B-tree Design

Data Structure for Ordered Collection

RAM

hard drive

Memory Type	approximate access time (ns)	cost per megabyte ($)
cache	5-20	10-75
main memory	60-120	0.50-5.00
secondary storage	20,000,000	.001-0.10

Table 29.1 Approximate access times in nanoseconds (ns) as compared with the cost for cache, main memory, and secondary storage (disk).

Designed for situation in which n (# elements or tags in a tagged collection) is so large that B-tree + n references to elements cannot fit in main memory.

2-4K bytes

Data is moved from disk to RAM in chunk called <u>page</u>

Page Fault

Virtual memory lets you store data in secondary storage (act as if it's in main memory)

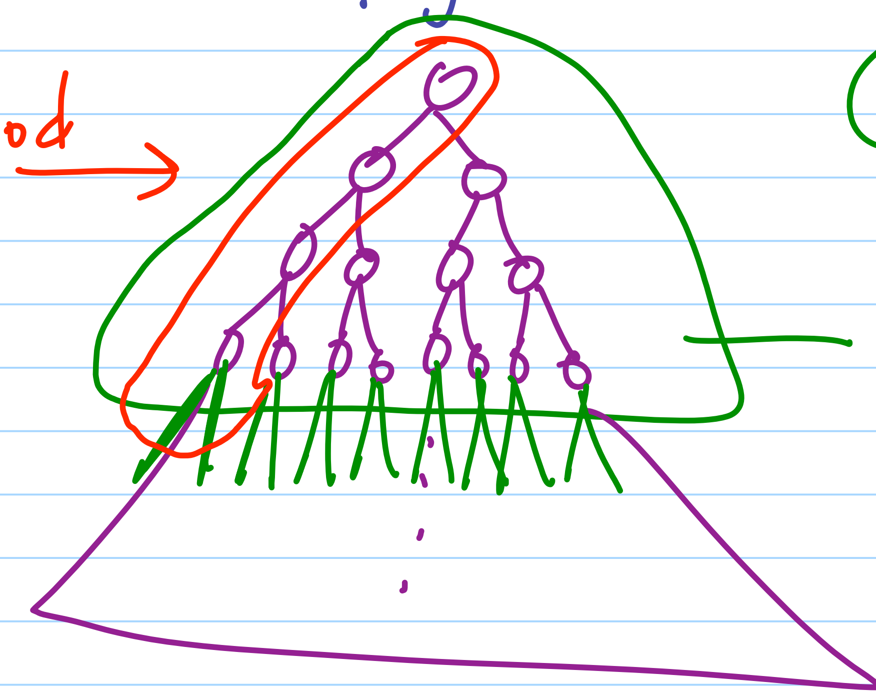
If program accesses data not in main memory, it's a page fault.

For a binary search tree, information we must store.

per tag

- tag (comparable) — date of a historical event
- location on disk (secondary storage) for the rest of information associated with event
- parent ref
- left child ref
- right child ref

Want to group nodes of binary search tree into a single B-tree node that fills a disk page

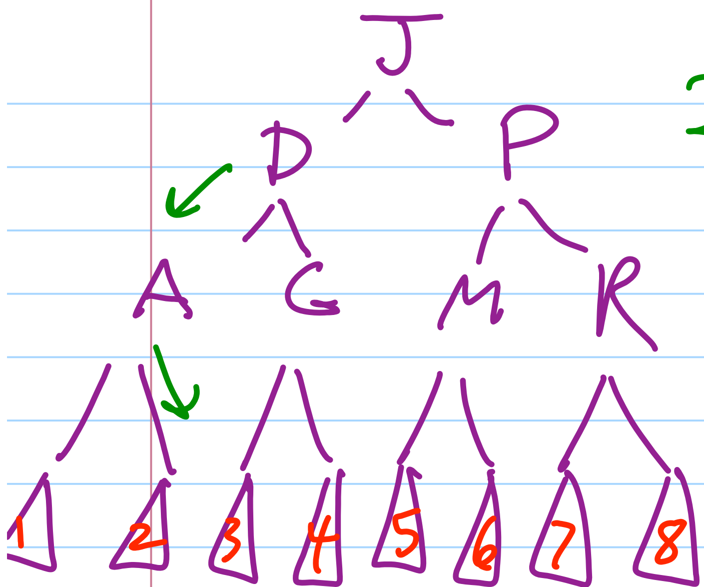


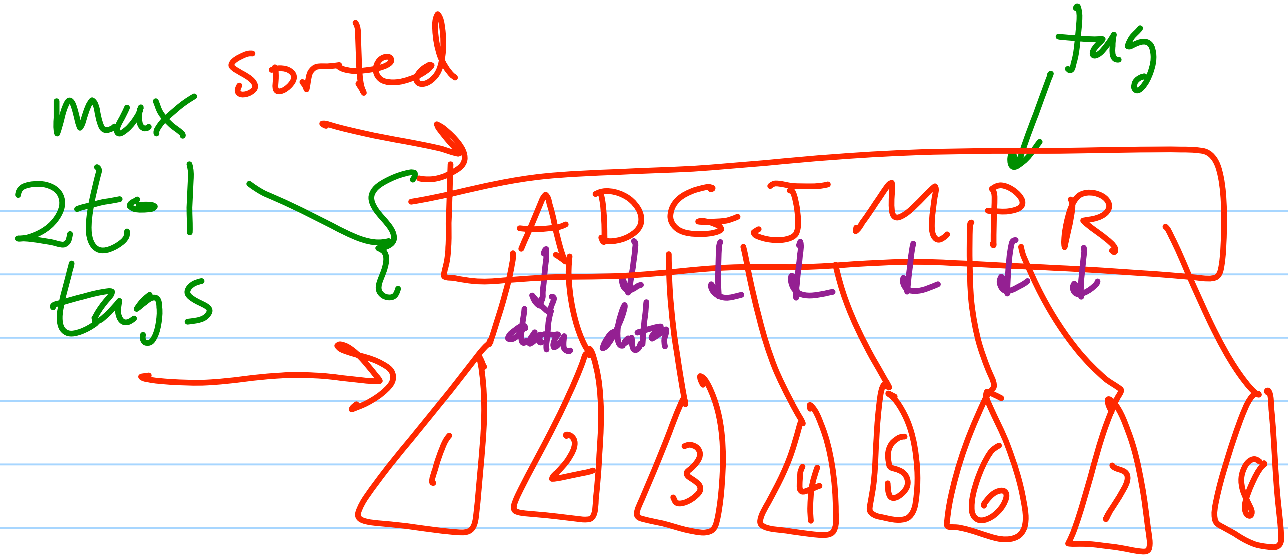
not a good choice because we may only look at one node. Usually, look at very few

Good

make this a disk page (go as many levels as we can fit)

$1/16^{th}$ of data left to consider

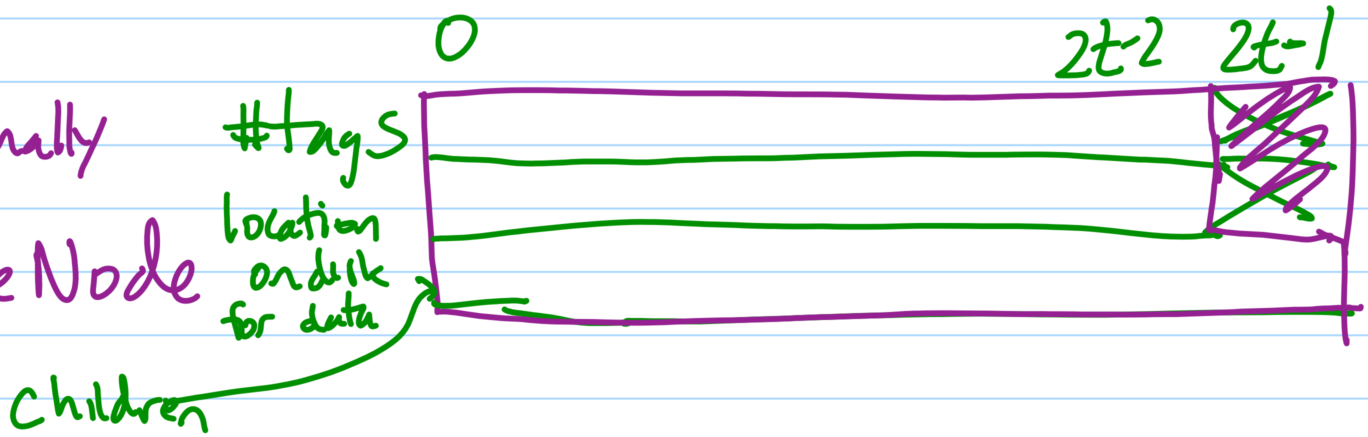




INORDER

Internally

BtreeNode



children is at most $2t$