

Start Recording

- Today
 - Finite State Machines
 - Complexity
 - Project 3 (due Tuesday, August 21 at 11:45 PM)
- Reminders
 - Project 2 due Tuesday, August 14 at 11:45 PM
 - Assignment 3 due Thursday, August 16 at 4:30 PM

Week 5

8/14	Project 2
8/16	Assignment 3

Week 6

8/21	Project 3
------	-----------

Week 7

8/28?	Self Quiz (optional)
8/30	Project 4

Week 8

9/4	Writing Assignment
9/6	Project 5

Finite State Machine

A machine that consists of...

A finite number of states

A starting state

An ending state(s)

States move to other states based on input condition

Can perfectly accept formal languages

Drawbacks?

- No concept of memory or history. All the machine “knows” is the current state.
- Hard to read as a human

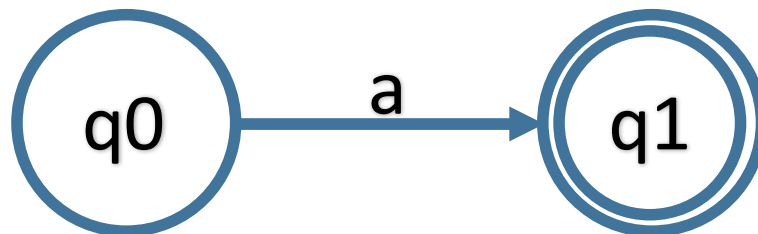
Regular Language / FSM (Ex 1)

Alphabet $\Sigma = \{a\}$

Language $L = \{a\}$

Regular Expression = $/a/$

FSM:



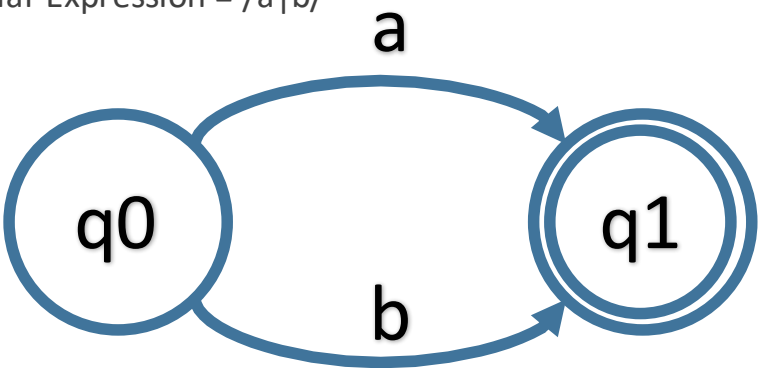
Regular Language / FSM (Ex 2)

Alphabet $\Sigma = \{a, b\}$

Language $L = \{a, b\}$

Regular Expression = $/a|b/$

FSM:



Alternatively:

$L1 = \{a\}$

$L2 = \{b\}$

$L3 = L1 \cup L2$

Regular Language / FSM (Ex 3)

Alphabet $\Sigma = \{a, b, c, d\}$

Language $L = \{ac, ad, bc, bd\}$

Regular Expression = $/(a|b)(c|d)/$

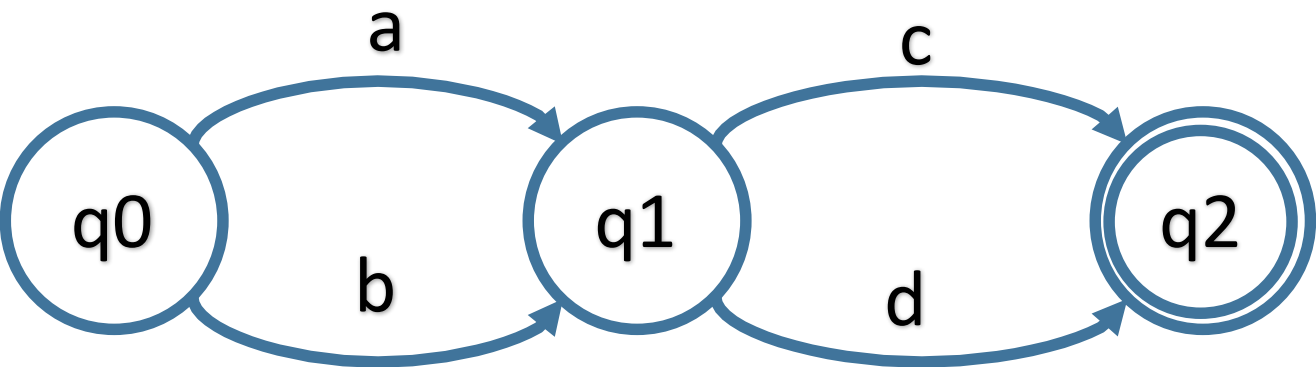
FSM:

Alternatively:

$L1 = \{a|b\}$

$L2 = \{c|d\}$

$L3 = L1 \cdot L2$



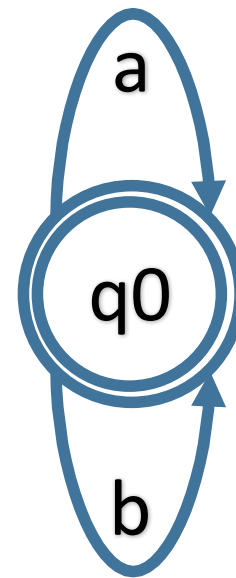
Regular Language / FSM (Ex 4)

Alphabet $\Sigma = \{a, b\}$

Language $L = \{\epsilon, a, b, aa, ab, ba, bb, aaa, \dots\}$

Regular Expression = $/(a|b)^*/$

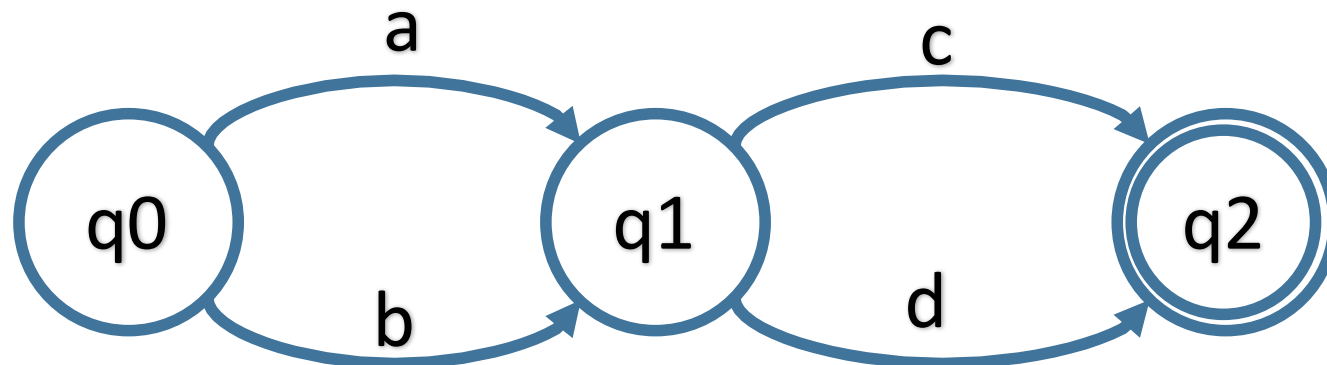
FSM:



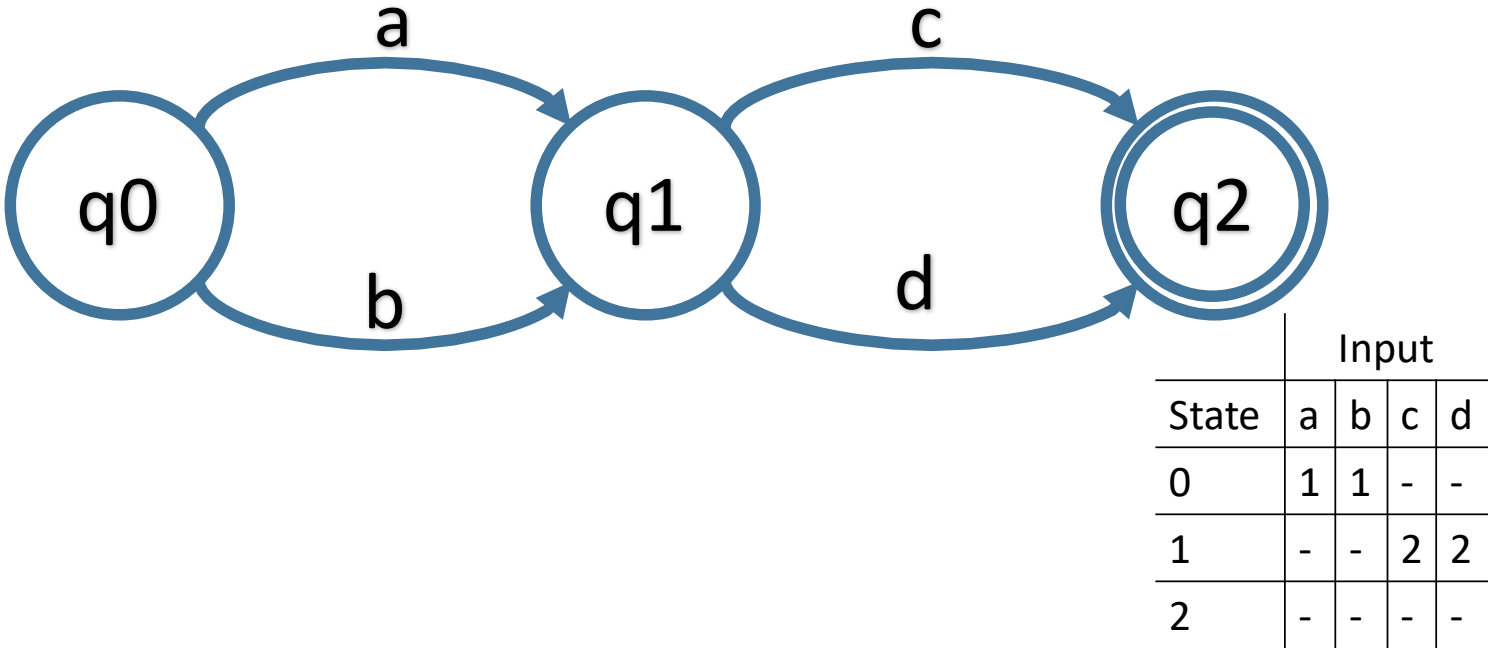
Regular Language / FSM

You can represent the FSM as a graph (human friendly)

We could also make it a table.



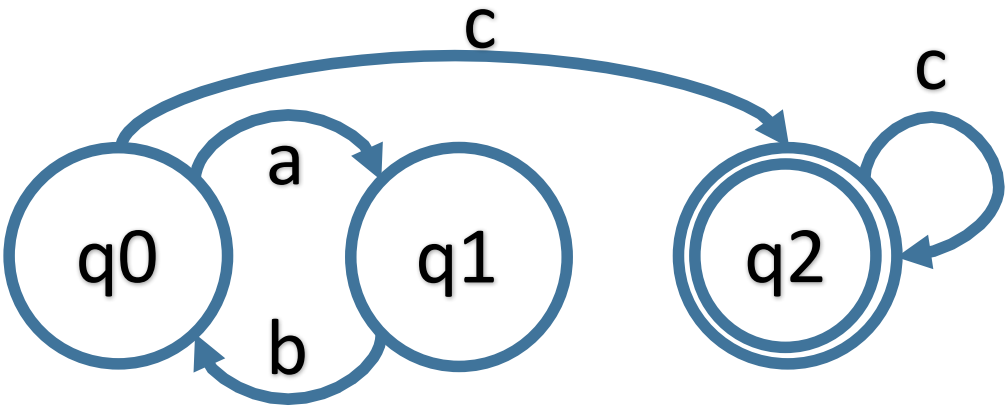
Regular Language / FSM



Regular Language / FSM

What does this FSM accept?
(Start state: 0, end state: 2)

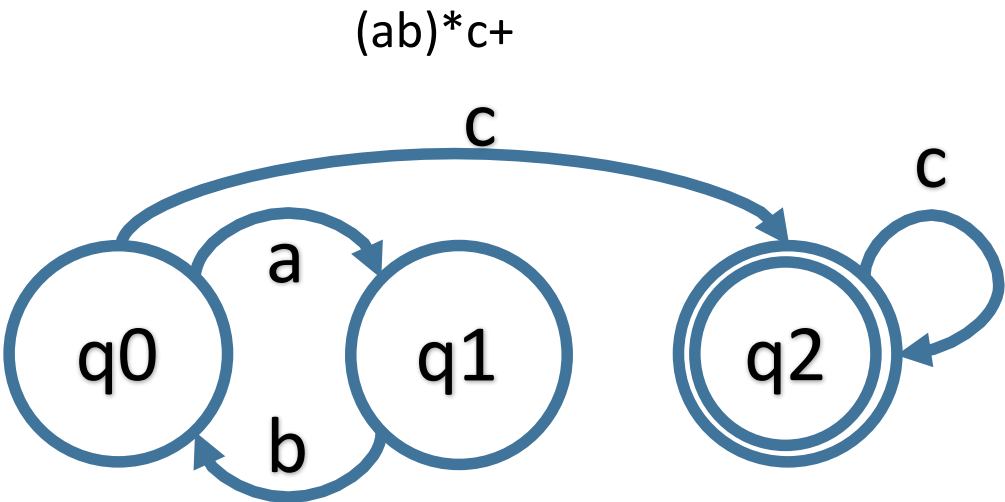
State	Input		
	a	b	c
0	1	-	2
1	-	0	-
2	-	-	2



Regular Language / FSM

What does this FSM accept?
(Start state: 0, end state: 2)

State	Input		
	a	b	c
0	1	-	2
1	-	0	-
2	-	-	2

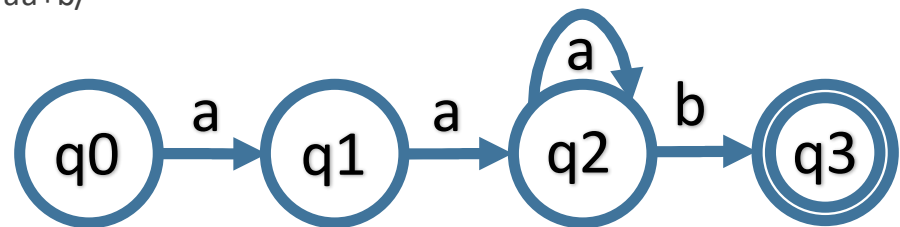


Non-Deterministic FSA (NFSA)

An FSA is an NFSA if:

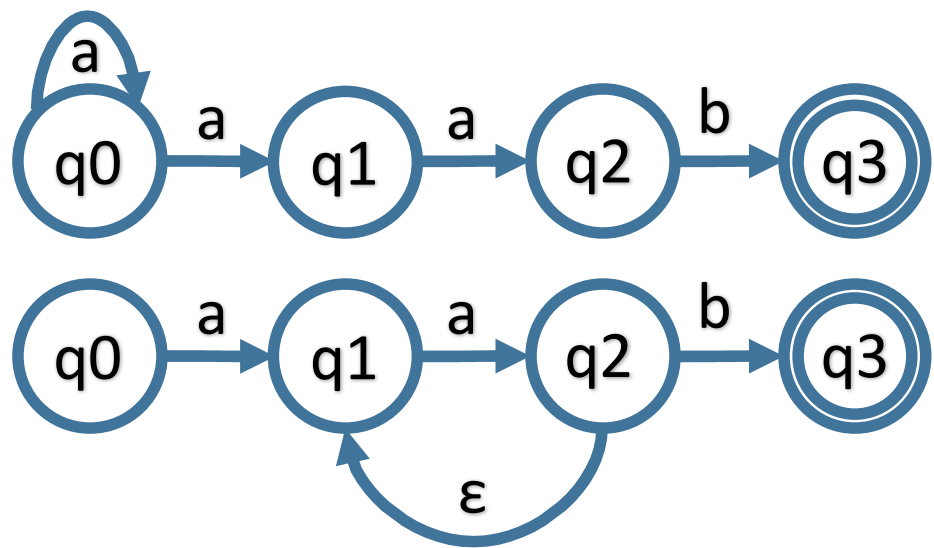
- There is any uncertainty about which arc to take.
- A single state has more than one arc with the same input
- ϵ -transitions (can transition w/o input)

Equivalent FSAs /aa+b/



Non-Deterministic FSA (NFSA)

Equivalent FSAs /aa+b/



Non-Deterministic FSA (NFSA)

They're the same as deterministic FSA (DFSA)

- An algorithm exists to transform an NFSA into a DFSA. (It's fairly straightforward.)

Who cares?

- Certain augmentations to FSAs break the symmetry (FSTs*, adding probability transitions)

Problems?

- Dead-ends
- Have to traverse many paths

Solutions?

- Check multiple paths
- Go back
- Look ahead/predict

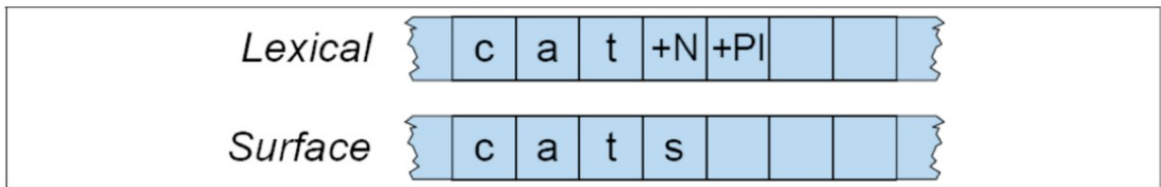
Finite State Transducer (FST)

It's an FSA + a bit extra

FSAs define regular languages, FSTs define regular *relations*

- two alphabets
- arcs with two labels: input/output, up/down

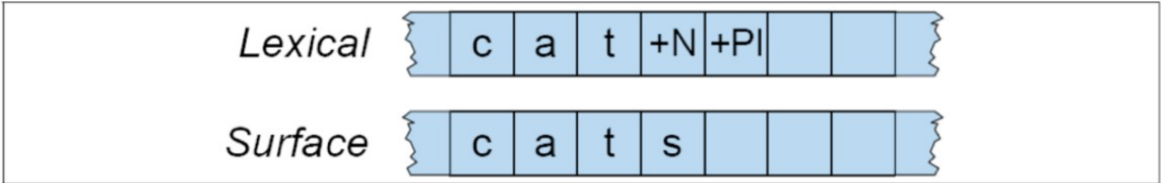
Inherently bidirectional



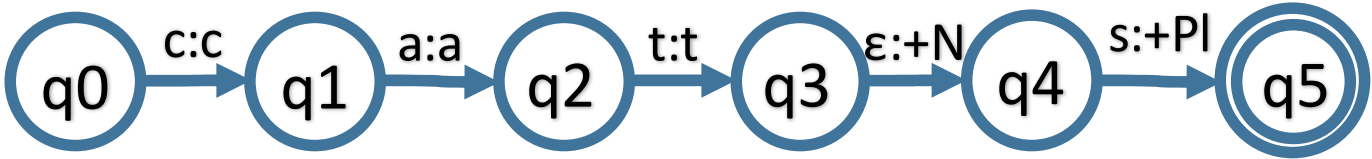
J&M text, Fig 3.12

- The FST arcs map one tape to the other

Finite State Transducer (FST)



J&M text, Fig 3.12



Regular Relation

Regular Language: a set of strings

Regular Relation: a set of pairs of strings

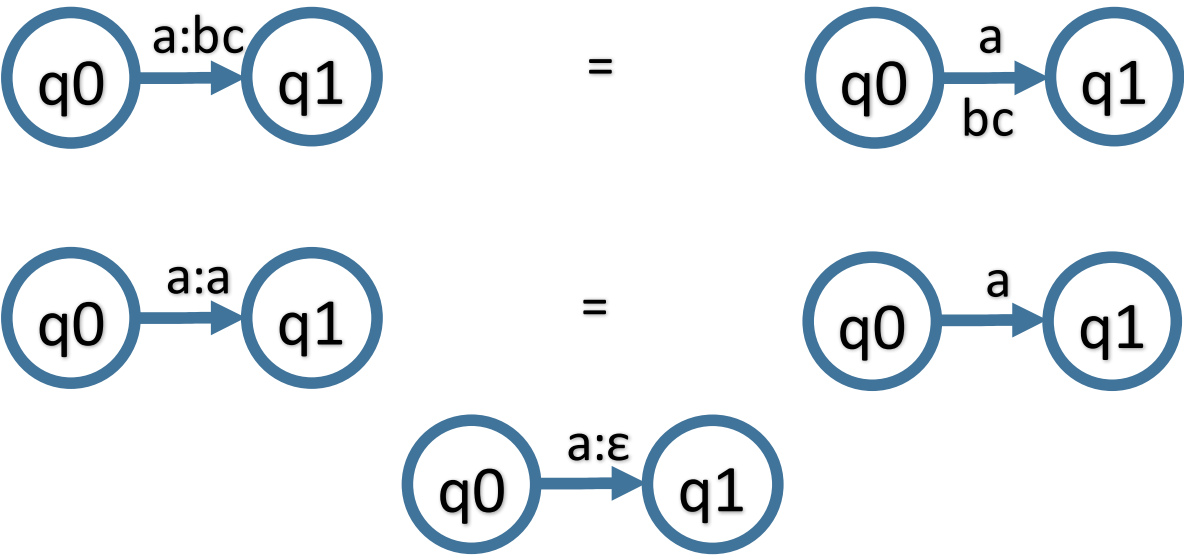
- Regular relation = {a:1, b:2, c:3, ...}
- Input alphabet = {a,b,c, ...}; output alphabet = {1,2,3, ...}

What would the alphabets look like for morphological relation?

- Input alphabet = characters = {a,b,c,d,e, ... λ , \acute{s} , \acute{c} , ... す , ...}
- Output alphabet = input alphabet + morphological units = {input alphabet, +PAST, +N, +PL, +SG, ...}

FST Conventions

Upper/lower



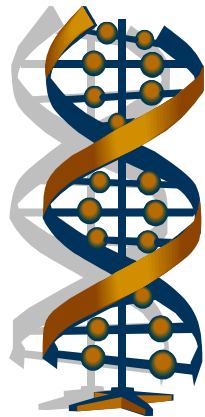
Algorithmic Complexity

- Algorithms can be characterized according to how their use of a resource scales with the size of the input(s).
- We typically look at two resources
 - space (memory consumption)
 - time (execution time)



Technically, these are independent *per algorithm*, but we often think of a family of algorithms that represent a spectrum of trade-offs between space and time

Example: space/time trade-off



$$f(x) = \begin{cases} 0 & \text{if } x = 'T' \\ 1 & \text{if } x = 'C' \\ 2 & \text{if } x = 'A' \\ 3 & \text{if } x = 'G' \\ -1 & \text{otherwise} \end{cases}$$

Example: space/time trade-off

4 time steps
0 memory

```
int f(char ch)
{
  → if (ch == 'T')
      return 0;
  → if (ch == 'C')
      return 1;
  → if (ch == 'A')
      return 2;
  → if (ch == 'G')
      return 3;
  return -1;
}
```

1 time step
256b “extra” memory

→ *// one-time initialization:*

```
int[] map = Enumerable.Repeat<int>(-1, 256).ToArray();
map['T'] = 0;
map['C'] = 1;
map['A'] = 2;
map['G'] = 3;
// ...
```

→

```
int f(char ch)
{
  return map[ch];
}
```

Best/average/worst case

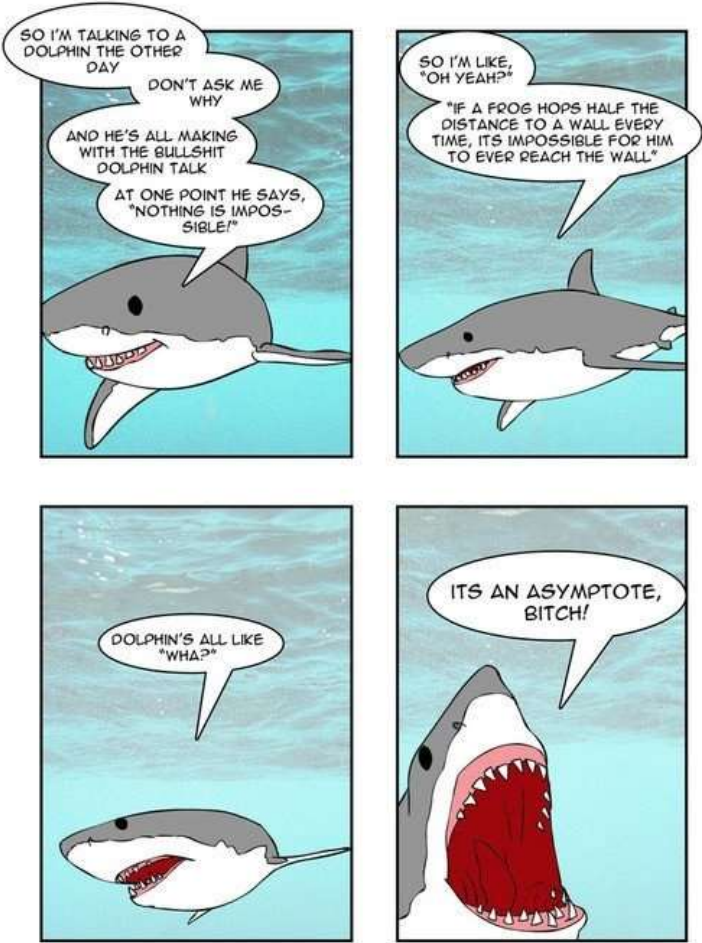
- If an algorithm's performance depends on more than just the size of the input, what is the possible range of complexity for a given input?
- Examples:
 - a **sort** algorithm may perform differently depending on the input ordering
 - intuitively, shouldn't sorting have complexity that's inversely proportional to the **entropy** of the input?
 - Xiaoming Li et al. (2007) *Optimizing Sorting with Machine Learning Algorithms*. IEEE
 - inserting into a **balanced tree** may perform differently depending on the current geometry

Asymptotic notations

- Express resource use in terms of input size
- Discard factors that become asymptotically irrelevant

Name	Notation	Expresses
Big-O	$f(n) = O(g(n))$	Upper bound
Big Omega	$f(n) = \Omega(g(n))$	Lower bound
Big Theta	$f(n) = \Theta(g(n))$	Upper and lower bound

- Usually when people say $O(n^2)$ “O of n-squared” etc. they actually mean $\Theta(n^2)$ “Theta of n-squared”
 - because usually, analysis of best case, worst case, and average case must be done separately
 - I will continue this tradition of abuse (i.e. on the next slide)



Complexity Classes

$O(k)$	constant	hash table
$O(\log n)$	logarithmic	binary search
$O(n)$	linear	naïve search (best and worst case)
$O(n \log n)$	log-linear	quicksort (best case)
$O(n^2)$	quadratic	naïve sort (best and worst case)
$O(n^3)$	cubic	parsing context-free-grammar (worst case)
$O(n^k)$	polynomial	2-SAT (boolean satisfiability)
$O(k^n)$	exponential	traveling salesman DP
$O(n!)$	factorial	naïve traveling salesman

Complexity scaling

- Consider an algorithm containing 8 million instructions
- State-of-the-art processor 2010 (1.48×10^{11} instructions per second)

$n = 12$ → $n = 350$

$O(1)$	54 μ s.
$O(\log n)$	58 μ s.
$O(n)$	649 μ s.
$O(n \log n)$	700 μ s.
$O(n^2)$	7.8 ms.
$O(n^3)$	93 ms.
$O(n^k), k = 8$	23 sec.
$O(k^n), k = 8$	1032 hours
$O(n!)$	2,231,000 centuries

$O(1)$	54 μ s.
$O(\log n)$	137 μ s.
$O(n)$	19 ms.
$O(n \log n)$	48 ms.
$O(n^2)$	6.6 sec.
$O(n^3)$	38.6 min.
$O(n^k)$	387 million years
$O(k^n)$	age of universe $\times 10^{294}$
$O(n!)$	age of universe $\times 10^{718}$

Hashing

- Hashing is crucial for indexing very large datasets
- HashSet: $O(1)$ test for membership
- “Dictionary” or “HashMap” or “Associative array” etc. $O(1)$ lookup
 - directly access one item based on the value of another
- All modern programming languages have hashing support
 - you usually won’t need to write your own hash functions

Hash function

A **hash function** f directly computes the index u of an element x in array g :

$$u = f(x), u \in \mathbb{Z}$$

$$g_u := x$$

The ideal function should have the following properties:

- $O(1)$ to compute
 - in practice, $f(x)$ is often $O(n)$ in the size of a single input
- Generate unique u for every x
- u is uniformly distributed over some range for all possible values of x

Hashing issues

- Collisions
- Quality of Hash function
- Perfect hash functions

Workhorse data structures

All modern languages have a number of off-the-shelf collection objects. For example C# offers the following, all strongly-typed:

<code>T[] (Array<T>)</code>	Fixed-size, ordered list of objects or values of type T
<code>List<T></code>	Dynamically-sized, ordered list of objects or values of type T
<code>Stack<T></code>	LIFO stack of objects or values of type T
<code>Queue<T></code>	FIFO queue of objects or values of type T
<code>LinkedList<T></code>	Doubly-linked list of objects or values of type T
<code>HashSet<T></code>	Hash table of objects or values of type T
<code>Dictionary<TKey,TValue></code>	Table of objects of type TValue hashed by objects of type TKey
<code>SortedDictionary<TKey,TValue></code>	Sorted table of objects of type TValue hashed by objects of type TKey
<code>ILookup<Tkey,Telement></code>	Multimap (an immutable dictionary that allows duplicate keys)

Optimization fever

“Early optimization is the root of much evil.”

- Donald Knuth (or, misattributed to Sir C. Anthony R. Hoare, developer of quicksort)

“Make everything as simple as possible, but not simpler.”

- Albert Einstein

The best—and uncontroversial—advice:
measure, measure, measure

Measuring performance

- Theoretical knowledge of your algorithms
 - Big-O complexity classes
 - Combining classes
- Actual stopwatch measurements

```
using System;  
using System.IO;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Diagnostics;  
  
class MyProgram  
{  
    static void Main(string[] args)  
    {  
        Stopwatch stopw = Stopwatch.StartNew();  
        // ... do stuff ...  
        Console.Error.WriteLine(stopw.ElapsedMilliseconds);  
    }  
}
```



Multithreading?

- This can be a real boost, but how well do you know your hardware?
- The 8-processor machines in the treehouse cluster –each support 8 Condor nodes
- To do 50-way multiprocessing on Condor, issue 50 single-threaded jobs



Immutable strings

- For modern programming languages, a key performance bottleneck is garbage collection
- Immutable strings provide many benefits, but GC activity can soar if you're not careful

```
String s = File.ReadAllText("chr1.dna");  
s = s.ToUpper();
```

- If you're going to be modifying a string, use mutable character arrays or special mutable “StringBuilder” objects instead

Binary trees

- Tree data structures are useful when you'll be doing a lot of inserting and deleting and you want to retain $O(\log n)$ access
 - Don't confuse with **decision tree classifiers**, which you'll get to implement in 572
- In practice, I have found that HashMaps (dictionaries) are better suited for the computational linguistics problems I've encountered
- However, one problem that cannot be solved with everyday data structures is the "all substrings" problem
 - For this, we'll look at a special kind of tree, the prefix trie, or simply "trie"

Trie

- Also known as “prefix trie”
- Pronounced either way: “tree” or “try”
- You will implement a trie for Project 4
- A trie is the most efficient way to search for multiple arbitrary-length substrings in a string

Example

- Identify an arbitrary number of string features in web browser user agent strings

```
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:2.0a1pre) Gecko/2008041102 Minefield/4.0a1pre
Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.9) Gecko/2008052906 Firefox/3.0
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
Mozilla/5.0 (compatible; OSS/1.0; Chameleon; Linux) MOT-U9/R6632_G_81.11.29R BER/2.0 Profile/MIDP-2.0 Configuration/CLDC-1.1
SAMSUNG-SGH-E250/1.0 Profile/MIDP-2.0 Configuration/CLDC-1.1 UP.Browser/6.2.3.3.c.1.101 (GUI) MMP/2.0
LG/U8130/v1.0
SonyEricssonC901/R1EA Browser/NetFront/3.4 Profile/MIDP-2.1 Configuration/CLDC-1.1 JavaPlatform/JP-8.4.2
ZTE-V8301/MB6801_V1_Z1_VN_F1BPa101 Profile/MIDP-2.0 Configuration/CLDC-1.1 Obigo/Q03C
Mozilla/5.0 (iPhone; U; CPU iPhone OS 3_0 like Mac OS X; en-us) AppleWebKit/420.1 (KHTML, like Gecko) Version/3.0 Mobile/1A542a Safari/419.3
Mozilla/5.0 (iPhone; U; CPU iPhone OS 4_0 like Mac OS X; en-us) AppleWebKit/532.9 (KHTML, like Gecko) Version/4.0.5 Mobile/8A293 Safari/6531.22.7
Mozilla/5.0 (iPod; U; CPU iPhone OS 3_1_1 like Mac OS X; en-us) AppleWebKit/528.18 (KHTML, like Gecko) Mobile/7C145
Mozilla/5.0 (iPad; U; CPU OS 3_2 like Mac OS X; en-us) AppleWebKit/531.21.10 (KHTML, like Gecko) Version/4.0.4 Mobile/7B367 Safari/531.21.10
SIE-S68/36 UP.Browser/7.1.0.e.18 (GUI) MMP/2.0 Profile/MIDP-2.0 Configuration/CLDC-1.1
SIE-EF81/58 UP.Browser/7.0.0.1.181 (GUI) MMP/2.0 Profile/MIDP-2.0 Configuration/CLDC-1.1
9700 Bold: BlackBerry9700/5.0.0.423 Profile/MIDP-2.1 Configuration/CLDC-1.1 VendorID/100
```

Problem statement

```
String[] targets =  
{ "Mozilla/5.0", "Trident/4.0", "Firefox/3",  
  "iPhone", "Mac OS X", "BlackBerry", "Safari" };  
  
String ua = "Mozilla/5.0 (iPod; U; CPU iPhone OS 3_1_1 like Mac OS X; en-us) AppleWebKit/528.18 (KHTML  
foreach (String feat in AllSubstrings(ua, targets))  
    Console.WriteLine(feat);  
  
//...  
  
IEnumerable<String> AllSubstrings(String s_in, IEnumerable<String> targets)  
{  
    yield return "hmm...";  
}
```

$O(n^2)$?

Naïve solution

- This is the first solution that should come to mind
 - It is appropriate for small inputs
 - It is easy to understand and trivial to implement
 - Test this on the input set. Maybe it's adequate
 - It provides a baseline for optimization work

```
IEnumerable<String> AllSubstrings(String s_in, IEnumerable<String> targets)
{
    for (int i0 = 0; i0 < s_in.Length; i0++)
        foreach (String t in targets)
            if (i0 + t.Length <= s_in.Length && s_in.Substring(i0, t.Length) == t)
                yield return t;
}
```

- Ok, what if it's not adequate?

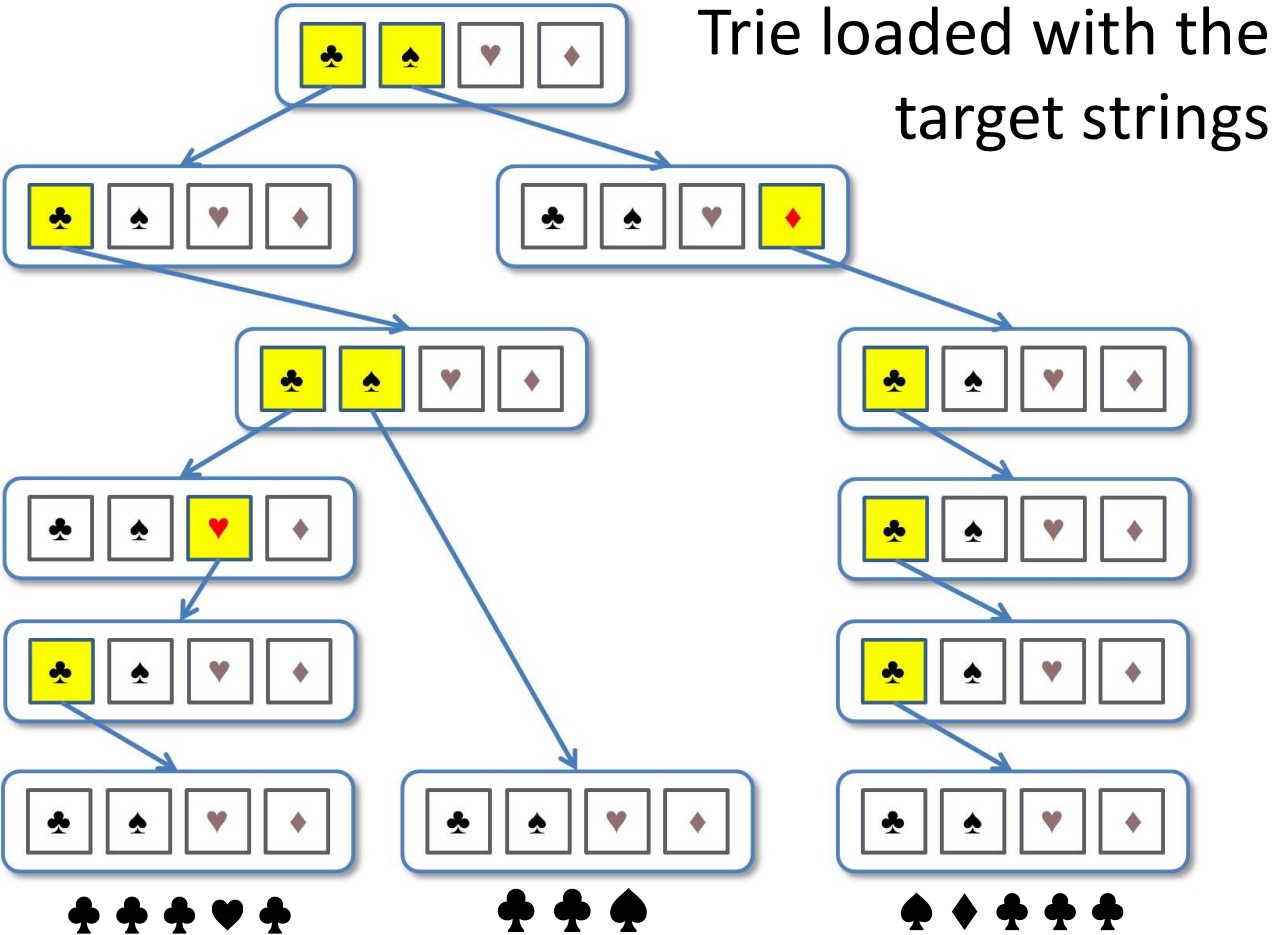
Using a trie for the all substrings problem

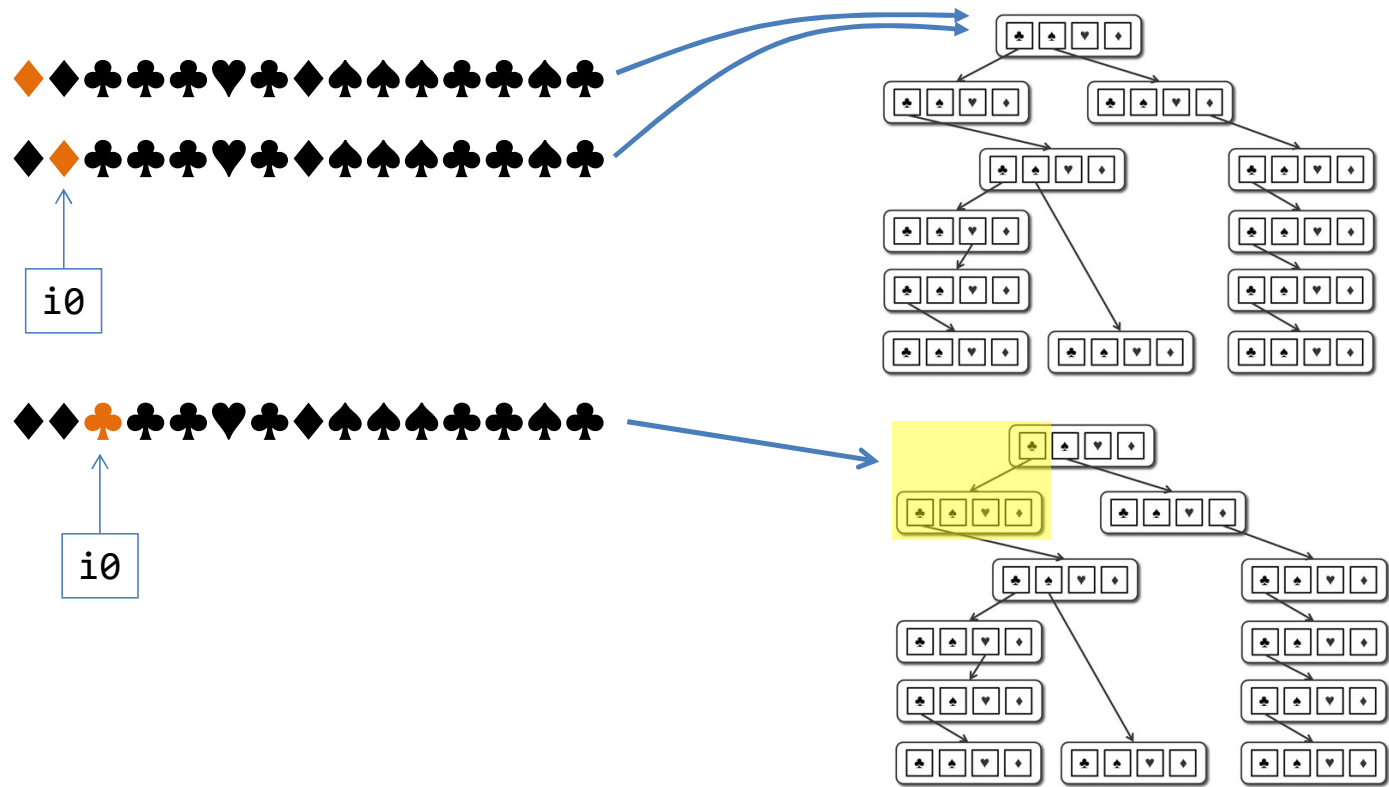
1. Build a trie from the target strings
 - The targets are all now stored implicitly in the trie, you can release them
2. Scan through the corpus once, using the trie to simultaneously check for all targets

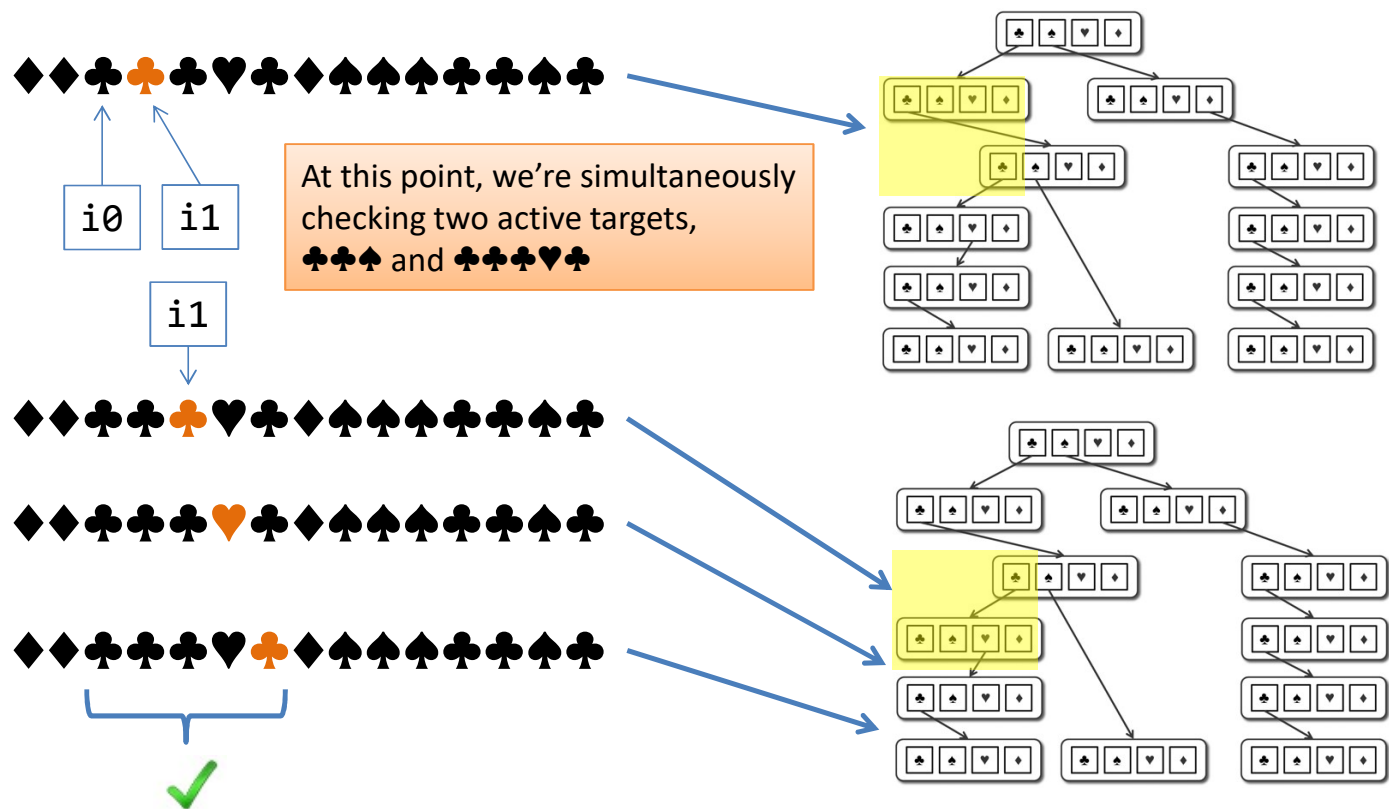
Trie example

```
String[] targets = { "♣♣♣", "♣♣♣♥♣", "♠♦♣♣♣" };  
  
String corpus = "♦♦♣♣♣♥♣♦♠♠♠♣♣♣♣";  
//           0           1  
//           012345678901234
```

- target “♣♣♣♠” found at position 11
- target “♣♣♣♥♣” found at position 2







Found: corpus.Substring(i0, i1 - i0 + 1)
Output and continue from i0 + 1

Walking through a trie

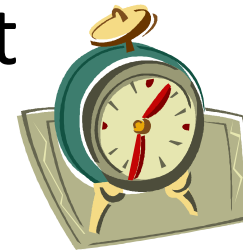
- To use a trie for finding all substrings, for each character c_i :
 - attempt to navigate the trie while moving forward $c_{j+1}...$, returning substrings, as they are found in the trie
 - After reaching the end of the trie, advance to next character c_{i+1} and repeat
- Recursion is not necessary because we always progress downwards from the trie root
 - You only need a reference to the “current” trie node
 - For this same reason, a trie node does not need to have a back-pointer to its parent node

Trie applications

- A trie guarantees that there is exactly one unique node for every possible prefix of every target
- If several targets share the same prefix, they share those trie nodes

Q: Considering this, what characteristics would a set of target strings have, such that a trie is a good candidate data structure?

Project 3: Tokenize Thai Text



- This project could be challenging
- 10-minute rule:
 - If you are stuck for 10 minutes, post your question to Canva
 - Other people have probably run into the same problem
 - Often asking the question helps you answer it for yourself

Project 3

- Goals
 - Implement an FST
 - Become familiar with non-English character encoding

Project 3

Divide each line of Thai text into syllables

คู่แข่งชั้นต่างก็คุมเชิงกัน
เขาเจียบไปครู่หนึ่งแล้วพุดขึ้น
เธอหันมาค้ำทรายขึ้นมาใหม่

Process each line, character by character

Change state according to the finite state machine

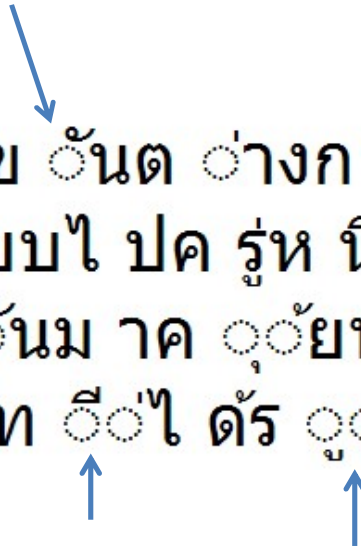
Emit spaces for states 7, 8, and 9

State #	Action	Transition
0	Accept V1/C1	V1 → 1 C1 → 2
1	Accept C1	C1 → 2
2	Accept C2/V2/T/V3/C3/V1/C1	C2 → 3 V2 → 4 T → 5 V3 → 6 C3 → 9 V1 → 7 C1 → 8
3	Accept V2/T/V3/C3	V2 → 4 T → 5 V3 → 6 C3 → 9
4	Accept T/V3/C3/V1/C1	T → 5 V3 → 6 C3 → 9 V1 → 7 C1 → 8
5	Accept V3/C3/V1/C1	V3 → 6 C3 → 9 V1 → 7 C1 → 8
6	Accept C3/V1/C1	C3 → 9 V1 → 7 C1 → 8
7	Break before previous character	→ 1
8	Break before previous character	→ 2
9	Break now	→ 0

Start state: 0
Accept (final) states: 7, 8, 9

Incorrect Thai tokenization

Some operating systems (i.e. Windows) will show a dotted circle for an invalid sequence of Unicode combining characters. This tells you that your result is not correct.



คู่แ ข่งข ั้นต ่างก ั้ค ุมเ ชิงก ั้น
เขาเ จียบไ ปค รุ่ห ึ่งแ ล้วพ ุดข ั้้น
เธอห ั้นม าค ุ้ยท รายข ั้้นม ่าใ หม่
ยินด ั้ท ั้ไ ด้ร ุ้จ ักค ุณ

Using HTML <meta> tag to specify encoding

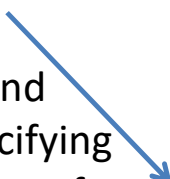
```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

Additional step for reading and writing:

- Save the file using the specified encoding
- Make sure your editor is set to read or write the encoding
- This requires using an editor that is capable of doing so

Additional step for using the page on a web server

- Configure the web server to send a matching HTTP header
- This is an out-of-band mechanism for specifying the content encoding of every single web page



```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
```

Programming language support for encodings

```
String s = "สวัสดีครับ";           // C# strings are always unicode
int i = s.Length;                   // number of code points: 10
Char ch = s[0];                     // always a 16-bit character.
                                    // In this case the value is 3626 (U+0E2A)

Byte[] bytes = Encoding.GetEncoding("TIS-620").GetBytes(s);
i = bytes.Length;                   // number of bytes: 10
byte b = bytes[0];                  // a byte. In this case, the value is 202 (\xCA)

bytes = Encoding.UTF8.GetBytes(s);
i = bytes.Length;                   // number of bytes: 30

s = Encoding.UTF8.GetString(bytes); // back to the original string
```

Project 3: Encoding issues

1. Your text editor
 - Can your **editor** create, display and save a file with wide literals (typ. UTF-8)?
2. Language support for wide characters
 - Does your **compiler** support wide literals in source code files?
 - If so, does it need/expect/reject a BOM? Or do you perhaps need to specify the source file's encoding on the compiler command line?
 - If not, does it support wide characters through via ASCII escape?
3. Utility programs:
 - Does your **FTP program** correctly understand UTF-8 files when it tries to convert line-endings from Windows to Unix? (try binary mode)
4. At runtime:
 - Does your **environment** have functions for reading unicode files?
 - Which input file are you trying to read? { UTF-8, UTF-16LE, TIS-620 }
 - Does your environment have functions for writing unicode files?
 - What output encoding are you trying to write? { UTF-8, UTF-16LE, TIS-620 }

Next Time...

- Corpus Linguistics