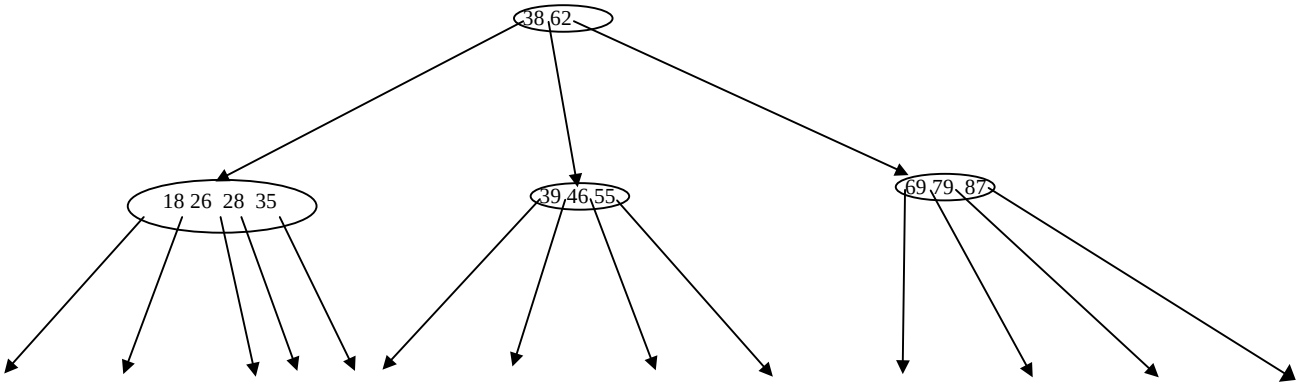
# CIS 671 Introduction to Database Systems II

## Autumn 2001 — Solutions modified 12/5

## Homework 9 — Due: Friday, November 30, start of class

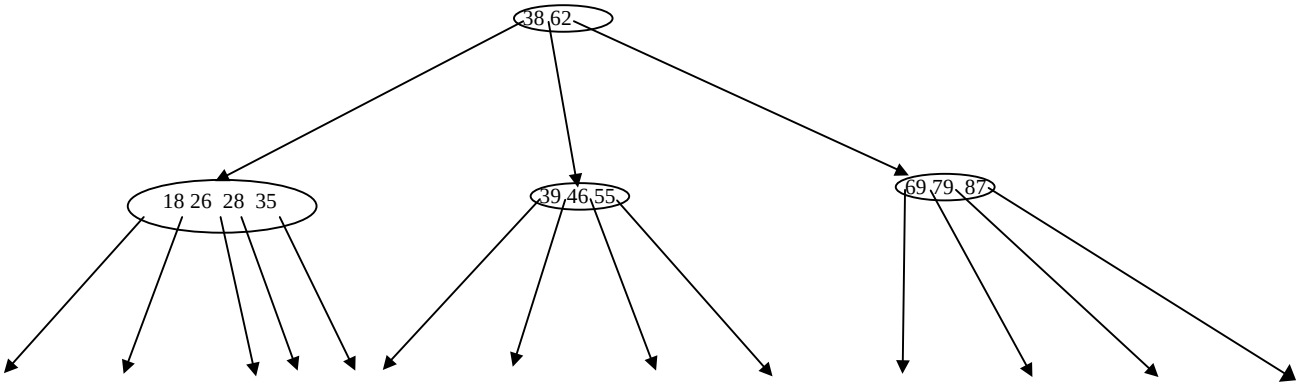
1. The following problems are about a B-tree of order 5. Also assume the *left* subtree is chosen for any change whenever there could be a choice between the left and right subtrees.

   a) Show the B-tree that results from inserting a record with key 57.



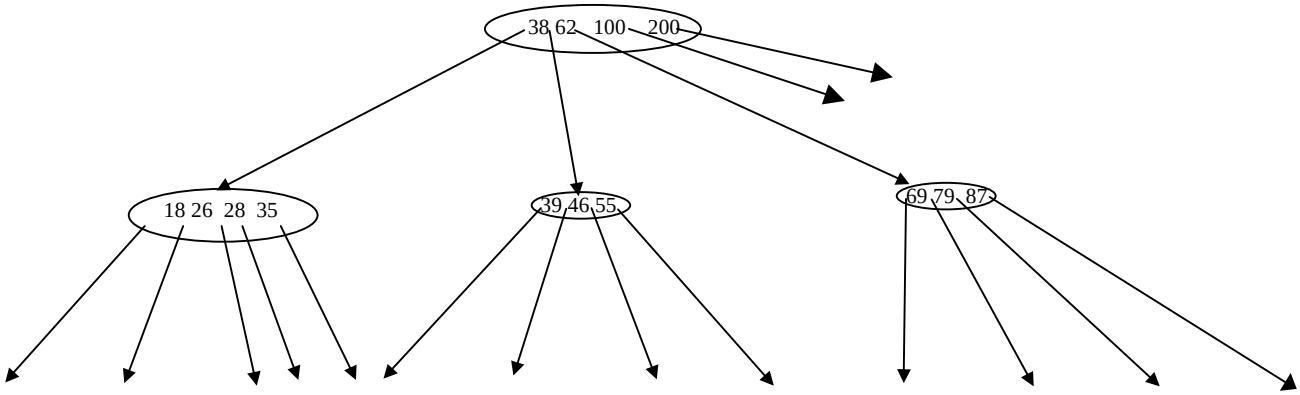

Just add to node with [39 46 55].

   b) Show the B-tree that results from inserting a record with key 37.



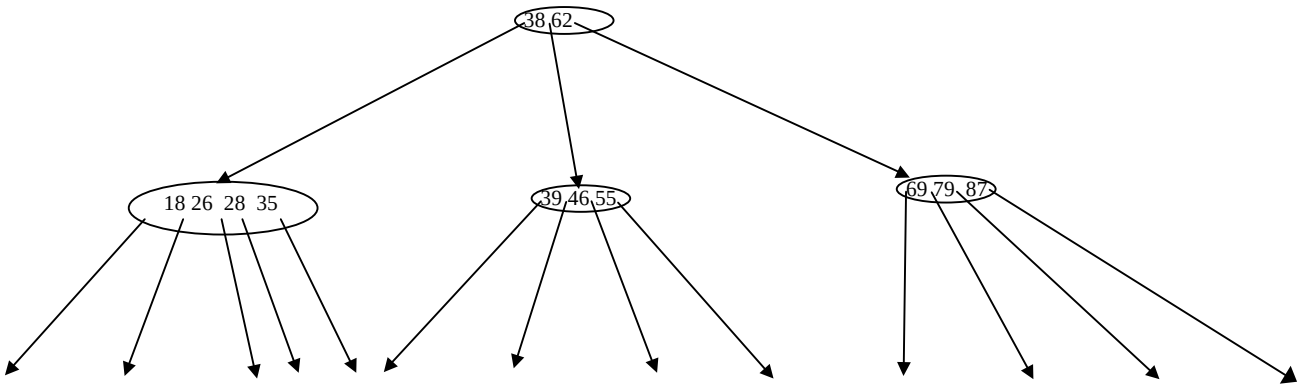

```
Insert into node with [18 26 28 35] yielding [18 26 28 35 37].
Too full so split into [18 26] and [35 37], promoting 28.
```

c) Show the B-tree that results from inserting a record with key 37.



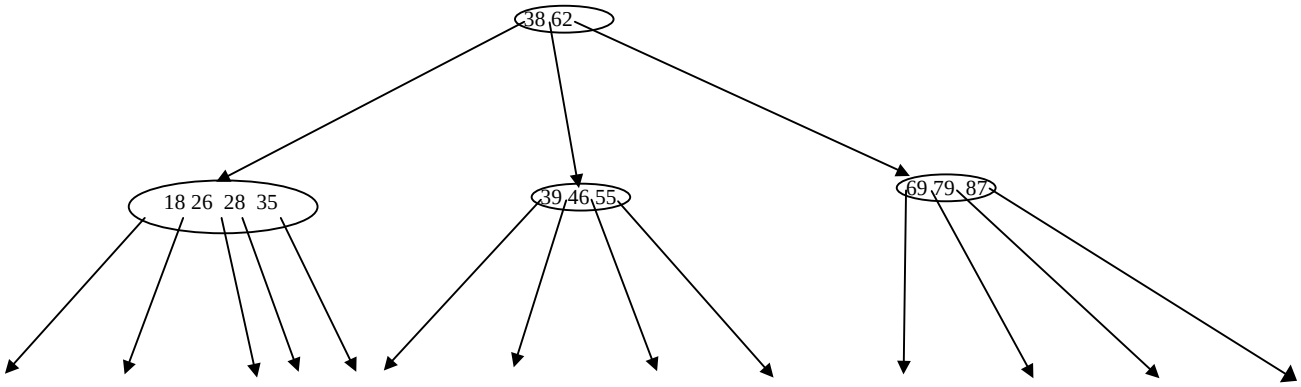

```
Insert into node with [18 26 28 35] yielding [18 26 28 35 37].
Too full so split into [18 26] and [35 37], promoting 28.
Insert 28 into node with [38 62 100 200] yielding [28 38 62 100 200].
Too full so split [28 38 62 100 200] into [28 38] and [100 200],
      making [62] a new root node.
```

d) Show the B-tree that results from deleting a record with key 55.



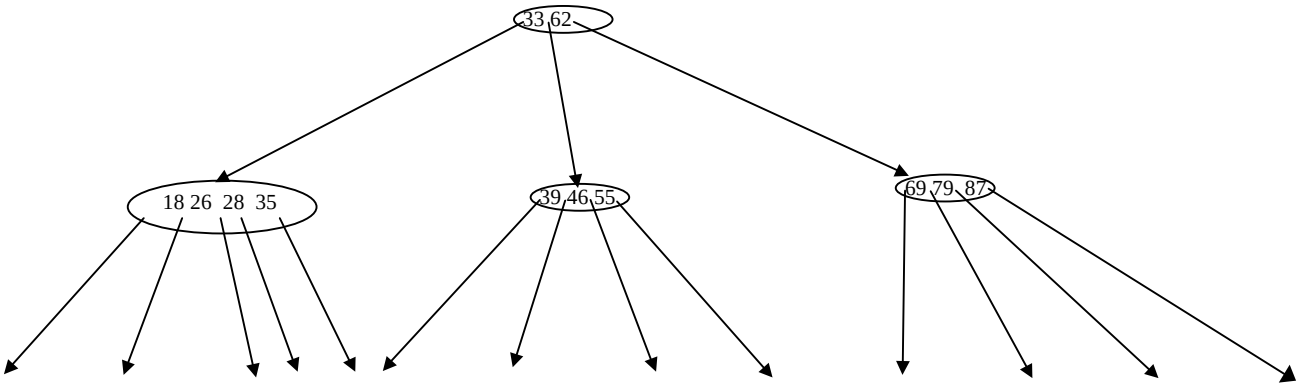

```
Just delete 55 from node with [39 46 55].
```

e) Show the B-tree that results from deleting records with keys 46 and 55.




```
Delete 46 & 55 from node with [39 46 55], yielding [39].
Not full enough.  Left sibling is full enough,
      so rotate giving [18 26 28] on left, [35 62] above and [38 39] for [39].
```

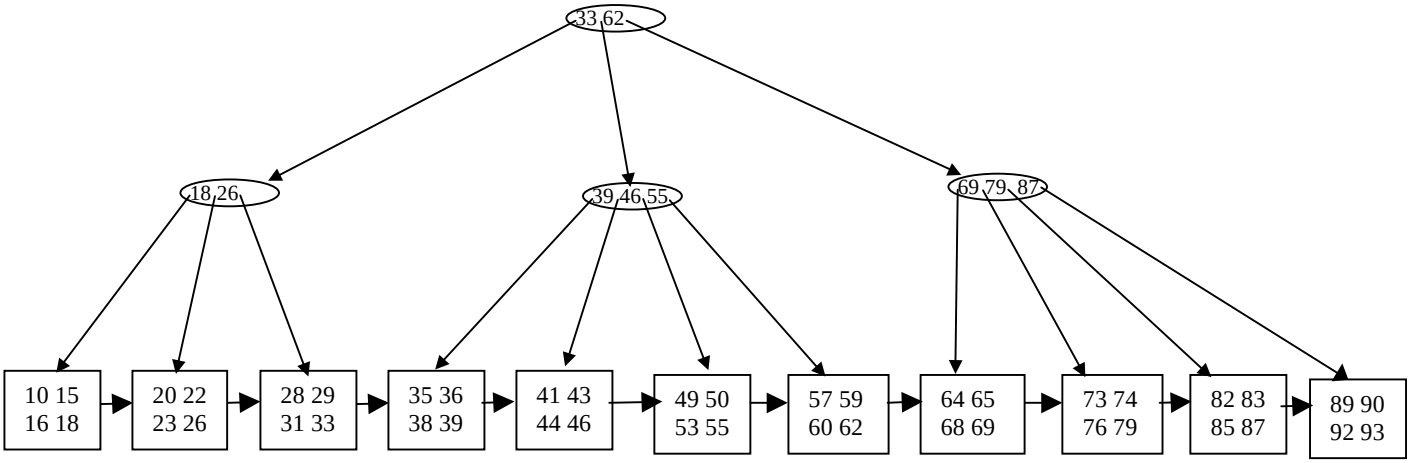
f) Show the B-tree that results from deleting records with keys 55, 79 and 87.




```
[39 46 55] becomes [39 46].
[69 79 87] becomes [69], not full enough.
[39 46] not full enough, so cannot rotate.
Combine [39 46] and [69] yielding  [39 46 62 69].
Delete 62 from root, yielding [33].
```

2. The following problems are about $B^+$-trees. Assume the index is a B-tree of order 4. Assume each leaf node holds at most 4 keys and record pointers. Also assume the *left* subtree is chosen for any change whenever there could be a choice between the left and right subtrees. Finally assume a leaf node splits to have 3 keys in the leftmost node and 2 keys in the rightmost node.

   a) Show the $B^+$-tree that results from inserting a record with key 37.




Revised Solution:

Inserting 37 gives leaf node of 35 36 37 38 39. Must split.
Split leaf node into 35 36 37 and 38 39.
Insert 37 into parent node 39 46 55.
Must split giving (37 39) and (55), promoting 46.
Insert 46 into root node.

Result:
Root node 37 46 62
Left node 18 26
Second node 37 39
Third node 55
Fourth node 69 79 87
Leaf nodes 35, 36 37 and 38, 39

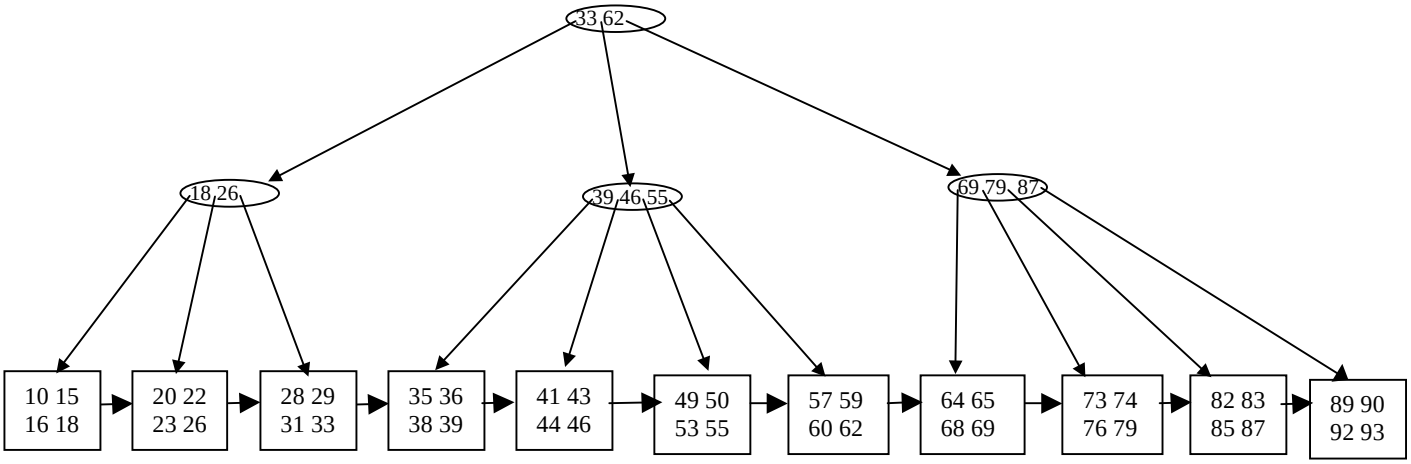
Assume a nonleaf node splits 2 and 1.

Original Solution – actually $B^*$-tree:

Inserting 37 gives leaf node of 35 36 37 38 39. Must split.
Split leaf node into 35 36 37 and 38 39.
Insert 37 into parent node 39 46 55.
Must rotate from middle node to left node.

Result:
Root node 37 62
Left node 18 26 33
Middle node 39 46 55
Leaf nodes 35, 36 37 and 38, 39

b) Show the $B^+$-tree that results from inserting a record with key 84.




Revised Solution:

Inserting 84 gives leaf node of 82 83 84 85 87. Must split.
Insert 84 into parent 69 79 87 giving 69 79 84 87. Too full. Thus split.
Split right node giving (69 79) and (87), promoting 84.
Root node becomes 33 62 84.

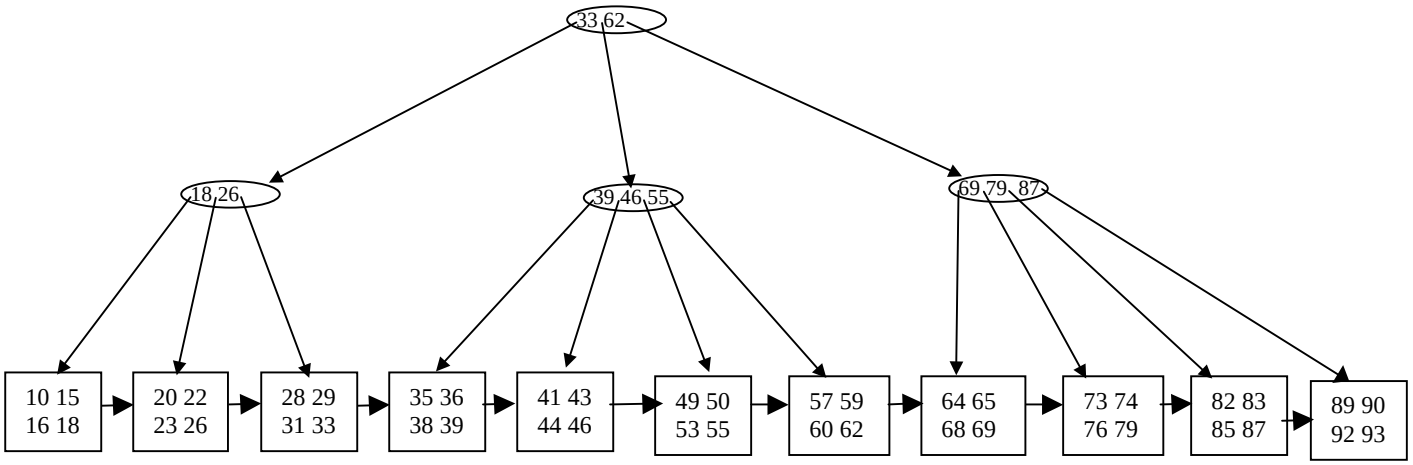
Root node 33 55 79
Left node 18 26
2nd node 39 46 55
3rd node 69 79
4th node 87
New/revised leaf nodes 82, 83, 84 & 85, 87

Original Solution – actually $B^*$-tree:

Inserting 84 gives block 82 83 84 85 87.
Must split.
Insert 84 into parent 69 79 87 giving 69 79 84 87. Too full.
Can't split. Must split right two B*-tree nodes.

Root node33 55 79
Left node 18 26
2nd node 39 46
3rd node 62 69
4th node 84 87
New/revised data blocks 82, 83, 84 & 85, 87

c) Show the $B^+$-tree that results from deleting the node with 28, 29, 31, 33.




```
Delete data block
Delete 33 from B*-tree
Move 26 up to root and delete 26
from (18 26)
Rotate 39 from (39 46 55)

Root node 39 62
Left node 18 26
2nd node 46 55
```

3. Assume you have a disk drive with the following characteristics:

| | |
|---|---|
| Bytes per sector | 512 |
| Sectors per track | 60 |
| Sectors per cluster | 4 |
| Tracks per cylinder | 16 |
| Cylinders | 4000 |
| Average seek time | 10 ms |
| Average rotational delay | 6 ms |
| Transfer rate | 3000 bytes/ms |

You have been asked to set up a $B^+$-tree file of 300,000 records, each of length 400 bytes using leaf nodes that hold 5 keys.

a) How much space, in cylinders, will the data blocks take? Explain your answer.

Since data may not span clusters need to determine records/cluster:

Floor( [512 Bytes/sector * 4 sectors/cluster] / [400 Bytes/record]) = Floor( 5.12)
= 5 records/cluster

$$\text{Cylinders/file} = \frac{[300{,}000 \text{ records/file}] * [4 \text{ sectors/cluster}]}{[5 \text{ records/cluster}] * 60 \text{ sectors/track}] * [16 \text{ tracks/cylinder}]}$$

= 250 cylinders

b) Assume a $B^+$-tree using a B-tree of order 50 for the index. How many levels will be needed in the index? Give both the best and worst cases. Explain your answers.

Number of data blocks and hence number of keys in B+-tree leaf nodes:
[300,000 records/file] / [5 records/cluster * 1 cluster/block] = 60,000 blocks/file

Min 3 and Max 5 keys per leaf node. Number of leaf nodes: Min = 60,000/5 =12,000 Max = 60,000/3 = 20,000.

Order 50 B-tree: Min subtrees except at root is 25.

| Level | Max | Min |
|---|---|---|
| 1 | 50 | 2 |
| 2 | 50**2 = 2,500 | 2 * 25 = 50 |
| ***3*** | 50**3 = ***125,000*** | 2 * 25**2 = ***1,250*** |
| 4 | - | 2 * 25**3 = 31,250 |

Min height is ***3*** since 2,500 < 20,000 < 125,000. Max height is ***3*** since 1,250 < 12,000 < 31,250.

c) How long does it take to retrieve a random data record? Give both the best and worse case.

Transfer time one block: [512 Bytes/sector * 4 sectors/cluster] = .68 ms
3,000 Bytes/ms

Time per retrieval seek + latency + transfer = 10 + 6 + .68 = 16.68 ms = 16 ms ignoring transfer time.
To retrieve a data record we must retrieve 3 internal nodes and one leaf node in the $B^+$-tree. Then we must retrieve the data block (cluster), i.e. there are 5 reads. Total time = 5 reads * 16 ms/read = ***80 ms***.
Note that we assume each read requires a seek. Assuming there are multiple users, that is likely since there will be "interference" among the reads of the various users.