Red-Black Trees

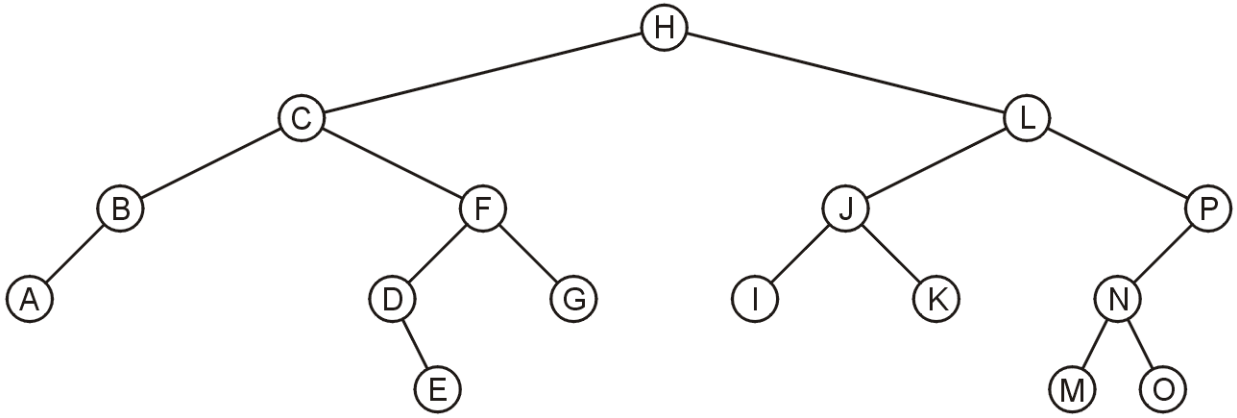
A red black tree “colours” each node within a tree either red or black

- This can be represented by a single bit
- In AVL trees, balancing restricts the difference in heights to at most one
- For red-black trees, we have a different set of rules related to the colours of the nodes

Red-Black Trees

Define a *null path* within a binary tree as any path starting from the root where the last node is not a full node

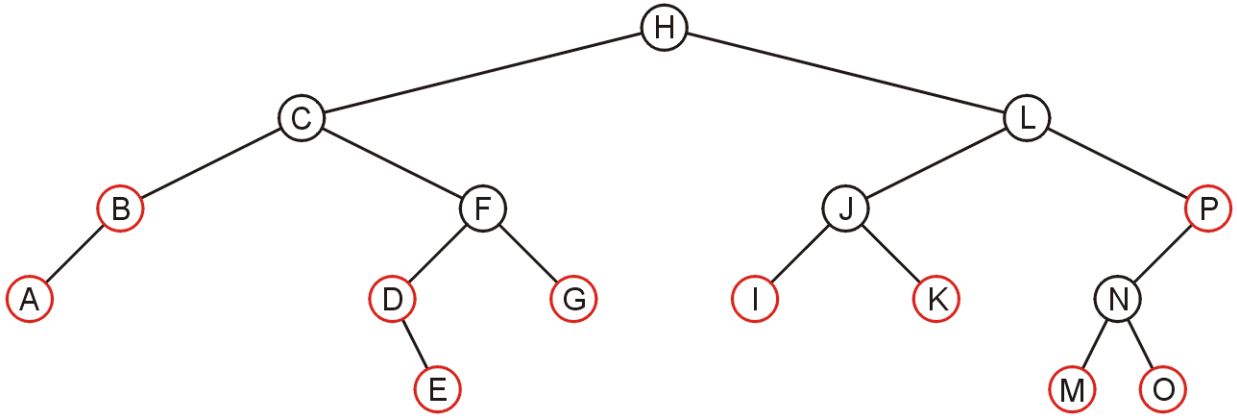
– Consider the following binary tree:



Red-Black Trees

All null paths include:

(H, C, **B**)
(H, C, B, **A**)
(H, C, F, **D**)
(H, C, F, D, **E**)
(H, C, F, **G**)
(H, L, J, **I**)
(H, L, J, **K**)
(H, L, **P**)
(H, L, P, N, **M**)
(H, L, P, N, **O**)



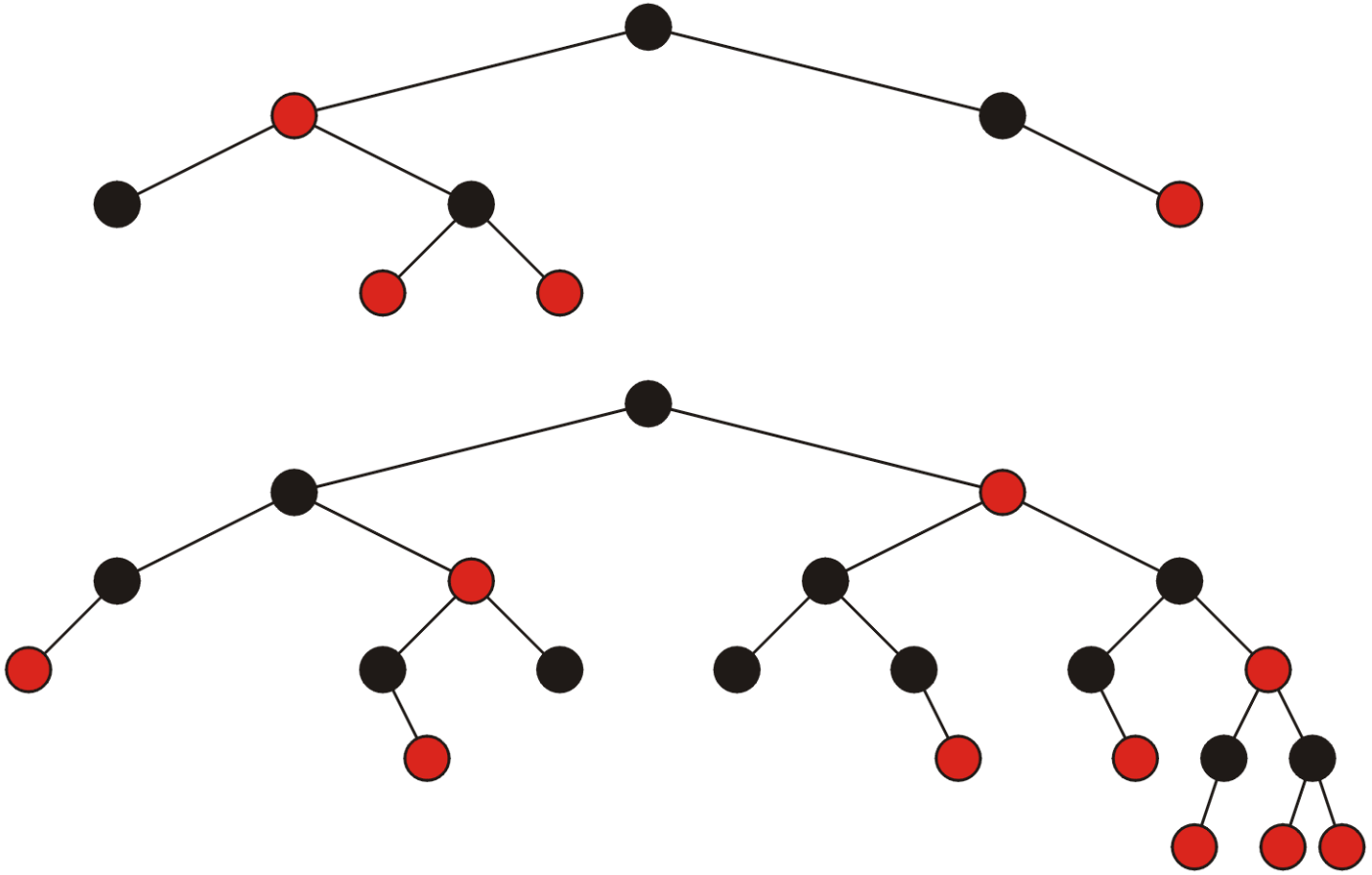
Red-Black Trees

The three rules which define a red-black tree are

1. The root must be black,
2. If a node is red, its children must be black, and
3. Each null path must have the same number of black nodes

Red-Black Trees

These are two examples of red-black trees:



Red-Black Trees

Theorem:

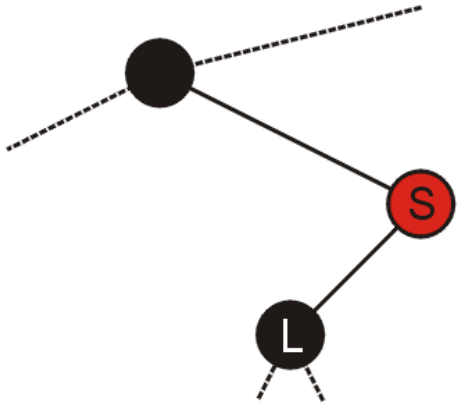
Every red node must be either

- A full node (with two black children), or
- A leaf node

Proof:

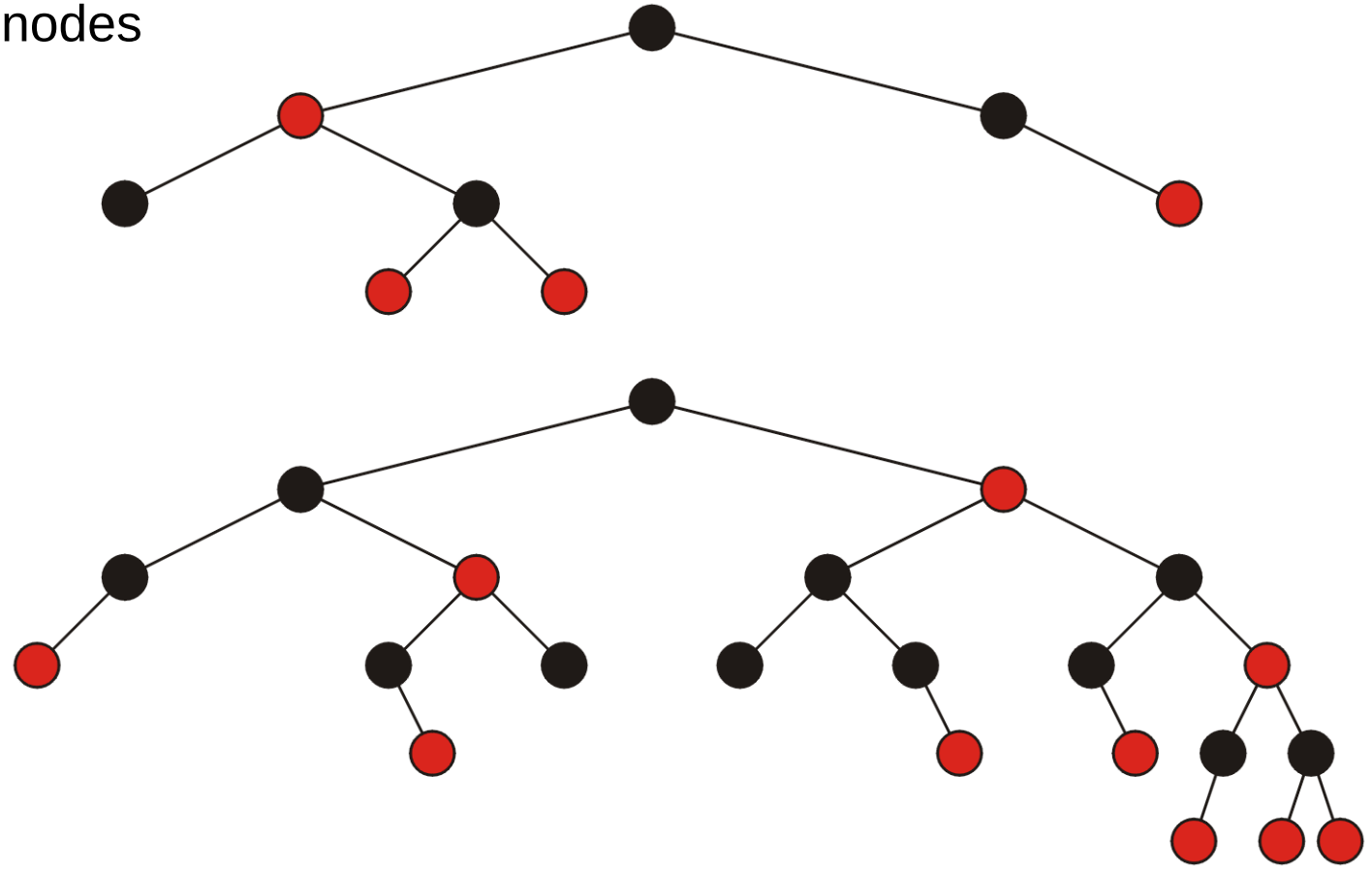
Suppose node S has one child:

- The one child L must be black
- The null path ending at S has k black nodes
- Any null path containing the node L will therefore have at least $k + 1$ black nodes



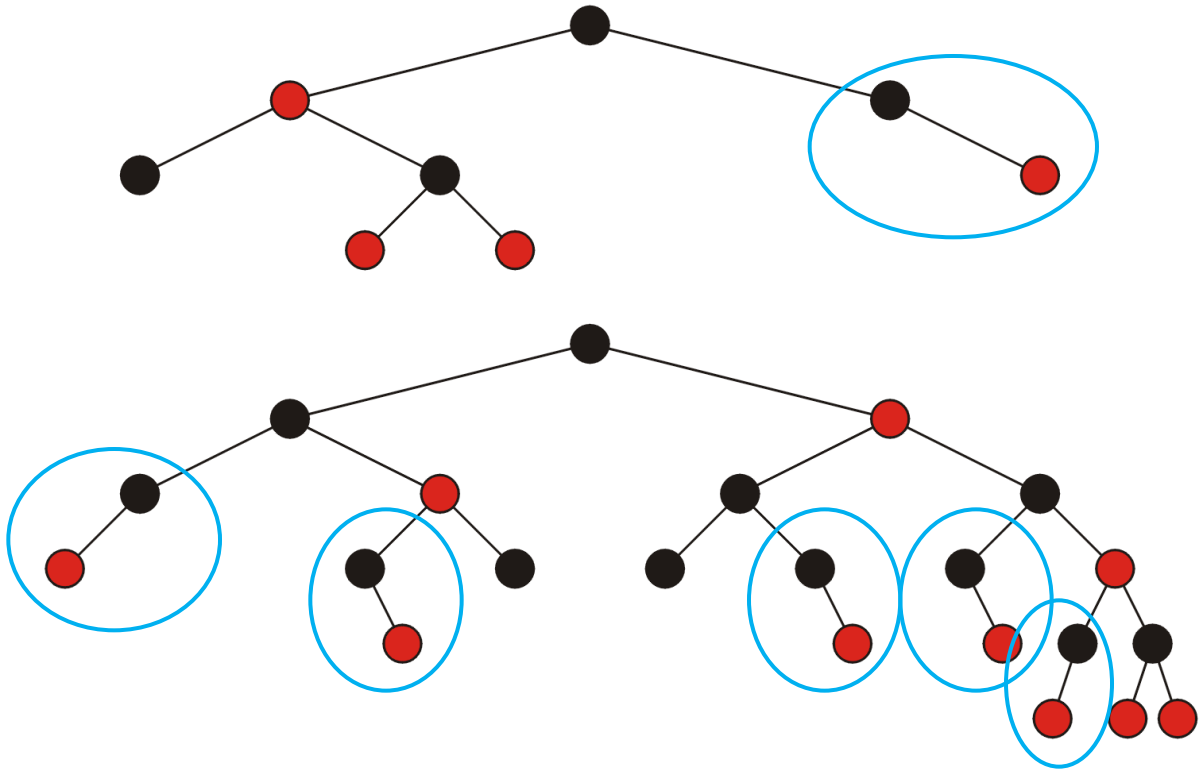
Red-Black Trees

In our two examples, you will note that all red nodes are either full or leaf nodes



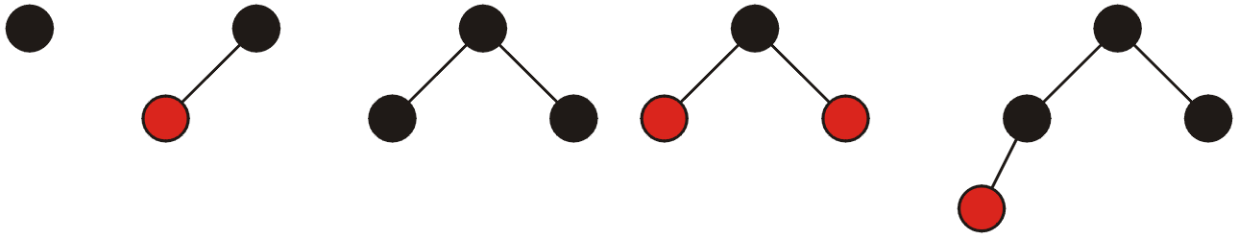
Red-Black Trees

Again, in our examples, the only nodes with a single child are black and the corresponding children are red



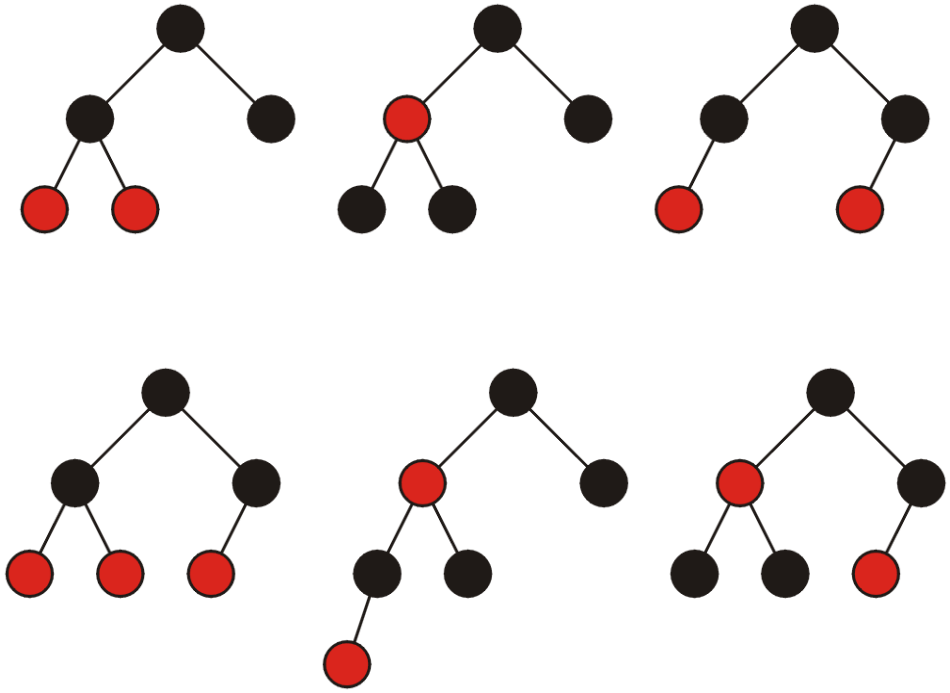
Red-Black Trees

All red-black trees with 1, 2, 3, and 4 nodes:



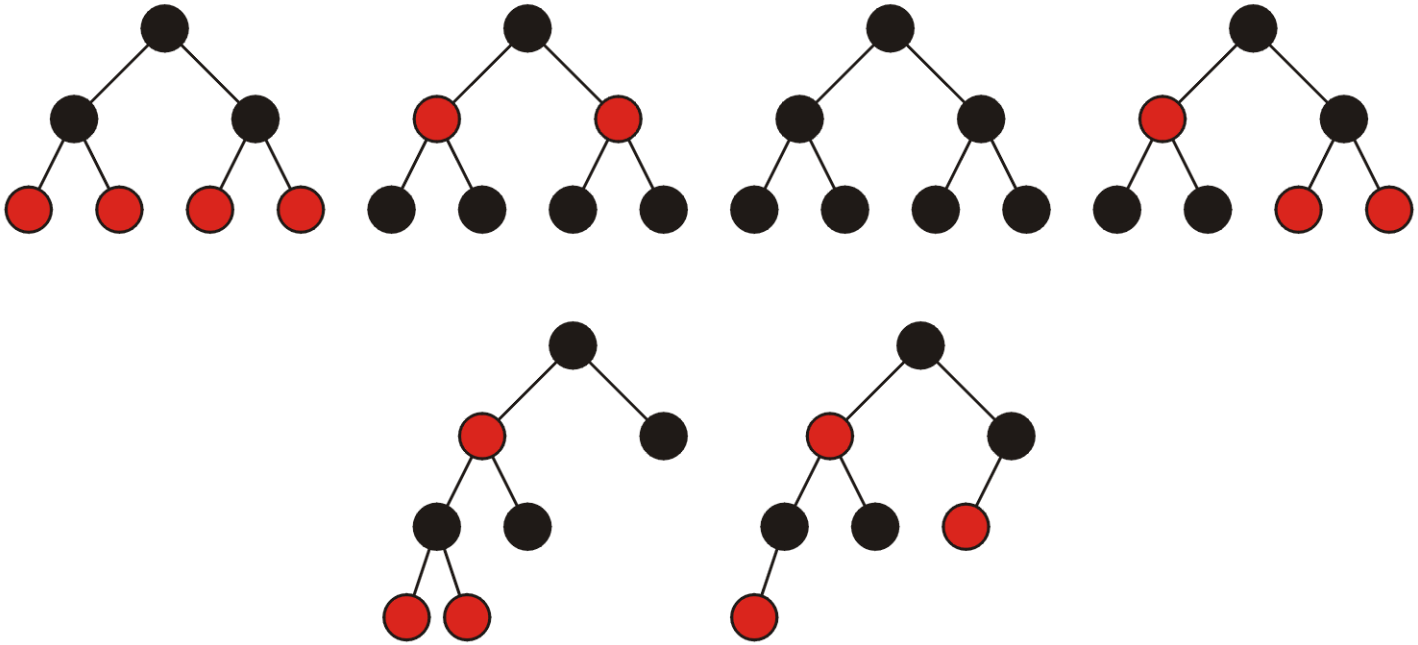
Red-Black Trees

All red-black trees with 5 and 6 nodes:



Red-Black Trees

All red-black trees with seven nodes—most are shallow:



Red-Black Trees

Every perfect tree is a red-black tree if each node is coloured black

A complete tree is a red-black tree if:

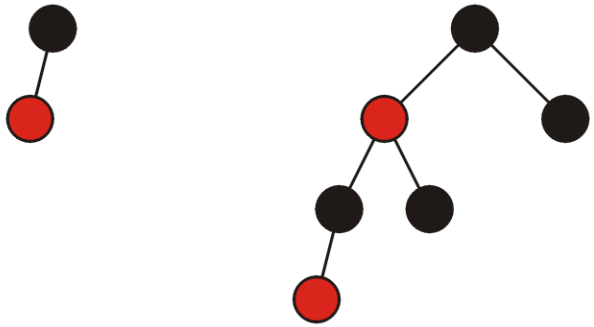
- each node at the lowest depth is coloured red, and
- all other nodes are coloured black

What is the worst case?

Red-Black Trees

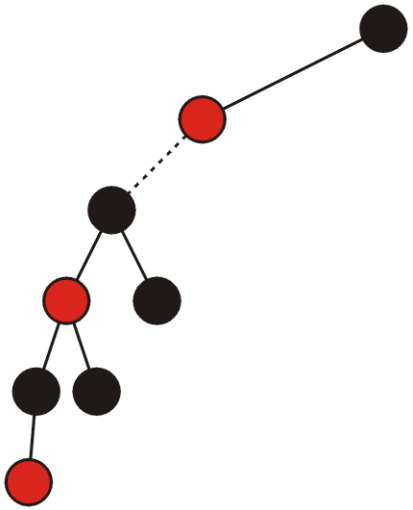
Any worst-case red-black tree must have an alternating red-black pattern down one side

The following are the worst-case red-black trees with 1 and 2 black nodes per null path (*i.e.*, heights 1 and 3)



Red-Black Trees

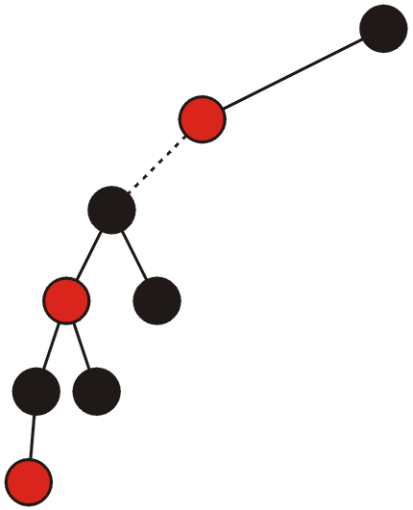
To create the worst-case for paths with 3 black nodes per path, start with a black and red node and add the previous worst-case for paths with 2 nodes



Red-Black Trees

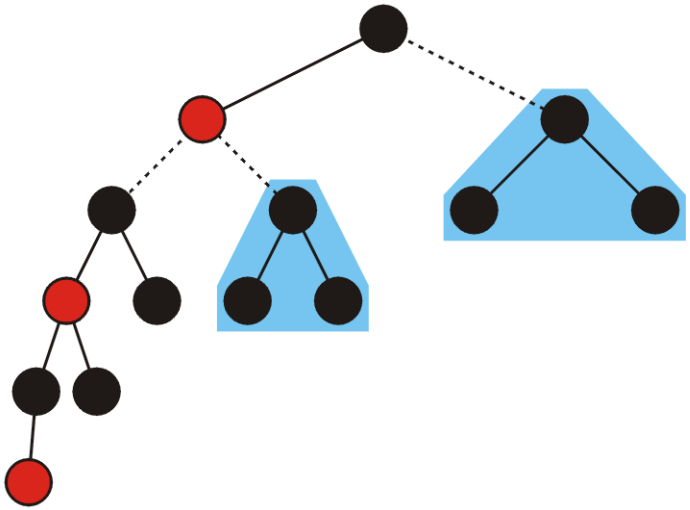
This, however, is not a red-black tree because the two top nodes do not have paths with three black nodes

- To solve this, add the optimal red-black trees with two black nodes per path



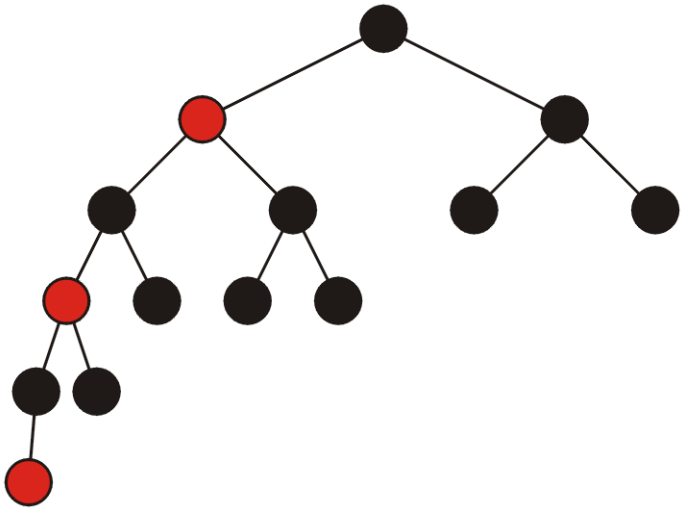
Red-Black Trees

That is, add two perfect trees with height 1 to each of the missing sub-trees



Red-Black Trees

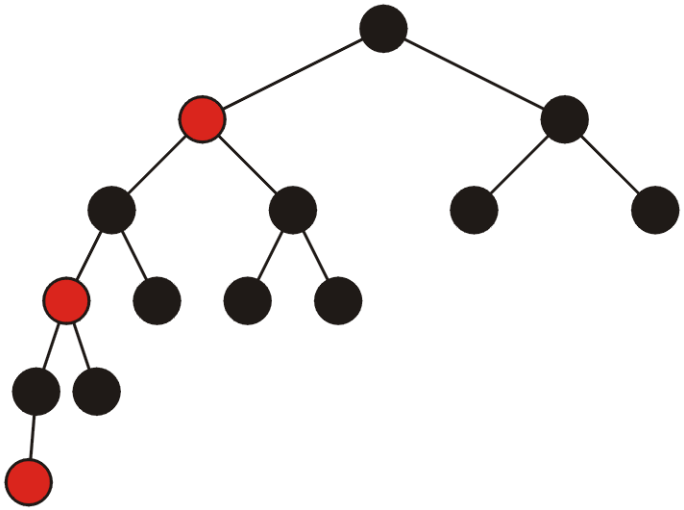
Thus, we have the worst-case for a red-black tree with three black nodes per path (or a red-black tree of height 5)



Red-Black Trees

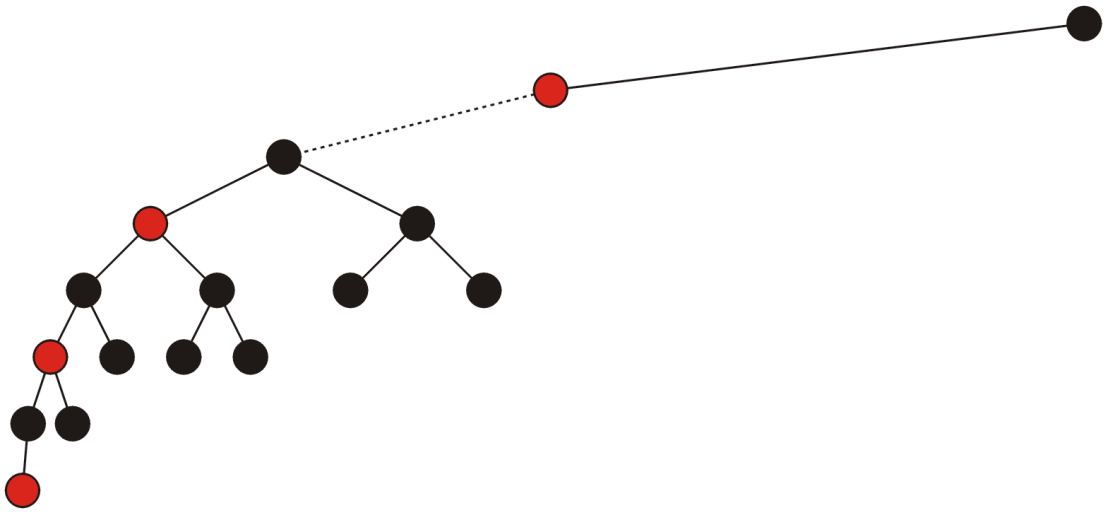
Note that the left sub-tree of the root has height 4 while the right has height 1

- Thus suggests that AVL trees may be better...



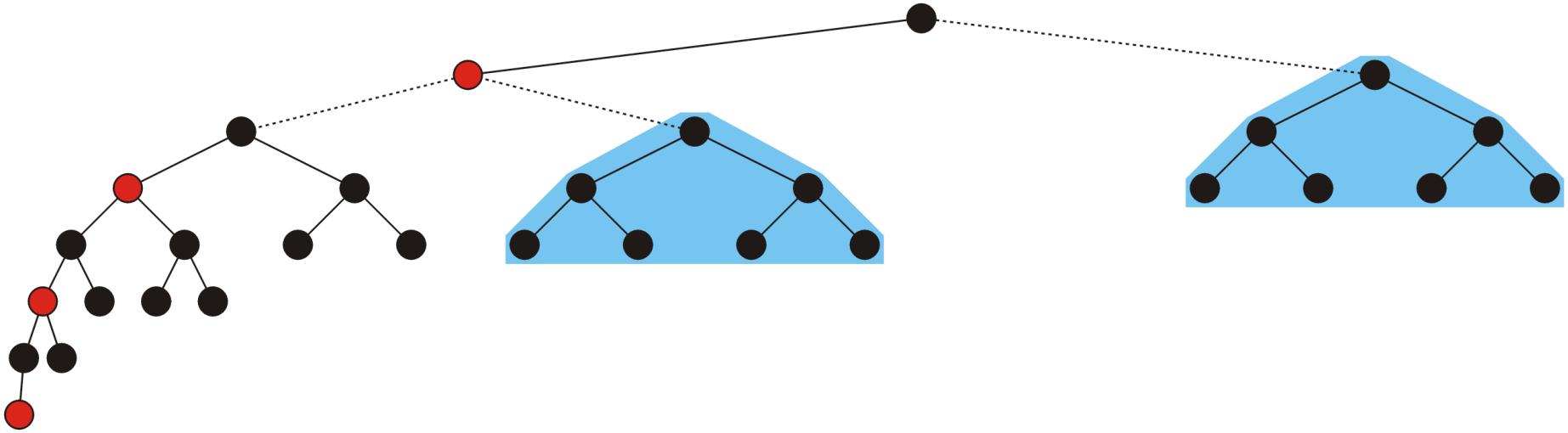
Red-Black Trees

To create the worst-case scenario for red-black trees with 4 black nodes per null path (*i.e.*, height 7), again, start with two nodes and add the previous worst-case tree:



Red-Black Trees

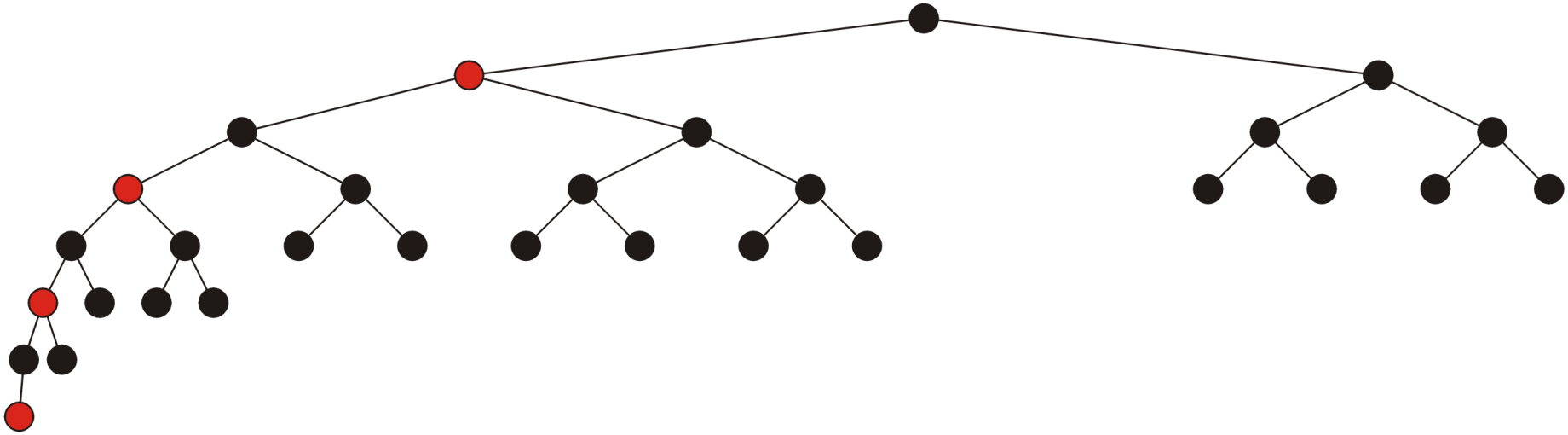
To satisfy the definition of the red-black tree, add two perfect trees of height 2:



Red-Black Trees

Thus, with 30 nodes, the worst case red-black tree has height 7

- The worst-case AVL tree with 33 nodes has height 6...



Red-Black Trees

This table shows the number of nodes in a worst-case trees for the given heights

Thus, an AVL tree with 131070 nodes has a height of 23 while a red-black tree could have a height as large as 31

Height	AVL Tree	Red-Black Tree
1	2	2
3	7	6
5	20	14
7	54	30
9	143	62
11	376	126
13	986	254
15	2583	510
17	6764	1022
19	17710	2046
21	46367	4094
23	121492	8190
25	317810	16382
27	832039	32766
29	2178308	65534
31	5702886	131070
33	14930351	262142

Red-Black Trees

Comparing red-black trees with AVL trees, we note that:

- Red-black trees require one extra bit per node
- AVL trees require one byte per node (assuming the height will never exceed 255)
  - aside: we can reduce this to two bits, storing one of -1, 0, or 1 indicating that the node is left heavy, balanced, or right heavy, respectively

Red-Black Trees

AVL trees are not as deep in the worst case as are red-black trees

- therefore AVL trees will perform better when numerous searches are being performed,
- however, insertions and deletions will require:
  - more rotations with AVL trees, and
  - require recursions from and back to the root
- thus AVL trees will perform worse in situations where there are numerous insertions and deletions

References

[1] Cormen, Leiserson, and Rivest, *Introduction to Algorithms*, McGraw Hill, 1990, Ch.14, p.261-80.

[2] Weiss, Data Structures and Algorithm Analysis in C++, 3rd Ed., Addison Wesley, §12.2, p.525-34.