Comparing Equally Often, the Number of Slow-Memory Reads Is Set by the Tree's Branching

— With 10^9 elements, binary search makes **29.897353** comparisons, and a tree of branching **100**, at depth **4.500000**, makes the same **29.897353** — the separator is whether one counts comparisons or page reads —

[Paper 759]

Yuuki YAMAGISHI (山岸 勇輝)
Yamagishi Complexity Torus Lab, Japan
September 2026

— *From thinking to syncing.* —

Abstract

Both binary search and a tree index take effort growing as the logarithm of the number of elements, and so are readily called the same.Comparisons and reads of slow memory are counted separately.No new theorem or law is claimed. First, the number of comparisons does not depend on the tree's branching——with 10^9 elements, 29.897353 for binary search and for a tree of branching 100 alike (Section 2). **Second, reads of slow memory differ greatly——with 100 elements per page, 24.253497 for binary search and 4.500000 for the tree** (Section 2). **Third, and this is the core. The ratio of reads approaches the ratio of logarithm bases——5.389666 at 10^9 elements and 6.016761 at 10^{18}, never exceeding $\log_2 100 = 6.643856$** (Section 3). **Fourth, small tables differ less——the ratio is 2.881285 at 10^3 elements** (Section 3). **Fifth, a trial shows the same direction——with 10^6 elements and 2000 searches, pages read average 13.212000 for binary search and 3.000000 for the tree** (Section 3). **Sixth, the separator is whether one counts comparisons or page reads——"both are logarithmic" says nothing about speed until one says what is counted** (Section 4). **Keywords**: binary search, B-tree, index, base of the logarithm, page reads, memory hierarchy.

Scope of this paper (scope note): **No new theorem or law is claimed——that binary search makes $\log_2 n$ comparisons, that a tree of branching B has depth $\log_B n$, and that external-memory indexes use trees of large branching are all known (Cormen et al. 2009, Bayer and McCreight 1972).Wall-clock time is not measured——only the number of reads is counted; the time per read and the device are not treated.Tree maintenance is not treated——insertion, deletion, node splitting and fill factor are not treated; only a full static tree is considered.One page size——100 elements per page is a value for comparison.Counts are kept apart——the trial's binary search counted equality and order as two comparisons, so its comparisons are not set beside the closed form; only reads are compared.Relation to earlier papers**: **Paper 702 showed that dividing does not always make it faster, and that what decides is not the division but the combining——its scope note stated that memory hierarchy, caches and parallel effects would not be included. There asymptotic exponents competed. Here, within the same $O(\log n)$, reads of slow memory make the base of the logarithm matter.Paper 705 is also cited——there, even at a hit rate of 0.99, heavy misses took more than half the time. Here the tree's branching decides how many of those heavy reads remain.**What is added is **setting comparisons that ignore the base beside reads that depend on it**, **showing that the ratio of reads approaches $\log_2 100 = 6.643856$ with size**, **counting reads in a trial**, and **placing the separator on comparisons against reads**.

1. Notation and the Question

Setting: **search a sorted table of n elements. Binary search makes $\log_2 n$ comparisons; a tree of branching B has depth $\log_B n$, and searching within each node by bisection also makes $\log_2 n$ comparisons. With B elements per page of slow memory, the number of reads is as follows.**

$$C_{\text{bin}} = \log_2 n = \log_B n \cdot \log_2 B, \qquad P_{\text{bin}} = \log_2 \frac{n}{B} + 1, \qquad P_{\text{tree}} = \log_B n, \qquad \frac{P_{\text{bin}}}{P_{\text{tree}}} \to \log_2 B$$

Take $B = 100$.

Question: **if both take logarithmic effort, are binary search and a tree index equally fast?**

2. Equal Comparisons, Unequal Reads

Computation 2.1: **with 10^9 elements, comparisons and reads were derived.**

quantity	binary search	tree of branching 100
comparisons	**29.897353**	**29.897353**
reads of slow memory	**24.253497**	**4.500000**

Observation 2.2

the number of comparisons does not depend on the tree's branching.The tree's comparisons equal $\log_2 n$ to within 3.553×10^{-15}, and are $1/(1.672389 \times 10^{7})$ of the linear average of 5×10^{8}——counted in comparisons, the two efforts do not differ.

Observation 2.3

reads of slow memory differ greatly.24.253497 for binary search, 4.500000 for the tree, a ratio of 5.389666 ——binary search reads a different page at every step until it lands in the last one.

3. The Ratio of Reads Approaches the Ratio of Bases

Computation 3.1: **the ratio of reads was derived for tables of various sizes.**

elements	ratio of reads (binary / tree)
10^{3}	2.881285
10^{6}	4.762571
10^{9}	5.389666
10^{12}	5.703213
10^{18}	6.016761
limit $\log_2 100$	6.643856

Observation 3.2

this is the core: the ratio of reads grows with size and approaches the ratio of logarithm bases.It rises from 2.881285 at 10^{3} to 6.016761 at 10^{18} and never exceeds 6.643856——within the same $O(\log n)$, counting reads leaves the difference of bases as a constant factor.

Computation 3.3: **in a table of 10^{6} elements, 2000 randomly chosen keys were searched, counting the pages binary search touched and the nodes the tree read.**

quantity	average
pages read by binary search	13.212000
nodes read by the tree	3.000000

> **Observation 3.4**
>
> **in the trial too, the tree reads fewer times than binary search touches pages.The tree read exactly 3 nodes each time; binary search touched 13.212000 pages on average——comparisons were counted differently in the trial, so only reads are set side by side.**

4. The Separator

> **Observation 4.1**
>
> **the separator is whether one counts comparisons or page reads.Counting comparisons, the base vanishes; counting reads, a difference remains that tends to the ratio of bases, 6.643856——"both are logarithmic" says nothing about speed until one says what is counted.**

> **Observation 4.2**
>
> **beside Paper 702.Paper 702 decided the speed of divide-and-conquer by comparing asymptotic exponents, and stated that memory hierarchy would not be included——this is the case of equal exponents, where the machine detail left out makes the base matter.**

> **Observation 4.3**
>
> **beside Paper 705 as well.Paper 705 showed heavy misses taking more than half the time even at a hit rate of 0.99——a read of slow memory is such a heavy miss, and the tree's branching reduces how many there are.**

Observation 4.4

what is not said.Wall-clock time, tree maintenance and cache details are not treated——what is shown is that comparisons and reads are different numbers, and the base matters only for the latter.

5. Roll Call and Closing

1. **comparisons 29.897353** (Computation 2.1): **the same for both.**
2. **binary-search reads 24.253497** (Computation 2.1): **a new page each step.**
3. **tree reads 4.500000** (Computation 2.1): **only the depth.**
4. **ratio 5.389666** (Observation 2.3): **at 10^9 elements.**
5. **2.881285 at 10^3** (Computation 3.1): **a small table.**
6. **6.016761 at 10^{18}** (Computation 3.1): **growing.**
7. **limit 6.643856** (Observation 3.2): **the core.**
8. **13.212000 pages in the trial** (Computation 3.3): **binary search.**
9. **3.000000 nodes** (Computation 3.3): **the tree.**
10. **comparisons or reads** (Observation 4.1): **the separator; Paper 702 and Paper 705.**

Closing: **Comparing equally often, the number of slow-memory reads is set by the tree's branching.With 10^9 elements, binary search and a tree of branching 100 both make 29.897353 comparisons, but with 100 elements per page they read 24.253497 and 4.500000 times, a ratio of 5.389666——binary search reads a different page at every step until it lands in the last one.The ratio of reads grows with size from 2.881285 at 10^3 to 6.016761 at 10^{18}, approaching $\log_2 100 = 6.643856$——within the same $O(\log n)$, counting reads leaves the difference of bases as a constant factor.In a trial with 10^6 elements, binary search touched 13.212000 pages and the tree read 3.000000 nodes.The separator is whether one counts comparisons or page reads——"both are logarithmic" says nothing about speed until one says what is counted.Placed among the earlier papers——Paper 702 compared exponents and left memory hierarchy out; this counts how that machine detail makes the base matter when exponents are equal. What reduces Paper 705's heavy misses is the tree's branching.To be honest——wall-clock time, tree maintenance and cache details are not treated.**

References

1. Bayer, R., & McCreight, E. (1972). Organization and maintenance of large ordered indexes. *Acta Informatica*.

2. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.

3. Yamagishi, Y. (2026). *Dividing Does Not Always Make It Faster: What Decides Is Not the Division but the Combining* [Paper 702]. Zenodo. 10.5281/zenodo.23014909

4. Yamagishi, Y. (2026). *Even at a Hit Rate of* **0.99***, More Than Half the Time Is Spent on Misses* [Paper 705]. Zenodo. 10.5281/zenodo.23014923

— From thinking to syncing. —

Yuuki YAMAGISHI (山岸 勇輝)

© 2026 Yamagishi Complexity Torus Lab, Japan · Licensed under Creative Commons Attribution 4.0 International (CC BY 4.0)