Practicing B+ Trees

Database Applications - Recitation 10

Zeinab Khalifa

March 26th, 2020

Exercise (1)

Preparing our B+ Tree

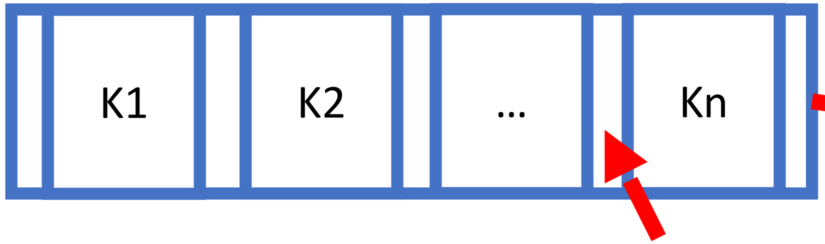
Consider the relation

`Student(sid: type, name: char, major: char, gpa: double)`

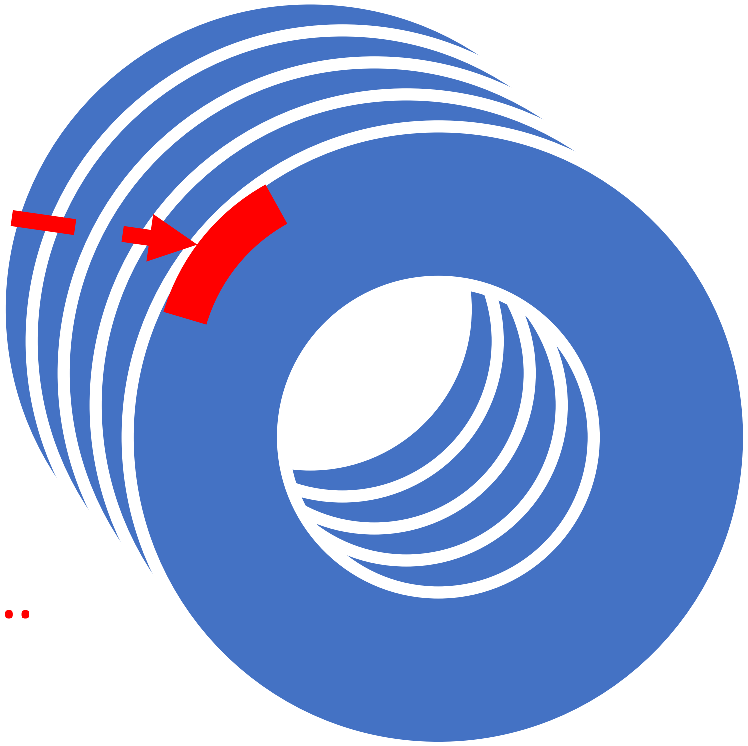
My index is on sid

Let's assume our mini disk has a block size of 64 bytes

A B+ Tree Page



(n+1) Pointers

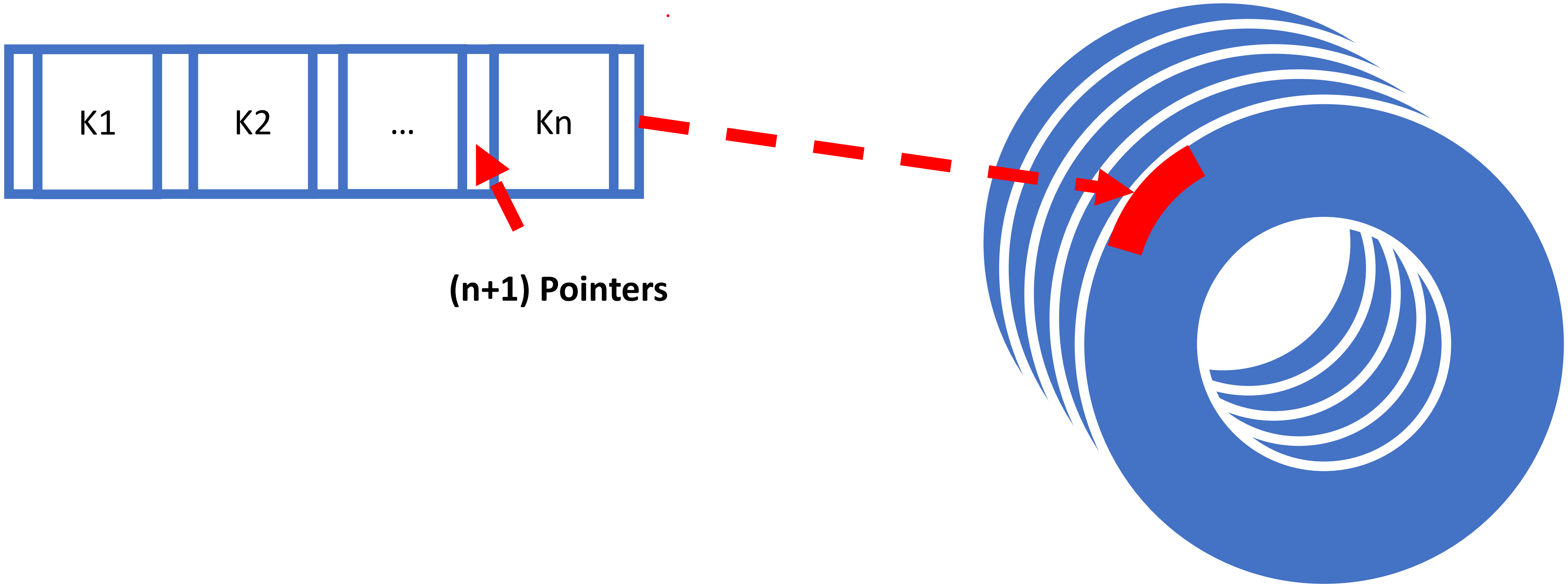


How can we build our B+ tree?

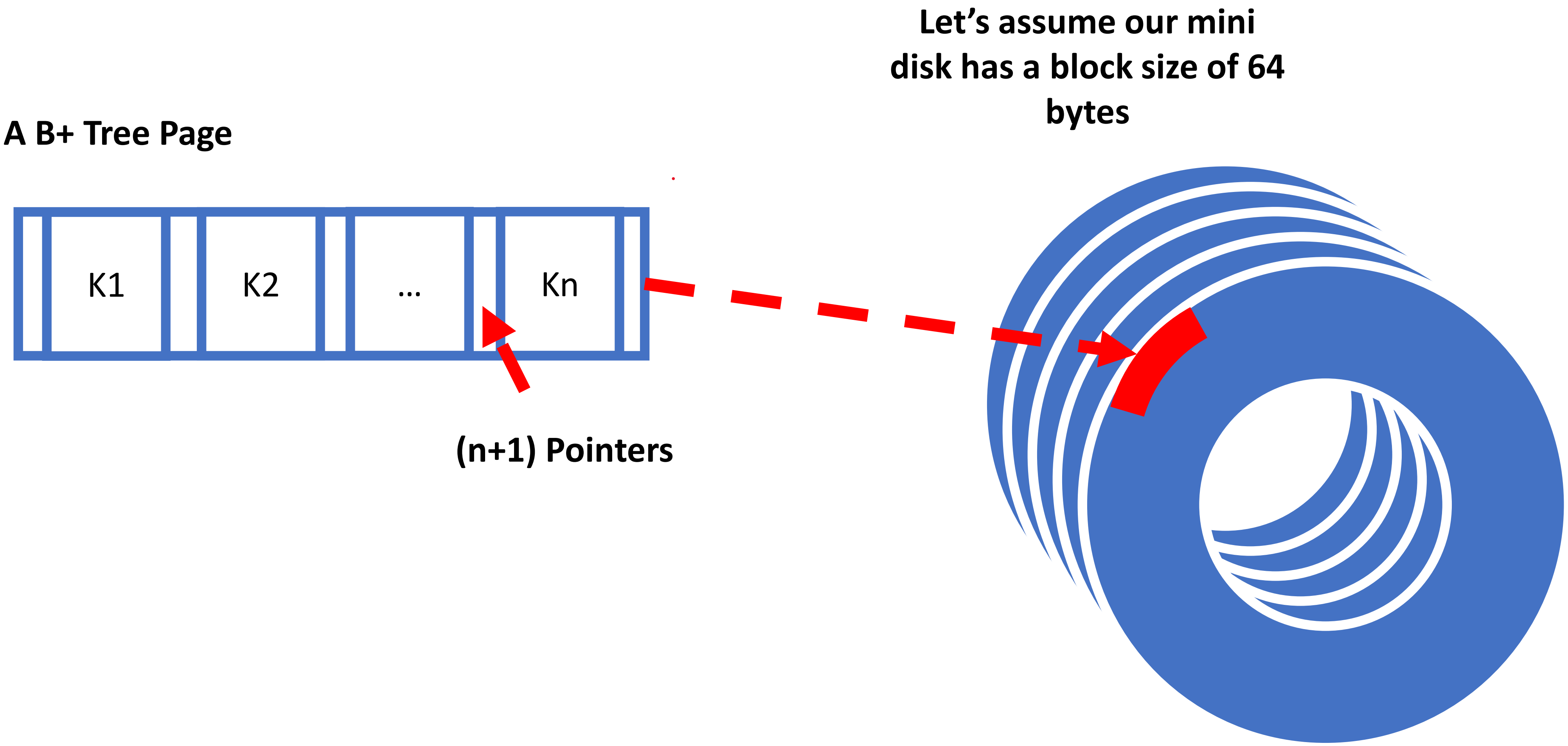
We need to know how many keys (order)…

Let’s assume our mini disk has a block size of 64 bytes

A B+ Tree Page

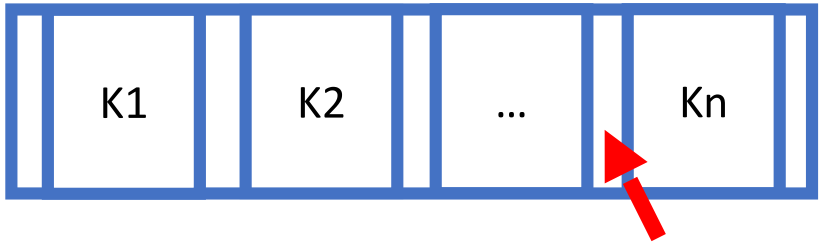


To fit a B+ tree page into a disk block of size 64 bytes



To fit a B+ tree page into a disk block of size 64 bytes

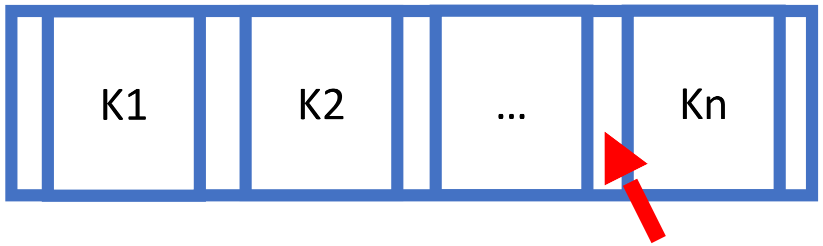
A B+ Tree Page



(n+1) Pointers

To fit a B+ tree page into a disk block of size 64 bytes

A B+ Tree Page

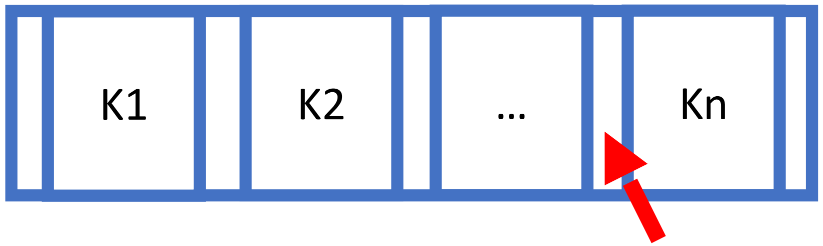
Size() $\leq$ 64

(n+1) Pointers

To fit a B+ tree page into a disk block of size 64 bytes

Assume that: sid/key size = 4 bytes and pointers are of size 8 bytes

A B+ Tree Page

Size(

) $\leq$ 64

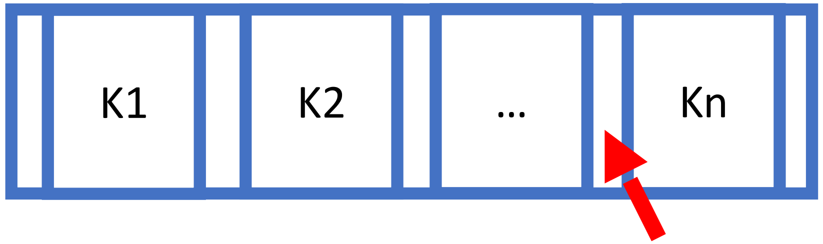
(n+1) Pointers

To fit a B+ tree page into a disk block of size 64 bytes

Assume that: sid/key size = 4 bytes and pointers are of size 8 bytes

A B+ Tree Page

Size(



) $\leq$ 64

(n+1) Pointers

$$4n + 8\,(n+1) \leq 64$$

Solving for n

$$n \leq 5.3$$

Maximum number of keys 2d = 5,

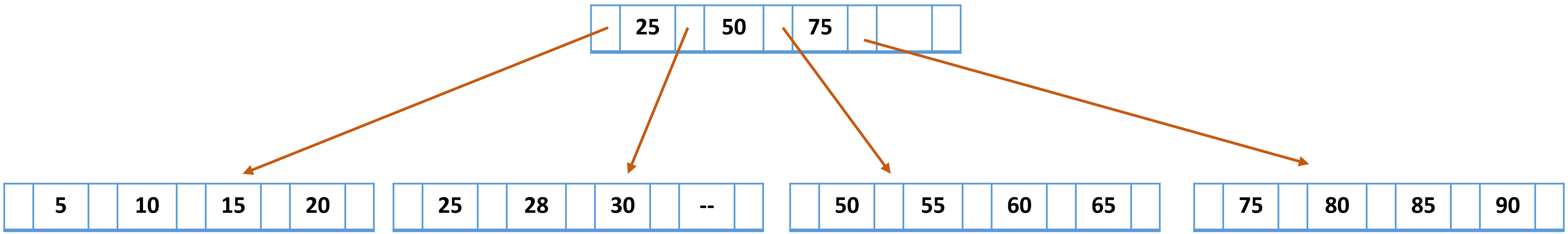
Tree order: d = 2

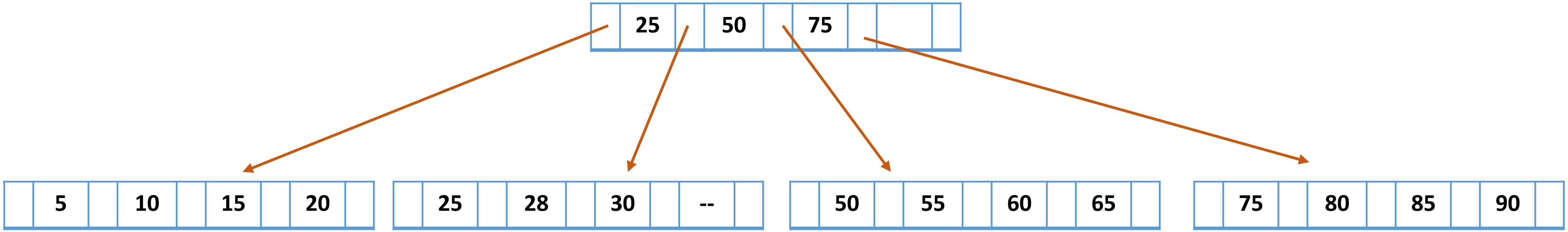
Exercise (2)

Let’s start populating our data in the Student relation

Insert into Student (SID, Name, Major, GPA) values
(5, “”, “”, “”),
(10,””,””,””),
(15,””,””,””),
(20,””,””,””),
(25,””,””,””),
(30,””,””,””),
…
(90,””,””,””),
(28,””,””,””)

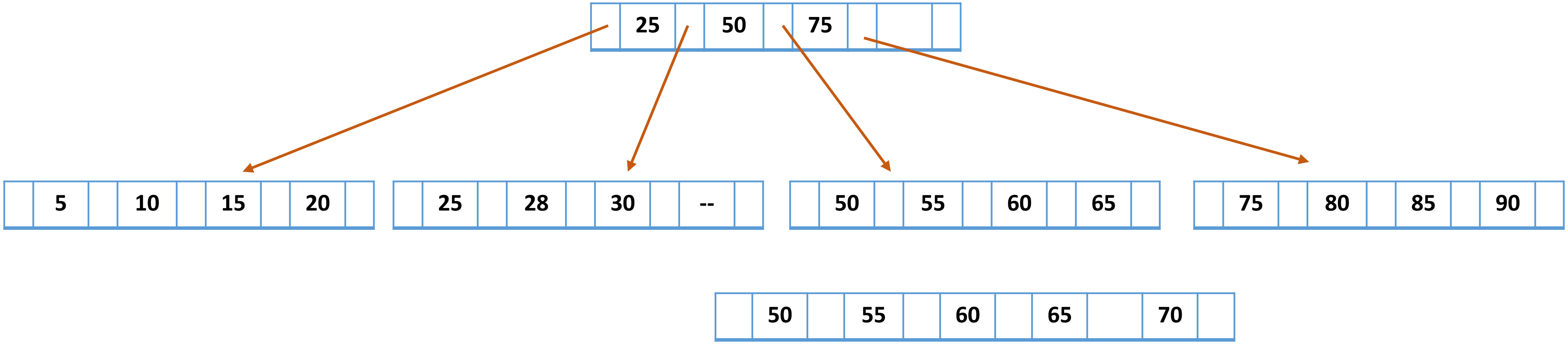
SID	Name	Major	GPA
5	James Smith	Computer Science	2.91
10	Michael Smith	Computer Science	3.22
15	Robert Smith	Biological Sciences	2.59
20	Maria Hernandez	Computer Science	3.00
25	Michael Garcia	Computational Biology	2.54
30	Maria Garcia	Information Systems	4.0
50	..	..	..
55	..	..	..
60	..	..	..
65	..	..	..
75	..	..	..
80	..	..	..
85	..	..	..
90	..	..	..
28	..	..	..


25
50
75
5
10
15
20
25
28
30
--
50
55
60
65
75
80
85
90



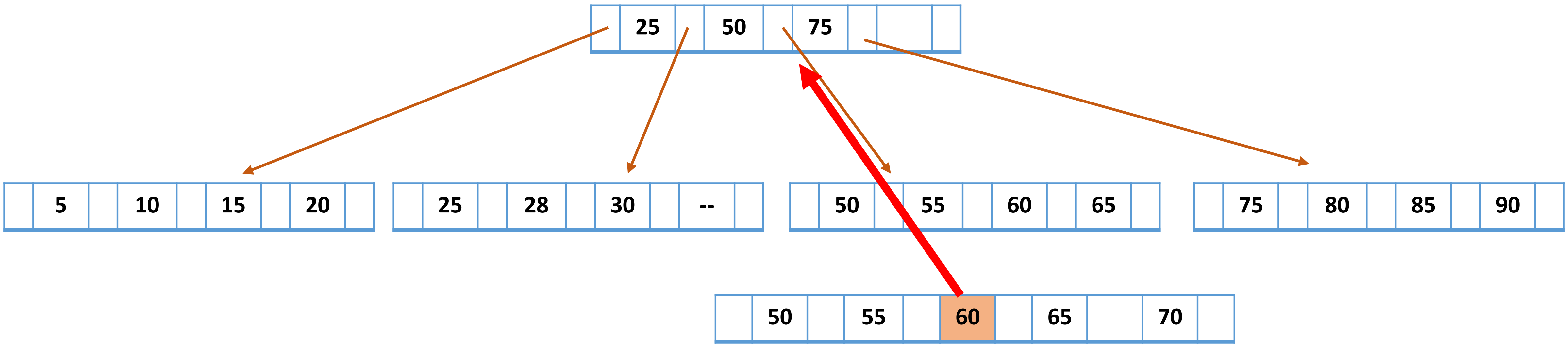
Now insert key 70

The leaf page is full but the index is not



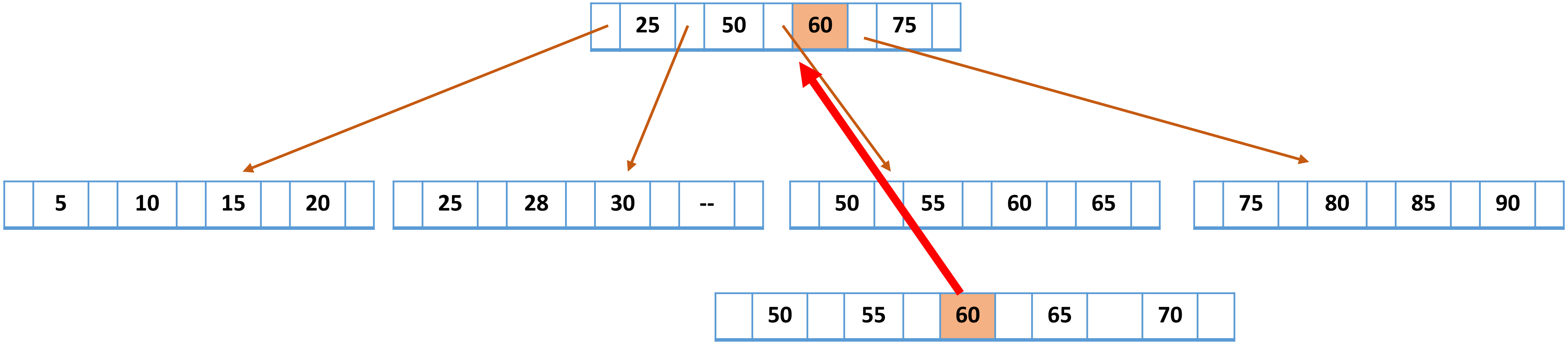
Now insert key 70

The leaf page is full but the index is not



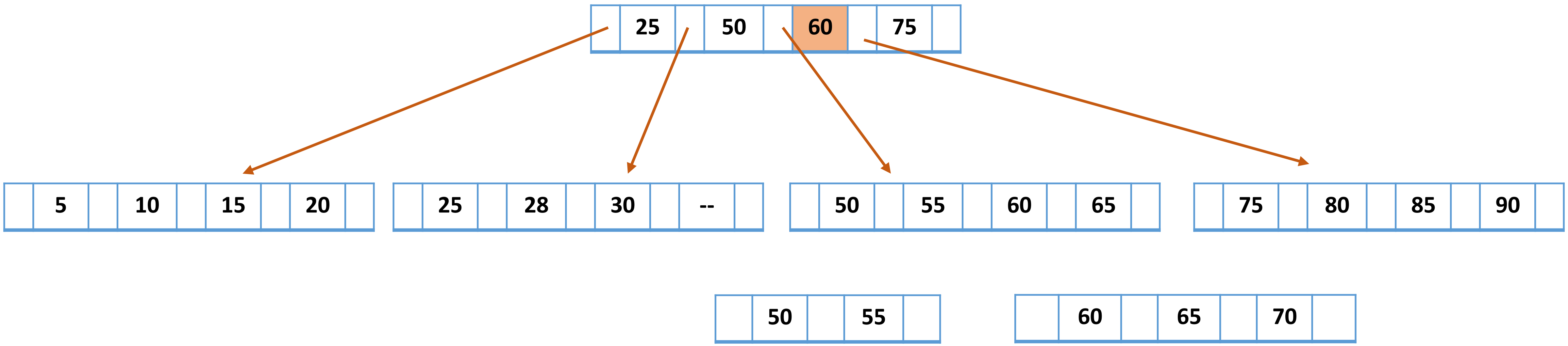
Now insert key 70

The leaf page is full but the index is not



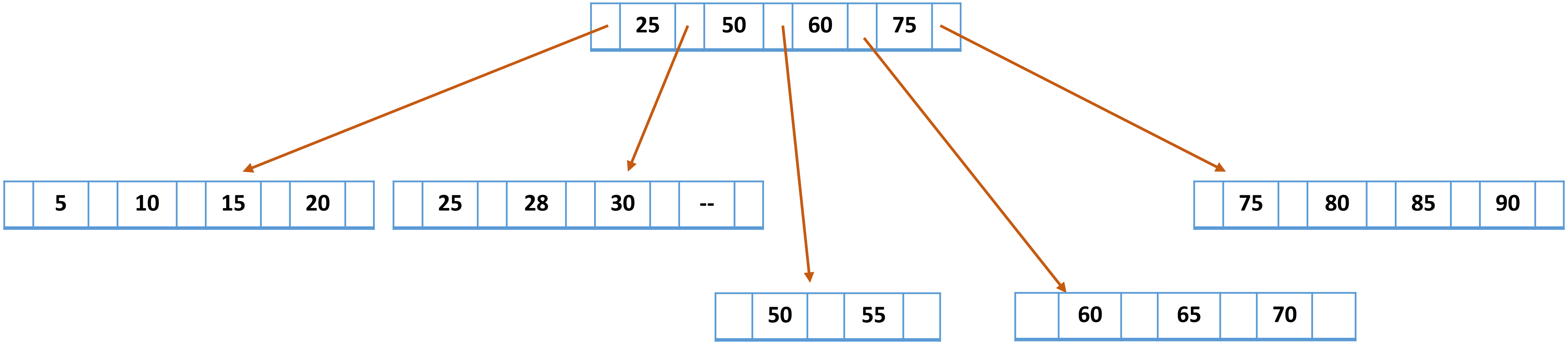
Now insert key 70

The leaf page is full but the index is not



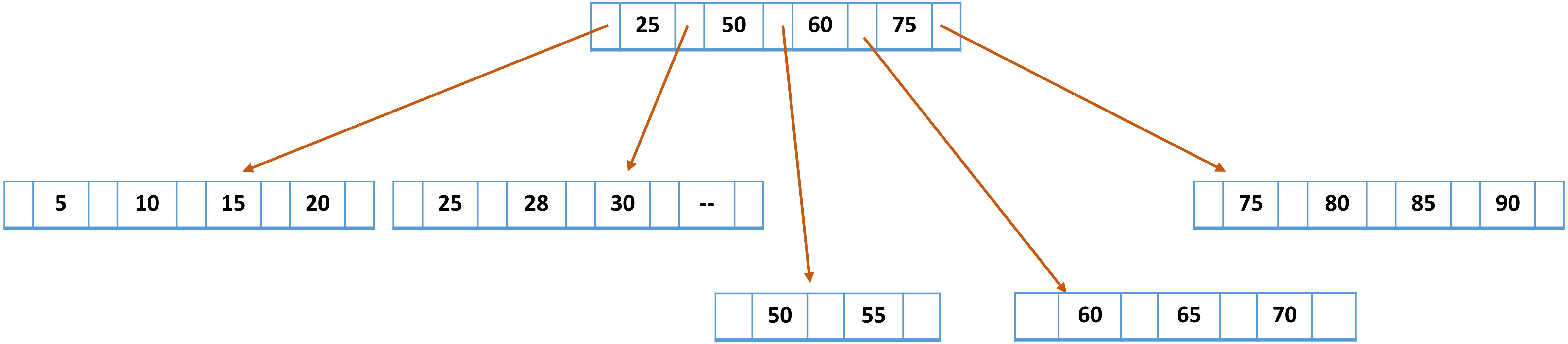
Now insert key 70

The leaf page is full but the index is not



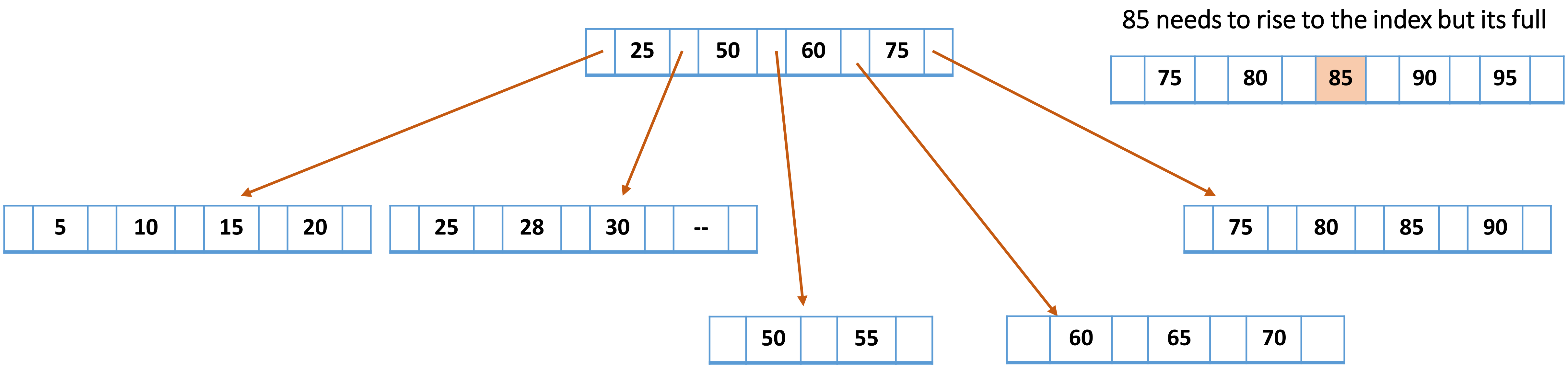
Now insert key 70

The leaf page is full but the index is not



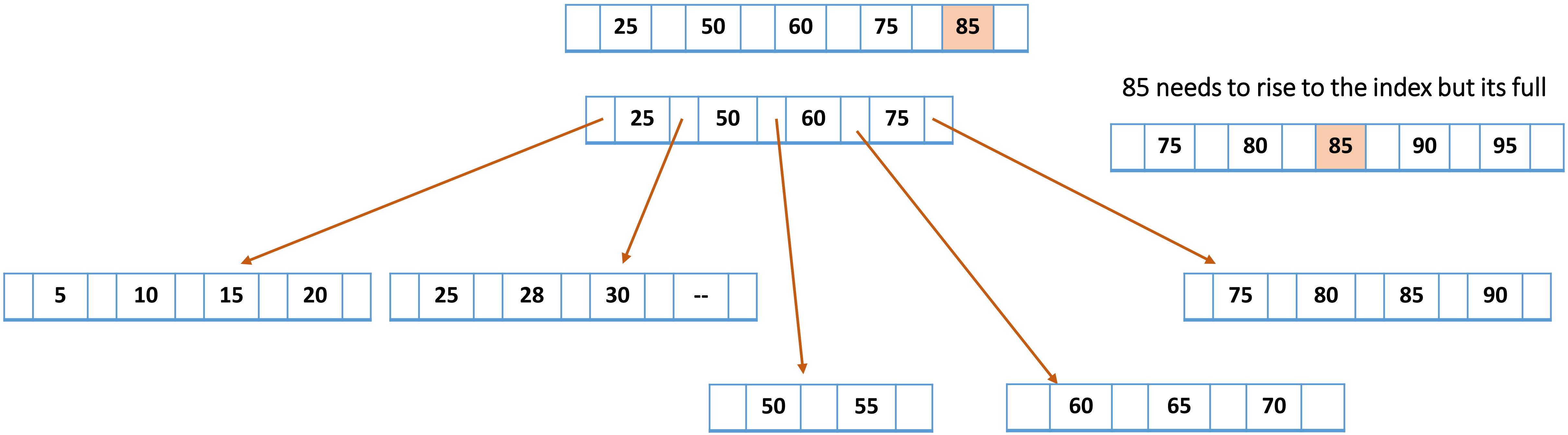
Now insert key 95

The leaf page and the index are full



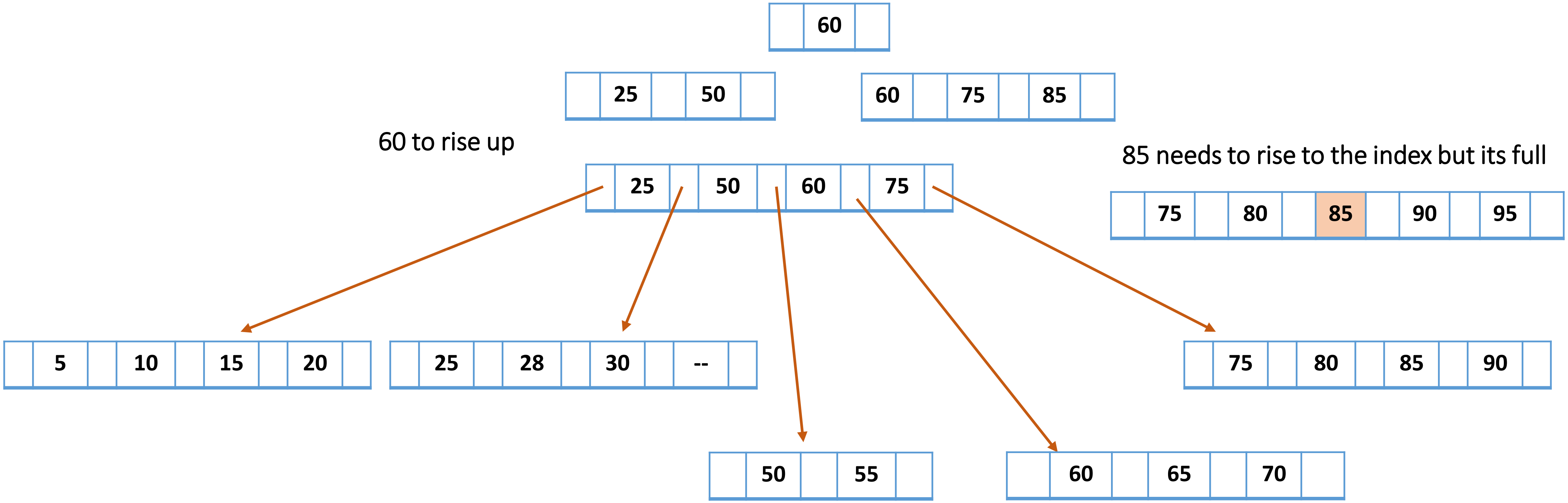
Now insert key 95

The leaf page and the index are full



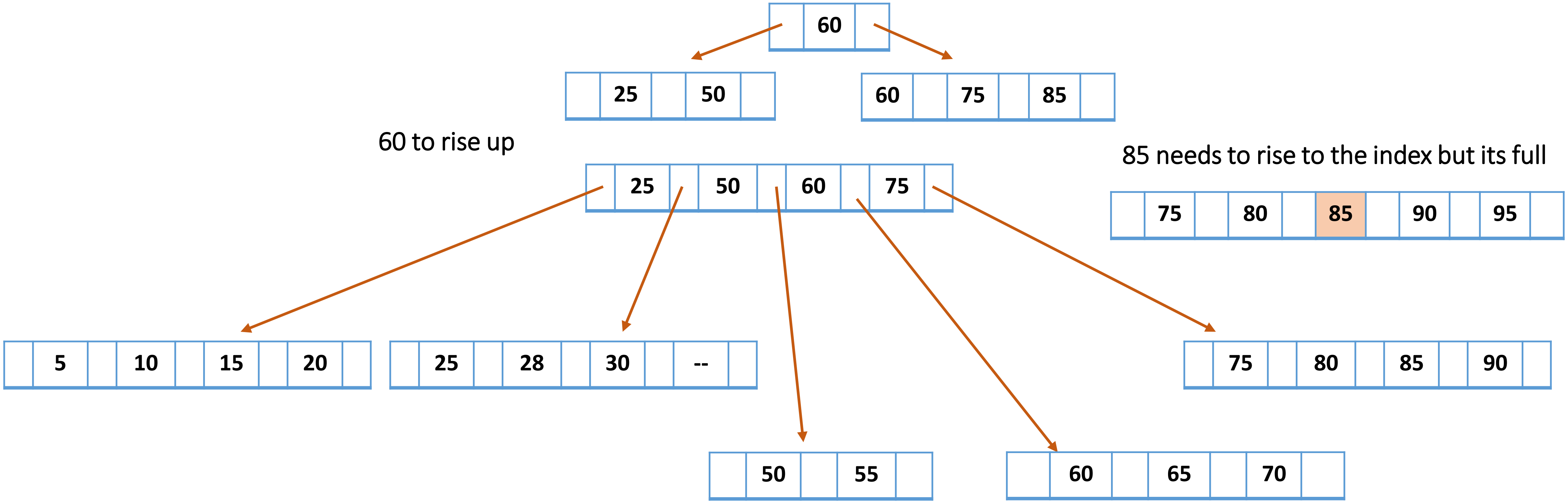
Now insert key 95

The leaf page and the index are full



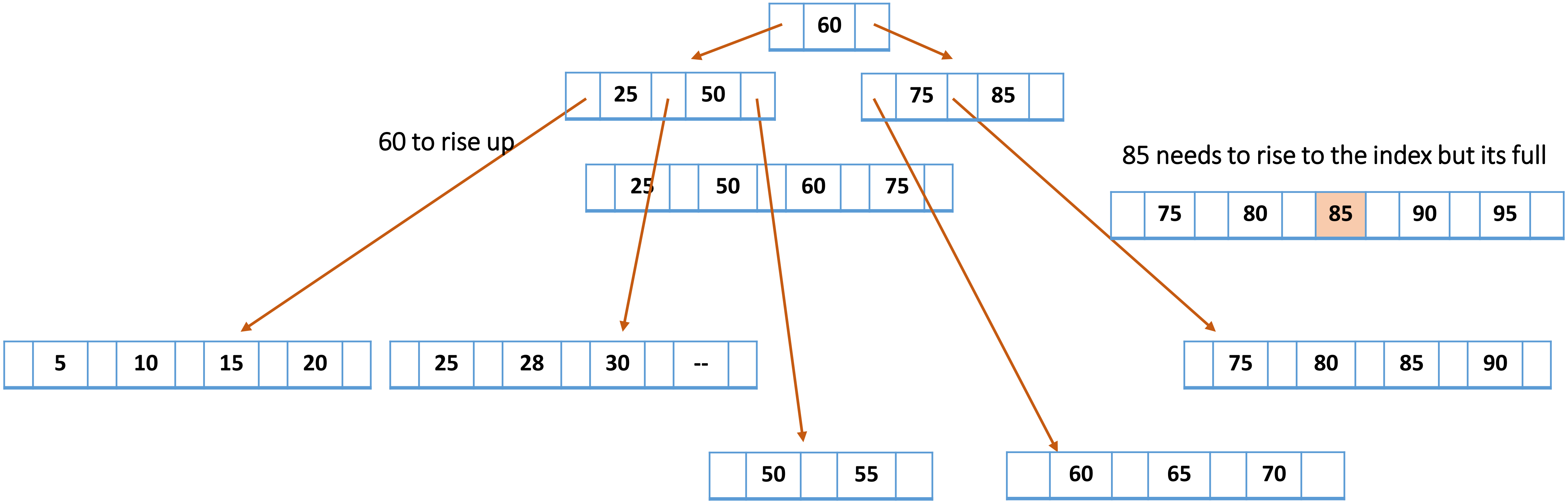
Now insert key 95

The leaf page and the index are full



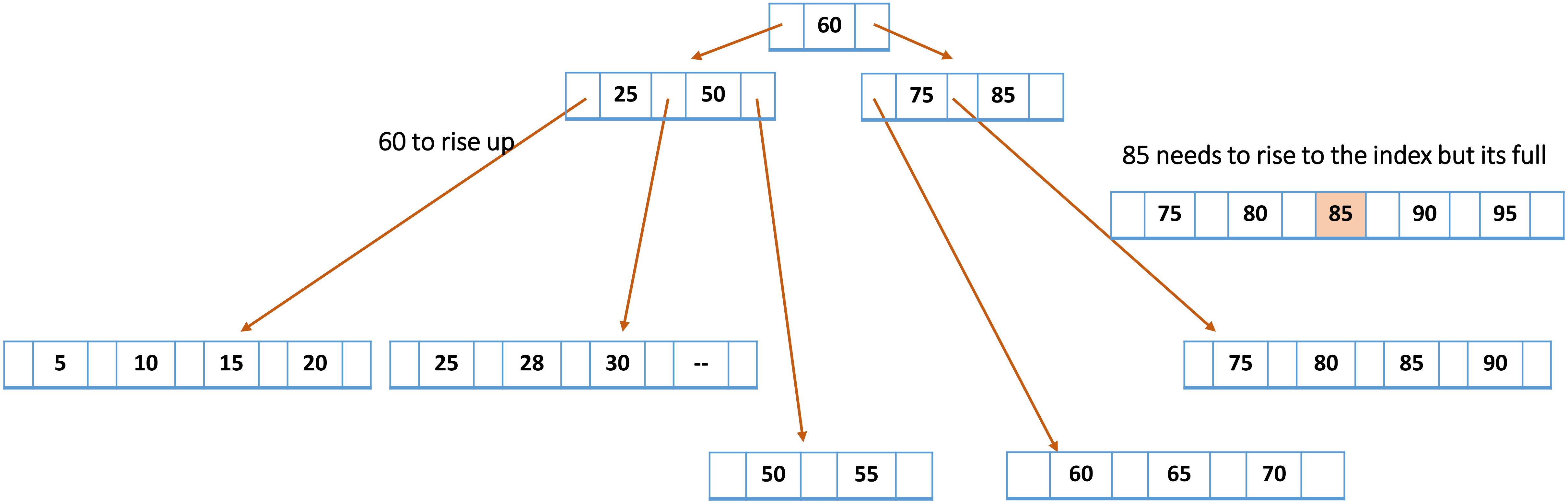
Now insert key 95

The leaf page and the index are full



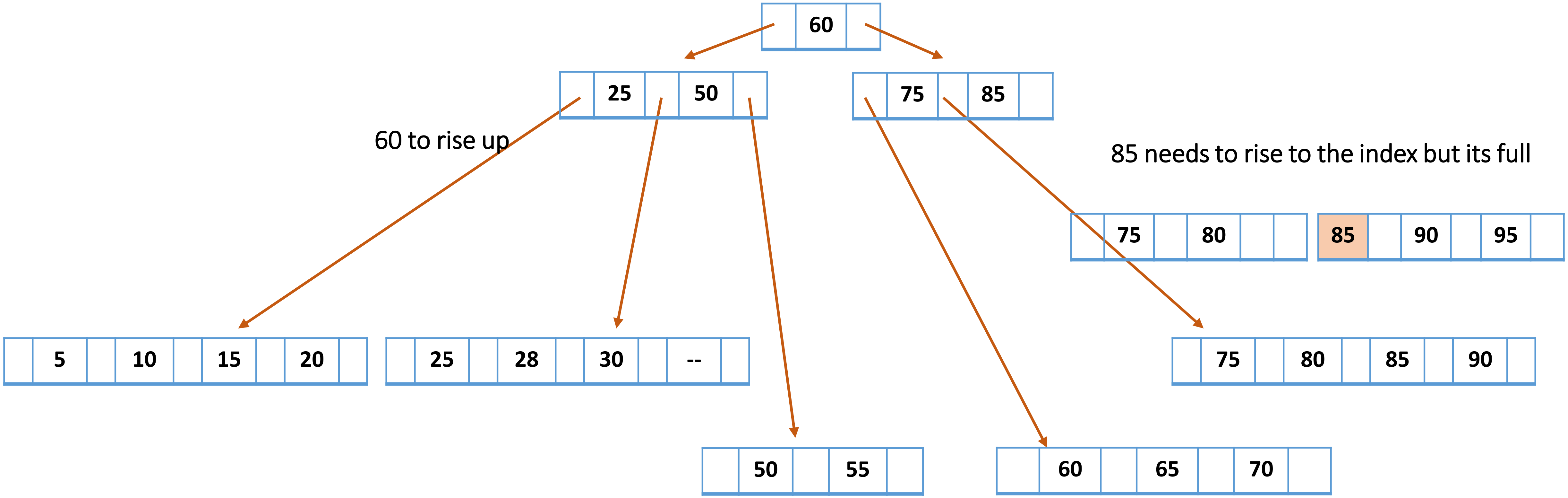
Now insert key 95

The leaf page and the index are full



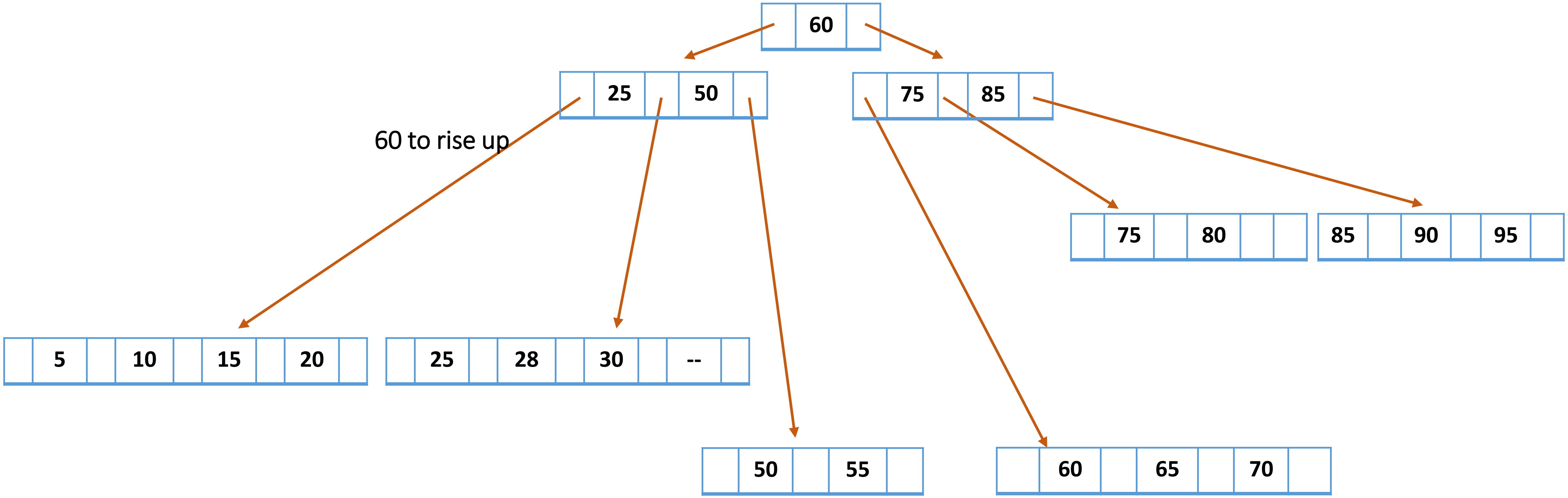
Now insert key 95

The leaf page and the index are full



Now insert key 95

The leaf page and the index are full



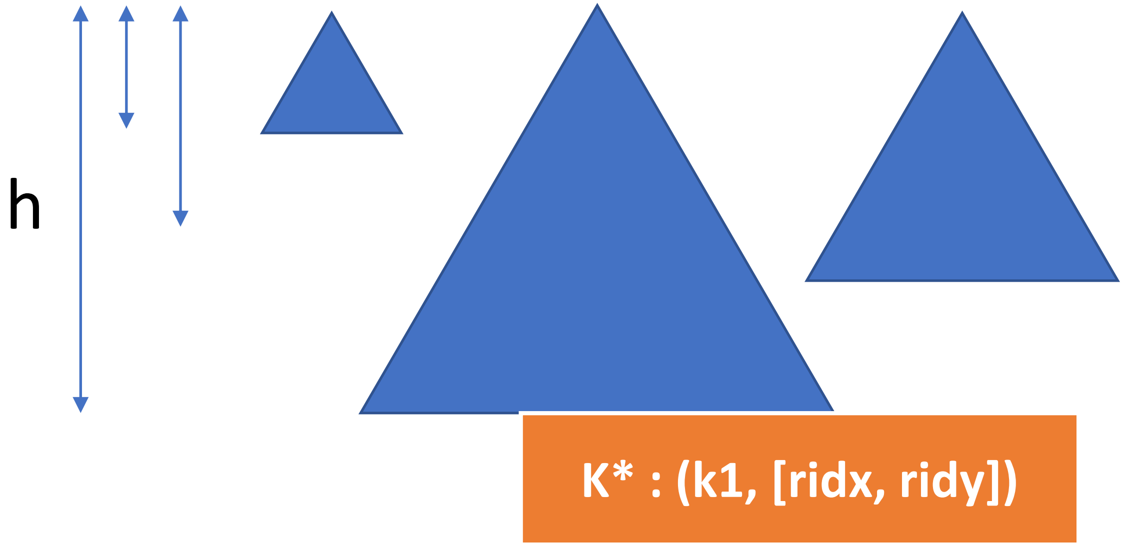
Now insert key 95

The leaf page and the index are full

Exercise (3)

Let’s play with numbers…

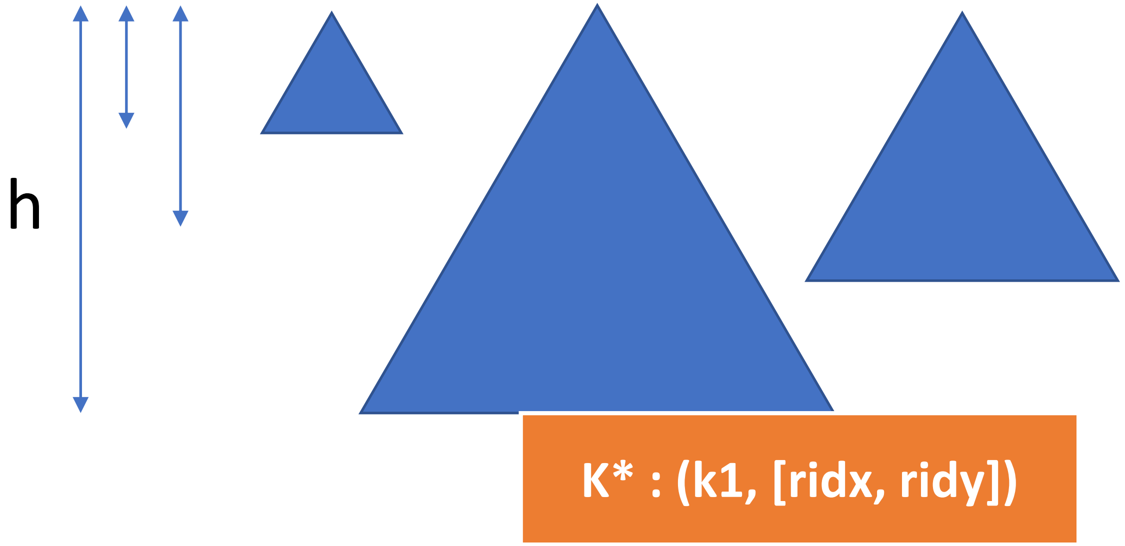
After inserting 6000 records, we are curious to know how high our tree has become!



Alternative 3

SID	Name	Major	GPA
5	James Smith	Computer Science	2.91
10	Michael Smith	Computer Science	3.22
15	Robert Smith	Biological Sciences	2.59
20	Maria Hernandez	Computer Science	3.00
25	Michael Garcia	Computational Biology	2.54
30	Maria Garcia	Information Systems	4.0
		…	
6000	Hammoud	Computer Science	4.0

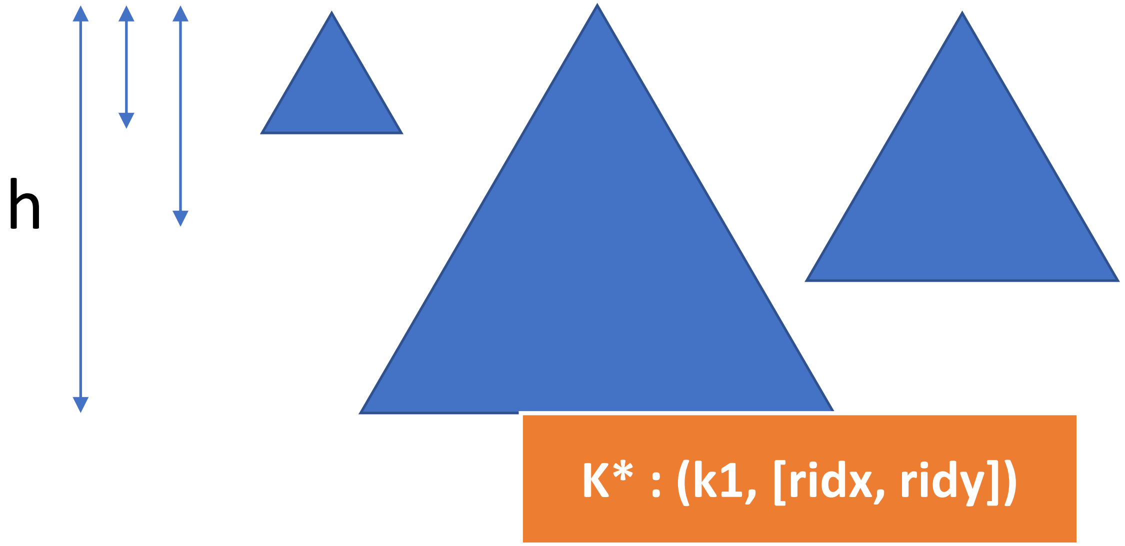
After inserting 6000 records, we are curious to know how high our tree has become!



- **Key size = 4 bytes**
- **Pointer Size = 8 bytes**
- **Disk block = 64 bytes.**
- **Avg(Size(rid-list)) = 2**
- **d = 2**
- **6000 total records**

SID	Name	Major	GPA
5	James Smith	Computer Science	2.91
10	Michael Smith	Computer Science	3.22
15	Robert Smith	Biological Sciences	2.59
20	Maria Hernandez	Computer Science	3.00
25	Michael Garcia	Computational Biology	2.54
30	Maria Garcia	Information Systems	4.0
		…	
6000	Hammoud	Computer Science	4.0

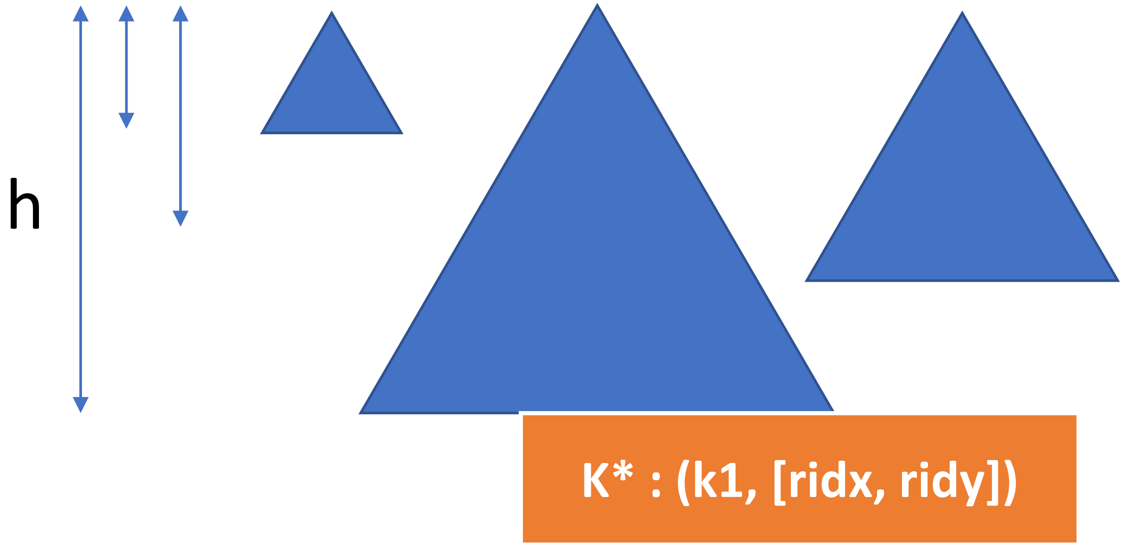
After inserting 6000 records, we are curious to know how high our tree has become!



- **Key size = 4 bytes**
- **Pointer Size = 8 bytes**
- **Disk block = 64 bytes.**
- **Avg(Size(rid-list)) = 2**
- **d = 2**
- **6000 total records**

After inserting 6000 records, we are curious to know how high our tree has become!

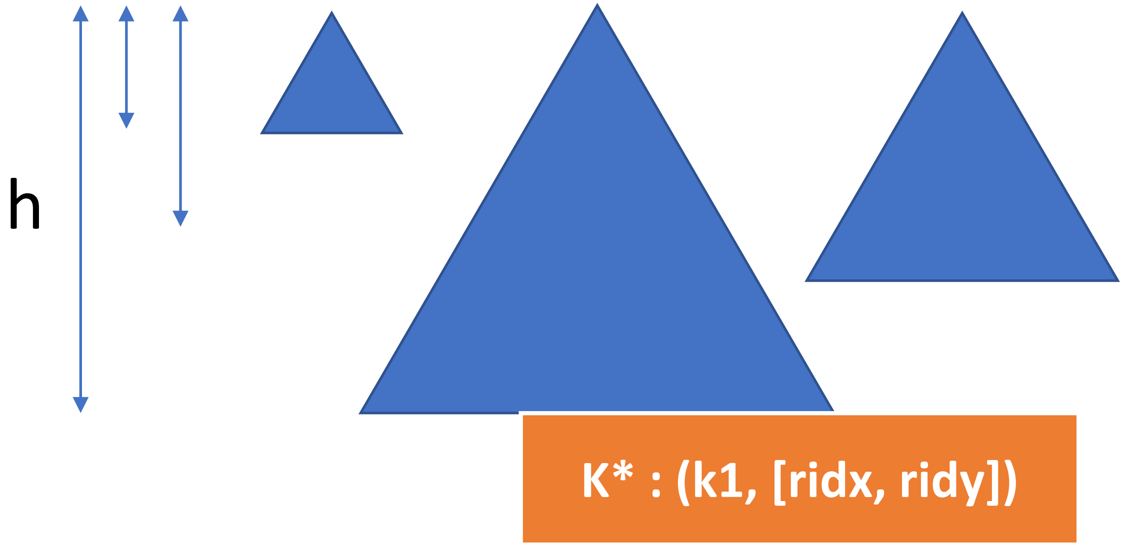
$$\log_{\#pointers}(\#leaves) + 1$$



- **Key size = 4 bytes**
- **Pointer Size = 8 bytes**
- **Disk block = 64 bytes.**
- **Avg(Size(rid-list)) = 2**
- **d = 2**
- **6000 total records**

After inserting 6000 records, we are curious to know how high our tree has become!

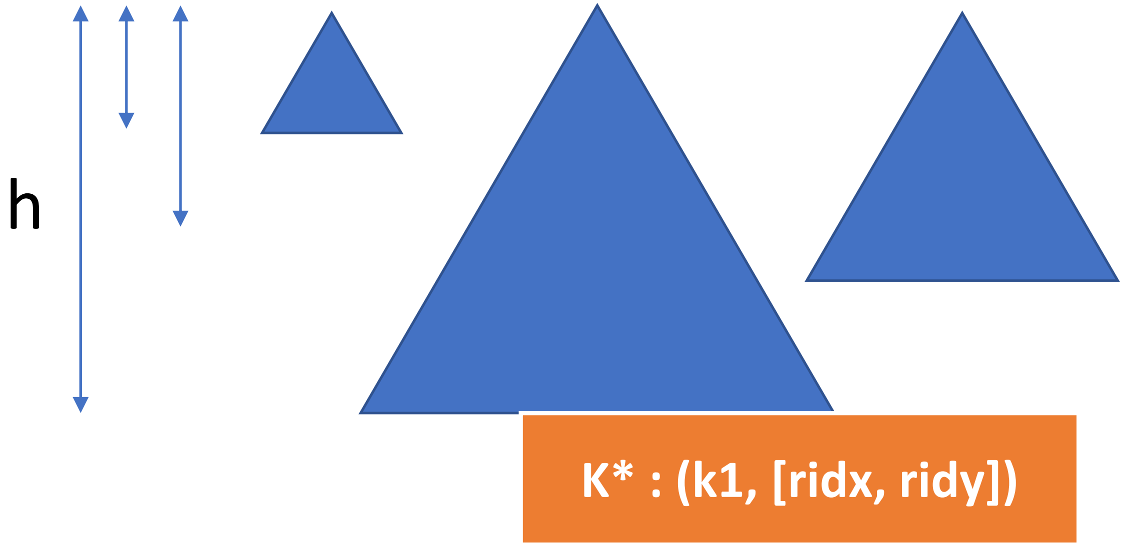
$$\log_5(\#leaves) + 1$$



- **Key size = 4 bytes**
- **Pointer Size = 8 bytes**
- **Disk block = 64 bytes.**
- **Avg(Size(rid-list)) = 2**
- **d = 2**
- **6000 total records**

After inserting 6000 records, we are curious to know how high our tree has become!

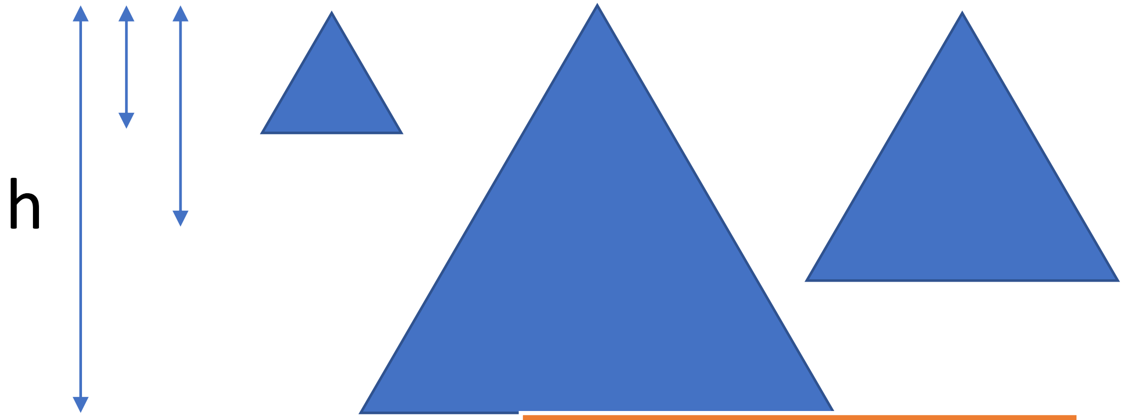
$$\log_5(\#leaves) + 1$$



- **Key size = 4 bytes**
- **Pointer Size = 8 bytes**
- **Disk block = 64 bytes.**
- **Avg(Size(rid-list)) = 2**
- **d = 2**
- **6000 total records**

After inserting 6000 records, we are curious to know how high our tree has become!

$$\log_5(\#leaves) + 1$$



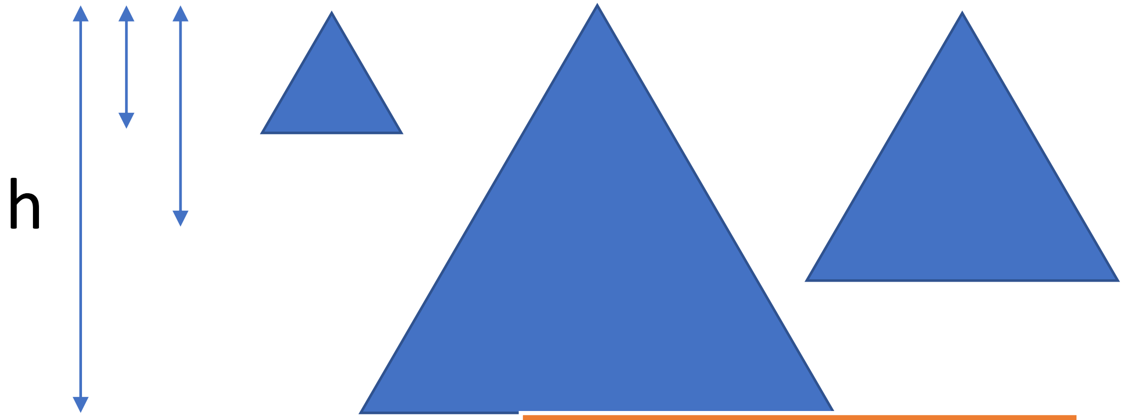
K* : (k1, [ridx, ridy])

- **Key size = 4 bytes**
- **Pointer Size = 8 bytes**
- **Disk block = 64 bytes.**
- **Avg(Size(rid-list)) = 2**
- **d = 2**
- **6000 total records**

Size of K* = 20 BYTES

After inserting 6000 records, we are curious to know how high our tree has become!

$$\log_5(\#leaves) + 1$$



K* : (k1, [ridx, ridy])

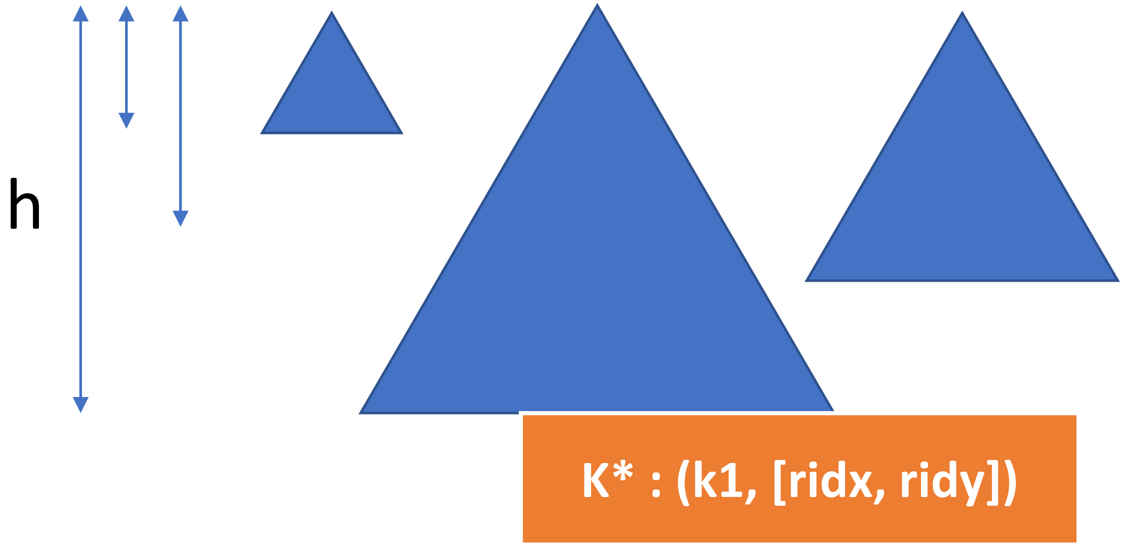
Size of K* = 20 BYTES

How many can we fit in 1 disk block/page?

- **Key size = 4 bytes**
- **Pointer Size = 8 bytes**
- **Disk block = 64 bytes.**
- **Avg(Size(rid-list)) = 2**
- **d = 2**
- **6000 total records**

After inserting 6000 records, we are curious to know how high our tree has become!

$$\log_5(\#leaves) + 1$$



- **Key size = 4 bytes**
- **Pointer Size = 8 bytes**
- **Disk block = 64 bytes.**
- **Avg(Size(rid-list)) = 2**
- **d = 2**
- **6000 total records**

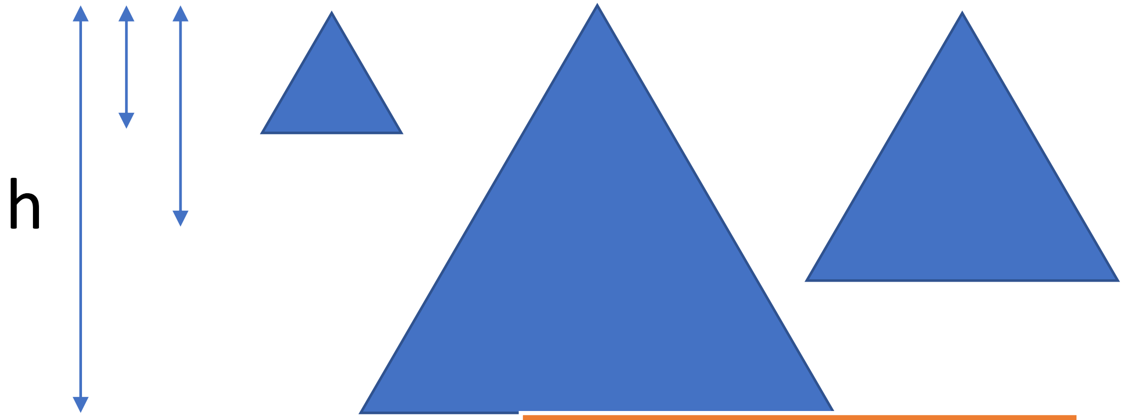
Size of K* = 20 BYTES

How many can we fit in 1 disk block/page?

$$\frac{64}{20} \approx 3\ \text{K*}$$

After inserting 6000 records, we are curious to know how high our tree has become!

$$\log_5(\#leaves) + 1$$



K* : (k1, [ridx, ridy])

- **Key size = 4 bytes**
- **Pointer Size = 8 bytes**
- **Disk block = 64 bytes.**
- **Avg(Size(rid-list)) = 2**
- **d = 2**
- **6000 total records**

Size of K* = 20 BYTES

How many can we fit in 1 disk block/page?

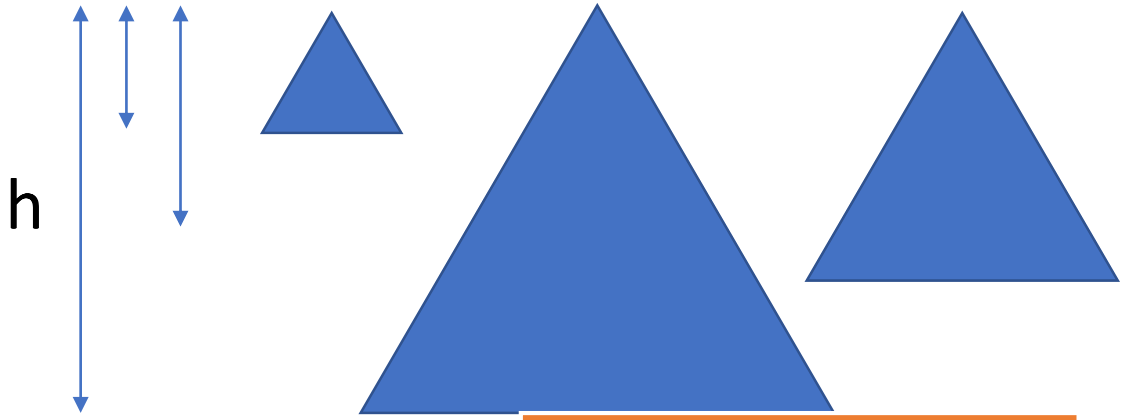
$\frac{64}{20} \approx 3$ K*

Each K* has 2 Records

Total = 3 * 2 = 6 Records in a leaf

After inserting 6000 records, we are curious to know how high our tree has become!

$\log_5(\#leaves) + 1$



K* : (k1, [ridx, ridy])

$$\frac{6000}{6} = 1000$$

Each K* has 2 Records

Total = 3 * 2 = 6 Records in a leaf

- **Key size = 4 bytes**
- **Pointer Size = 8 bytes**
- **Disk block = 64 bytes.**
- **Avg(Size(rid-list)) = 2**
- **d = 2**
- **6000 total records**

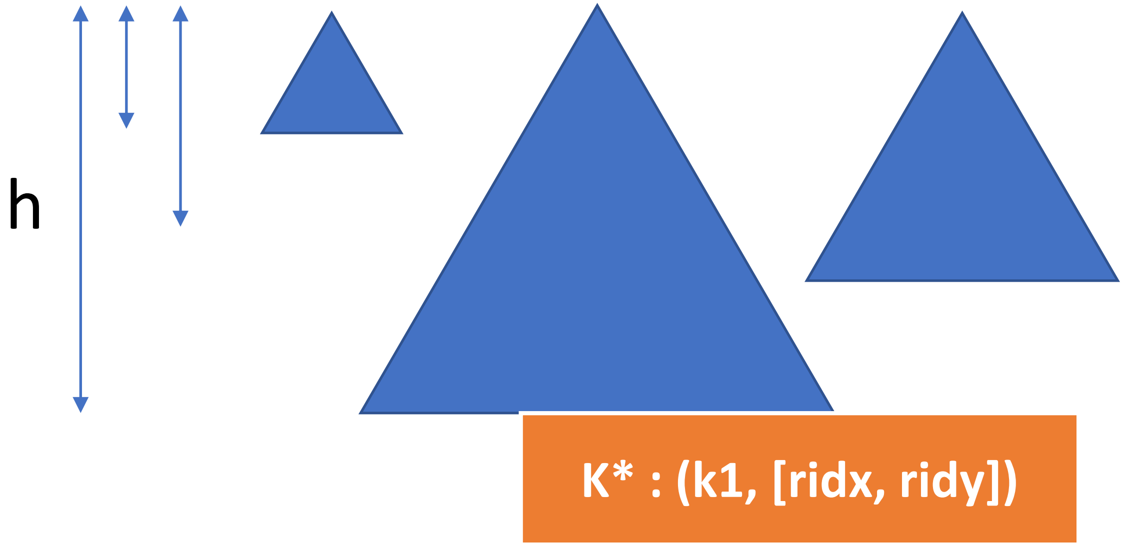
Size of K* = 20 BYTES

How many can we fit in 1 disk block/page?

$\frac{64}{20} \approx 3$ K*

After inserting 6000 records, we are curious to know how high our tree has become!

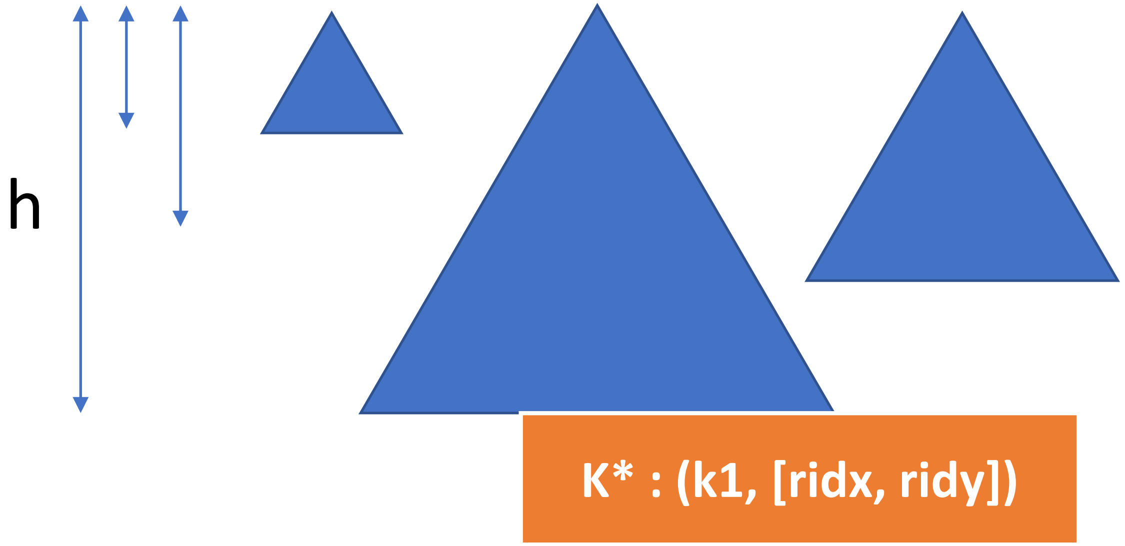
$$\log_5(\#leaves) + 1$$



- Key size = 4 bytes
- Pointer Size = 8 bytes
- Disk block = 64 bytes.
- Avg(Size(rid-list)) = 2
- d = 2
- 6000 total records

After inserting 6000 records, we are curious to know how high our tree has become!

$$\log_5(1000) + 1 = 5.2 \ \approx 6 \ levels$$



- **Key size = 4 bytes**
- **Pointer Size = 8 bytes**
- **Disk block = 64 bytes.**
- **Avg(Size(rid-list)) = 2**
- **d = 2**
- **6000 total records**

Exercise (4)

Happy students! ☺

Happy students! ☺

We decided to increase the GPA of each student by 0.5 for all students with GPA < 4.00. Accordingly, we wrote this query..

```
UPDATE Students SET GPA=GPA+0.5 WHERE GPA < 4.00
```

Happy students! ☺

We decided to increase the GPA of each student by 0.5 for all students with GPA < 4.00. Accordingly, we wrote this query..

```
UPDATE Students SET GPA=GPA+0.5 WHERE GPA < 4.00
```

Oopps!

After running this query, we found that all students ended up with a GPA 4.00.
Why do you think this happened?
What are some possible solutions?