Interpolation Search

Review talk, joint work with: S. Sioutas, C. Makris, C. Zaroliagis, T. Tsakalidis, K. Tsichlas

Results published at:
Conferences: ESA'03, ISAAC '05 & 09, ICALP'06, ICDT'10
Journals: JDA, Algorithmica

Talk :
Alexis C. Kaporis, Dept. Information & Communication Systems, U. Aegean, Karlovassi, Samos

Interpolation Search

BUT….

Interpolation Search

BUT…. What is this “Interpolation” ?

Interpolation Search

BUT…. What is this “Interpolation” ?

Easy things *should* come first:

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: Serial search

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: **Serial** search

Suppose that Nature likes us...

Interpolation Search

BUT…. What is this “Interpolation” ?

Easy things *should* come first: **Serial** search

target!

File = [3, 5, 1, 0, 7, 9, 12, 16, 4]

Interpolation Search

BUT…. What is this “Interpolation” ?

Easy things *should* come first: **Serial** search

target!

File = [3, 5, 1, 0, 7, 9, 12, 16, 4]

1 step!

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: **Serial** search

target!

File = [3, 5, 1, 0, 7, 9, 12, 16, 4]

1 step! But...

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: **Serial** search

Target!

File = [3, 5, 1, 0, 7, 9, 12, 16, 4]

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: **Serial** search

Target!

File = [3, 5, 1, 0, 7, 9, 12, 16, 4]

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: **Serial** search

Target!

File = [3, 5, 10, 7, 9, 12, 16, 4]

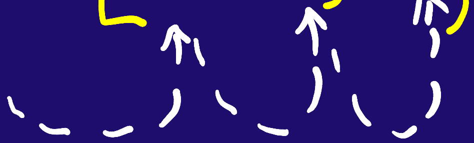
Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: **Serial** search

Target!

File = [3, 5, 1, 0, 7, 9, 12, 16, 4]



Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: **Serial** search

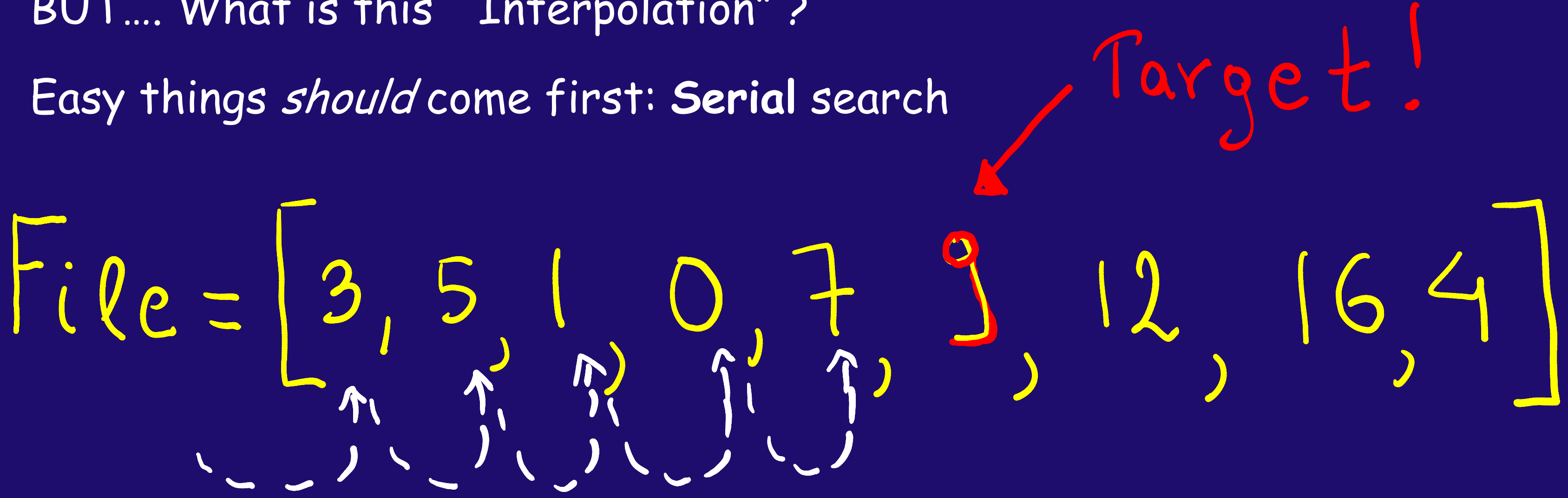
Target!

File = [3, 5, 1, 0, 7, 9, 12, 16, 4]

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: **Serial** search



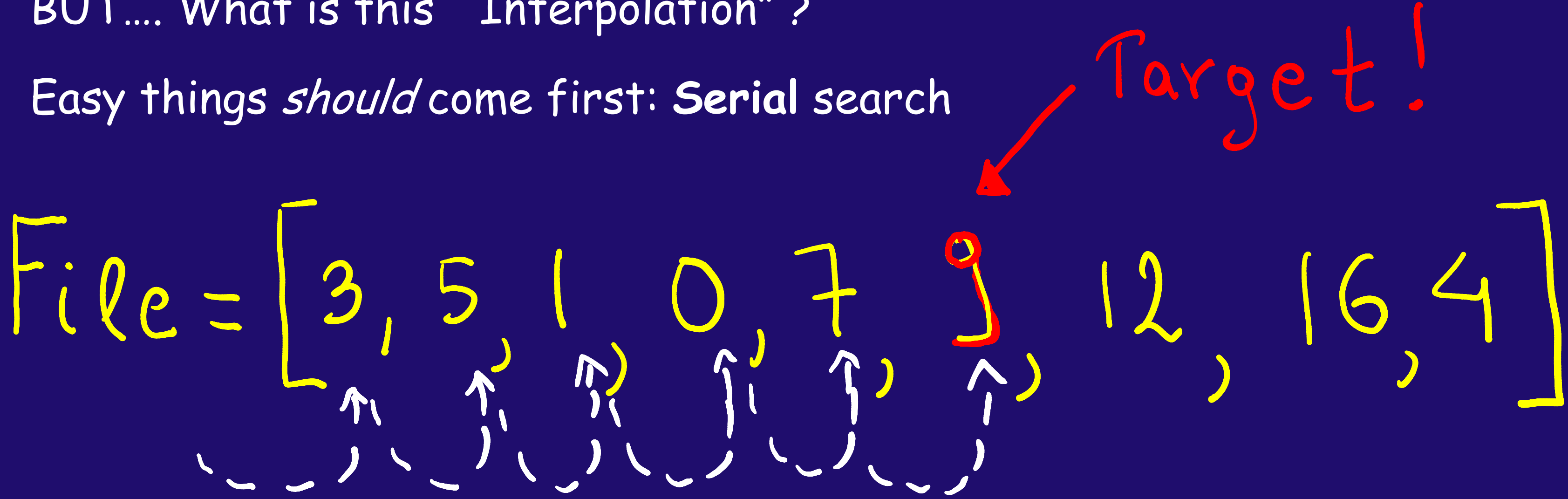
Target!

File = [3, 5, 1, 0, 7, 9, 12, 16, 4]

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: **Serial** search



Target!

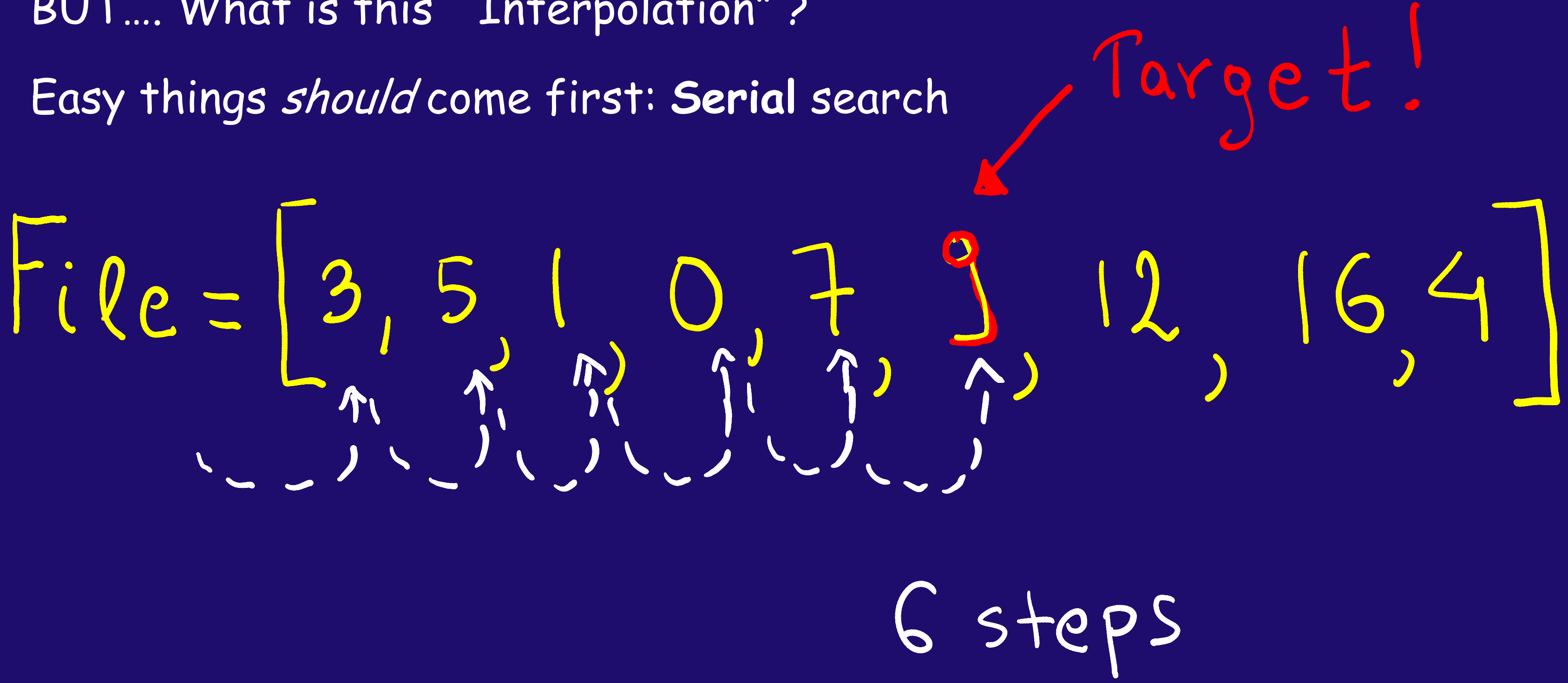
File = [3, 5, 1, 0, 7, 9, 12, 16, 4]

6 steps

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: **Serial** search



File = [3, 5, 1, 0, 7, 9, 12, 16, 4]

Target!

6 steps

Deteriorates as Θ(n), as file size n increases

Interpolation Search

BUT…. What is this “Interpolation” ?

Easy things *should* come first: **Serial** search

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: **Serial** search

Suppose that she (Nature) still likes us

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: **Serial** search

Target!

~~File = [3, 5, 1, 0, 7, 9, 12, 16, 4]~~

Target!

File = [0, 1, 3, 4, 5, 7, 9, 12, 16]

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: **Serial** search

~~Target!~~

~~File = [3, 5, 1, 0, 7, 9, 12, 16, 4]~~

Target!

File = [0, 1, 3, 4, 5, 7, 9, 12, 16]

«Εν αρχη ην η Τάξη», aka «assume **ordered** file»

Interpolation Search

BUT.... What is this "Interpolation" ?

Easy things *should* come first: **~~Serial~~** search -> **Binary** search

Target!

~~File = [3, 5, 1, 0, 7, 9, 12, 16, 4]~~

Target!

File = [0, 1, 3, 4, 5, 7, 9, 12, 16]

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: ~~**Serial**~~ search -> **Binary** search

Target!

File = [0, 1, 3, 4, 5, 7, 9, 12, 16]

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: **~~Serial~~** search -> **Binary** search

Target!

File = [0, 1, 3, 4, 5, 7, 9, 12, 16]

Halves the file per recursive step

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: ~~Serial~~ search -> **Binary** search

Target!

File = [0, 1, 3, 4, 5, 7, 9, 12, 16]

5 < 9

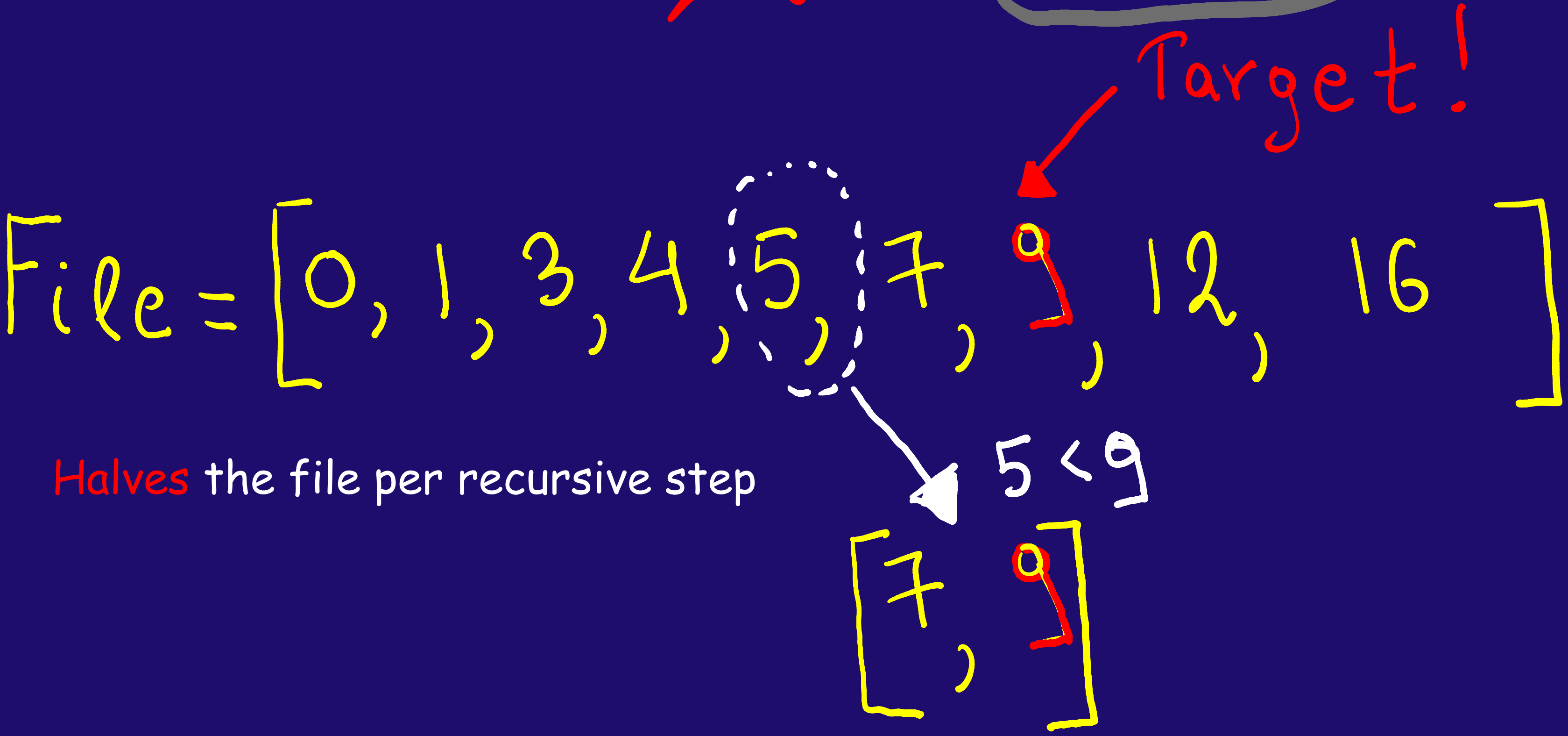
Halves the file per recursive step

[7, 9, 12, 16]

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: ~~Serial~~ search -> Binary search

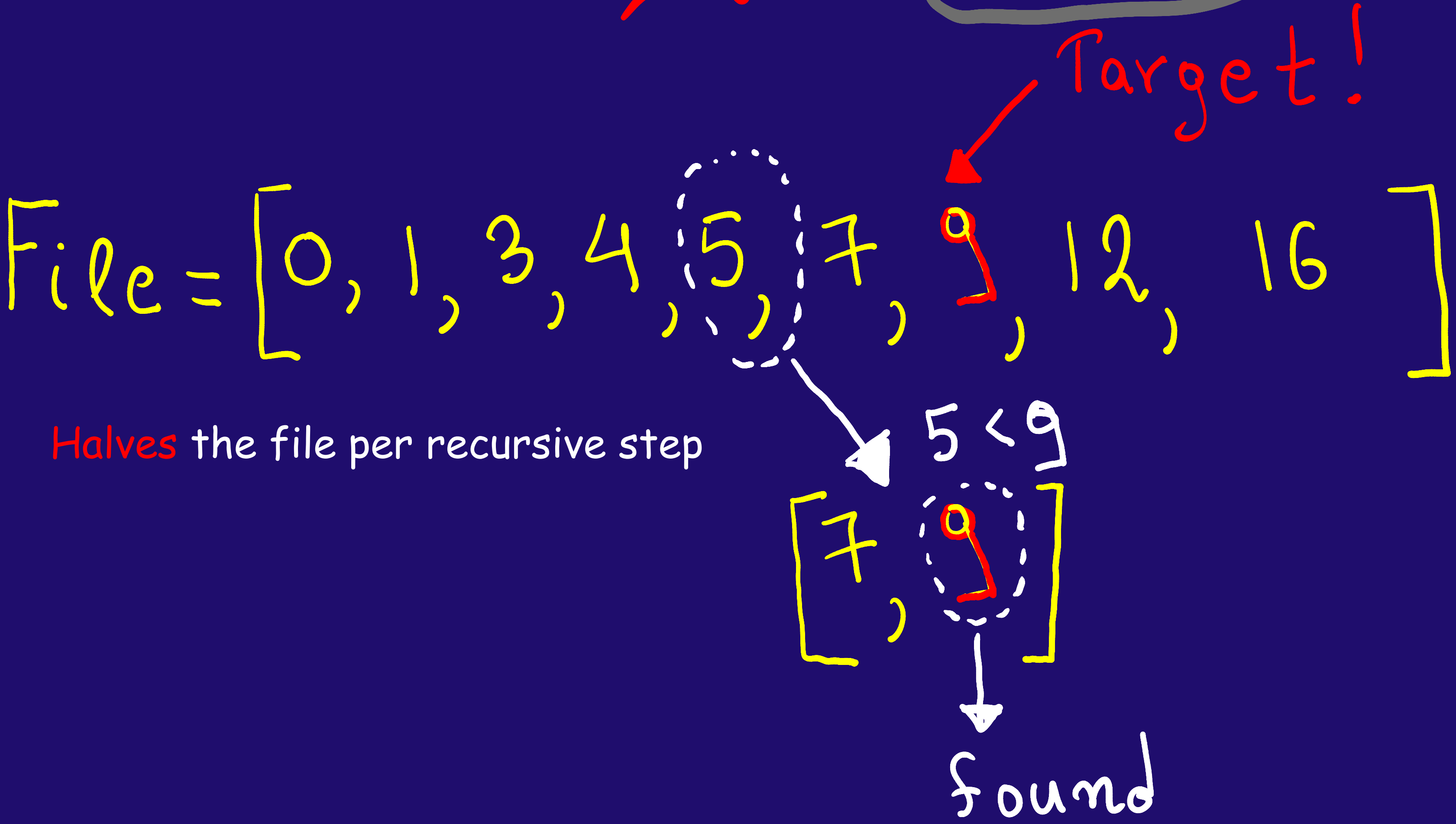


Halves the file per recursive step

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: ~~Serial~~ search -> Binary search



Halves the file per recursive step

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: ~~Serial~~ search -> Binary search

Target!

File = [0, 1, 3, 4, 5, 7, 9, 12, 16]

5 < 9

[7, 9]

Halves the file per recursive step

$n \rightarrow n/2 \rightarrow n/2^2 \rightarrow \dots n/2^i \rightarrow \dots 1$

Interpolation Search

BUT…. What is this "Interpolation" ?

Easy things *should* come first: ~~**Serial**~~ search -> **Binary** search

Target!

File = [0, 1, 3, 4, 5, 7, 9, 12, 16]

$5 < 9$

[7, 9]

Halves the file per recursive step

$$n \to n/2 \to n/2^2 \to \dots n/2^i \to \dots 1$$

Interpolation Search

BUT…. What is this “Interpolation” ?

Easy things *should* come first: ~~Serial~~ search -> Binary search

Target!

File = [0, 1, 3, 4, 5, 7, 9, 12, 16]

$5 < 9$

[7, 9]

Halves the file per recursive step

$n \to n/2 \to n/2^2 \to \dots n/2^i \to \dots 1$

Interpolation Search

BUT…. What is this "Interpolation"?

Easy things *should* come first: ~~Serial~~ search -> Binary search

Target!

File = [0, 1, 3, 4, 5, 7, 9, 12, 16]

$5 < 9$

[7, 9]

Halves the file per recursive step

$$n \rightarrow n/2 \rightarrow n/2^2 \rightarrow \dots n/2^i \rightarrow \dots 1$$

Interpolation Search

BUT…. What is this "Interpolation"?

Easy things *should* come first: ~~Serial~~ search -> Binary search

Target!

File = [0, 1, 3, 4, 5, 7, 9, 12, 16]

5 < 9

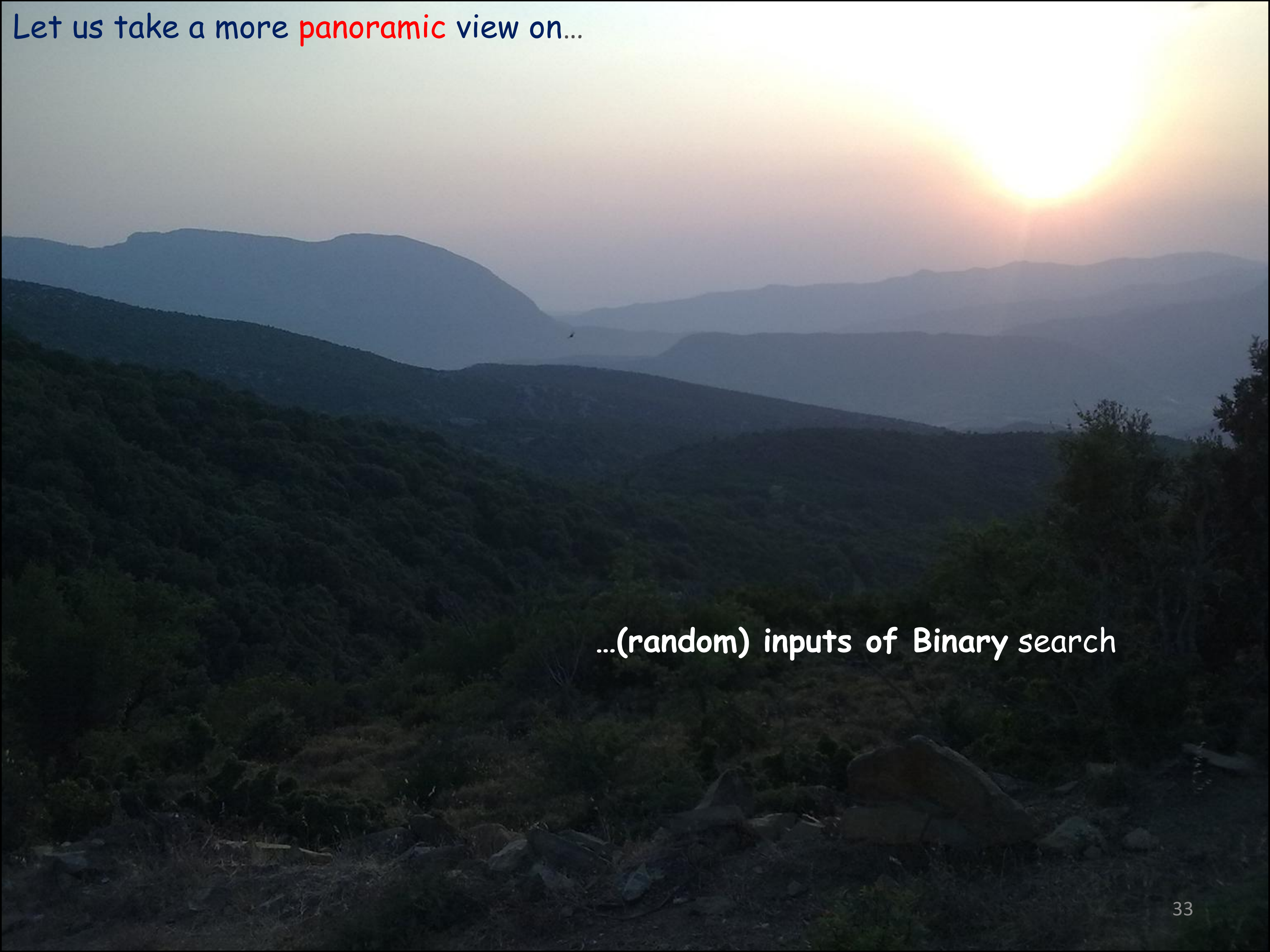
[7, 9]

Halves the file per recursive step

$$n \to n/2 \to n/2^2 \to \dots n/2^i \to \dots 1, \quad i = \log n$$

Deteriorates as Θ(logn), as file size n increases

Let us take a more panoramic view on…



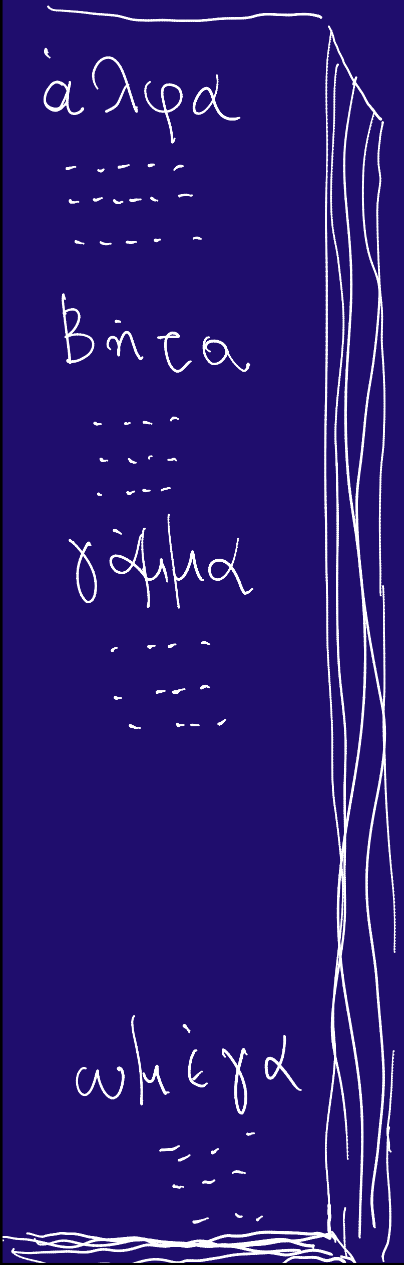
…(random) inputs of Binary search

Let us take a more panoramic view on…

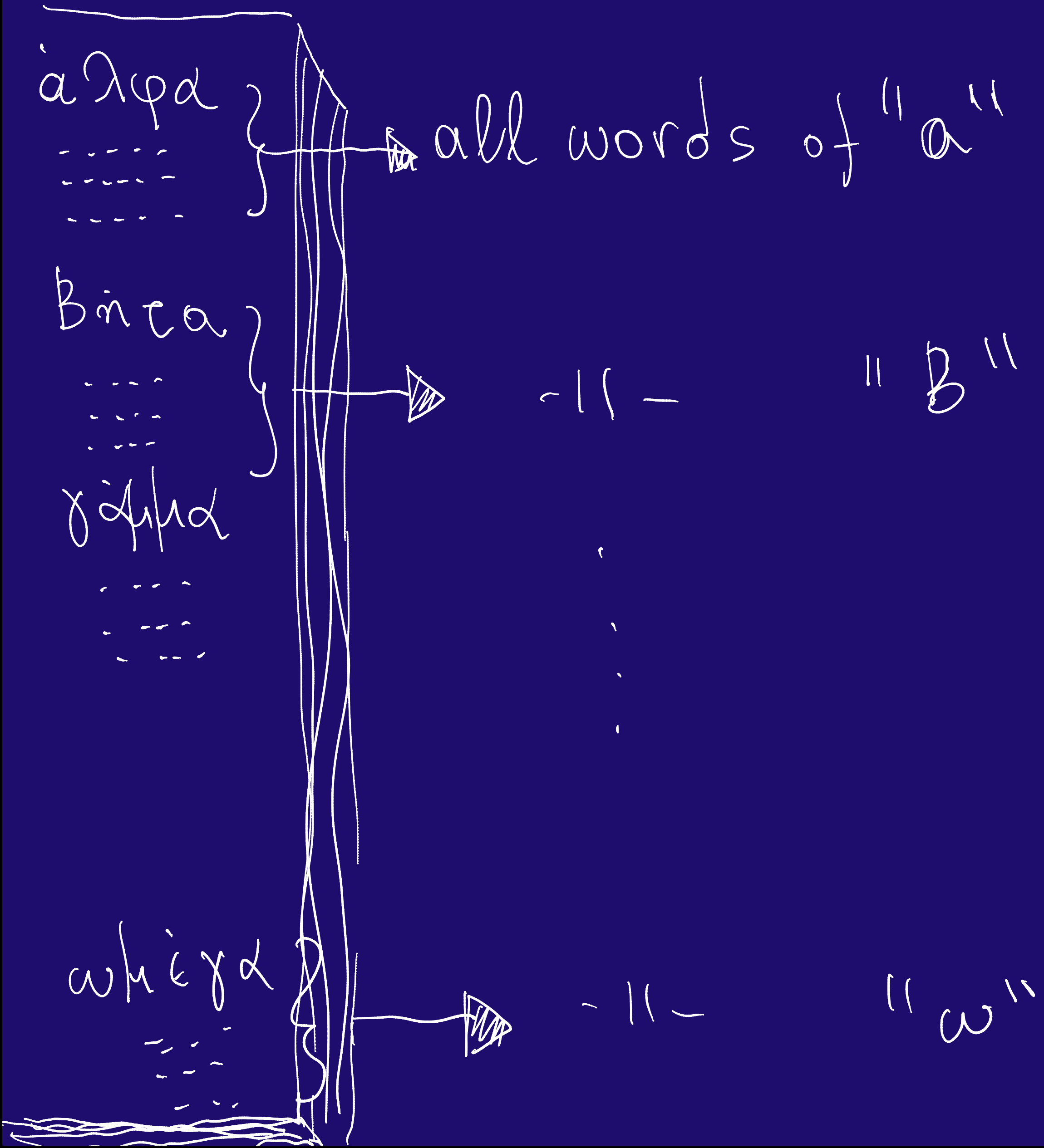
…(random) inputs of Binary search

…not a Πανάκεια wrt input distribution

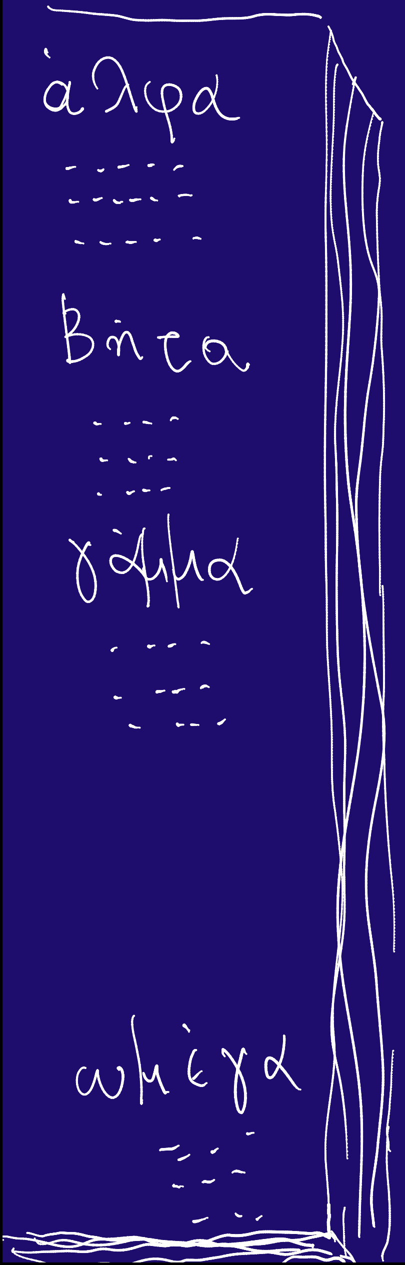
Knuth's Example: a lexicon is a very nice and very ordered file



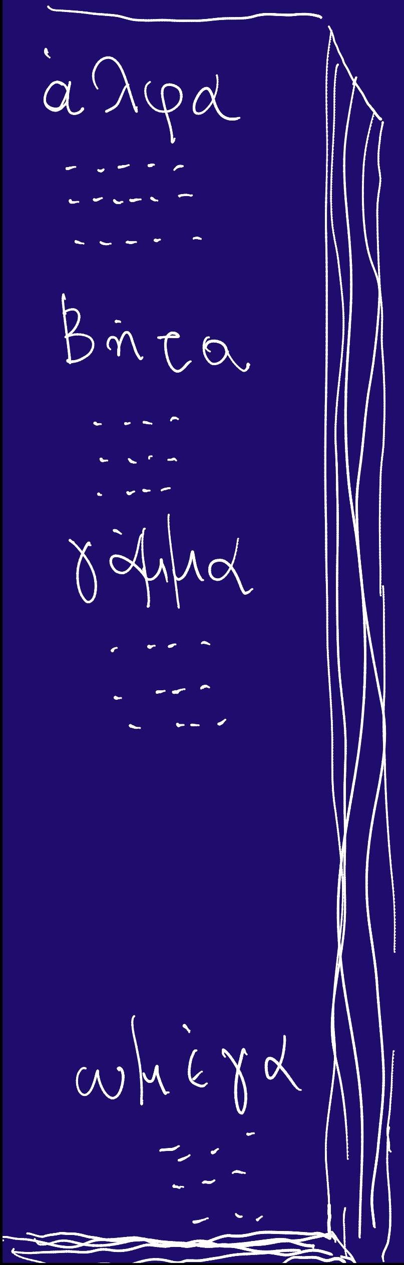
Knuth's Example: a lexicon is a very nice and very ordered file



Knuth's Example: a lexicon is a very nice and very ordered file



Knuth's Example: a lexicon is a very nice and very ordered file

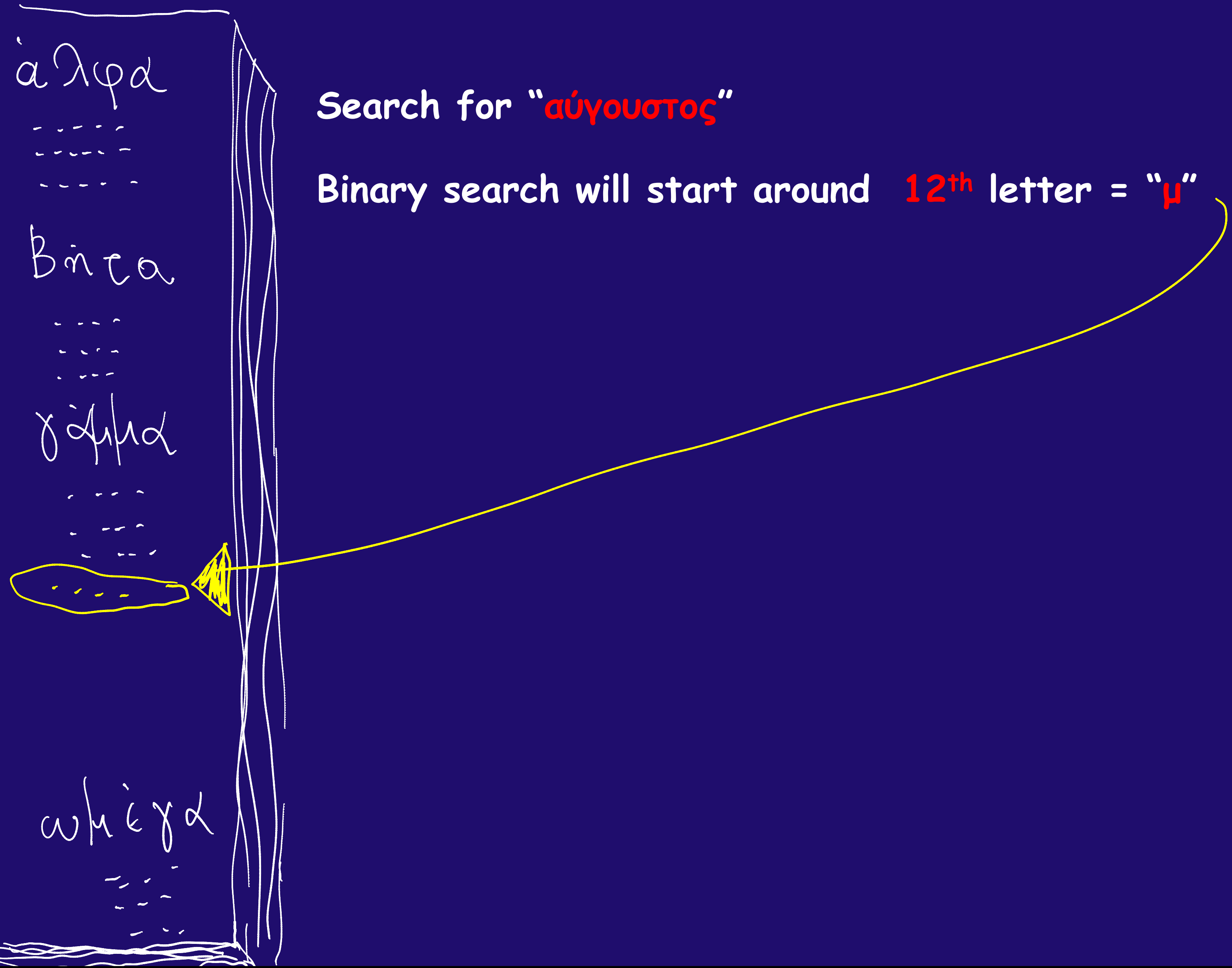


Search for "αύγουστος"

Knuth's Example: a lexicon is a very nice and very ordered file

Search for "αύγουστος"

Binary search will start around 12th letter = "μ"

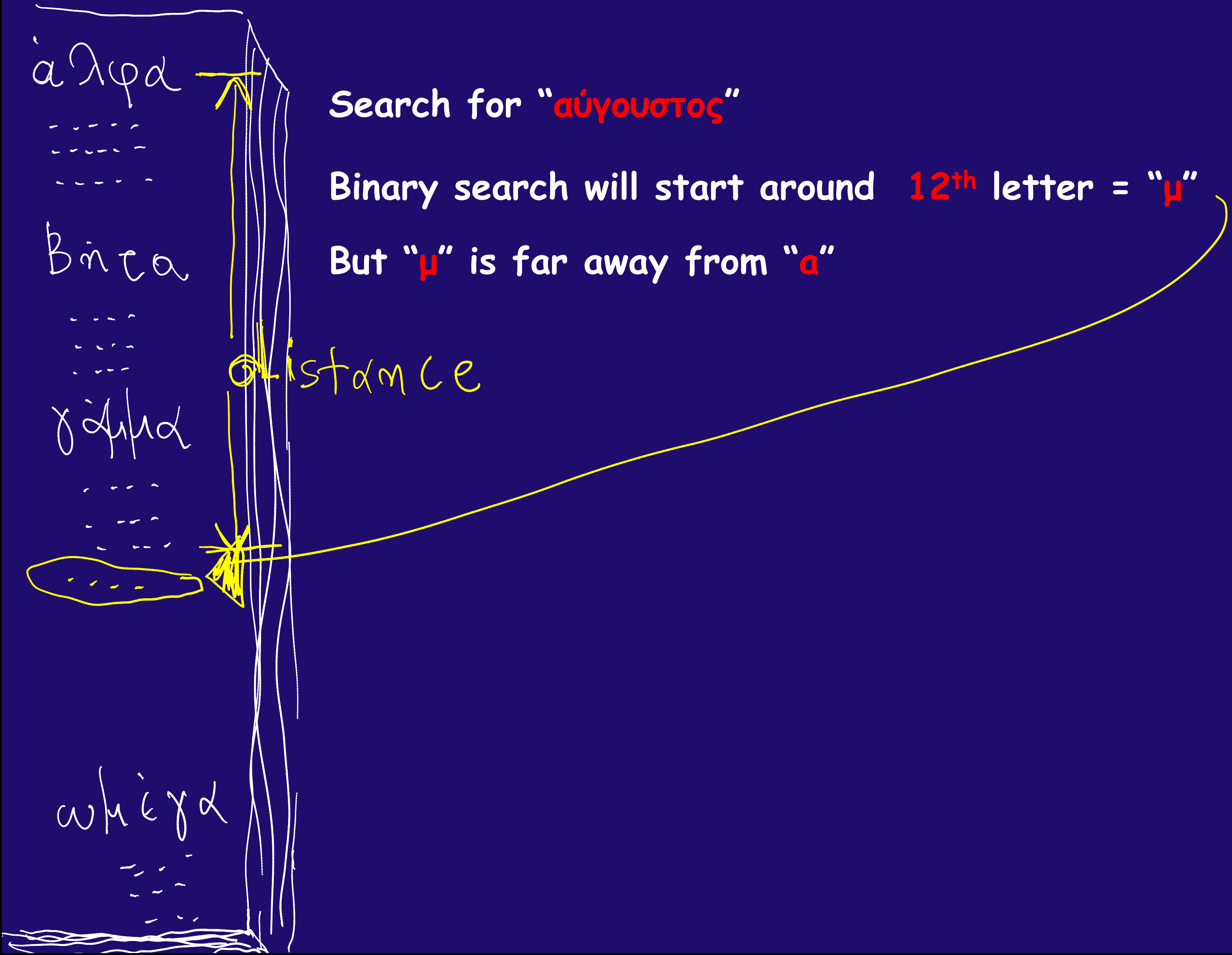


Knuth's Example: a lexicon is a very nice and very ordered file

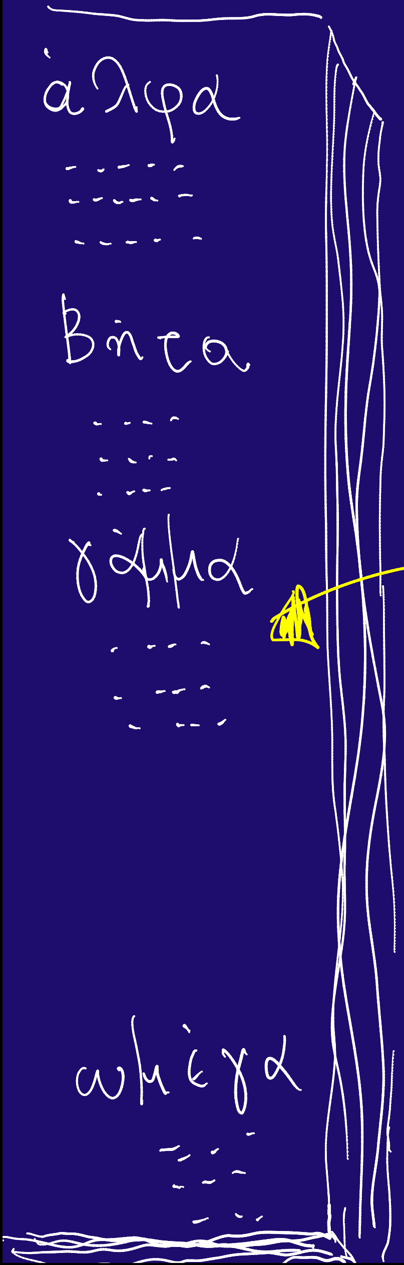
Search for "αύγουστος"

Binary search will start around 12th letter = "μ"

But "μ" is far away from "α"



Knuth's Example: a lexicon is a very nice and very ordered file



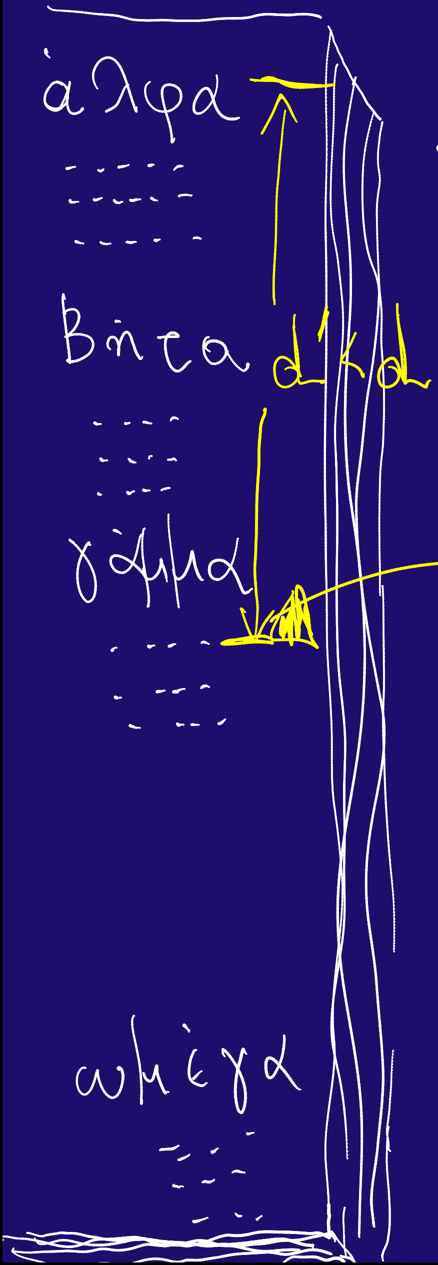
Search for "αύγουστος"

Binary search will start around 12th letter = "μ"

But "μ" is far away from "α"

Binary search will get as closer as 6th letter = "ζ"

Knuth's Example: a lexicon is a very nice and very ordered file



Search for "αύγουστος"

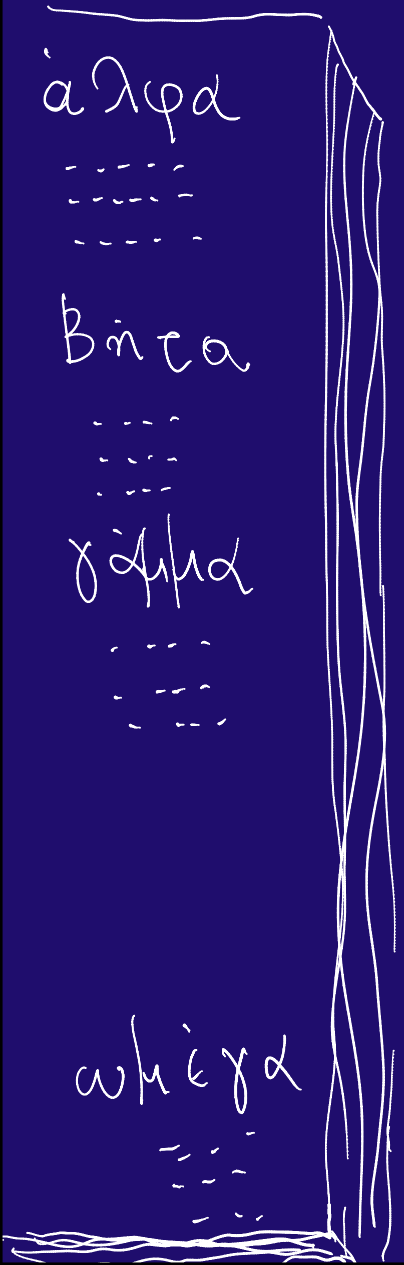
Binary search will start around 12th letter = "μ"

But "μ" is a far away from "α"

Binary search will get as closer as 6th letter = "ζ"

But "ζ" is also away from "α"

Knuth's Example: a lexicon is a very nice and very ordered file



Search for "αύγουστος"

Binary search will start around 12th letter = "μ"

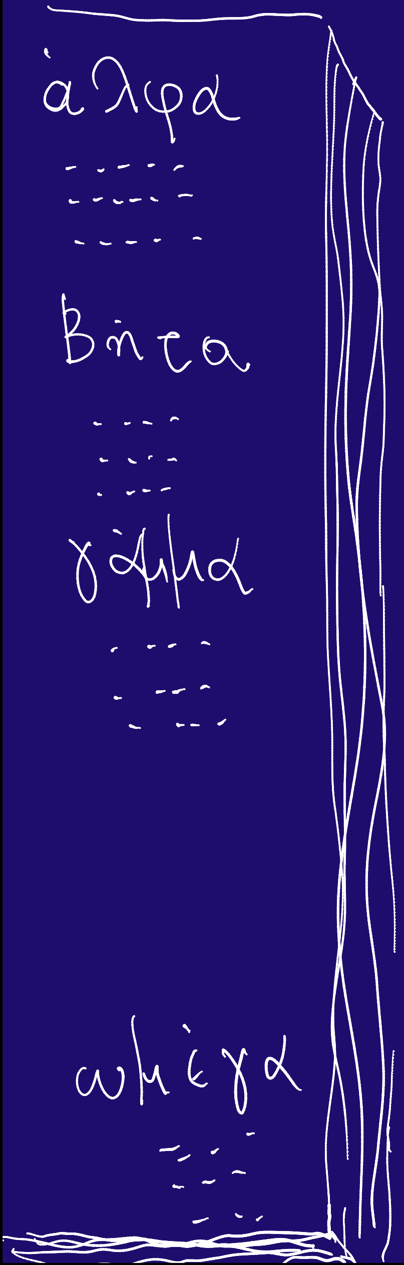
But "μ" is far away from "α"

Binary search will get as closer as 6th letter = "ζ"

But "ζ" is also away from "α"

Binary search keeps walking to 3rd letter = "γ"

Knuth's Example: a lexicon is a very nice and very ordered file



Search for "αύγουστος"

Binary search will start around 12th letter = "μ"

But "μ" is far away from "α"

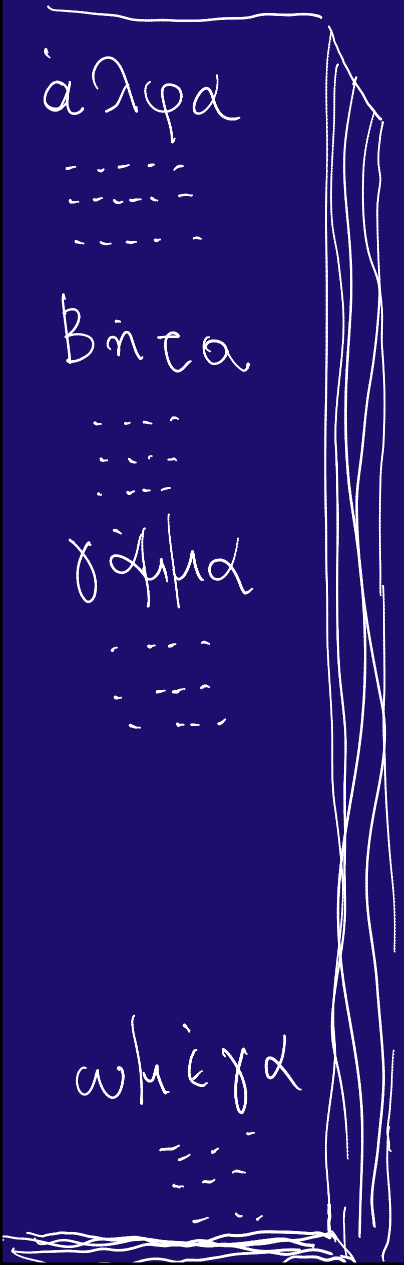
Binary search will get as closer as 6th letter = "ζ"

But "ζ" is also away from "α"

Binary search keeps walking to 3rd letter = "γ"

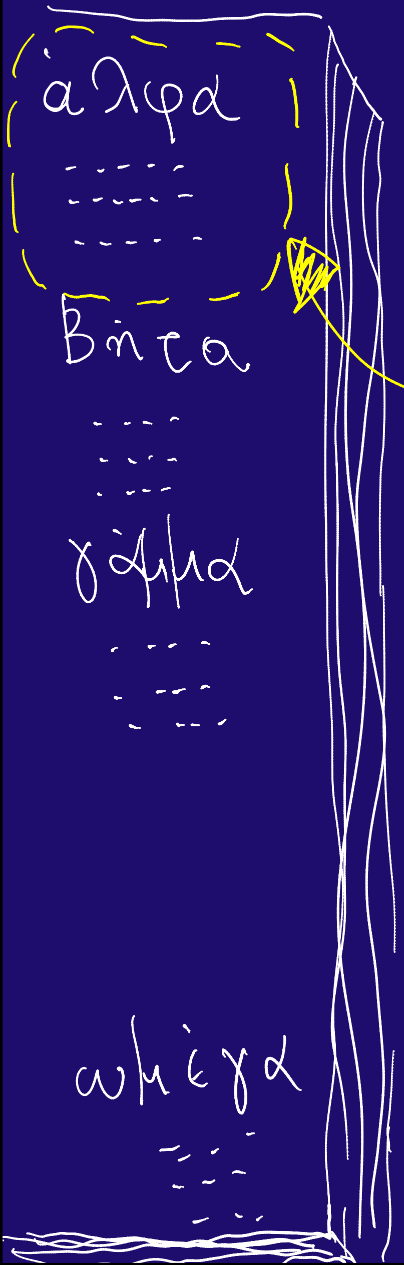
And the story goes on …

Knuth's Example: a lexicon is a very nice and very ordered file



How a secretariat (≠ Computer Scientist) would search for "αύγουστος"?

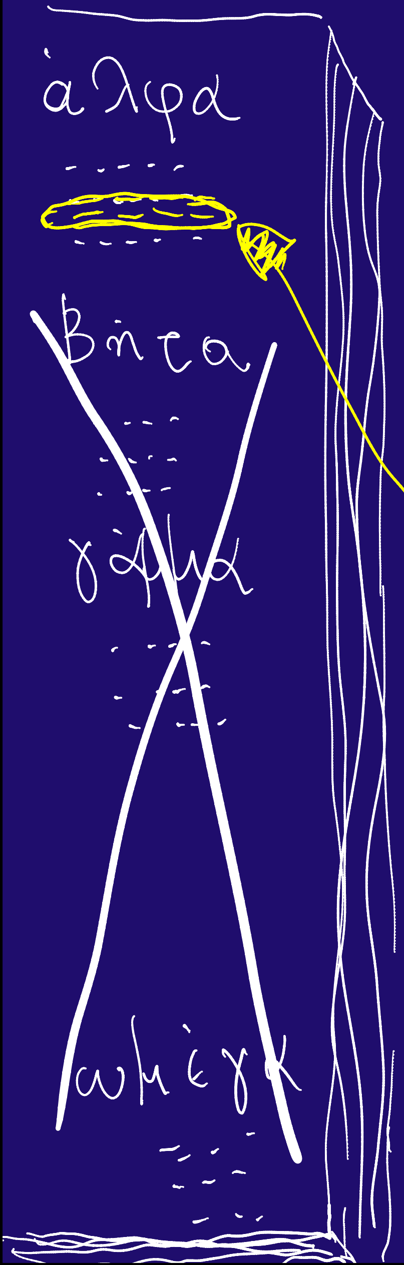
Knuth's Example: a lexicon is a very nice and very ordered file



How a secretariat (≠ Computer Scientist) would search for "αύγουστος"?

She expects "αύγουστος" to be near the front pages

Knuth's Example: a lexicon is a very nice and very ordered file

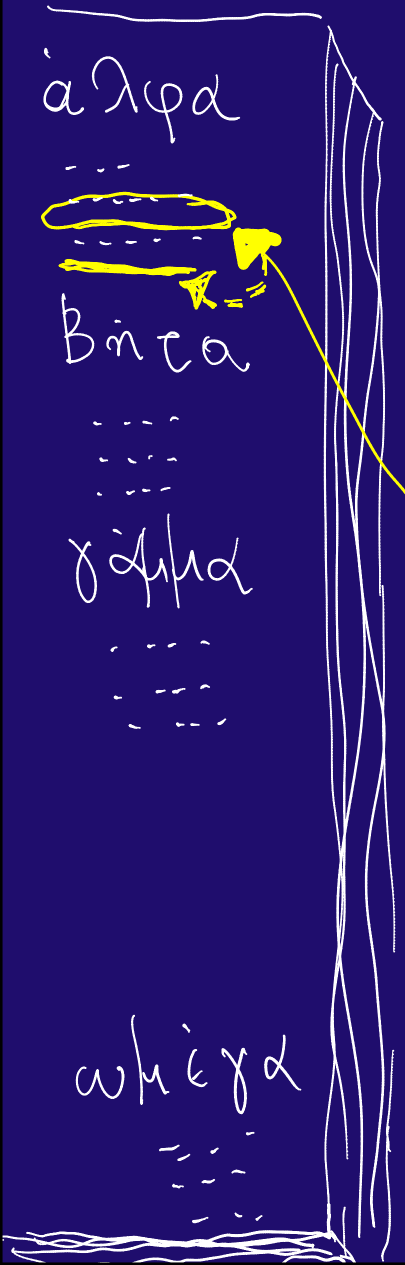


How a secretariat (≠ Computer Scientist) would search for “αύγουστος”?

She expects “αύγουστος” to be near the front pages

and opens the lexicon near the front pages, locating “αναψυχη” < “αύγουστος”

Knuth's Example: a lexicon is a very nice and very ordered file



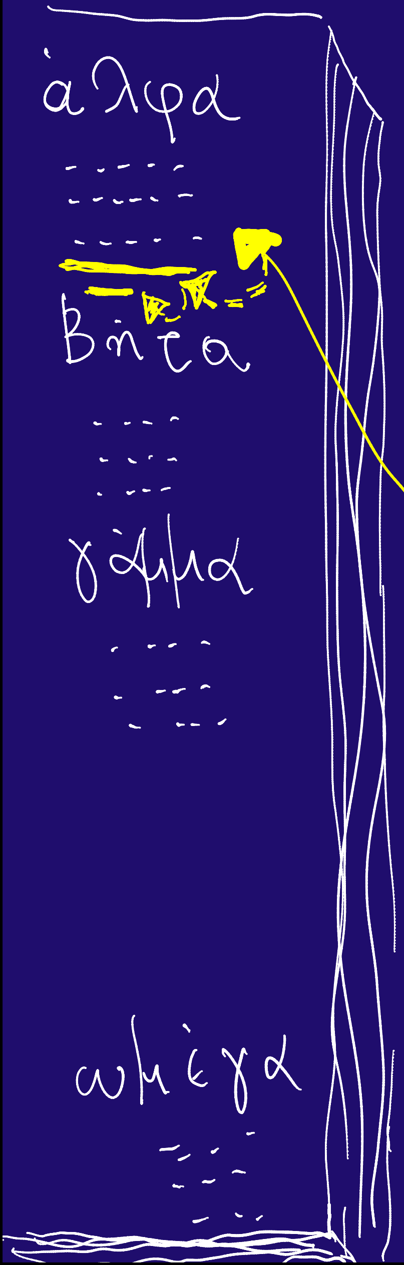
How a secretariat (≠ Computer Scientist) would search for "αύγουστος"?

She expects "αύγουστος" to be near the front pages

and opens the lexicon near the front pages, locating "αναψυχη" < "αύγουστος"

She observes "ν" is close to "ύ" and opens lexicon near to the current page, locating "αυγό"

Knuth's Example: a lexicon is a very nice and very ordered file



How a secretariat (≠ Computer Scientist) would search for "αύγουστος"?

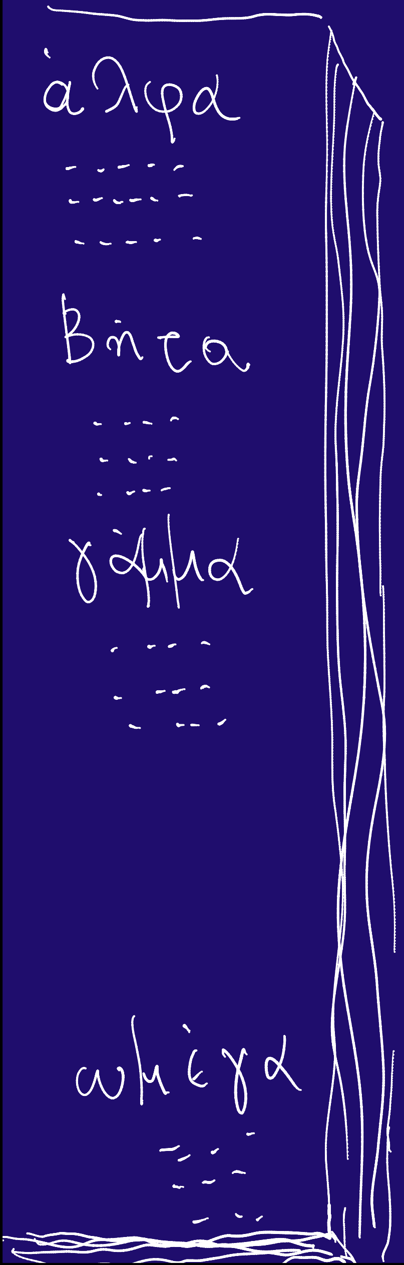
She expects "αύγουστος" to be near the front pages

and opens the lexicon near the front pages, locating "αναψυχη" < "αύγουστος"

She observes "ν" is close to "ύ" and opens lexicon near to the current page, locating "αυγό"

Now she is really close, within seconds locates "αύγουστος"

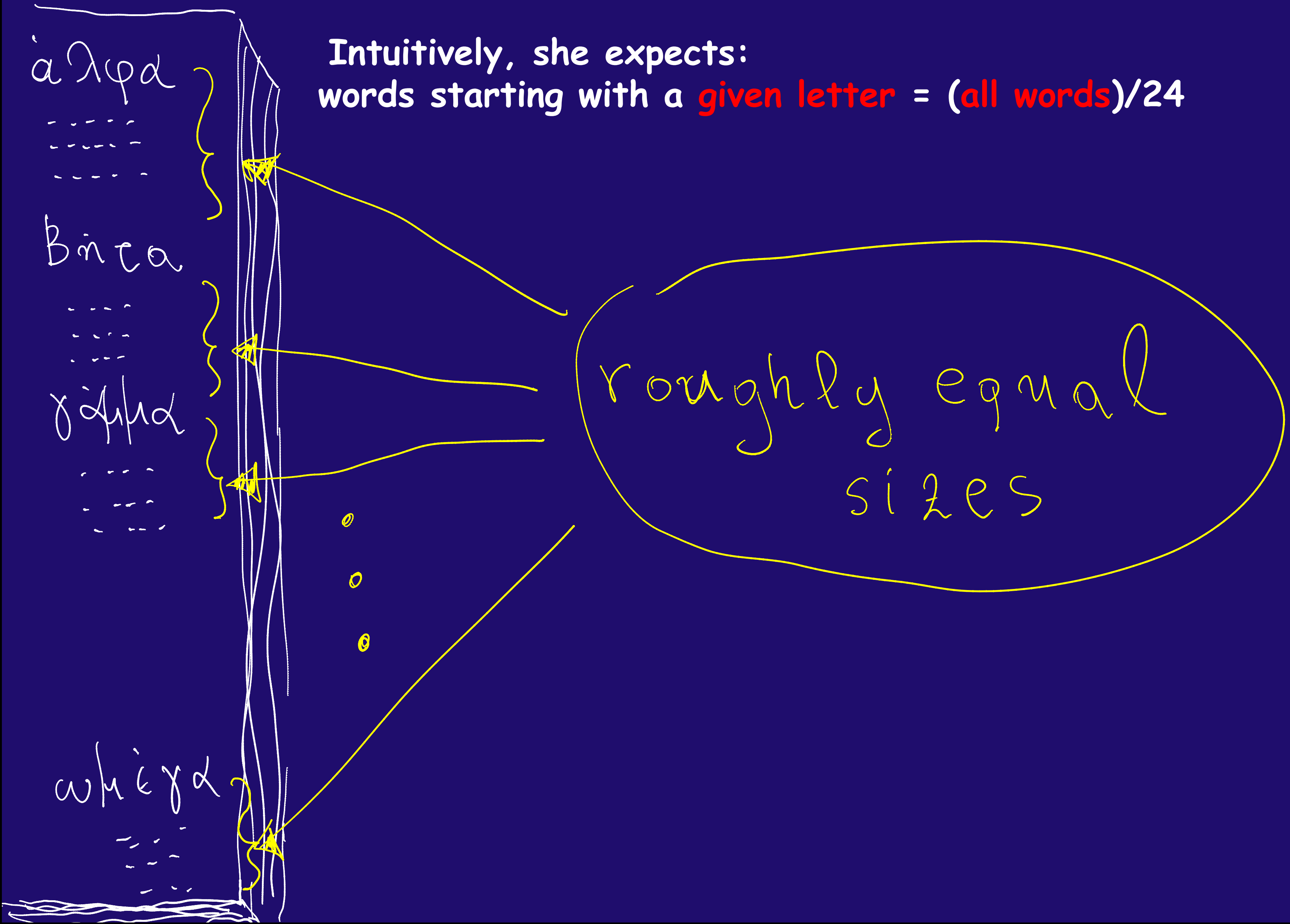
Knuth's Example: a lexicon is a very nice and very ordered file



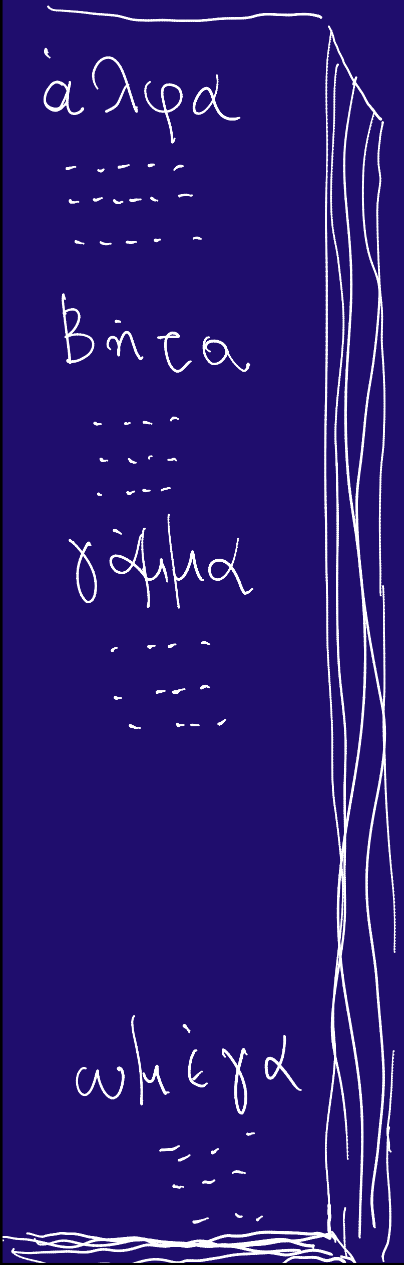
Intuitively, she expects:
words starting with a given letter = (all words)/24

Knuth's Example: a lexicon is a very nice and very ordered file

Intuitively, she expects:
words starting with a given letter = (all words)/24



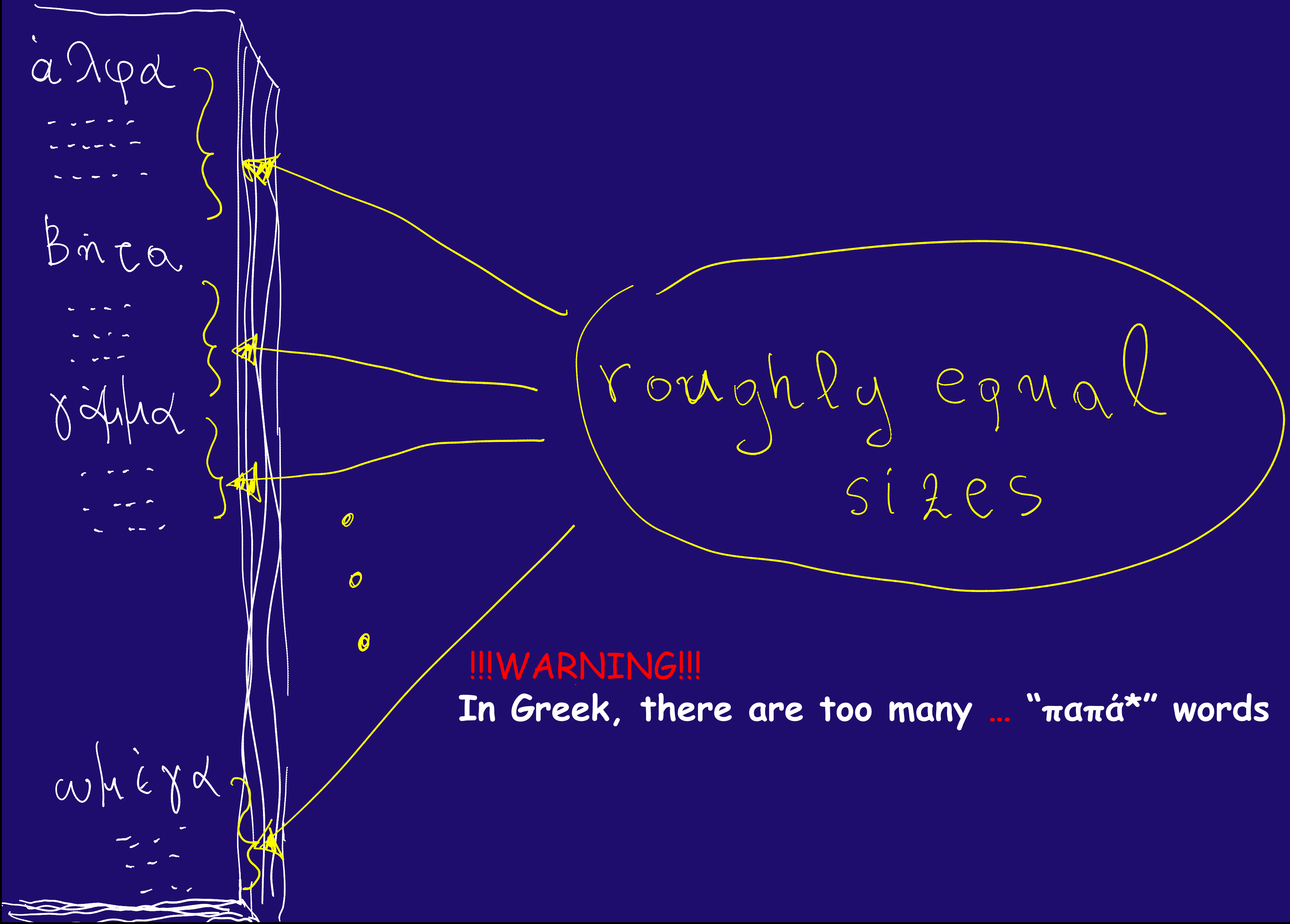
Knuth's Example: a lexicon is a very nice and very ordered file



Intuitively, she expects:
words starting with a given letter = (all words)/24

So, she expects to find "αύγουστος" at:
1/24-th part of the file

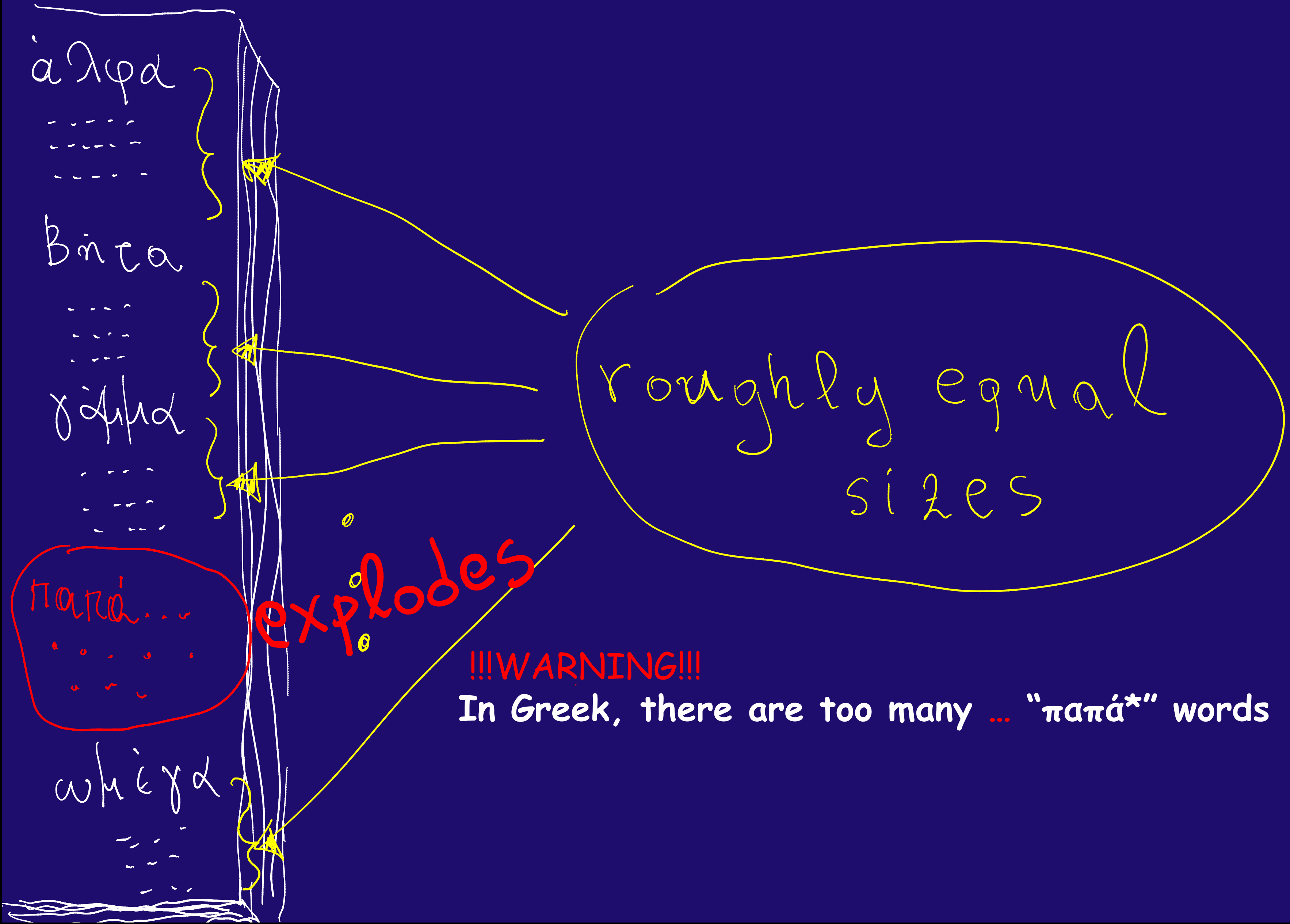
Knuth's Example: a lexicon is a very nice and very ordered file



!!!WARNING!!!

In Greek, there are too many … "παπά*" words

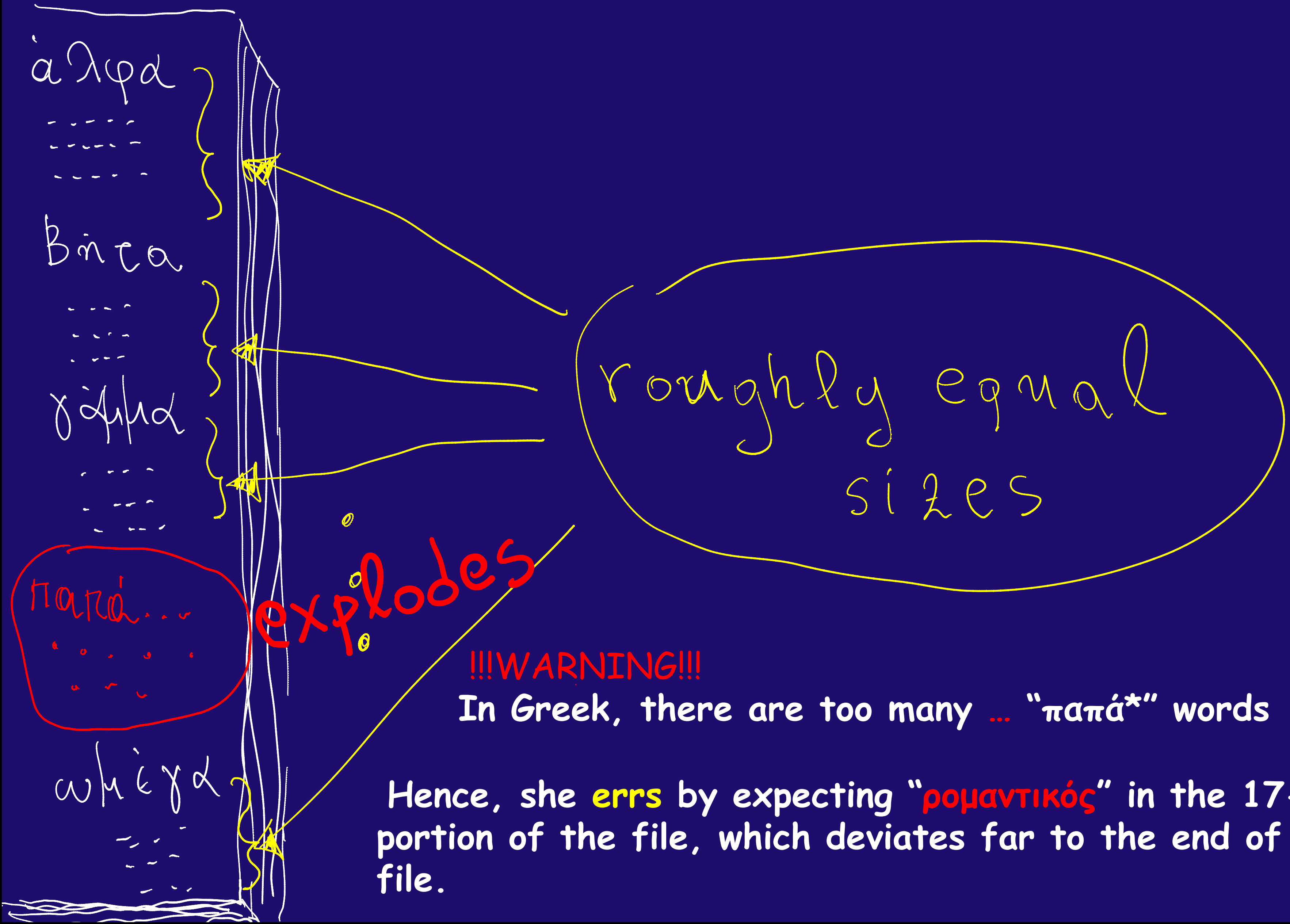
Knuth's Example: a lexicon is a very nice and very ordered file



!!!WARNING!!!

In Greek, there are too many … "παπά*" words

Knuth's Example: a lexicon is a very nice and very ordered file



!!!WARNING!!!

In Greek, there are too many … "παπά*" words

Hence, she **errs** by expecting "ρομαντικός" in the 17-th portion of the file, which deviates far to the end of file.

So, this is (a variant of) interpolation

Morals: uniformity matters (at least at present…)

W. Peterson (1957), the “father” of the Interpolation Search (IS)

W. Peterson (1957), the "father" of the Interpolation Search (IS)

He observed extremely fast experimental performance, but, he only managed to prove order of log(n) search time. (The same as binary search!)

W. Peterson (1957), the “father” of the Interpolation Search (IS)

He observed extremely fast experimental performance, but, he only managed to prove order of log(n) search time. (The same as binary search!)

Knuth “επικήρυξε” the analysis of IS in his famous list of 10 most important problems in searching. Thus, IS became the “protagonist” of at least 10 years of intensive research.

W. Peterson (1957), the "father" of the Interpolation Search (IS)

He observed extremely fast experimental performance, but, he only managed to prove order of log(n) search time. (The same as binary search!)

Knuth "επικήρυξε" the analysis of IS in his famous list of 10 most important problems in searching. Thus, IS became the "protagonist" of at least 10 years of intensive research.

A great amount of papers focused on analyzing IS rigorously on UNIFORM input distribution:

W. Peterson (1957), the "father" of the Interpolation Search (IS)

He observed extremely fast experimental performance, but, he only managed to prove order of log(n) search time. (The same as binary search!)

Knuth "επικήρυξε" the analysis of IS in his famous list of 10 most important problems in searching. Thus, IS became the "protagonist" of at least 10 years of intensive research.

A great amount of papers focused on analyzing IS rigorously on UNIFORM input distribution:

Yao & Yao (1976),
G. Gonnet (1977)
Perl, Reingold (1977),
Perl, Itai, Avni (1978)
Gonnet, Rogers & George (1985)

They concluded to Θ(loglogn) performance, exponentially better than Binary Search

What about NON uniform input distributions?

What about NON uniform input distributions?

D. Willard (1985),
Demaine, Jones & Patrascu (2004)

What about Dynamic files? (insert/delete keys)

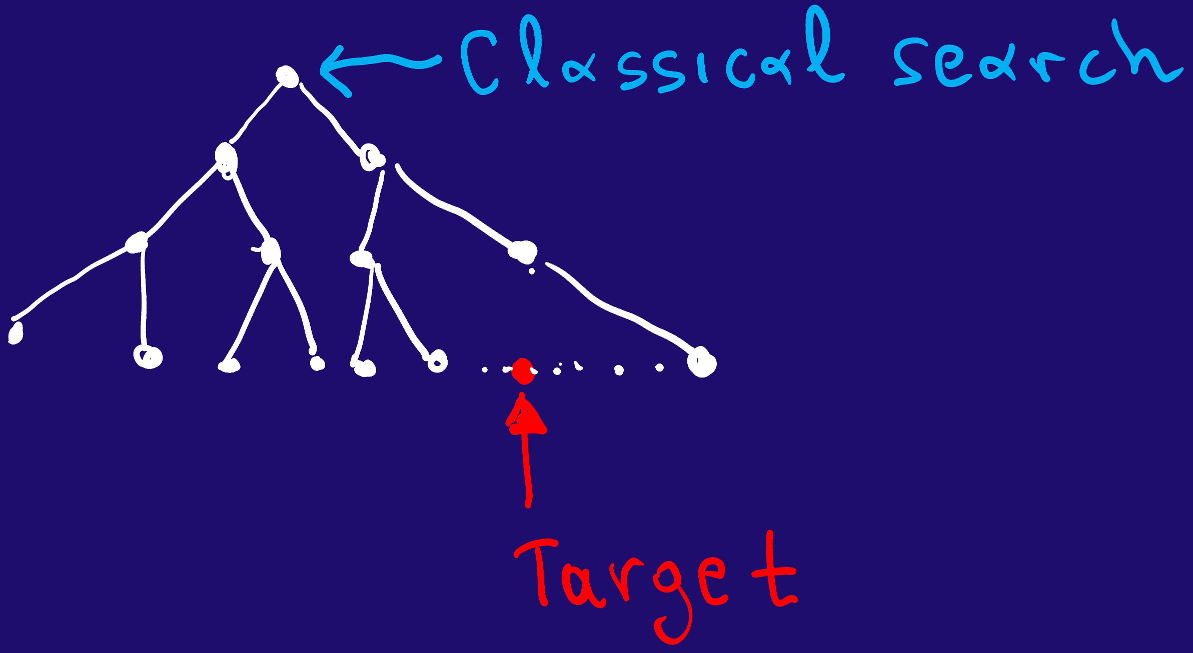
What about Dynamic files? (insert/delete keys)

Mehlhorn & Tsakalides (19??), also extended input distributions
Andersson & Mattson (1993), more extended input distributions

What about removing n from loglog(n)?

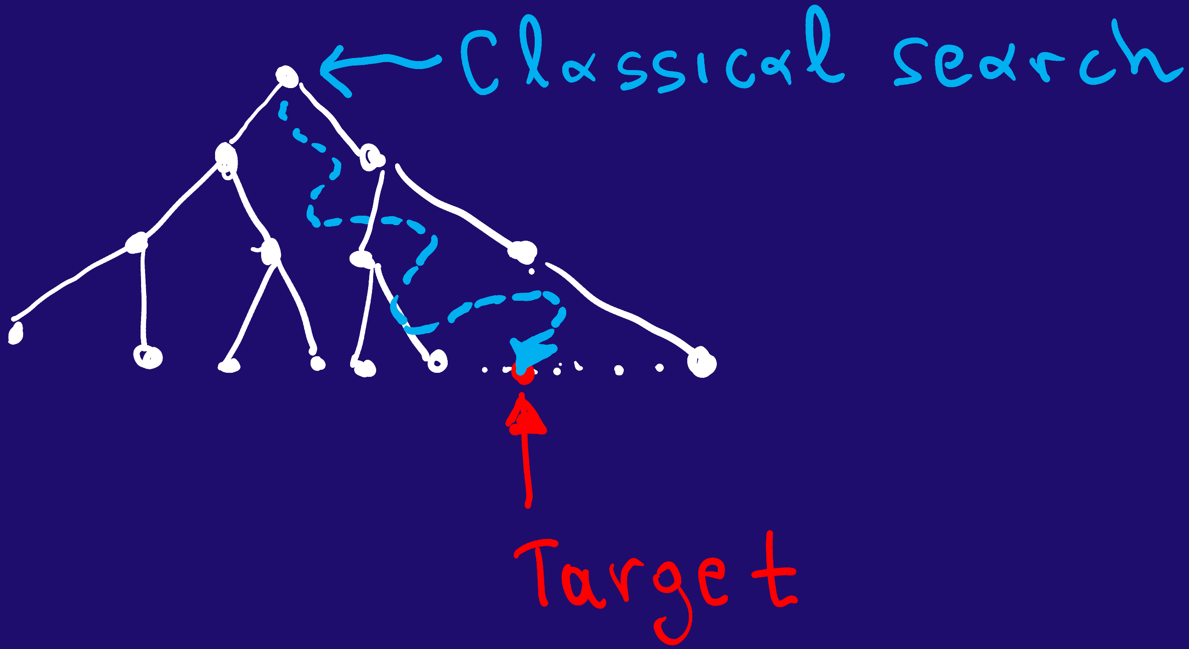
What about removing n from loglog(n)?

Kaporis, Makris, Sioutas, Tsakalidis, Tsichlas & Zaroliagis (2003) showed order of loglog(d),
d= distance(target key, finger key)



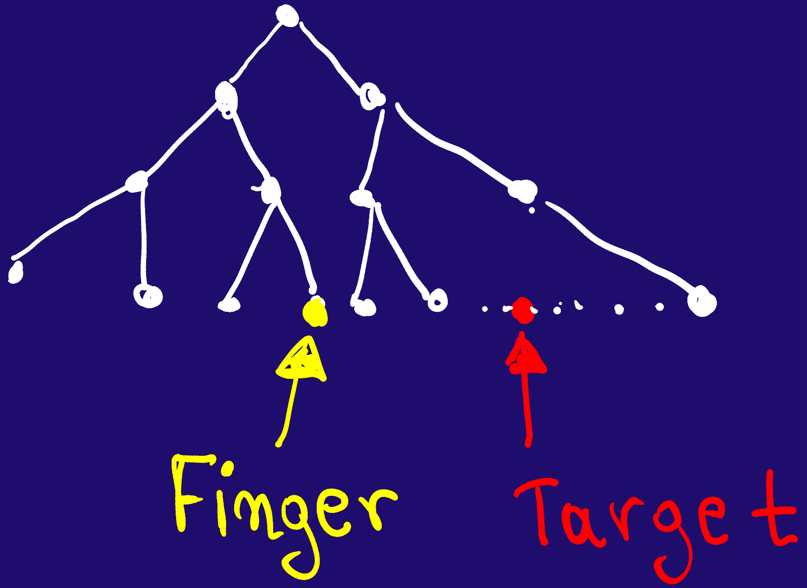
What about removing n from loglog(n)?

Kaporis, Makris, Sioutas, Tsakalidis, Tsichlas & Zaroliagis (2003) showed order of loglog(d),
d= distance(target key, finger key)



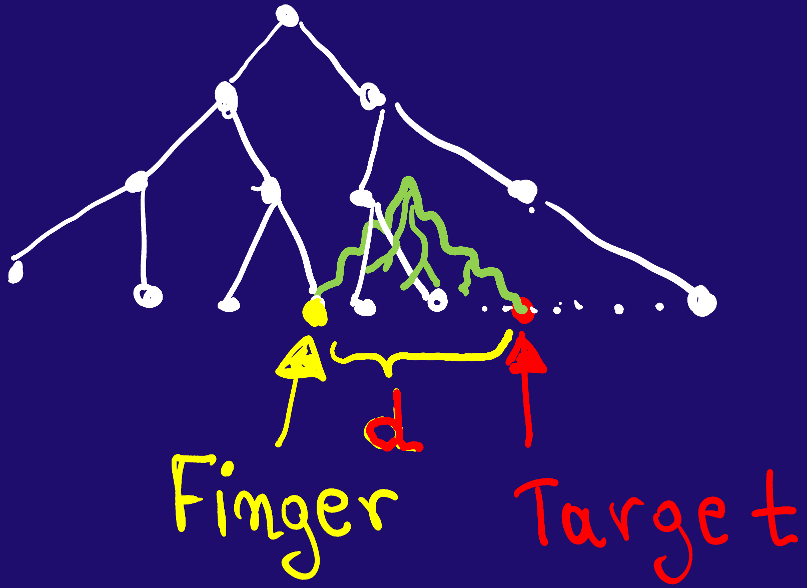
What about removing n from loglog(n)?

Kaporis, Makris, Sioutas, Tsakalidis, Tsichlas & Zaroliagis (2003) showed order of loglog(d),
d= distance(target key, finger key)



What about removing n from loglog(n)?

Kaporis, Makris, Sioutas, Tsakalidis, Tsichlas & Zaroliagis (2003) showed order of loglog(d),
d= distance(target key, finger key)



All previous results assumed continuous input distributions, with keys real numbers in an interval [a, b]

All previous results assumed continuous input distributions, with keys real numbers in an interval [a, b]

This assumption greatly simplifies the analysis, but, it does not hold on real-world data structures, governed by discrete distributions

All previous results assumed continuous input distributions, with keys real numbers in an interval [a, b]

This assumption greatly simplifies the analysis, but, it does not hold on real-world data structures, governed by discrete distributions

Why not removing this “innocent” assumption???

All previous results assumed continuous input distributions, with keys real numbers in an interval [a, b]

This assumption greatly simplifies the analysis, but, it does not hold on real-world data structures, governed by discrete distributions

Why not removing this "innocent" assumption???

Because, strong experimental evidence showed that all existing versions of IS behave poorly on finite keys, e.g., alphabetic tables, Perl & Gabriel (1992)

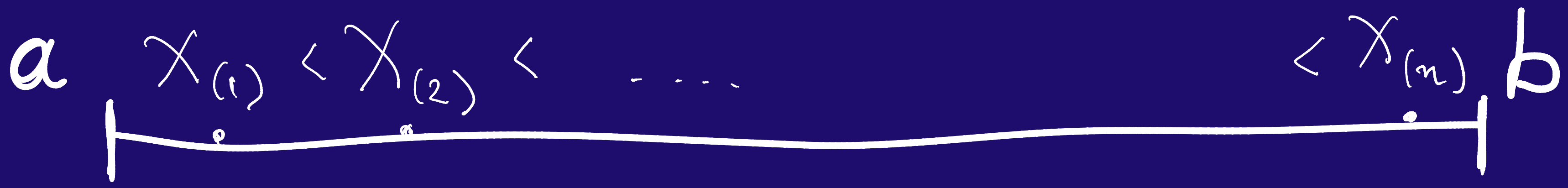
All previous results assumed continuous input distributions, with keys real numbers in an interval [a, b]

This assumption greatly simplifies the analysis, but, it does not hold on real-world data structures, governed by discrete distributions

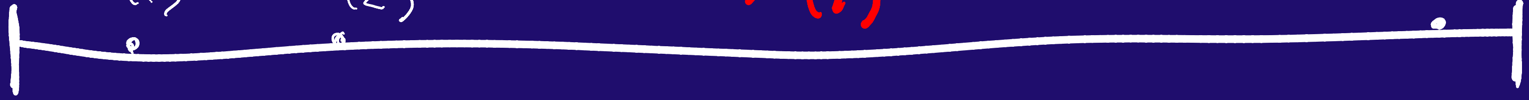
Why not removing this “innocent” assumption???

Because, strong experimental evidence showed that all existing versions of IS behave poorly on finite keys, e.g., alphabetic tables, Perl & Gabriel (1992)

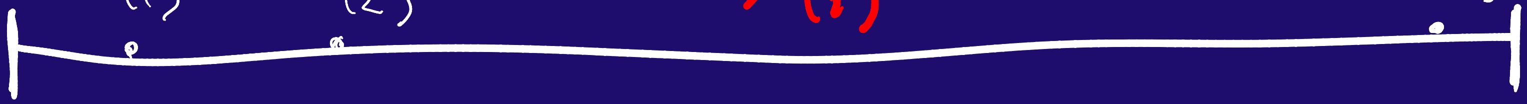
Frankly, IS was misled by repetitions of finite keys
(recall: key repetition has probability 0 in a continuous distribution)


a
$X_{(1)} < X_{(2)} < \dots\dots < X_{(n)}$
b

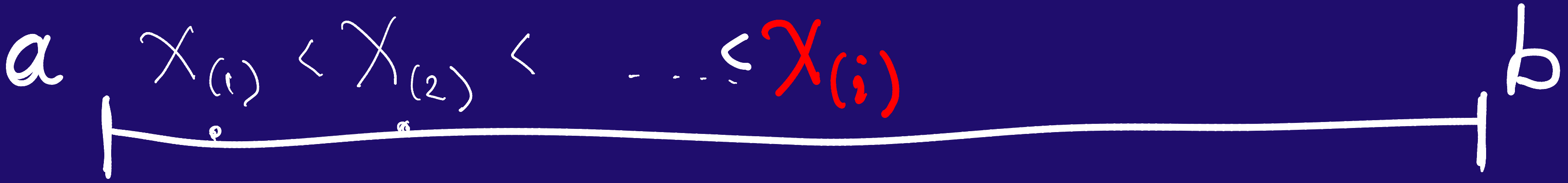
$$a \quad X_{(1)} < X_{(2)} < \cdots < X_{(i)} < \cdots < X_{(n)} \quad b$$



$$a \quad x_{(1)} < x_{(2)} < \cdots < x_{(i)} < \cdots < x_{(n)} \quad b$$



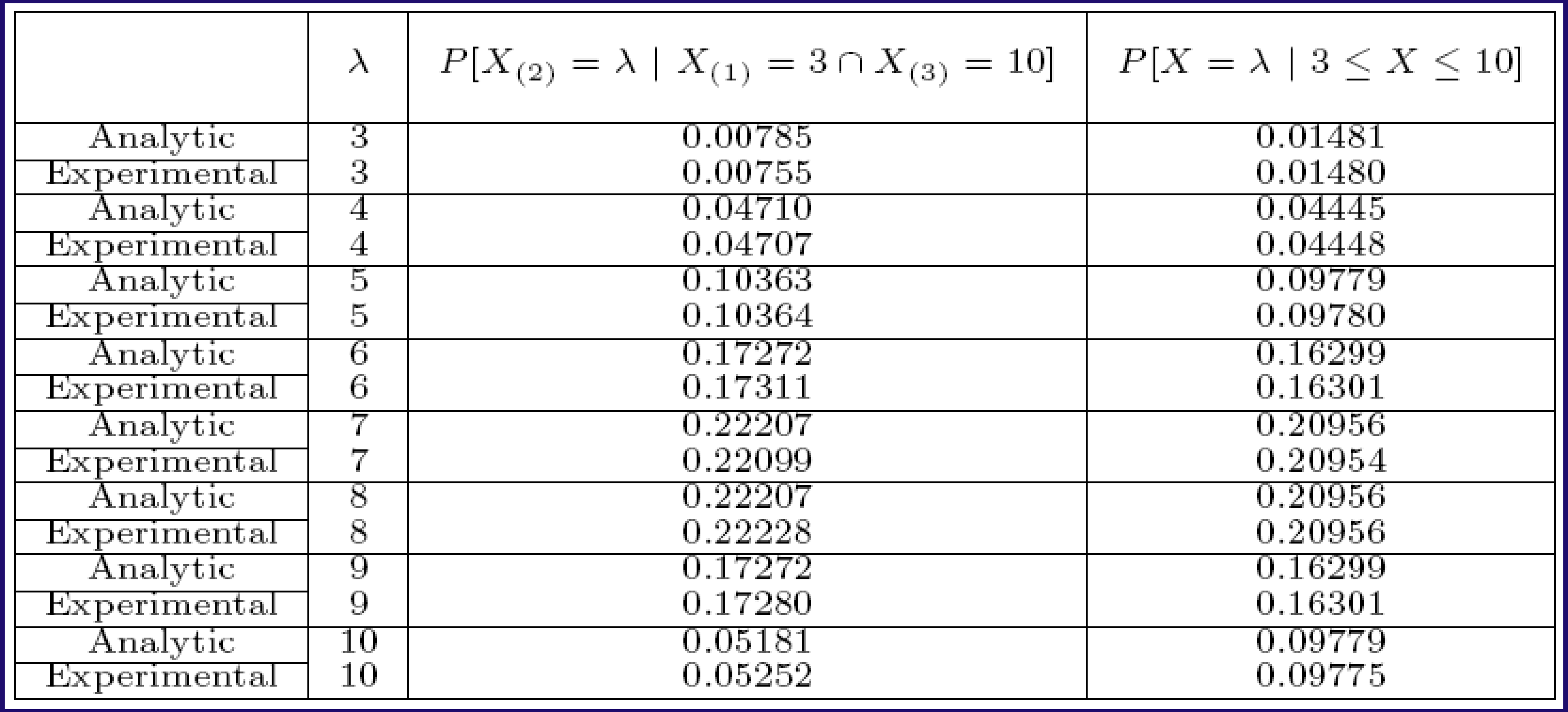
$$x_{(i)} > \text{target}$$



$X_{(1)} < X_{(2)} < \dots < X_{(i)}$

$a \quad X_{(1)} < X_{(2)} < \dots < X_{(i)} \quad b$

is λ random?



	λ	$P[X_{(2)} = \lambda \mid X_{(1)} = 3 \cap X_{(3)} = 10]$	$P[X = \lambda \mid 3 \leq X \leq 10]$
Analytic	3	0.00785	0.01481
Experimental	3	0.00755	0.01480
Analytic	4	0.04710	0.04445
Experimental	4	0.04707	0.04448
Analytic	5	0.10363	0.09779
Experimental	5	0.10364	0.09780
Analytic	6	0.17272	0.16299
Experimental	6	0.17311	0.16301
Analytic	7	0.22207	0.20956
Experimental	7	0.22099	0.20954
Analytic	8	0.22207	0.20956
Experimental	8	0.22228	0.20956
Analytic	9	0.17272	0.16299
Experimental	9	0.17280	0.16301
Analytic	10	0.05181	0.09779
Experimental	10	0.05252	0.09775