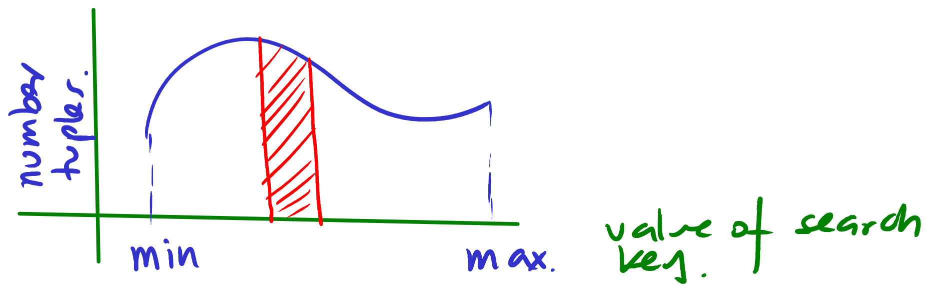
Selectivity of matching a primary key: $\frac{1}{|R|}$

Selectivity of matching a nonprimary key:

We need a distribution.



We need to compute proportion of the tuples that match the predicate.

Example:

- Relation $R(\underline{a}, b)$
- Assume a uniform distribution of values of b, min 1, max 100.

$\text{Selectivity}(b = 10) = \frac{1}{100}$.

$\text{Selectivity}(b > 20) = \frac{79}{100}$

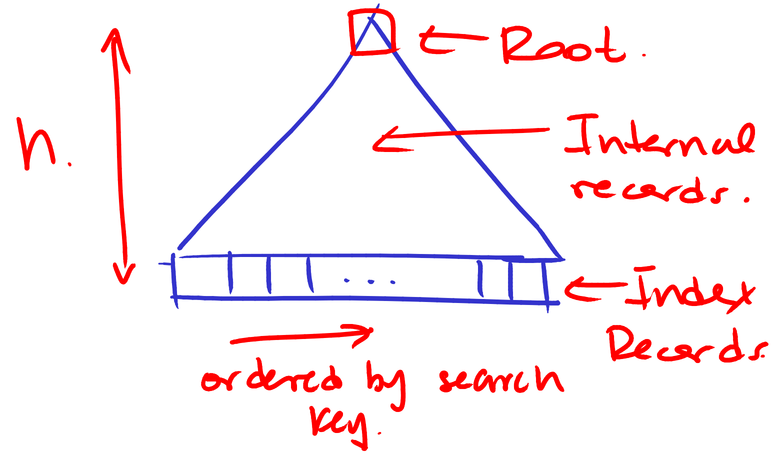
$\text{Selectivity}(b > 20 \text{ and } b < 30) = \frac{9}{100}$

$\text{Selectivity}(b > 200) = \emptyset$

Indexes.

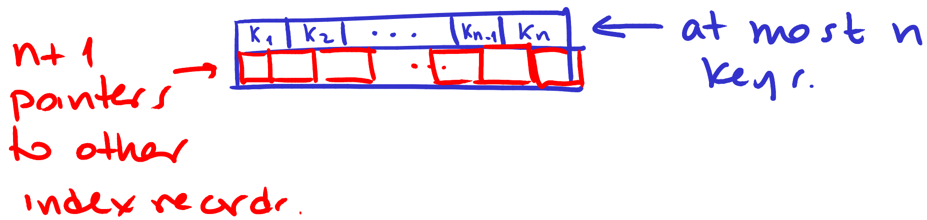
B+-tree

- Automatically Balanced
- Index records are at the leaves



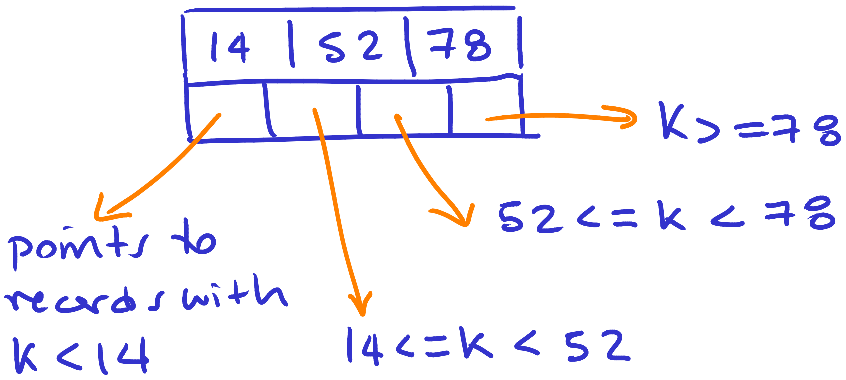
- Every record is a block.
- Index records form a list
  - They can be traversed in the order of the search key

Internal records

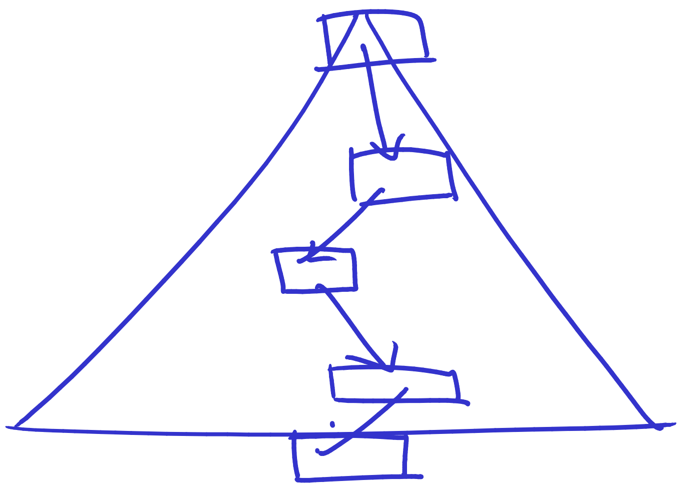


Example.

Assume $n = 3$



Index is traversed from root to leaf



We assume root is always in memory hence, to reach the leaf we read h blocks.

$$\#\text{blocks in index} = \left\lceil \frac{\#\text{matching tuples}}{n \cdot \text{fill}} \right\rceil = \frac{.5 \cdot 10^6}{10^2} = .5 \cdot 10^4$$

$$\text{Cost} = h - 1 + 0.5 \cdot 10^4 + 10^6$$

$$\approx 10^6 !!$$

Approximately 20 times more than Sparse index!!

How do we know how many tuples match a given query?

$$\sigma_p R$$

$$\text{Selectivity}(p) = \frac{\#\text{ expected matching tuples}}{|R|}$$

$\Rightarrow$ selectivity $(p) \triangleq$ probability that a tuple in R matches predicate P.

Example.

$R(\underline{a}, b)$ $|R| = 10^6$ tuples.

heap: 10 tuples per block

Dense index on $R(a)$

Effective index records per block: 100

Assume $h = 4$

a) $\sigma_{a=5} R$

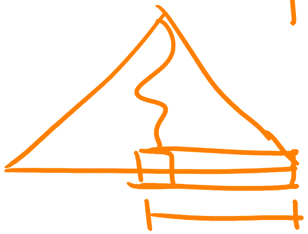
if tuple exists: cost = $h+1$.

if tuple does not exist: cost = h.

Similar to sparse!!

b) $\sigma_{a>10} R$

Assume 50% of tuples match.



blocks in index.

We need to scan 50% of them.

Cost of index:

- Cost of reaching the leafs
- Cost of reading the matching records.

Example:

1) Assume $R(\underline{a}, b)$

$\sigma_{a=5} R$

Only one or zero matching tuples

⇒ We must traverse the index h.

The leaf either

- contains $a=5$ or not

Cost of index = h of index.

2) $\sigma_{a>0} R$

What if all tuples match?

- We traverse index (cost h)

Reads first leaf.

- We must read all leaves of index

Cost of index = h + # leaves − 1

To be able to compute the cost of an index we need:

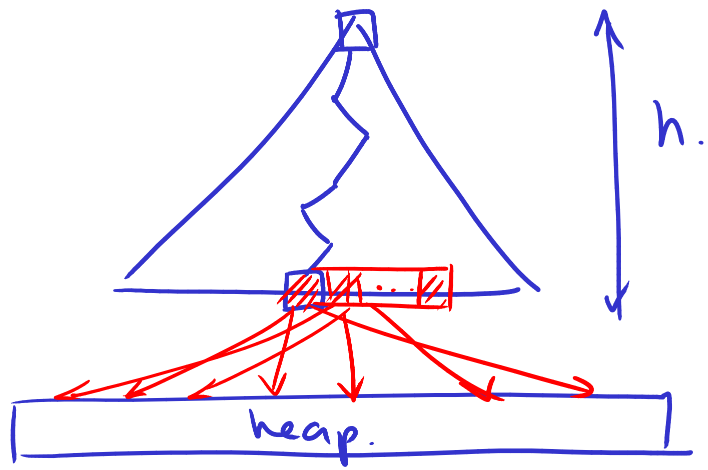
Calculate # of leaf blocks of index proportional to # index records per block.

# of index records depends upon
a) Type of index
Sparse vs. Dense
b) Number of tuples in Rel.

index records per block depends upon:
a) size of search key
b) occupancy rate
=> How much waste space is there in the index (to keep it balanced).

We assume that occupancy rate of inner nodes and leaves is the same as internal nodes.

Dense Index.



Cost:

$h - 1$ + # leaves in Index
+ 1 block per matching tuple.

- Traverse tree to first leaf : h
- Might need to read $\emptyset$ or more leafs.
- For every tuple in result, read one block.

$$h - 1 + \left\lceil \frac{\#\text{matching tuples}}{n \cdot \text{fill}} \right\rceil$$

$$+ \ \#\text{matching tuples}.$$

$$h + \left\lceil \frac{\#\text{ matching tuples}}{\text{tuples per block}} \right\rceil$$

Example.

$R(\underline{a}, b)$ $\quad |R| = 10^6$ tuples.

heap: 10 tuples per block

Sparse index on $R(a)$

Assume $h = 3$

a) $\sigma_{a=5} R$

if tuple exists: cost $= h + 1$.

if tuple does not exist: cost $= h$.

b) $\sigma_{a>10} R$

Assume 50% of tuples match query:

$\Rightarrow$ We must scan 50% of sorted heap.

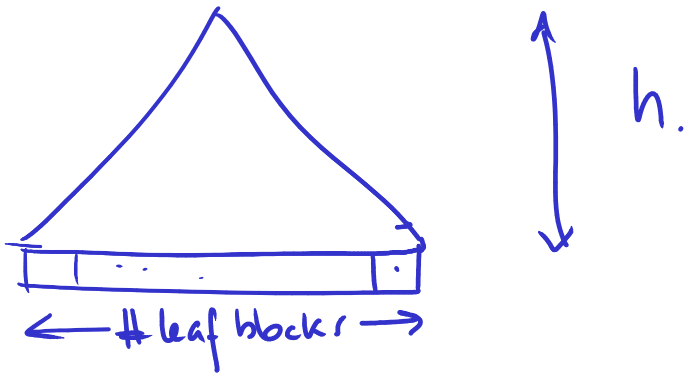
$$\text{Cost} = h + \frac{B(R)}{2}$$

$$B(R) = \frac{10^6}{10} = 10^5$$

Cost $= 3 + 0.5 \cdot 10^5$ blocks.

Approx: $0.5 \cdot 10^5$ blocks :

Hence, height of tree depends upon # of leaf records.



n = max number of keys per record.

fill = occupancy rate (between 0–1, but usually around ½ to ¾)

$$\#\text{leaf blocks} = \left\lceil \frac{\#\text{ index records}}{n \cdot \text{fill}} \right\rceil$$

For h, we simplify calculations:

$$h = \left\lceil \log_{n \cdot \text{fill}} (\#\text{index records}) \right\rceil$$

Example:

Assume $n = 150$, $fill = \frac{2}{3}$

How many index records can we store in an index of height 1, 2, 3, 4, 5, 6.

$$h = \left\lceil \log_{n \cdot fill} \#indexRecords \right\rceil$$

Let us worry about max #index Records

$$\Rightarrow \quad h \simeq \log_{n \cdot fill}(\#indexRecords)$$

$$n \cdot fill^{h} \approx \#indexRecords$$

h	#index Records	
1	100	$= 10^2$
2	100^2	$= 10^4$
3	100^3	$= 10^6$
4	100^4	$= 10^8$
5	100^5	$= 10^{10}$

With 5 block reads we can find a leaf with a given search key in an index of 10 giga-records!!

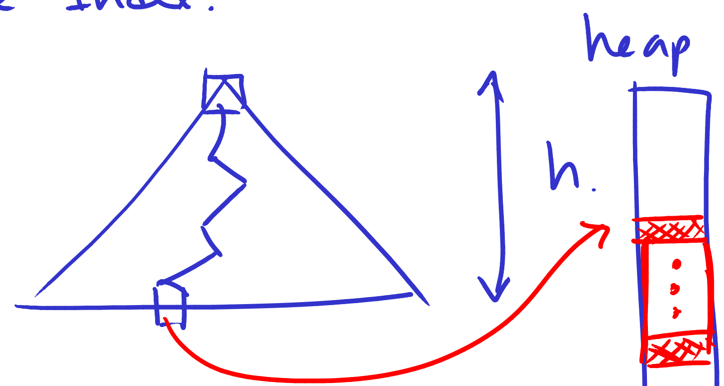
How many search keys do we need to store?

- Sparse index: $B(R)$
- Dense index: $|R|$

Sparse index is marginally shorter than dense index.

Cost of Using an Index

Sparse Index.



Cost of using index
$h + \#\text{blocks read in heap}$.

- We only read one record in index
- We read 1 or more blocks from heap