Database Technology

Tutorial I B-Trees

Alessandro Artale

Department of Computation – UMIST

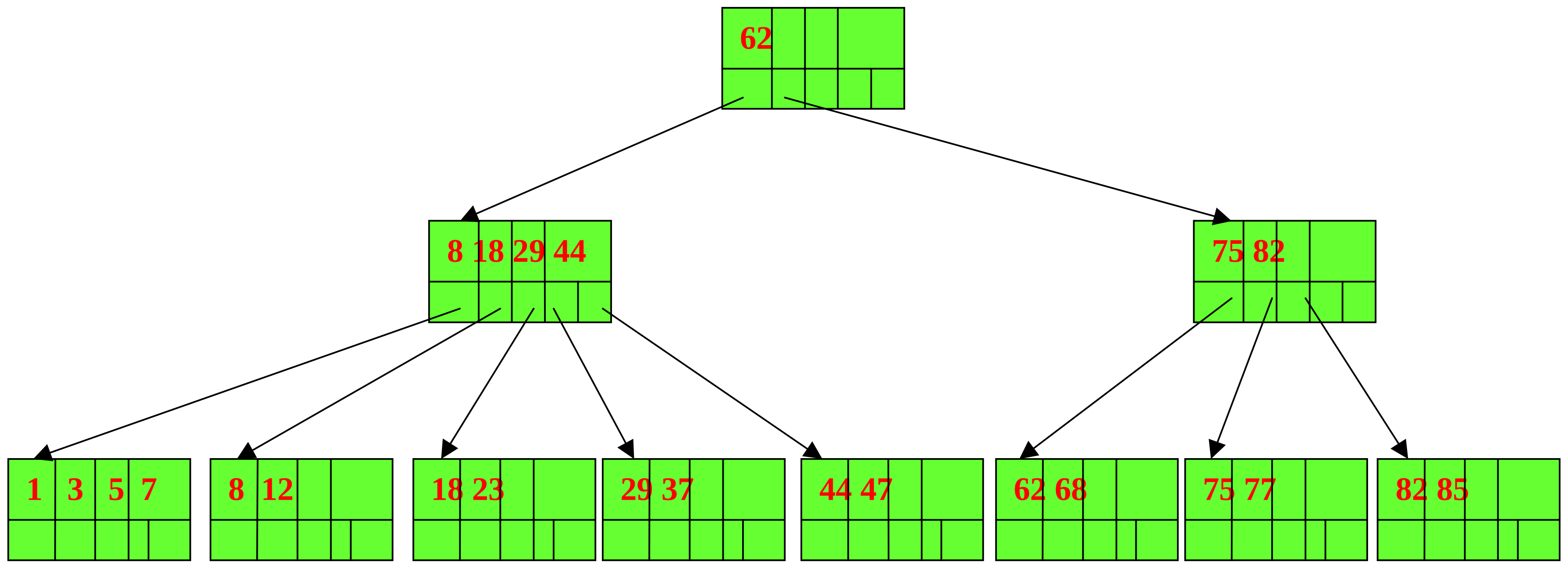
Room: MSS-E18

artale@co.umist.ac.uk

http://www.co.umist.ac.uk/∼artale/

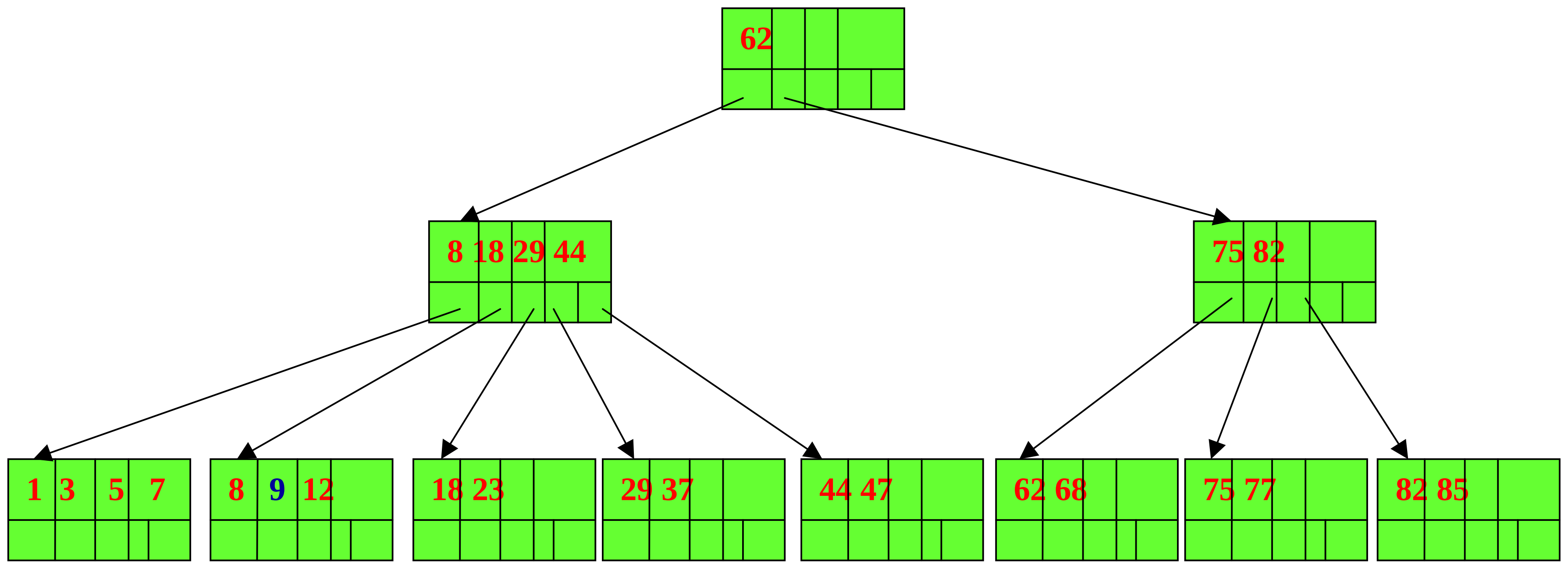
2001/2002 – Second semester

EXERCISE 1. Given the following B-Tree:



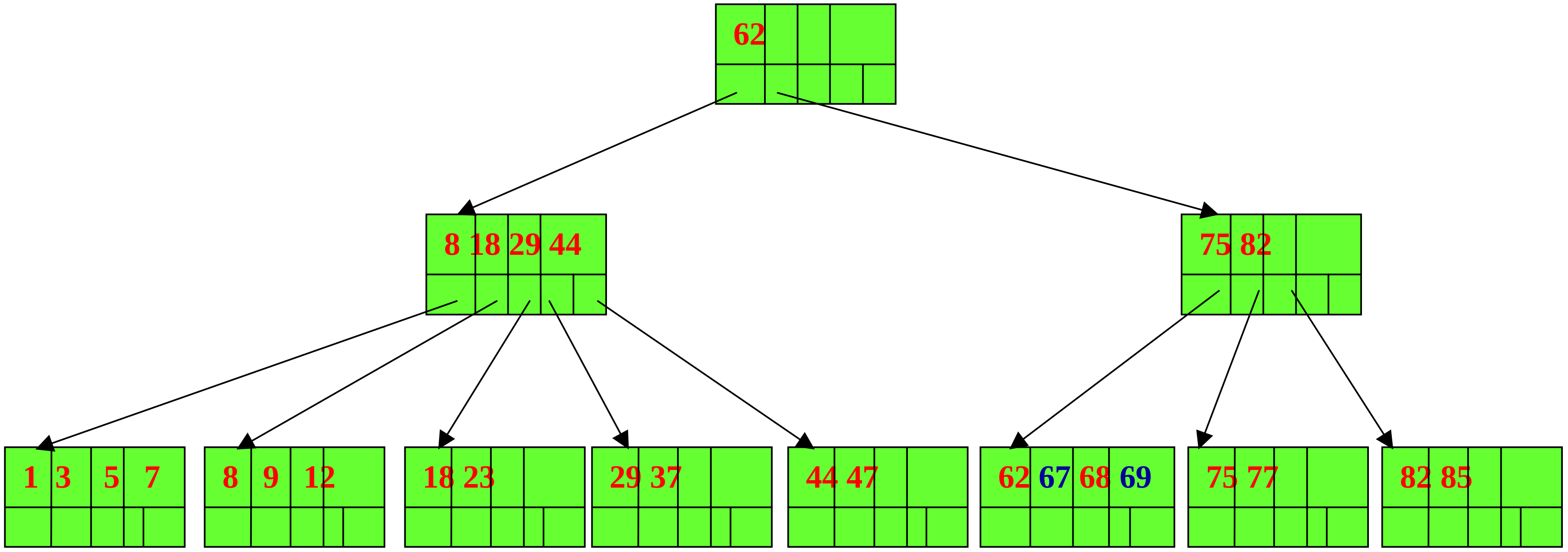
Answer the following questions by showing the resulting B+Tree. When a node split, show the B+Tree before and after the splitting. All the questions modify the B+Tree obtained in the previous question (e.g., question 2 modifies the B+Tree resulting from question 1).

1. Insert the new key with value 9.



2. Insert the new keys with the following values: 67, 69,71

B+Tree before the splitting: Insertion of keys 67, 69.



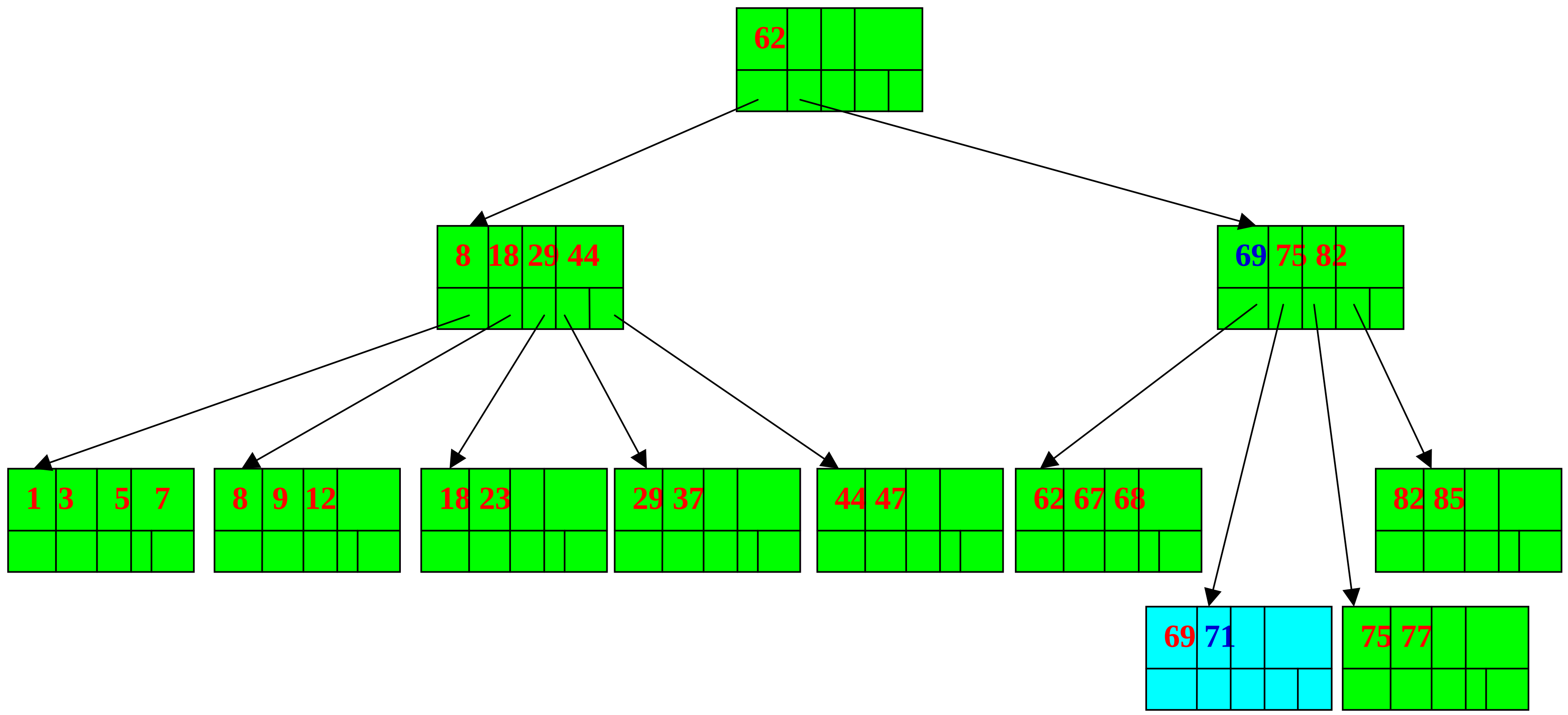
B-Tree Insertion: Splitting Leaves

Let N be a leaf whose capacity is n keys, and we need to insert the $(n+1)$ key-pointer pair.

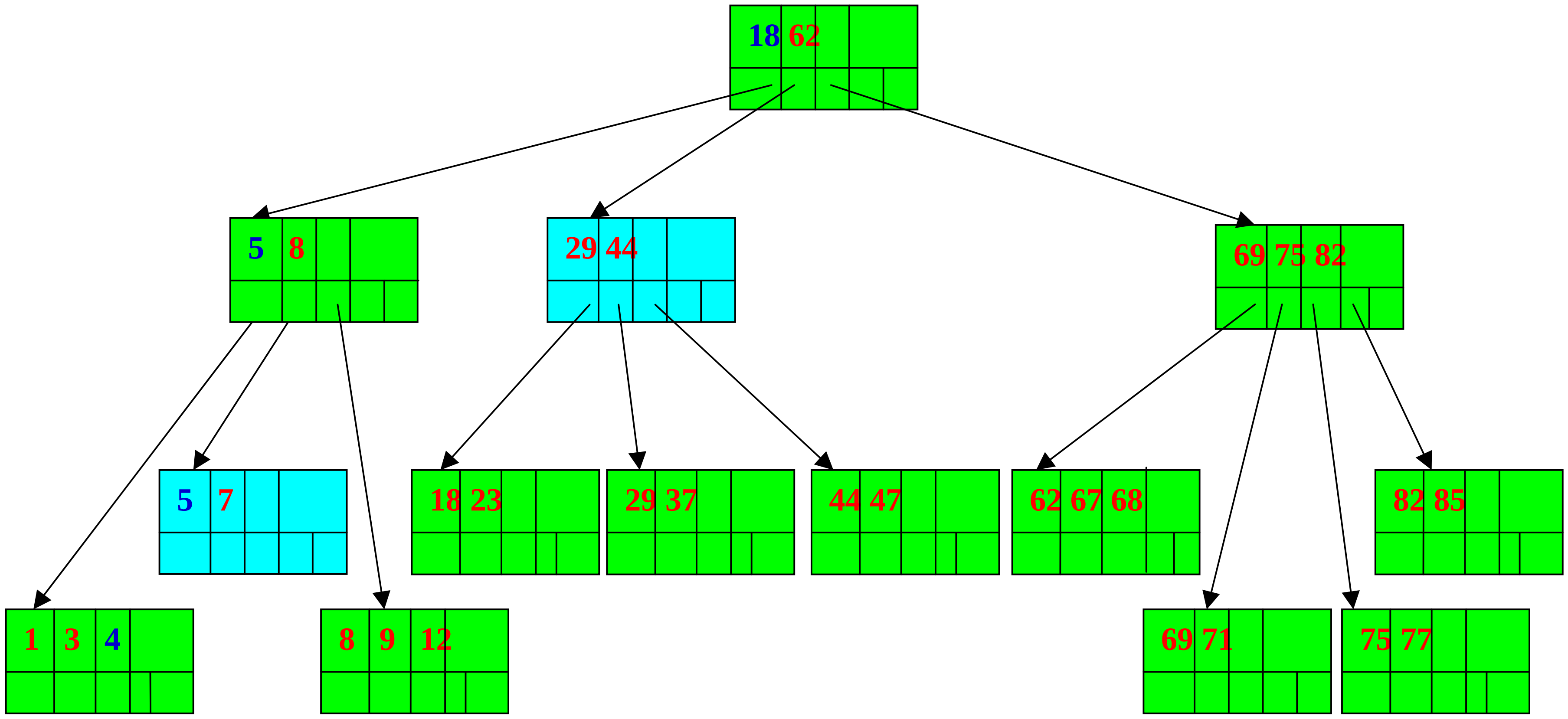
1. Create a new sibling node M, to the right of N;
2. The first $\lceil \frac{n+1}{2} \rceil$ key-pointer pairs remain with N, while the other move to M.
3. The first key of the new node M is also inserted at the parent node.

Note: At least $\lfloor \frac{n+1}{2} \rfloor$ key-pointer pairs for both of the splitted nodes.

B+Tree after the split: insertion of key 71. Splitting leaves.



3. Insert the new key with value 4. Splitting leaves and interior nodes.



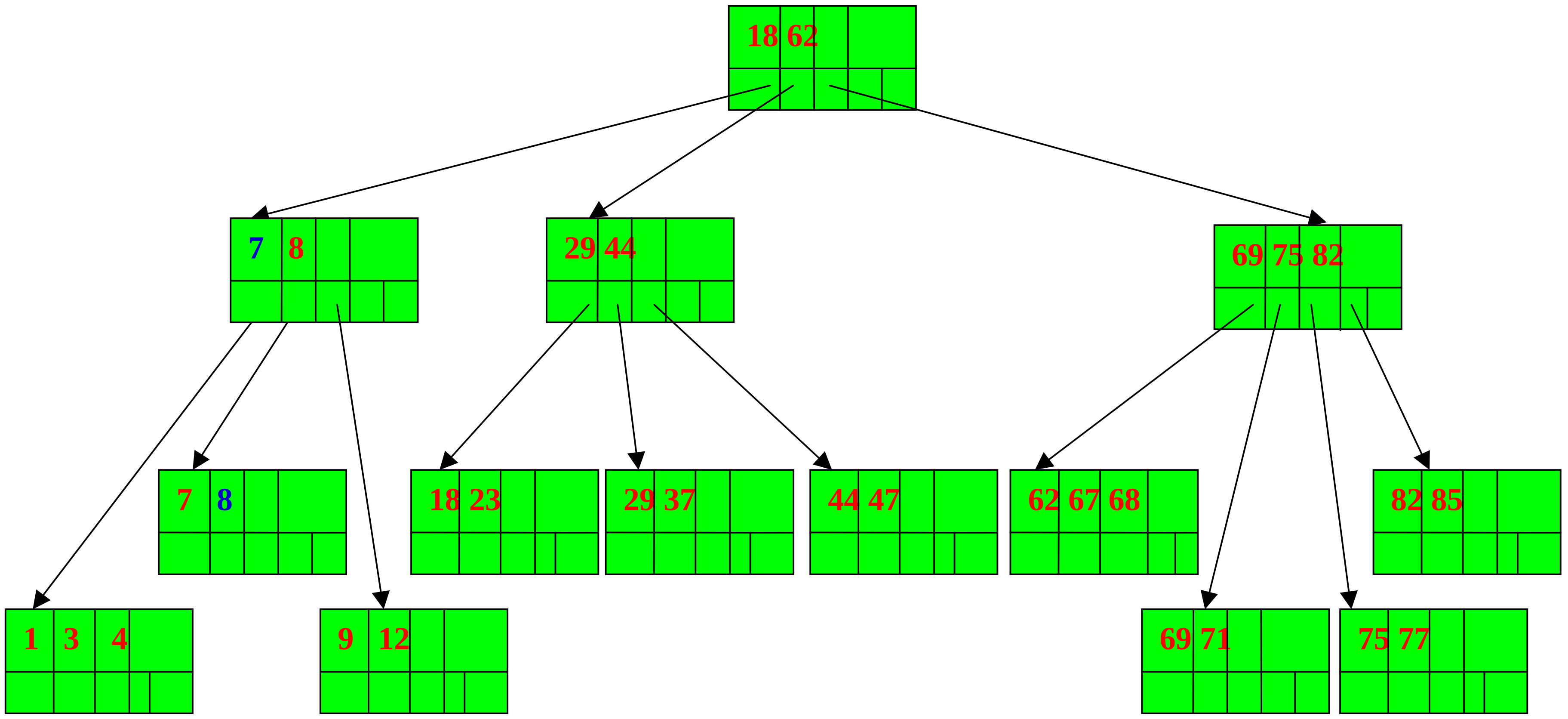
B-Tree Insertion: Splitting Interior Nodes

Let N be an interior node whose capacity is n keys and $(n+1)$ pointers, and N has been assigned the new pointer $(n+2)$ because of a node splitting at the inferior level.

1. Create a new sibling node M, to the right of N;
2. Leave at N the first $\lceil \frac{n+2}{2} \rceil$ pointers, and move the other to M;
3. The first $\lceil \frac{n}{2} \rceil$ keys stay with N, while the last $\lfloor \frac{n}{2} \rfloor$ keys move to M. Since there are $(n+1)$ keys there is one key in the middle (say it K_l) that doesn't go with neither N nor M, but:
   - K_l is reachable via the first of M's children;
   - K_l is used by the common parent of N and M to distinguish the search between those two nodes.

Note: At least $\lceil \frac{n+1}{2} \rceil$ pointers for both of the splitted nodes.

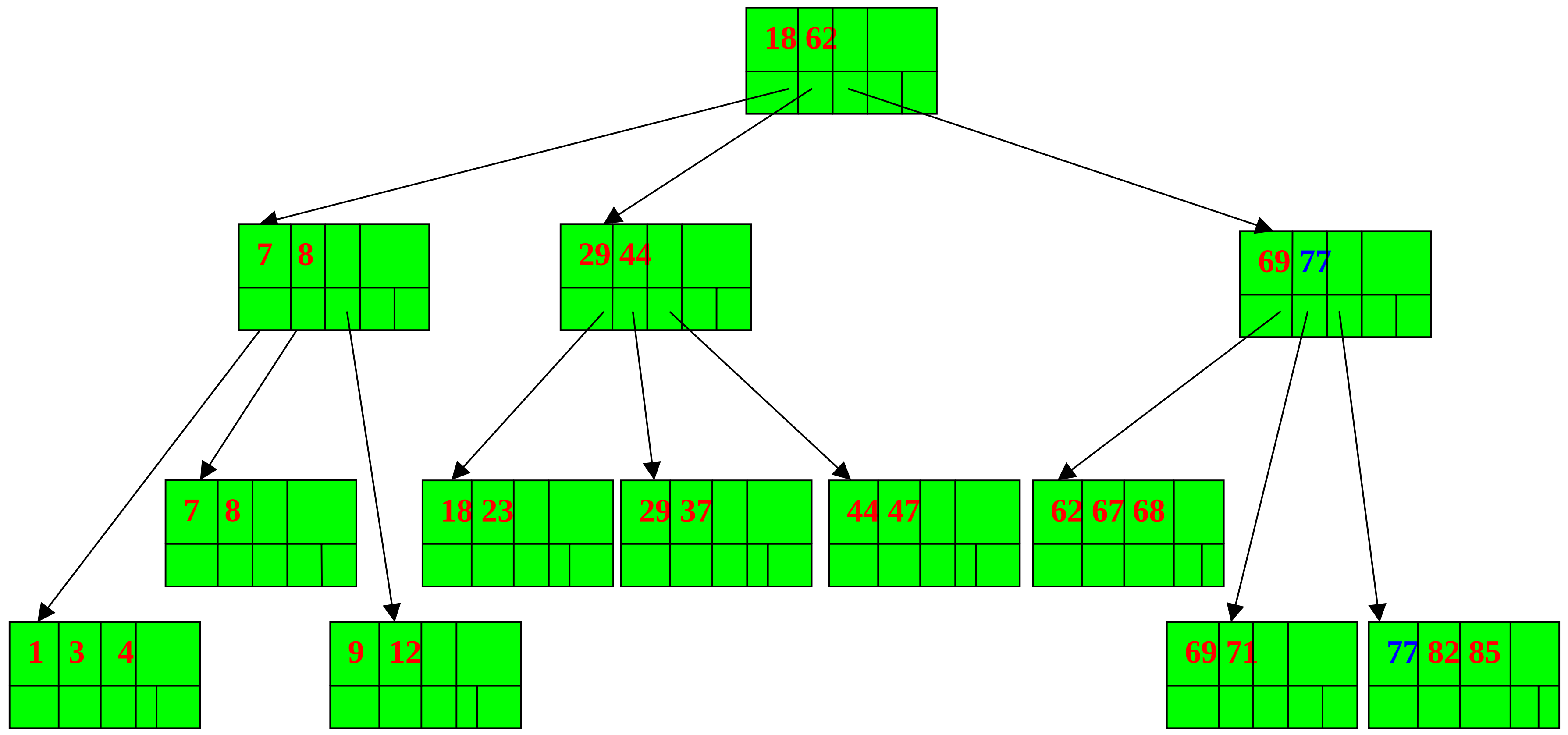
4. Delete the key with value 5. Borrowing one sibling key.



B-Tree Deletion

- Start a search for the key being deleted;
- Delete the record from the data file and the key-pointer pair from the leaf of the B-tree;
- If the lower limit of keys and pointers on a leaf is violated then two cases are possible:
  1. Look for an adjacent sibling that is above lower limit and “steal” a key-pointer pair from that leaf, keeping the order of keys intact. Make sure keys for the parent are adjusted to reflect the new situation.
  2. Hard case: no adjacent sibling can provide an extra key. Then there must be two adjacent siblings leaves, one at minimum, one below minimum capacity. Just enough to merge nodes deleting one of them. Keys at the parent should be adjusted, and then delete a key and a pointer. If the parent is below the minimum capacity then we recursively apply the deletion algorithm at the parent.

5. Delete the key with value 75. Coalescing leaves.



EXERCISE 2.

Given a relation, R, there is a need for building a B+Tree as a dense index on its primary key. You have the following data:

1. The number of tuples in R, T(R) = 10,000,000
2. The primary key for R is made by a single attribute that requires 15 bytes to be stored.
3. Each B+Tree node is stored in a disk block with size 4KB (i.e., 4096 bytes). To store a pointer 8 bytes are needed.

Given the above data answer the following questions:

a. Compute the maximum number, N, of keys that can be allocated in a single B+Tree node.

N*15+(N+1)*8 = 4096; i.e., N≦round-down(4088/23). Then Nmax=177

b. Compute the number of levels and the number of nodes of the B+Tree.

round-up(10,000,000/177) = 56,498	**Leaves**
round-up(56,498/178) = 318	**Internal nodes**
round-up(318/178) = 2	**Internal nodes (next level)**
1 + 2 + 318 + 56,498 = 56,819	**Total number of nodes**
L =4	**Number of Levels**