Exercise (B+-tree index)

Suppose we have a relation r = (A,B,C), with A primary key.
Assume

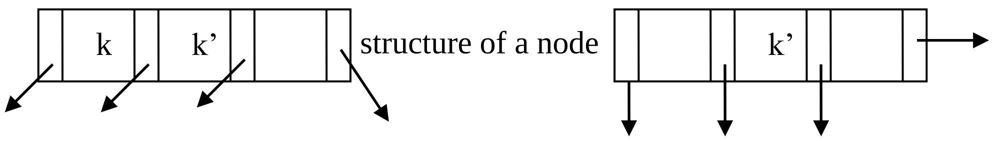
nr = 100.000	number of records in the relation
Lr = 50 byte	size of a record (fixed length record)
LA = 6 byte	size of attribute A
Lp = 4 byte	size of a pointer
Lb = 1000 byte	size of a block

Heap file organization

1. Show the number of leaves of a B+-tree index on search-key A
2. Cost in terms of number of block transfers from disk of the queries:

    1) select * from r where A=xxx;
    2) select * from r where 2.000 <= A < 3.000;
       assuming A uniformly distributed on the interval [1; 500.000]
    3) select * from r where B = xxxx;
       where B is not a key

Point 1



nonleaf node leaf node

Heap file organization

We have a B+-tree secondary index. The index is dense, with an entry in the leaves for every search-key value in the file. Since A is a key of the relation, the number of search-key values in the leaves of the B+tree is equal to the number of records in the file (100.000).

We evaluate the maximum number of (key, point) in a node (blocking factor of the index, named f_I)

$$f_I = \left\lfloor \frac{Lb - Lp}{(LA+Lp)} \right\rfloor = \left\lfloor \frac{1000-4}{6+4} \right\rfloor = 99$$

m = 100	fanout of the nodes: max number of pointers in a node
m-1 = 99	number of search-key values

$$\left\lceil \frac{100.000}{99} \right\rceil = 1011$$ Minimum number of leaf nodes in the B+-tree

We evaluate the minimum number of (key, point) in a node.

$\lceil m/2 \rceil = 50$	minimum number of pointers in a node
$\lceil m/2 \rceil - 1 = 49$	number of search-key values

$$\left\lceil \frac{100.000}{49} \right\rceil = 2041$$ Maximum number of leaf nodes in the B+-tree

Number of leaves: 1011 <= n_{leaves} <= 2041

==

Let h be the height of a B+-tree, it can be shown that

Full nodes:

1	level 1
|	
m	level 2
|	
m*m	level 3
……………….	
m*m … *m => m^{h-1}	level h

- number of blocks (nodes) is:

$$1 + m + m^2 + \dots + m^{h-1} = \quad (m^h - 1) / (m-1)$$

- number of search-key values is:

$m^h - 1$ (number of nodes * number of values in the node)

Given the number of leaves, the height of the B+tree can be computed as follows:

$$n_{leaves} = m^{h-1}$$

$$h-1 = \log_m (n_{leaves})$$

$$h = 1 + \log_m (n_{leaves})$$

Half full nodes:

- number of blocks (nodes) is:

$$1 + 2 + 2\lceil m/2 \rceil + \dots + 2\lceil m/2 \rceil^{h-2} =$$

$$= 1 + 2\,\frac{\lceil m/2 \rceil^{h-1} - 1}{\lceil m/2 \rceil - 1}$$

- number of search-key values is:

$2\lceil m/2 \rceil^{h-1} - 1$ (number of nodes * min number of values in the node)

- height of the tree

$h = 1 + \log_{\lceil m/2 \rceil} (n_{leaves})$

==

Point 2

Hight of the B+-tree

$1 + \log_{100} (1011) <= h <= 1+\log_{50}(2041)$
h = 3
Worst-case scenario. h=3

Point 2.1
select * from R where A=xxx

Cost of the query:

C = height of the B+-tree + 1 block for the file
C = 3 + 1 = 4

Point 2.2

select * from R where 2.000 <=A<3.000

- Cost of the query using the index

fs = 1.000/500.000 = 1/500 **selectivity factor of the query**

Let h be the height of the B+-tree

$$C = (h-1) + \lceil fs * n_{leaves} \rceil + \lceil fs * n_r \rceil$$

Number of leaf node transfers:

$$\lceil fs * n_{leaves} \rceil = \lceil 1/500 * 2041 \rceil = 5$$

Number of file block transfers:

$$\lceil fs * n_r \rceil = \lceil 1/500 * 100.000 \rceil = 200$$
(heap file organization, a block transfer for each record)

C= 2 + 5 + 200 = 207

- Cost of sequential scan of the file

Number of blocks of the file: 5000
Worst case cost: 5000 and the best case cost is 1. On average, we have:$(n_b + 1)/2$ = 2.500
C' = 5000

Cost of the query: min(C, C') = min(205, 5.000) = 205

Point 2.3
select * from r where B = xxxx;
No index on B. Moreover B is not a key. We estimate $C = n_b$
$C = 5.000$

Exercise (B+-tree index)
Same exercise, assuming **sequential file organization on search key A**.

Point 1

Sparse index. We have number of values in the index equal to number of blocks of the file. We evaluate the number of blocks in the file.

$$f_r = \left\lfloor \frac{Lb}{Lr} \right\rfloor \qquad f_r = \left\lfloor \frac{1000}{50} \right\rfloor = 20$$

blocking factor of the relation r max number of records in a block of the file

$$n_b = \left\lceil \frac{nr}{f_r} \right\rceil \qquad n_b = \left\lceil \frac{100.000}{20} \right\rceil = 5.000$$

number of blocks of the file

$$\left\lceil \frac{5.000}{99} \right\rceil = 51$$

Minimum number of leaf nodes in the B+-tree

$$\left\lceil \frac{5.000}{49} \right\rceil = 103$$

Maximum number of leaf nodes in the B+-tree

Number of leaves: $51 <= n_{leaves} <= 103$

Point 2

$1 + \log_{100} (51) <= h <= 1+\log_{50}(103)$
$2 <= h <= 3$

Worst-case scenario. h=3

Point 2.1

select * from R where A=xxx

- Cost of the query using the index
  C = height of the B+-tree + 1 block for the file
  C = 3 + 1 = 4

- Cost of the query using binary search
  $C' = \lceil \log_2 n_b \rceil = \lceil \log_2 5.000 \rceil = 13$

Cost of the query: min(C, C') = min(4, 13) = 4

Point 2.2

select * from R where 2.000 <=A<3.000

- Cost using the index:
  fs = 1/500

$$C = (h-1) + \lceil fs * n_{leaves} \rceil + \lceil fs * n_b \rceil$$

Number of leaves transfers:

$$\lceil fs * n_{leaves} \rceil = \lceil 1/500 * 103 \rceil = 1$$

Number of file block transfers:

$$\lceil fs * n_b \rceil = \lceil 1/500 * 5000 \rceil = 10$$

(sequential file organization, records are stored in search-key order in the blocks)

C = 2 + 1 +10 = 13

Point 2.3

select * from r where B = xxxx;

No index on B. Moreover B is not a key. We estimate $C = n_b$
C = 5.000