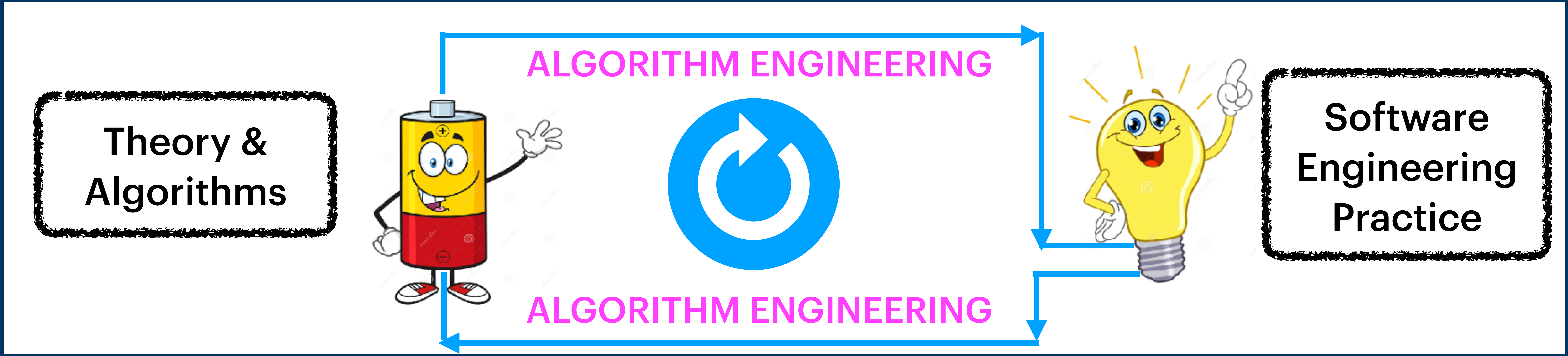


ALGORITHM ENGINEERING

Lecture 5:
Design of Experiments - II

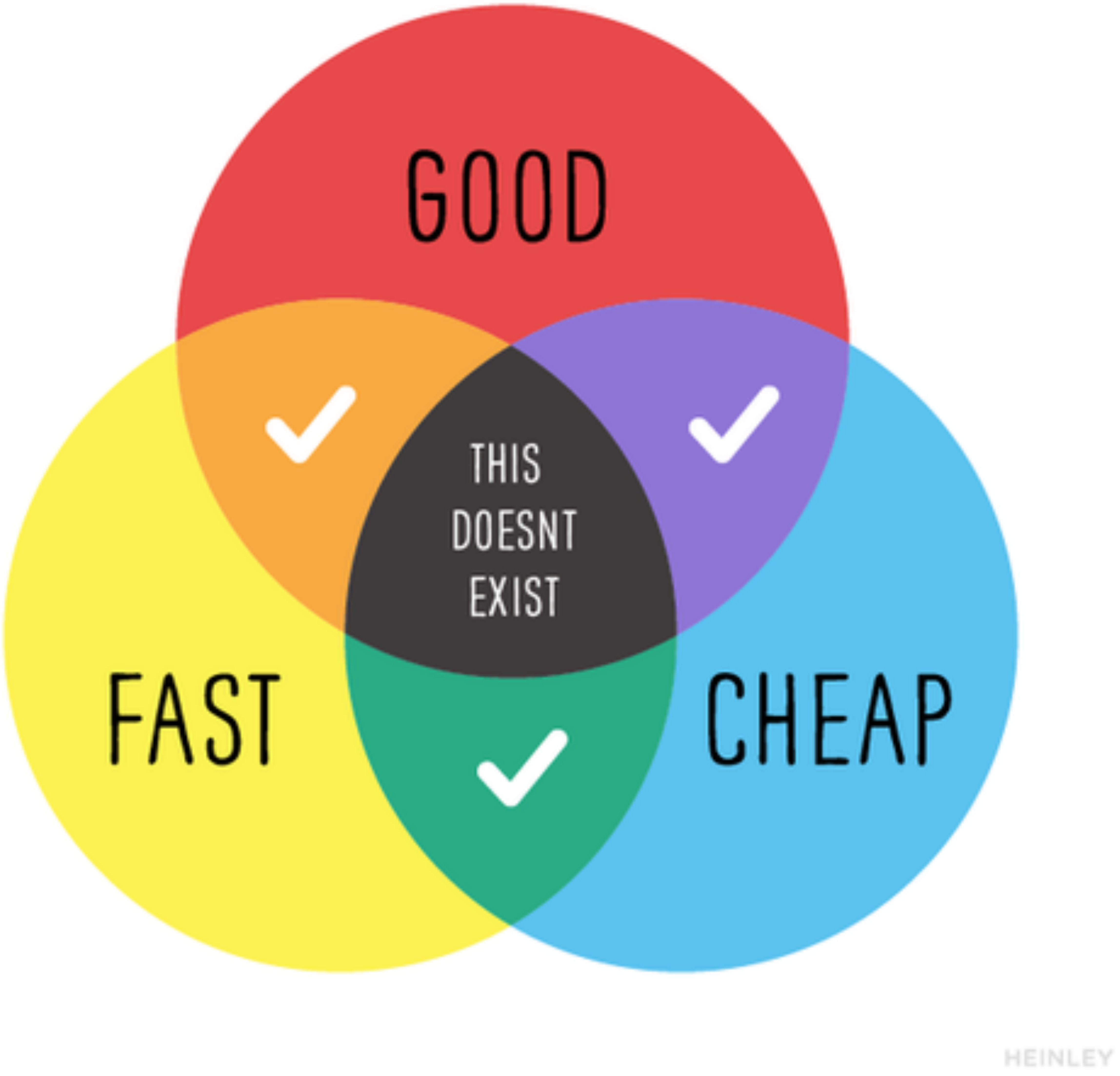
M. Oğuzhan Külekci - kulekci@itu.edu.tr

What to measure ?

Performance indicators :

- Time
- Space
- Solution quality

Usually, select two :)



Time Performance

Accuracy vs. Precision

Speed of light:

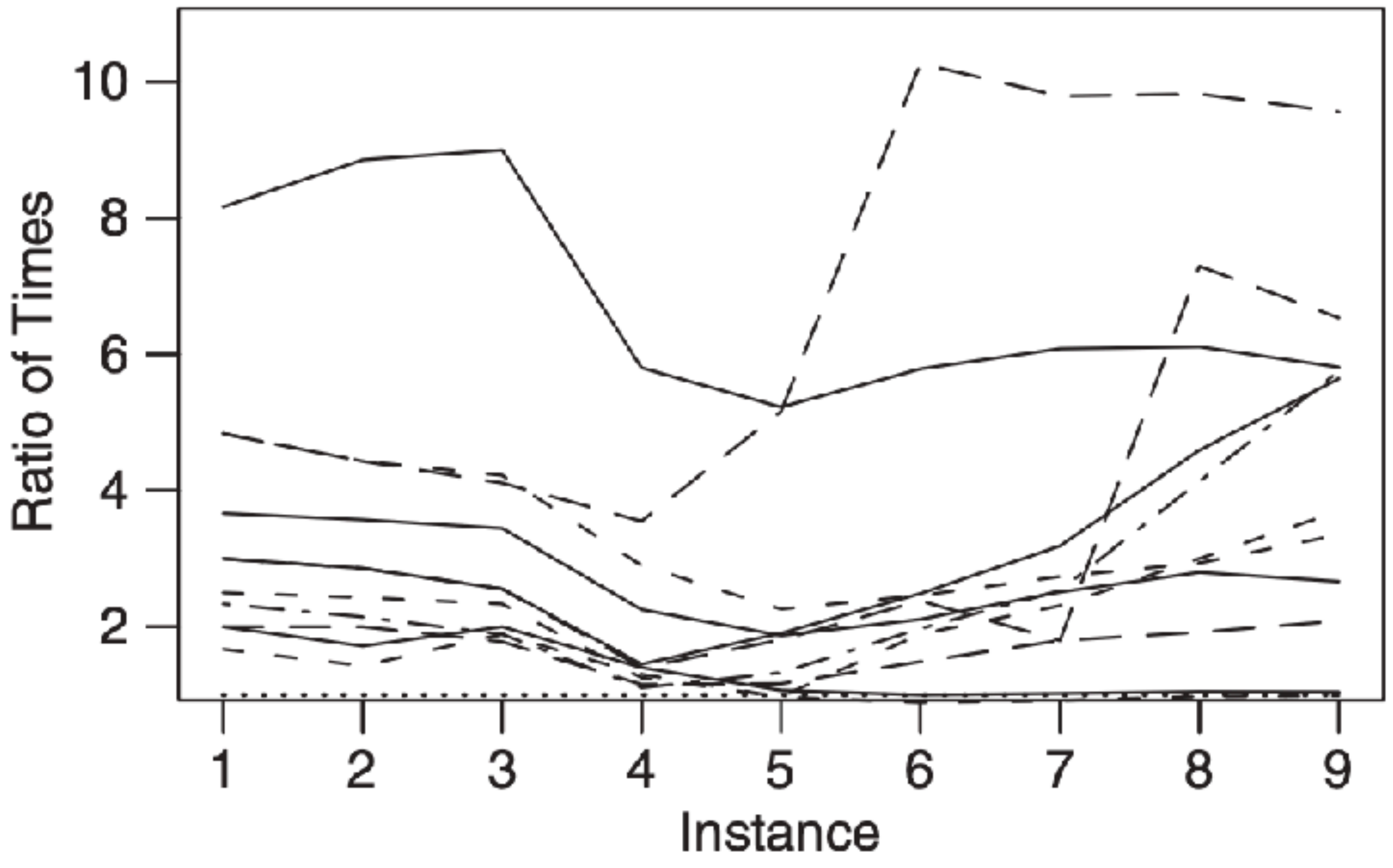
299,792.458 km/s

Year	Result
1879	$299,910 \pm 50$km/s
1926	$299,796 \pm 4$km/s
1935	$299,774 \pm 11$km/s

- Accuracy: How much close to the truth
- Precision: How much variation

Time Performance

Number of operations and Elapsed time



Accurate and general, but not precise

- **Dominant cost:** How many times the dominant operation performed, commonly in asymptotic notation, e.g., number of comparisons is $O(n \log n)$

Precise, but not accurate nor general

- **CPU time:** How much time elapse during the execution ?

Many repeats are required to improve confidence

A case study

Random text generation

ALL'S WELL THAT ENDS WELL ACT II Troy. DON JOHN And he, If I heard of all the giddiness of the pairs of Beatrice between her song and till they've swallowed me are in sin in jest. I his characters; He'd Lay on't. Perdita I think you beat thee. Fie on him: 'Tis false. Soft, swain, Pompey surnamed the PRINCESS, KATHARINE, ROSALINE, MARIA, KATHARINE, BOYET, ROSALINE, MARIA, and let him keep not o'erthrown by mis-dread.

From a given text, generate a new text that seems as human-written as possible.

2-Word Key	Suffix	2-Word Key	Suffix
this is	a	this is	a
is a	test	is a	test
a test	this	a test	of
test this	is	test of	the
this is	only	of the	emergency
is only	a	the emergency	broadcasting
only a	test	emergency broadcasting	system
a test	this	broadcasting system	this
test this	is	system this	is

this is a test this is only a test this is a test of the emergency broadcasting system

```
Text:        this is a test this is only a test
Indices:     0123456789012345678901234567890123
             0         1         2         3
```

Dictionary data structure

Key	Suffix	Index
a test	this	8
a test	this	28
is a	test	5
is only	a	20
only a	test	23
test this	is	10
test this	is	30
this is	a	0
this is	only	15

Figure 3.3. Words sorted by key. After sorting, the `word` array contains the indices on the right.

Automatic Text Generation

```
Text:        this is a test this is only a test
Indices:     0123456789012345678901234567890123
             0         1         2         3
```

Key	Suffix	Index
a test	this	8
a test	this	28
is a	test	5
is only	a	20
only a	test	23
test this	is	10
test this	is	30
this is	a	0
this is	only	15

```
// Text Generation Loop
// phrase = initialize to first k words of T

print (phrase);
for (wordsleft = m-k ;  wordsleft > 0;  wordsleft--) {
    // Lookup with binary search
    lo = -1;
    hi = nword;
    while (lo+1 != hi) {
        mid = lo+(hi-lo)/2;
        if (wordcmp(word[mid], phrase) < 0)  lo=mid;
        else hi = mid;
    }
    // hi = index of leftmost key that matches phrase

    // Random Select
    for (i = 0; wordcmp(phrase, word[hi+i]) == 0; i++)
        if (rand() % (i+1) == 0)
            p = skip(word[hi+i],k);  // get next suffix
    print (p);
    phrase = updatePhrase(phrase,  p) ;
}
```

n input words, m output word, k key-size

1. Construction of dictionary (initialization cost)

2. Binary search on the dictionary

3. Random selection among the results

$$\boxed{Cost = c_1 \cdot n \log n + c_2 \cdot m \log n + c_3 \cdot mn}$$

1 2 3

Automatic Text Generation

Analysis of the dominant cost function

$$Cost = c_1 \cdot n \log n + c_2 \cdot m \log n + c_3 \cdot mn$$

- Is this function correct?

- **Put counters in the software to count how many times the dominant operations is called.**
- Run the program on different instances with different m,n,k to observe how those counts change.
- Verify your hypothesis

Input file is the concatenation of the files below

File	Text	n
huckleberry	*Huckleberry Finn*, by Mark Twain	112,493
voyage	*The Voyage of the Beagle*, by Charles Darwin	207,423
comedies	Nine comedies by William Shakespeare	337,452
total	Combined comedies, huckleberry, voyage	697,368

Design points:

$k = 1,\ n \in \{10^5, 2 \cdot 10^5, 4 \cdot 10^5\},\ m \in \{10^5, 2 \cdot 10^5, 4 \cdot 10^5\}$

Read n words from the input file and run the program.

Doubling experiments:

STEP 1: check $n \log n$ term:

	$n = 10^5$	2×10^5	4×10^5
qcount/n	12.039	13.043	14.041

STEP 2: Check $m \log n$ term

	$n = 10^5$	$n = 2 \times 10^5$	$n = 4 \times 10^5$
$m = 10^5$	133,606	143,452	153,334
$m = 2 \times 10^5$	267,325	287,154	306,553
$m = 4 \times 10^5$	534,664	574,104	613,193

STEP 2: Check mn term

	$n = 10^5$	$n = 2 \times 10^5$	$n = 4 \times 10^5$
$m = 10^5$	1,234,334	2,298,021	4,500,283
$m = 2 \times 10^5$	2,367,523	4,617,355	9,009,292
$m = 4 \times 10^5$	4,933,294	9,394,869	18,284,907

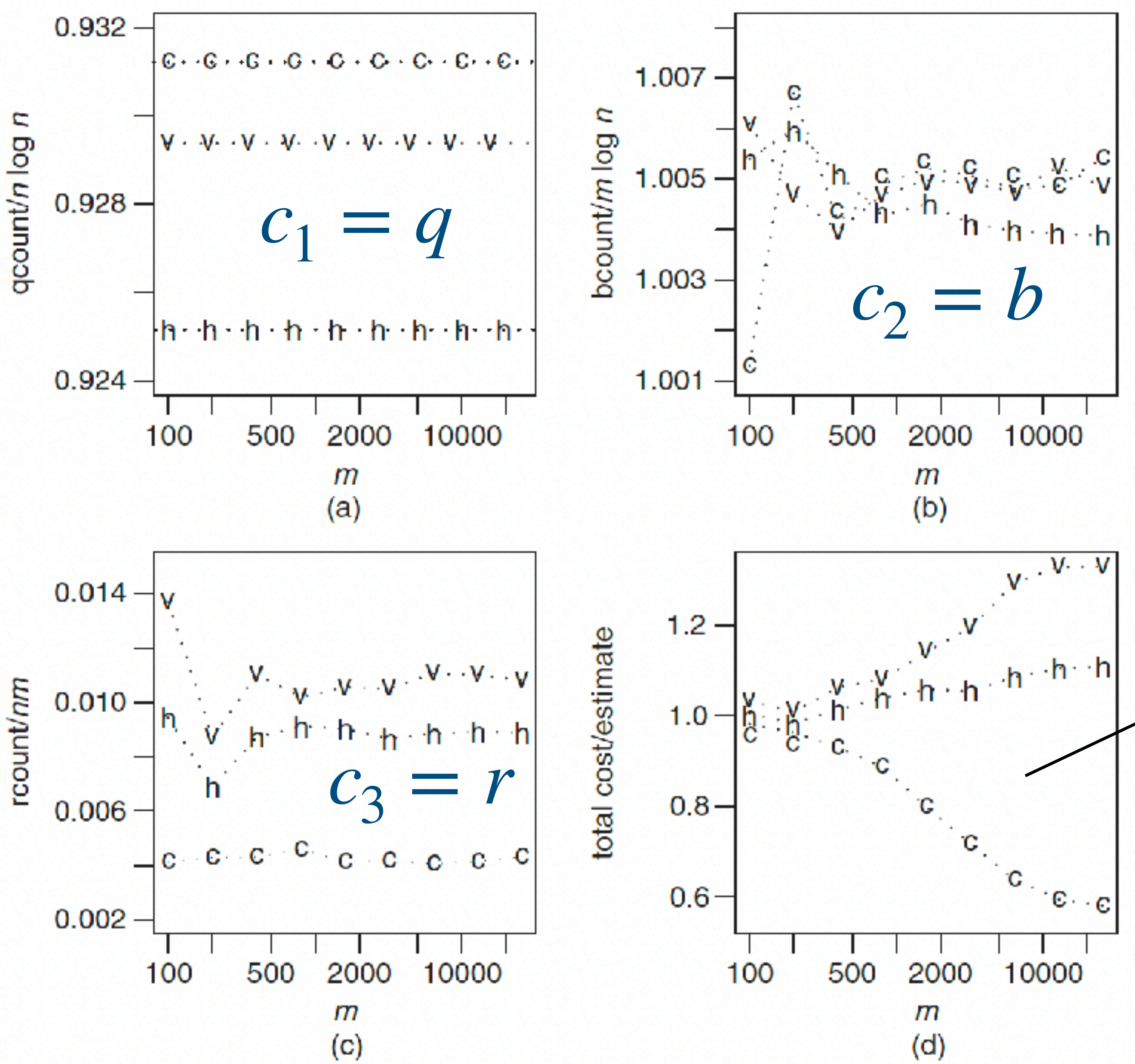
Automatic Text Generation

Analysis of the dominant cost function

File	Text	n
huckleberry	*Huckleberry Finn*, by Mark Twain	112,493
voyage	*The Voyage of the Beagle*, by Charles Darwin	207,423
comedies	Nine comedies by William Shakespeare	337,452
total	Combined comedies, huckleberry, voyage	697,368

$$\boxed{Cost = c_1 \cdot n \log n + c_2 \cdot m \log n + c_3 \cdot mn}$$

- What are the coefficients?



$$q = \text{qcount}/n \log_2 n$$

$$b = \text{bcount}/m \log_2 n$$

$$r = \text{rcount}/nm$$

$$W(n,m) = 0.9286 n \log_2 n + 1.0045 m \log_2 n + 0.0079 nm$$

Compare estimates with actual happenings? What you see? Why ?
- stems from the last term, c_3 ?

We assumed k=1 is fixed, so what happens if k varies?
READ THE STORY FROM THE Mcgeouch's book !!!

Clocks and Timers

- Two ways to measure time performance
    - Real time: Elapsed time, wall-clock time
    - CPU time: Time spent on processor to execute the program
- *Real time, system time, user time : see time command in Linux*
- Elapsed time measurement by cycle count or clock time
- CPU time measurement by interval time, hyper-threading effect !

Which one to use, and possible effects and scenarios. Please read them from the book.

Automatic Text Generation

Measuring time

Time command on linux

user_time+system_time

Real time

	Platform	Test	CPU	Wall
1	HP	Unoptimized, light	43.02	43.02
2	HP	Optimized, light	27.96	28.19
3	HP	Optimized, heavy 1	36.02	43.63
4	HP	Optimized, heavy 9	37.64	43.41

	Platform	Test	CPU	Wall
1	Mac	Unoptimized, light	108.15	115.18
2	Mac	Optimized, light	67.33	79.08
3	Mac	Optimized, heavy 1	96.97	630.06
4	Mac	Optimized, heavy 9	100.38	649.48

Two platforms, different timings, how to compare/analyse?

Automatic Text Generation

Measuring time

System	CPU user	CPU sys	Wall	Instr Count	Cycle Time
HP Opt	27.61	0.35	28.19	84.8×10^9	28.27
HP Unopt	43.96	0.40	43.35	128.7×10^9	42.93
Mac Opt	55.53	1.22	59.79	115.9×10^9	53.66
Mac Unopt	92.41	1.22	95.55	237.9×10^9	110.16

$$\boxed{\frac{84.8 \cdot 10^9}{3 \cdot 10^9}}$$

HP: 3Ghz, Mac: 2.2 Ghz

We can use instructions inside the code to measure the cycle, instruction count, time

Code Profilers

Extremely useful to understand the dynamics of execution

GPROF

```
$ gcc -pg markov.c -o markov
$ markov 1 1000000 <comedies.txt
$gprof

Flat profile:
Each sample counts as 0.01 seconds.
  %     cumulative   self                self      total
 time     seconds   seconds     calls   ns/call   ns/call   name
 64.98     13.31     13.31 1628788583    8.17      8.17     wordcmp
 34.38     20.36      7.04                                  main
  1.08     20.58      0.22                                  frame_dummy
  0.20     20.62      0.04                                  sortcmp
  0.00     20.62      0.00   3000000     0.00      0.00     skip
  0.00     20.62      0.00   1000000     0.00      0.00     writeword
```

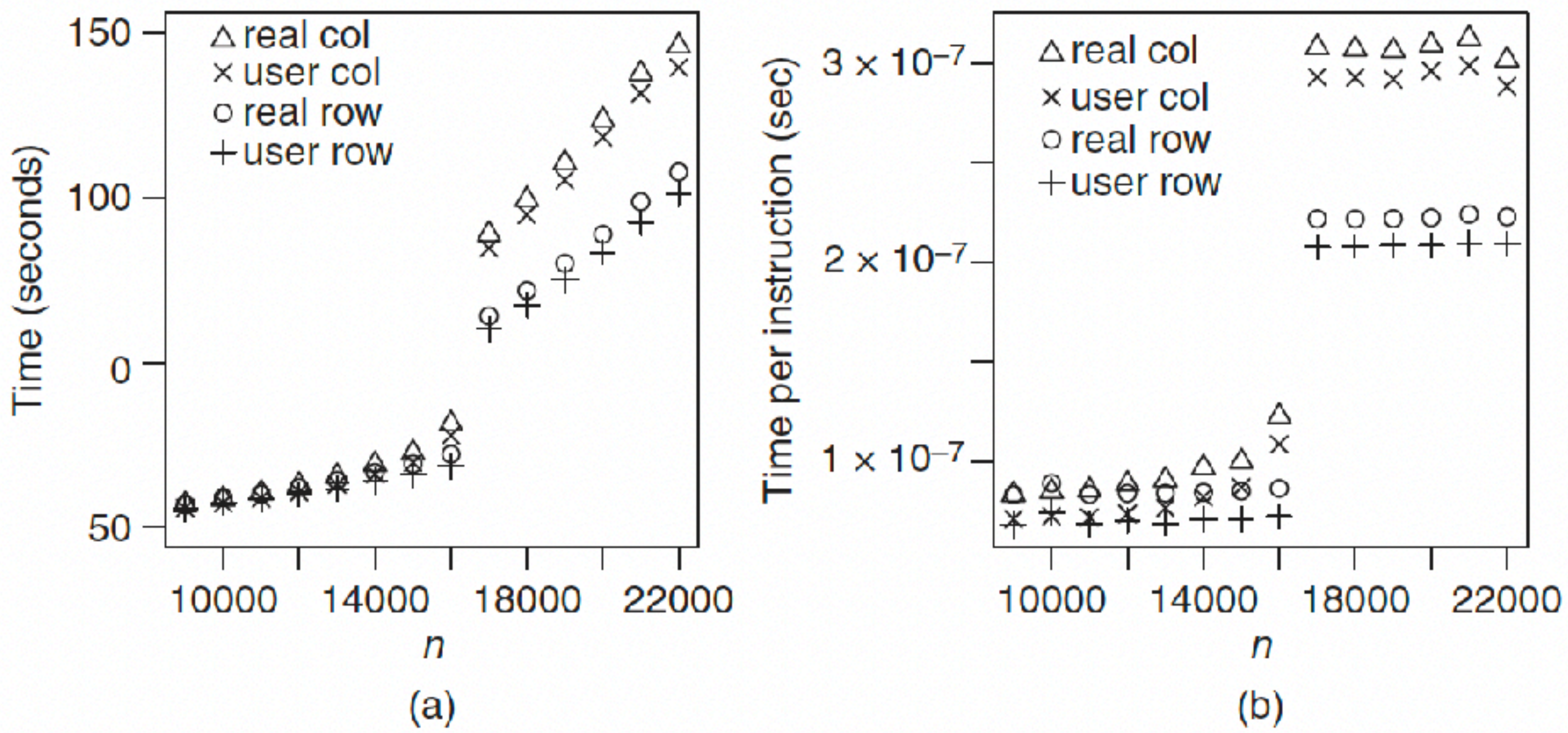
CACHEGRIND

```
gcc  -g  markov.c -o markov
valgrind  --tool=cachegrind  markov 1 100000 <comedies.txt
cg_annotate  cachegrind.out.1234   --auto=yes
```

```
                     int wordcmp (char *p, char* q)
  32,668,088          {
  16,334,044           int n = k;
 196,291,370           for ( ; *p == *q; p++, q++){
  91,500,879                if (*p == 0 && --n == 0)
   6,268,506                     return 0;
                        }
  53,461,943            return  *p - *q;
  16,334,044          }
```

Memory Consumption

We discussed memory hierarchy a lot in the previous lectures…



Row-major

```
for (r=0; r<n; r++)
    for (c=0; c<n; c++)
        C[r][c] = A[r][c] + B[r][c];
```

Column-major

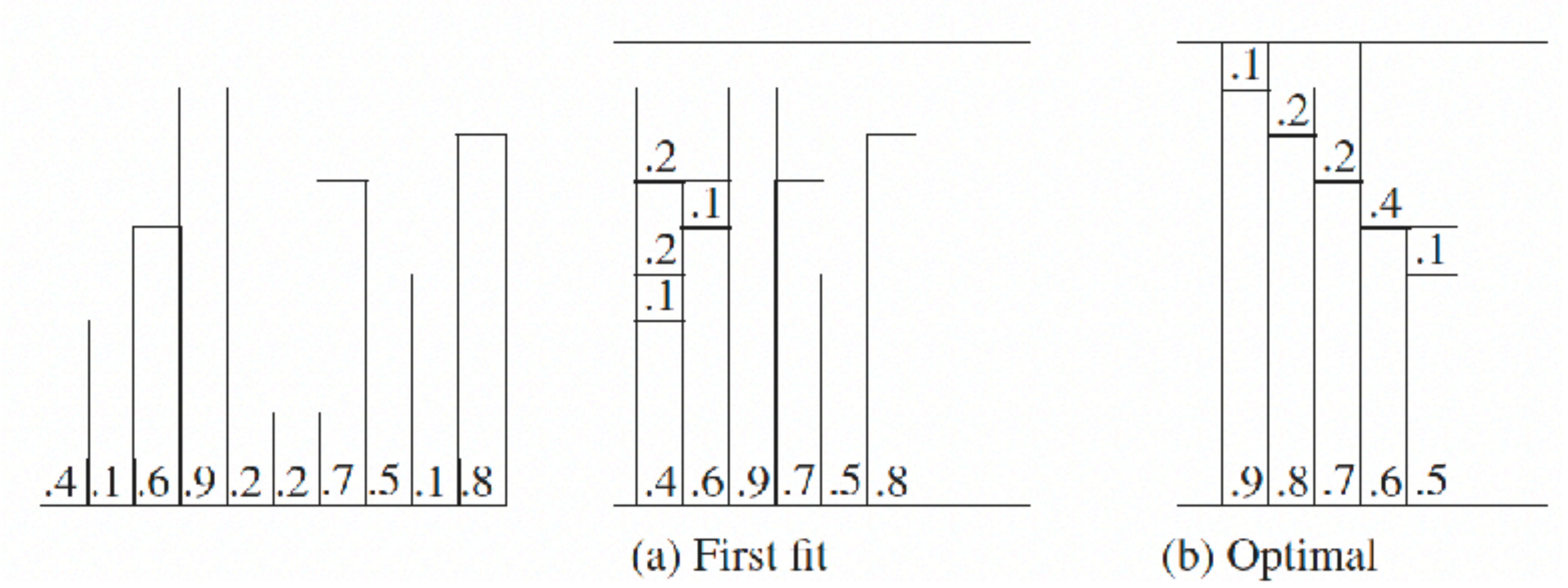
```
for (c=0; c<n; c++)
    for (r=0; r<n; r++)
        C[r][c] = A[r][c] + B[r][c];
```

Solution Quality

- **Combinatorial optimization problems** are NP-hard in general
- **Numerical problems** that approximate something
- Heuristics vs. algorithmic performance guarantees
- Need to compare produced solution with the optimal (?)
- **Performance indicator** by definition of the problem

Case Study

Bin-packing (Knapsack Problem)



6 packs used

5 packs used

Measuring the solution quality performance of first-fit experimentally:

- Generate n weights randomly in the range $[\ell, u], s.t.\ \ 0 < \ell < u \leq 1$
- Compute how many packs does first-fit generate?
- Compute the optimal solution for the generated sequence
- Repeat the same procedure t times and take the average

$$\overline{R} = \frac{1}{t} \sum_{i=1}^{t} \frac{F(L_i)}{O(L_i)},$$

PROBLEM : Computing optimum solution is NP-hard !

Case Study

Biased, easy-to-compute estimators instead of unbiased, hard-to-compute ones

PROBLEM : Computing optimum solution is NP-hard !

Try to find some lower or upper bounds that are easier to compute, and measure the performance according to them

$$\overline{R} = \frac{1}{t}\sum_{i=1}^{t}\frac{F(L_i)}{O(L_i)},$$

$$? < O(L_i) < ?$$

Two alternatives for bin-packing:

- $S(L_i)$: Ceil of the um of the numbers in the list (each item is in [0,1], $S(L_i) \le O(L_i)$
- $H(L_i)$: Number of items larger than 0.5, since each should be placed in a separate bin, $S(L_i) \le H(L_i) \le O(L_i)$

Compute R ratio according to S and H indicators !

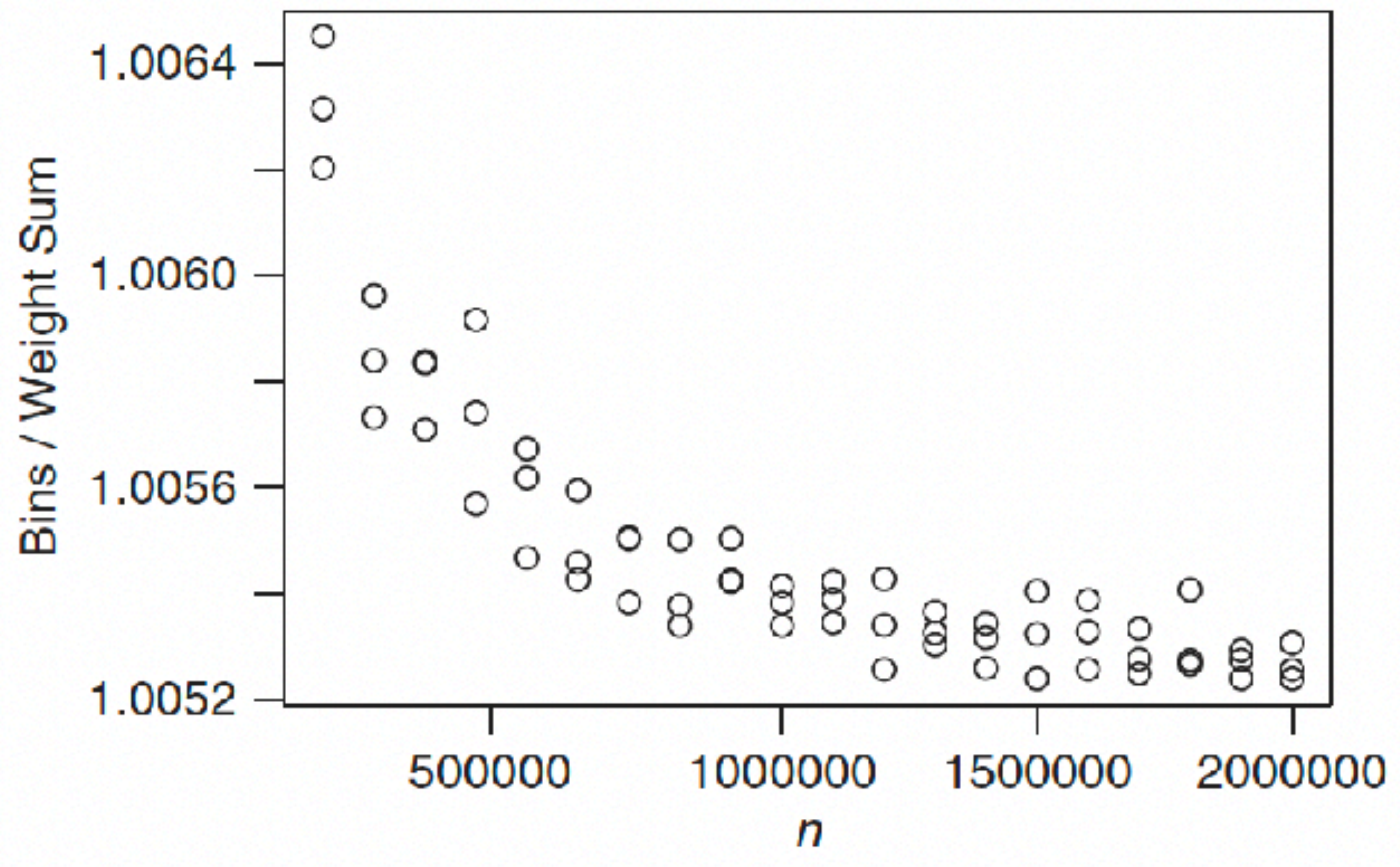


Figure 3.12. First fit. Measurements of $R_1(L_i)$ for parameters $l = 0$, $u = 0.5$, three random trials at each design point.

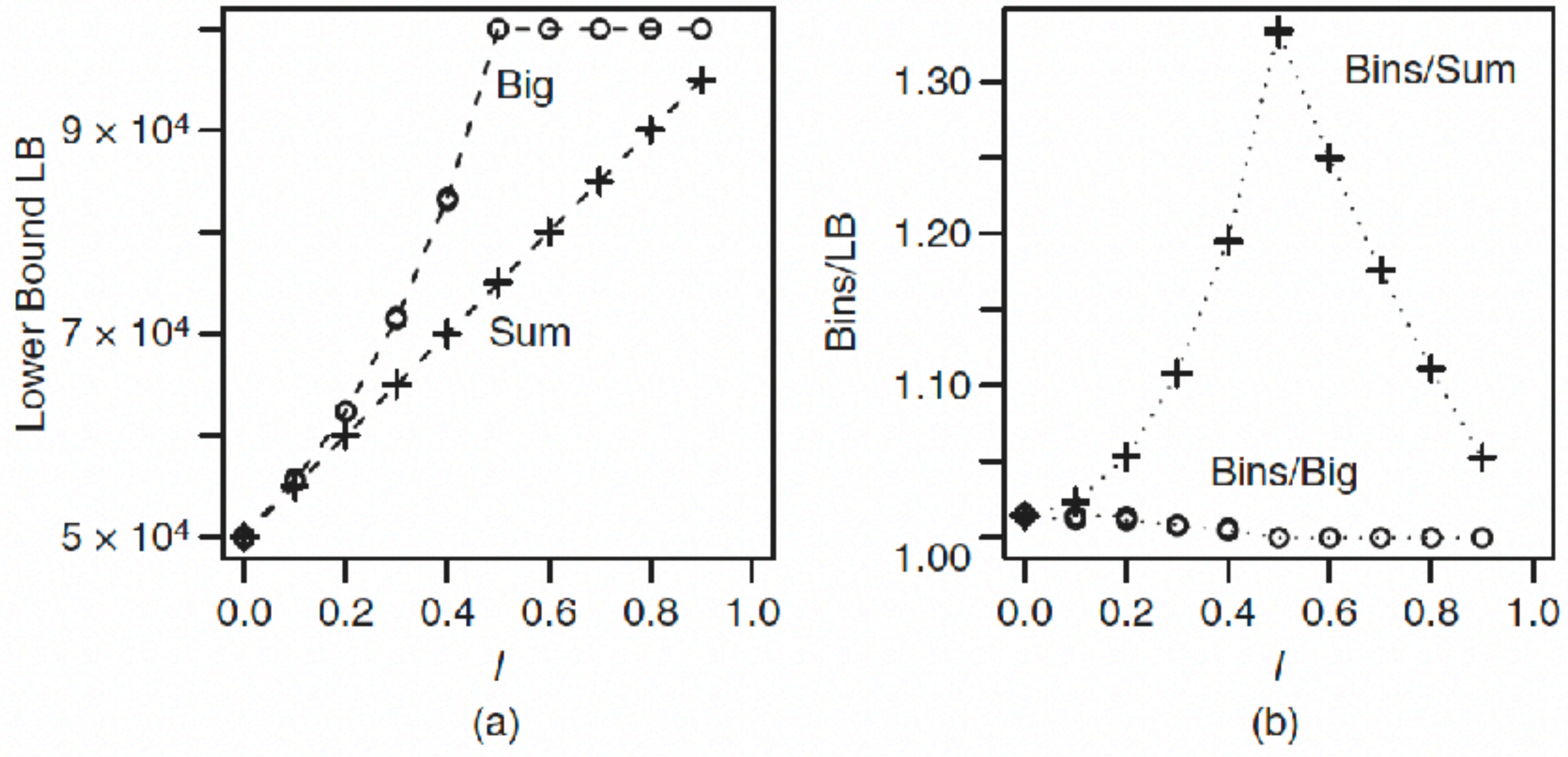


Figure 3.13. Lower bounds on packings. Panel (a) compares weight sums $S(L_i)$ to the number of large weights $H(L_i)$ in three trials at each design point $n = 100000$, $u = 1.0$, and ℓ, as shown on the x-axis. Panel (b) compares the ratios $R_1(L_i)$ and $R_2(L_i)$ from these lower bounds.

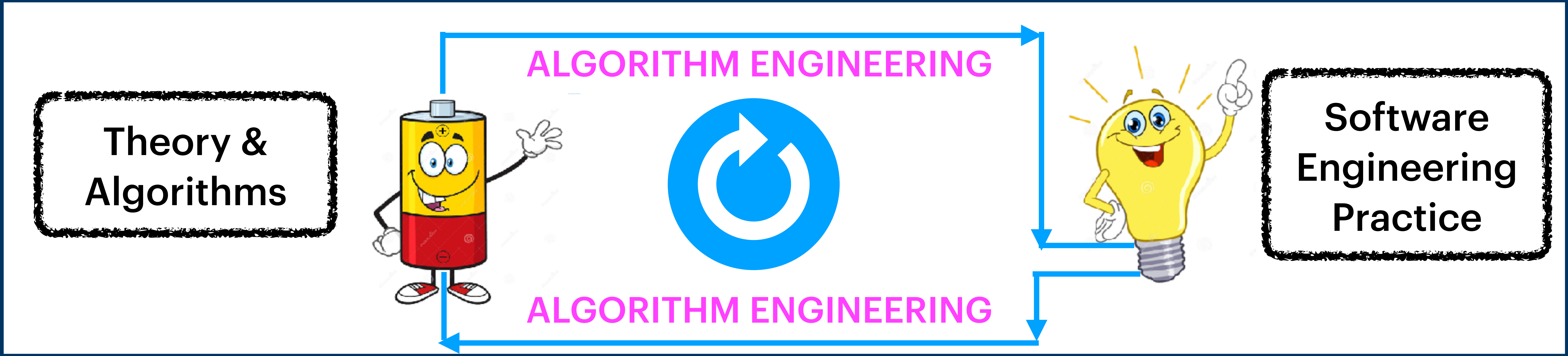
Reading assignment

`MCGEOCH, Chapter 3 What to measure`

`MS10, Chapter 8.2 Experiments / Planning experiments`

Next week we will have a quiz at the end of the lecture !

NEXT LECTURE ...



ALGORITHM ENGINEERING

Lecture 6:
Implementation Phase - I

M. Oğuzhan Külekci - kulekci@itu.edu.tr