B^+ Trees

What makes a good index?
- useful
- small
- make finding quicker

2 types of indicies:
- check existence: hash
- range queries: orded index

Things to consider
- Access type: single elt vs range
- Access time: how long to find something
- Insertion time:
- Deletion time:
- Space

def **Search key** as attribute used to lookup a record

About B^+ tree:
- balanced tree w/ every path from the root to a leaf has same length
- each non-leaf node have between $\lceil \frac{n}{2} \rceil$ and n children (where n is fixed for the tree)

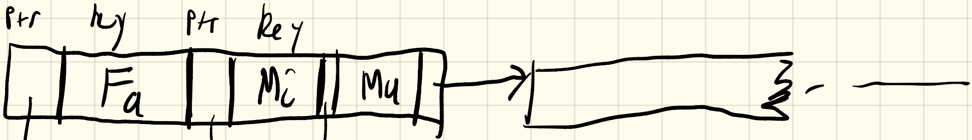
Structure of a leaf

- up to $n-1$ search keys $K_1, \ldots, K_{n-1}$ w/ $K_i < K_j$ if $i < j$

- up to n pointers $P_1, \ldots, P_n$ w/ P_i points to a record w/ search key K_i

- each leaf points to "neighbor" leaf

$n=4$



Records:

id	name	dept	course #
20	Sh	CS	447
21	Mu	CS	451
21	Fa	CS	432
23	Pa	CS	127
24	Mi	CS	540

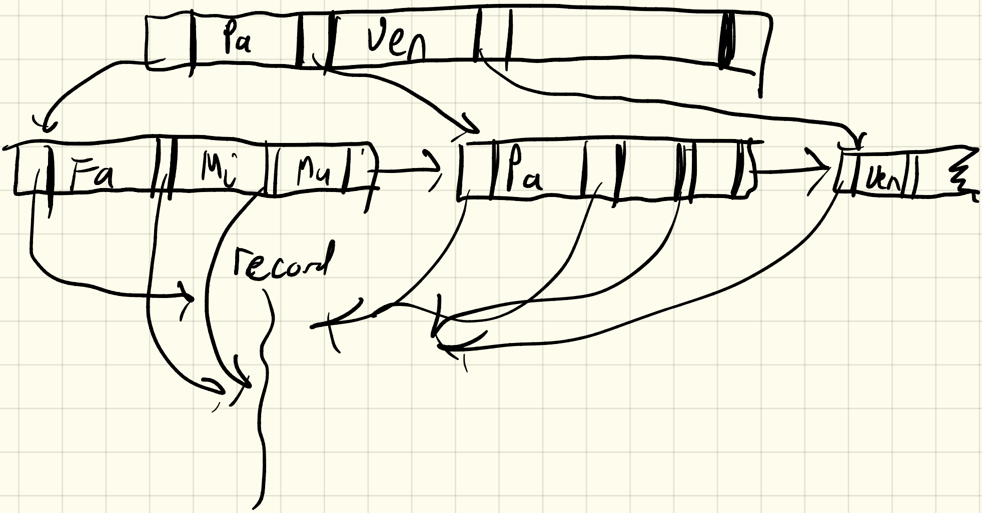
- each leaf may contain between $\lceil \frac{n-1}{2} \rceil$ an $n-1$ values

Non leaf nodes

- similar struct to leafs

But:

- ptrs are to other nodes
- no ptr to sibling
- each node holds between $\lceil \frac{n}{2} \rceil$ and n ptrs
  \# of ptrs sometimes called fanout



Bin search tree vs B-tree

consider table w/ 1,000,000 records

Bin search tree: height $\log_2(1{,}000{,}000) \simeq 20$

B-tree $n=100$: height $\log_{50}(1{,}000{,}000) \simeq 4$