Chapter 7 Strings

Fei Tan

Chaifetz School of Business
Saint Louis University

Computer Science Fundamentals
(Source: brilliant.org)

January 13, 2019

The Road Ahead...

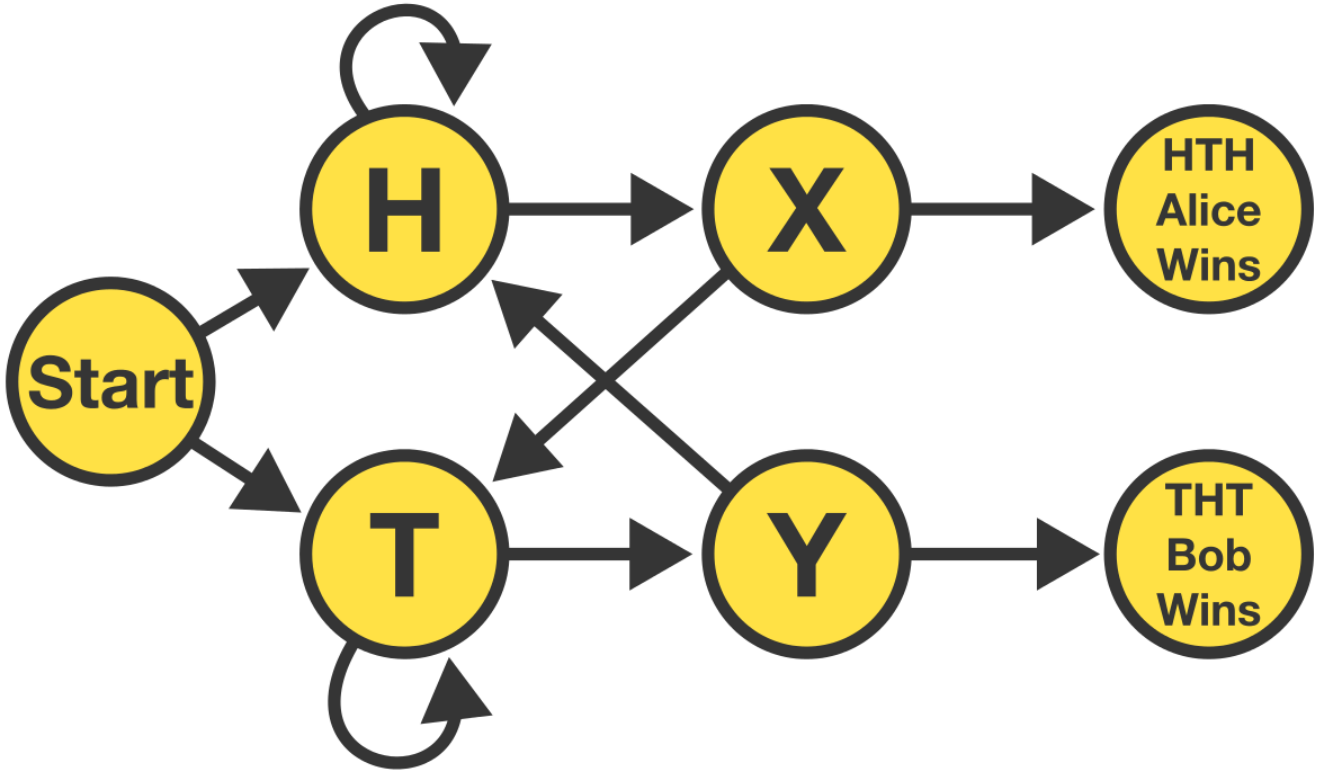
- Manipulating and searching strings have given rise to a lot of great codes over the years
- What we'll accomplish
  - use finite state machines to efficiently search strings
  - discover how to store words efficiently in a trie

Data Structure: Strings

- Representation in C language
  - continuous blocks of bytes in memory
  - 1 byte = 8 bits ('0' or '1'); 1 character takes 1 byte
  - terminated by character \0 = '00000000'
- Shallow vs. deep copy

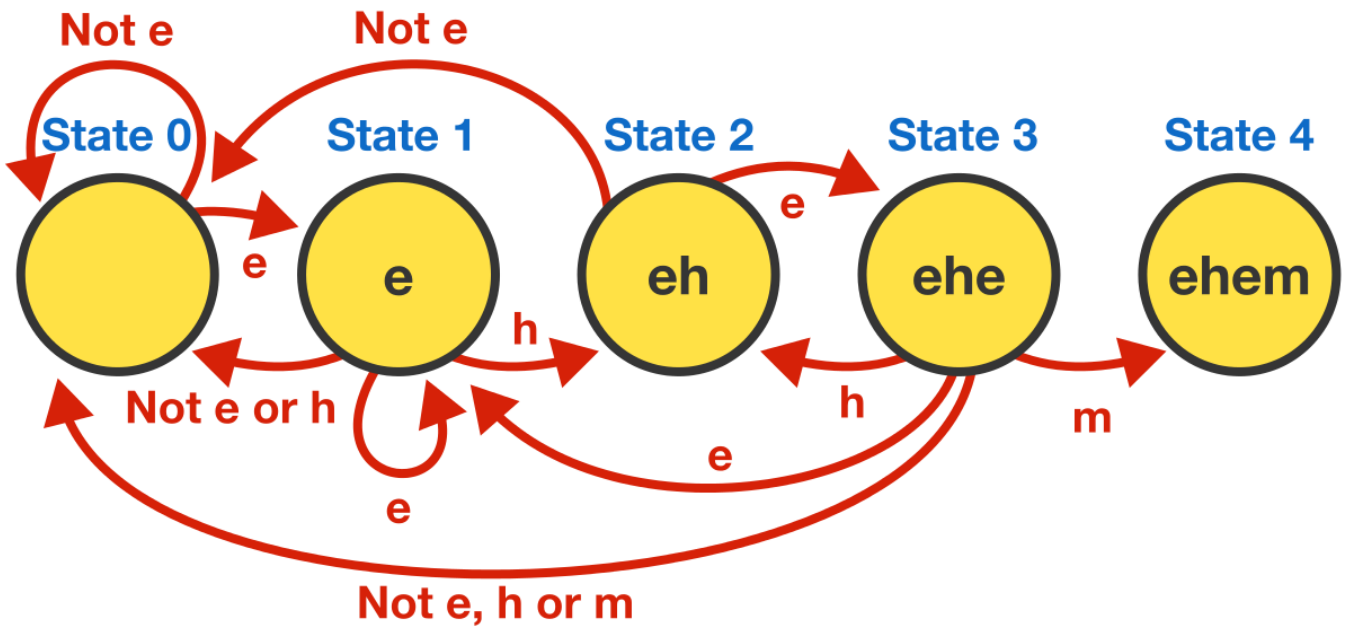
```
str1 = "hello"
str2 = shallowcopy(str1) # O(1) for string
    of length n; str2 points to same stri
str1 = "goodbye"
print(str2) #goodbye

str1 = "hello"
str2 = deepcopy(str1) # O(n) for string of
    length n; str2 represents different stri
str2 = "goodbye"
print(str1) #hello
```

Algorithm: Substring Search



- Deterministic finite automaton (DFA)
  - a set of states to track progress: '' (start), 'H', 'T', 'HT' (=X), 'TH' (=Y), 'HTH', 'THT'
  - why not 'HH' (='H'), 'TT' (='T')?

Constructing DFA



- Example: search for string 'ehem'
  - search stri of length M, source stri of length N
  - time complexity: $O(M + N)$

Python Code

Representation using adjacency lists

```
class Graph:
    def __init__(self,size):
        self.size = size
        #label n vertices 0 through n-1
        self.vertices = [[] for i in range(size)]

    #each vertex has a list of vertices it is
        connected to
    def addEdge(self,i,j):
        if not (i in self.vertices[j]):
            self.vertices[j].append(i)
            self.vertices[i].append(j)

    def numEdges(g):
        count = 0
        for i in range(g.size):
            count = count + len(g.vertices[i])
        return int(count/2)
```