LING/C SC/PSYC 438/538

Lecture 11

Sandiway Fong

Today's Topics

- Homework 7 note
- *The lines of code that changed everything*

Getting deeper into Perl regex

- *capture*
- *backreferences*
- *shortest vs. greedy matching*
- *nondeterminism* (*backtracking*)

Recall last lecture: Digital Advertising

- *If you needed any more proof of the absolutely lucrative nature of Search*

Sherman Antitrust Act violation?

... suppressed competition by engaging in anticompetitive conduct. Monopolization offenses may be prosecuted criminally or civilly.

*The government has argued that Google, which has some 90% of the search market, illegally paid $10 billion annually to smartphone makers like Apple (AAPL.O) and wireless carriers like AT&T (T.N) and others to be the default in search on their devices in order to stay on top.

*The clout in search then makes Google a heavy hitter in the lucrative advertising market, boosting its profits.

*DuckDuckGo has around 2.5% of the online search engine market because it has not been able to win a default position on devices made by big companies.

Apple and Google

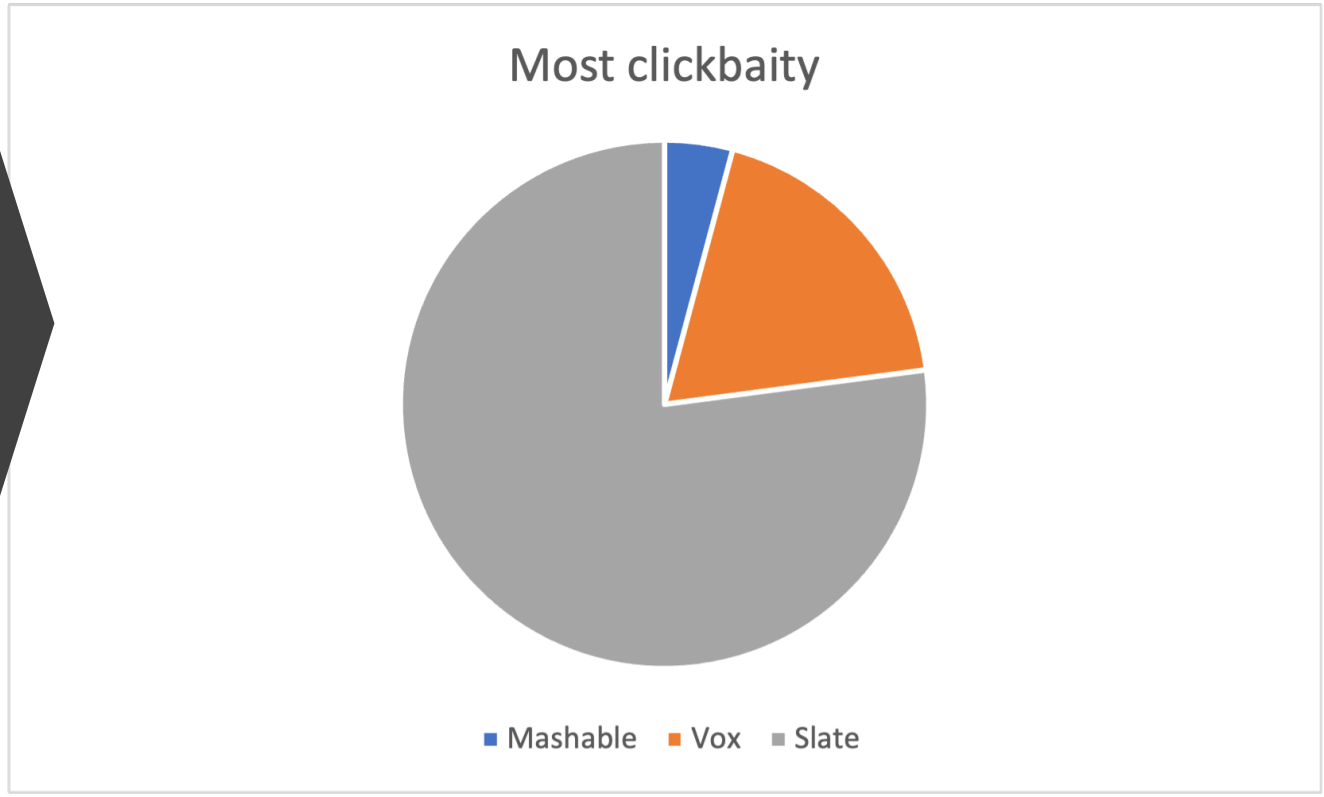
Apple won't make a Google search rival, says Cue

William Gallagher | *Sep 26, 2023*

18 Facebook Twitter Reddit

Apple executive Eddy Cue is expected to testify in court that the company has no plan to make an "Apple Search" engine, because its deal with Google is the best for users.

What the class thought



ChatGPT

Shared Chat • Default (GPT-3.5)

Analyzing Clickbait Titles

September 21, 2023

Can you determine whether an article has a clickbait title, based on the title, and the article content

Yes, I can help analyze an article title and its content to determine if it exhibits characteristics typical of clickbait.

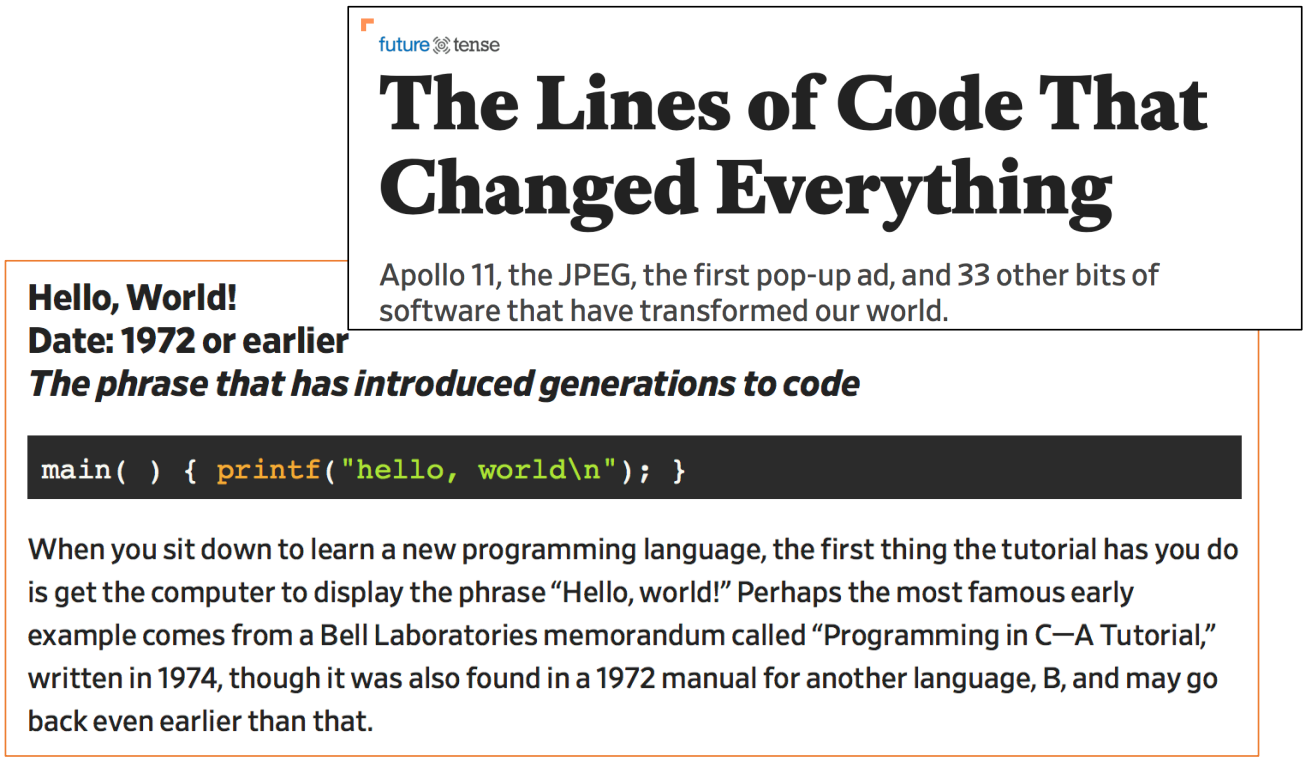
Clickbait is designed to attract attention and entice visitors to click on a link to a particular web page. Titles that are clickbait often exhibit some of these features:

ChatGPT: 6 Factors

1. **Exaggerated or misleading information:** If the title makes an extreme claim that isn't substantiated by the article content, it's probably clickbait.
2. **Withholding key information:** Titles like "You won't believe what happened next!" or "This one weird trick..." are designed to make readers curious, but they don't actually provide specific information.
3. **Strong emotional appeals:** Words like "shocking", "incredible", or "horrifying" that are used to evoke strong reactions but aren't supported by the article's content.
4. **Listicles with little substance:** While not all list articles (listicles) are clickbait, titles like "10 amazing ways to..." can be if the article doesn't deliver meaningful content.
5. **Pop culture references or celebrity names that are irrelevant:** If the title drops a well-known name or current event but doesn't actually discuss them in depth in the article, it's a sign of clickbait.
6. **Sensationalism:** Overhyping mundane information or events.

Interesting read

- https://slate.com/technology/2019/10/consequential-computer-code-software-history.html



future tense

The Lines of Code That Changed Everything

Apollo 11, the JPEG, the first pop-up ad, and 33 other bits of software that have transformed our world.

Hello, World!
Date: 1972 or earlier
The phrase that has introduced generations to code

```
main( ) { printf("hello, world\n"); }
```

When you sit down to learn a new programming language, the first thing the tutorial has you do is get the computer to display the phrase "Hello, world!" Perhaps the most famous early example comes from a Bell Laboratories memorandum called "Programming in C—A Tutorial," written in 1974, though it was also found in a 1972 manual for another language, B, and may go back even earlier than that.

Perl code and History

The Code That Made a T-Shirt Illegal
Date: Circa 1995
Language: Perl
One of the earliest examples of code as activism

```
#!/bin/perl -s-- -export-a-crypto-system-sig -RSA-3-lines-PERL
$m=unpack(H.$w,$m."\0"x$w),$_=`echo "16do$w 2+4Oi0$d*-^1[d2%Sa
2/d0<X+d*La1=z\U$n%0]SX$k"[$m*]\EszlXx++p|dc`,s/^.|\W//g,print
pack('H*',$_)while read(STDIN,$m,($w=2*$d-1+length($n)&~1)/2)
```

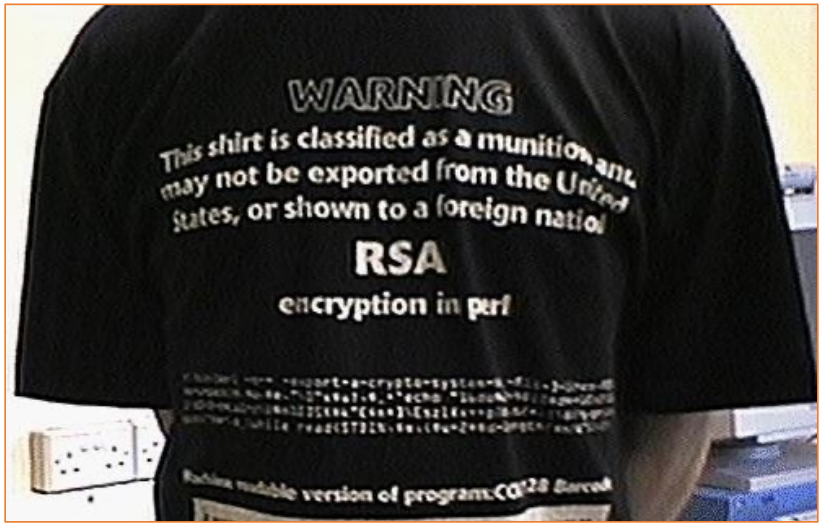
Munitions T-Shirt Homepage

"WARNING: This shirt is classified as a munition and may not be exported from the United States, or shown to a foreign national," the shirt warned. For a time, the United States government treated strong encryption like surface-to-air missiles: too dangerous to fall into the hands of America's foes. The idea made a kind of sense when encryption lived in heavy, expensive devices, but a lot less sense when the State Department tried to tell cryptography researchers in the 1990s they couldn't post their code on the internet. But the RSA encryption algorithm—one of the basic building blocks of modern cryptography—is elegant enough that it can be written out in just four dense lines of Perl code … short enough to fit on a T-shirt. The original shirts are now collector's items; the export controls, although not completely gone, have been substantially pared back. —*James Grimmelmann, professor of law at Cornell Tech and Cornell Law School*

Perl code and History

http://www.cypherspace.org/adam/rsa/rsa-details.html

```
#!/bin/perl -sp0777i<X+d*lMLa^*lN%0]dsXx++lMlN/dsM0<j]dsj
$/=unpack('H*',$_);$_=`echo 16dio\U$k"SK$/SM$n\EsN0p[lN*1
lK[d2%Sa2/d0$^Ixp"|dc`;s/\W//g;$_=pack('H*',/((..)*)$/)
```



Perl regex: Unicode and \w

- `\w is [0-9A-Za-z_]`

Definition is expanded for Unicode:

use **pragma:** https://perldoc.perl.org/open.html

```
use utf8;
use open qw(:std :utf8);
my $str = "school école École šola trường स्कूल škole โรงเรียน";
@words = ($str =~ /(\w+)/g);
foreach $word (@words) { print "$word\n" }
```

list context

```
bash-3.2$ perl regex_utf.perl
school
école
École
šola
trường
स्कूल
škole
โรงเรียน
```

```
school
cole
cole
ola
tr
ng
kole
```

Chapter 2: JM

RE	Match	Example Patterns Matched
*	an asterisk "*"	"K*A*P*L*A*N"
\.	a period "."	"Dr. Livingston, I presume"
\?	a question mark	"Why don't they come and lend a hand?"
\n	a newline	
\t	a tab	

Figure 2.8 Some characters that need to be backslashed.

Why is a backslash needed?

- ∗ means zero or more repetitions of the previous char/expr
- . means any single character
- ? means previous char/expr is optional (zero or one occurrence)

Chapter 2: JM

- Precedence of operators
  - Example: Column 1 Column 2 Column 3 …
  - `/Column [0–9]+ *`/
  - `/(Column [0–9]+ *)*/` space
  - `/house(cat(s|)|)/` (| = disjunction; ? = optional)
- Perl:
  - the regex matched by within the pair of parentheses **(…)** is stored/**captured** in global variables $1 (and $2 and so on).
  - (?: …) group but exclude from $*n* variable storage
- Precedence Hierarchy:

Parenthesis	`()`
Counters	`* + ? {}`
Sequences and anchors	`the ^my end$`
Disjunction	`|`

Perl regex

- Recall earlier lecture about time?

`http://perldoc.perl.org/perlretut.html`

```
1.  # extract hours, minutes, seconds
2.  if ($time =~ /(\d\d):(\d\d):(\d\d)/) {    # match hh:mm:ss format
3.      $hours = $1;
4.      $minutes = $2;
5.      $seconds = $3;
6.  }
```

returns 1 (true) or "" (empty if false)

A shortcut: **list** context for matching

```
1.  # extract hours, minutes, seconds
2.  ($hours, $minutes, $second) = ($time =~ /(\d\d):(\d\d):(\d\d)/);
```

returns a list

Chapter 2: JM

Backreferences

Closely associated with the matching variables `$1` , `$2` , ... are the *backreferences* `\1` , `\2` ,... Backreferences are simply matching variables that can be used *inside* a regexp. This is a really nice feature; what matches later in a

- s/([0-9]+)/<\1>/ ⟸ what does this do?

Backreferences give Perl regexs more expressive power than **Finite State Automata** (FSA)

The number operator can be used with other numbers. If you match two different sets of parenthesis, \2 means whatever matched the *second* set. For example,

```
/the (.*)er they (.*), the \1er we \2/
```

will match *The faster they ran, the faster we ran* but not *The faster they ran, the faster we ate*. These numbered memories are called **registers** (e.g., register 1, register 2,

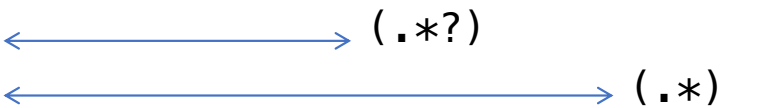
Shortest vs. Greedy Matching

- default behavior
  - in Perl regex matching:
    - *take the longest possible matching string*
    - *and see if it works*
    - *if so, ok*
    - *if not, backtrack (take a shorter match and try again)*
  - aka **greedy matching**
    - *This behavior can be changed, see next slide*

Shortest vs. Greedy Matching

from http://www.perl.com/doc/manual/html/pod/perlre.html

- Example:

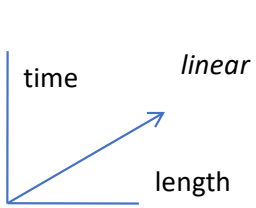


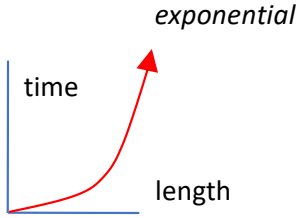

```
$_ =  "The food is under the bar in the barn.";
if ( /foo(.*?)bar/ ) {
        print "matched <$1>\n";
}
```

- Output:
  - greedy (.*): `matched <d is under the bar in the >`
  - shortest (.*?): `matched <d is under the >`
- Notes:
  - ? immediately following a repetition operator like * (or +) makes the operator work in non-greedy mode
  - + immediately following a repetition operator makes it **non-backtracking** greedy

Regex: exponential time

- Regex search is supposed to be fast
  - but searching is not necessarily proportional to the length of the string (or corpus) being searched
  - in fact, Perl RE matching can can take exponential time (in length)



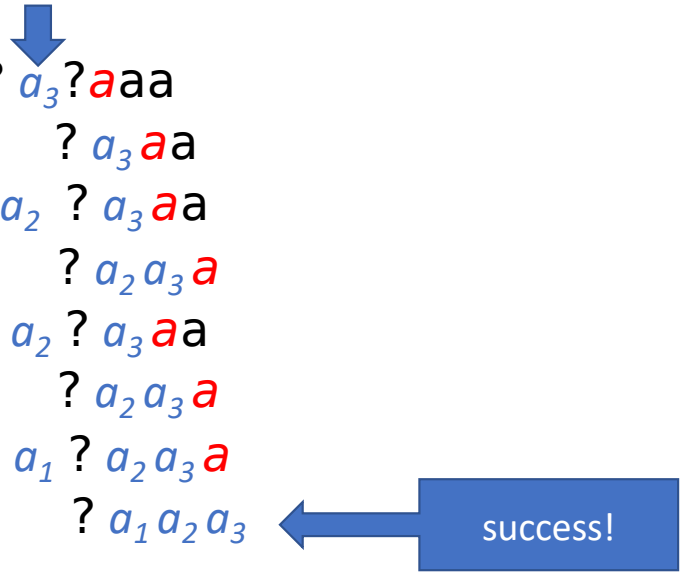


- **non-deterministic**
  - *may need to backtrack (revisit last choice point) if it matches incorrectly part of the way through*
  - Let's consider `a?a?a?aaa` matching against the string *aaa*

Regex: exponential time

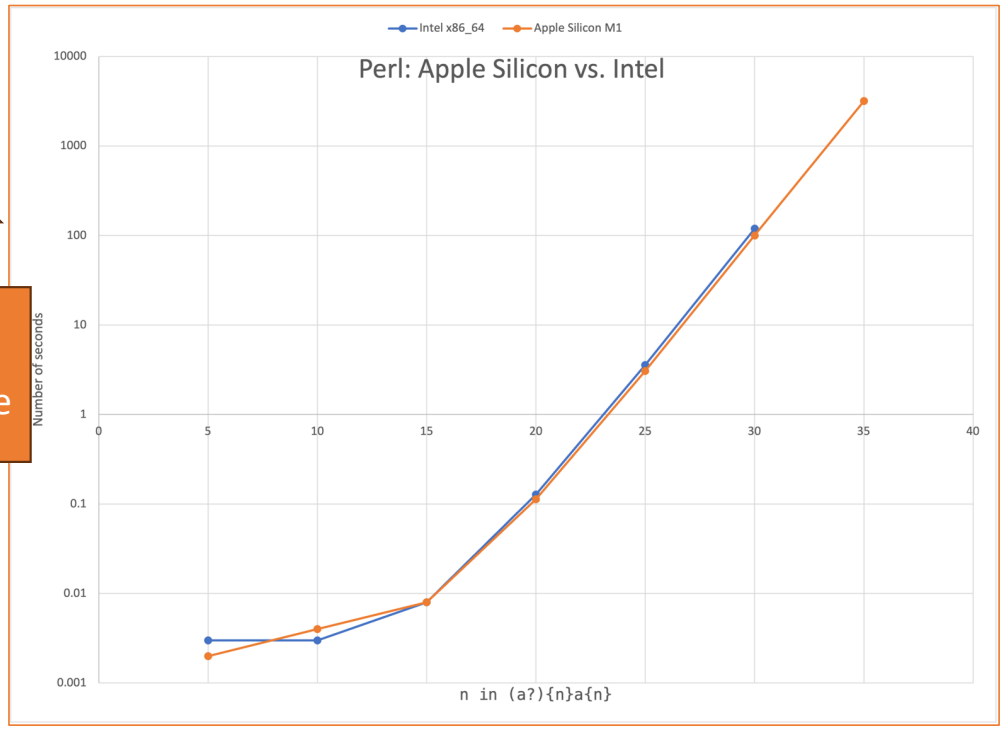
- Consider a?a?a?aaa matching against the string *aaa*
  - For expository purposes: $a_1\ a_2\ a_3$
  - red *a* = failure to match (causes backtracking)

Tries:



Perl implementations

- Now consider scaling up a?a?a?aaa, i.e. $(a?)^n a^n$ matching against a^n



- perl 5, version 28, subversion 3 (v5.28.3) built for darwin-thread-multi-2level
  - /opt/local/bin/perl: Mach-O 64-bit executable x86_64
- perl 5, version 34, subversion 1 (v5.34.1) built for darwin-thread-multi-2level
  - /opt/local/bin/perl: Mach-O 64-bit executable arm64

Reference:
https://swtch.com/~rsc/regexp/regexp1.html

Regex: exponential time

- regex $(a?)^n a^n$ matching against a^n for a range of values for *n*

```
time perl -e '$n = shift; $na = "a" x $n; print $na =~ /(a?){$n}a{$n}/' 25
real 0m3.201s
user 0m3.190s
sys 0m0.007s
```

- Note:
  - `shift` defaults to working on @ARGV, that's how $n gets 25 above.

shift ARRAY

shift

Shifts the first value of the array off and returns it, shortening the array by 1 and moving everything down. If there are no elements in the array, returns the undefined value. If ARRAY is omitted, shifts the `@_` array within the lexical scope of subroutines and formats, and the `@ARGV` array outside a subroutine and also within the lexical scopes established by the `eval STRING`, `BEGIN {}`, `INIT`