explanation: observe state 0. we go to state 1 only if we have an ‘a’. inputs ‘b’ or ‘c’ (b | c) mean that we remain in the state. Similar rules apply for the other states. Note that if we have a ‘c’ in state 3, we proceed to state 4, which is our desired (end) state.

**Improved Algorithm of pattern matching:** The Knuth-Morris-Pratt (KMP) Algorithm

* this algorithm prevents the previous worst-case of our brute force pattern checker by keeping track of what has been matched so far and does not revisit characters that we know will match
* ex:

String: aaaaay

pattern:aaaay  
  
the KMP algorithm would match the first four characters and then fail at the fifth character. Instead of going back and checking the ‘a’ characters from the second position of our String, it will check the 5th character of our String and will only have to check one more character (‘y’) to find the pattern

**See powerpoint for implementation**.

**Baeza-Yates-Gonnet (BYG) Algorithm [aka Bitap algorithm]**

* uses an array of bits (array size = size of pattern); a 1 in any position corresponds to a match in that corresponding index in the pattern
* uses bitwise operations to find match; also avoids redundant checking just like KMP algorithm
* Complexity ***O***(n), where n is the length of the text that we are searching

**Boyer-Moore** **Algorithm**

**fantastic resource:** <https://www.youtube.com/watch?v=4Xyhb72LCX4>

followed by: <https://www.youtube.com/watch?v=Wj606N0IAsw>