CSC 225

Algorithms and Data Structures: I

Rich Little
rlittle@uvic.ca
ECS 516

Building up a heap bottom-up

- n standard insert-operations for a heap result in $O(n \log(n))$ time.
- Can we build up a heap for n given elements faster? Is $O(n)$ possible?

Bottom-Up Heap (pgs. 176-178)

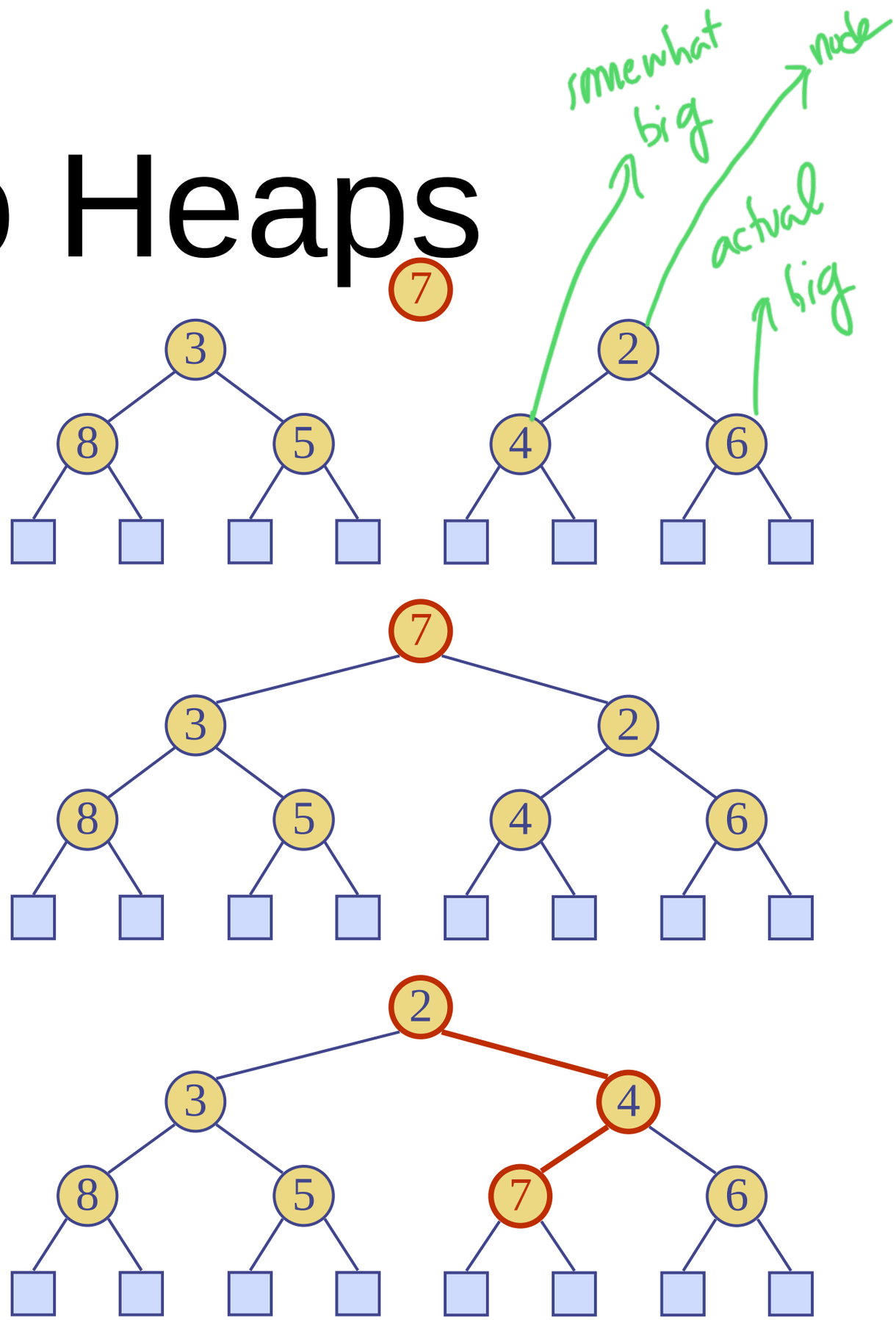
```
Algorithm BottomUpHeap(S):
    Input: A list S storing m keys
    Output: A heap T storing the m keys

    if S is empty then
        return external node
    remove the first key, k, from S
    split S in half, lists S_1 and S_2
    T_1 ← BottomUpHeap(S_1)
    T_2 ← BottomUpHeap(S_2)
    T ← merge(k,T_1,T_2)
    DownHeap(T,root)
    return T
```

Merging Two Heaps

- We are given two heaps and a key *k*

- We create a new heap with the root node storing *k* and with the two heaps as subtrees

- We perform downheap to restore the heap-order property



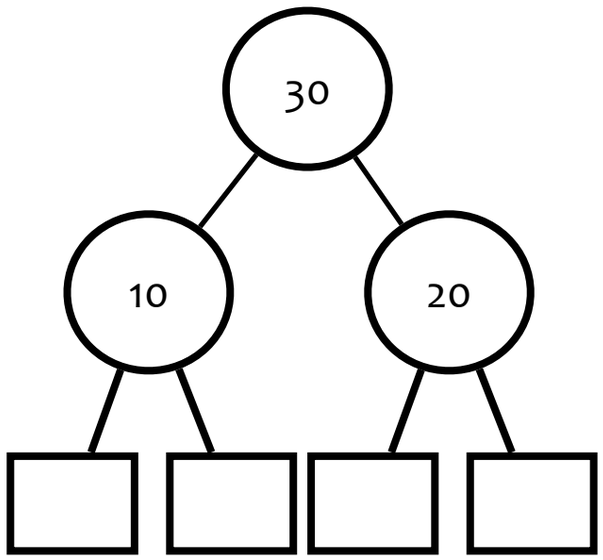
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120

150 70 30 10 20 60 40 50 | 140 100 80 90 130 110 120

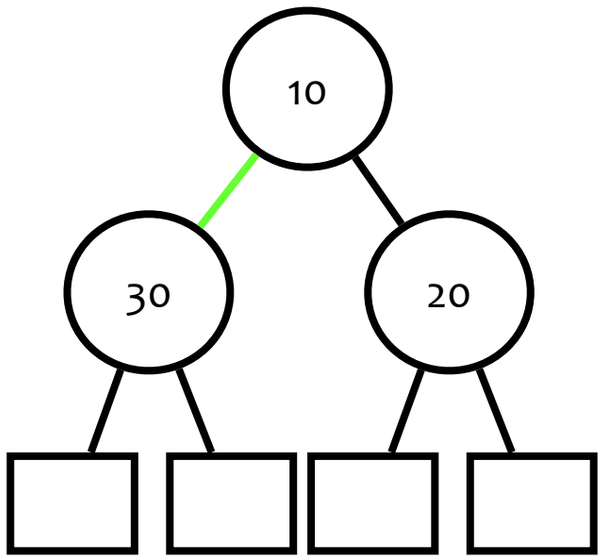
150 70 30 10 20 60 40 50 | 140 100 80 90 130 110 120

150 70 30 10 20 | 60 40 50 | 140 100 80 90 130 110 120

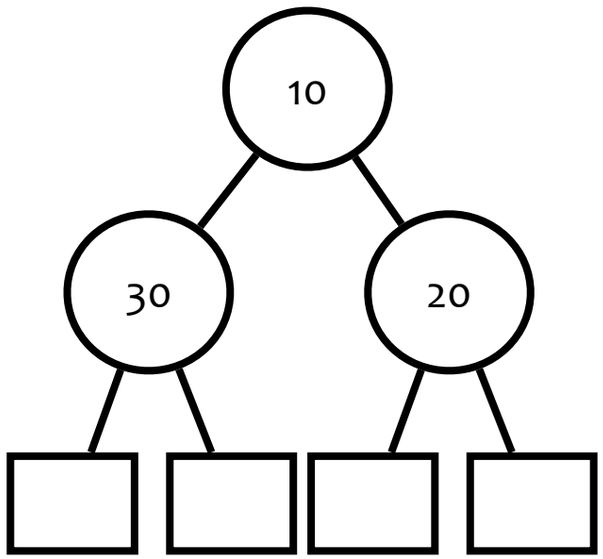
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



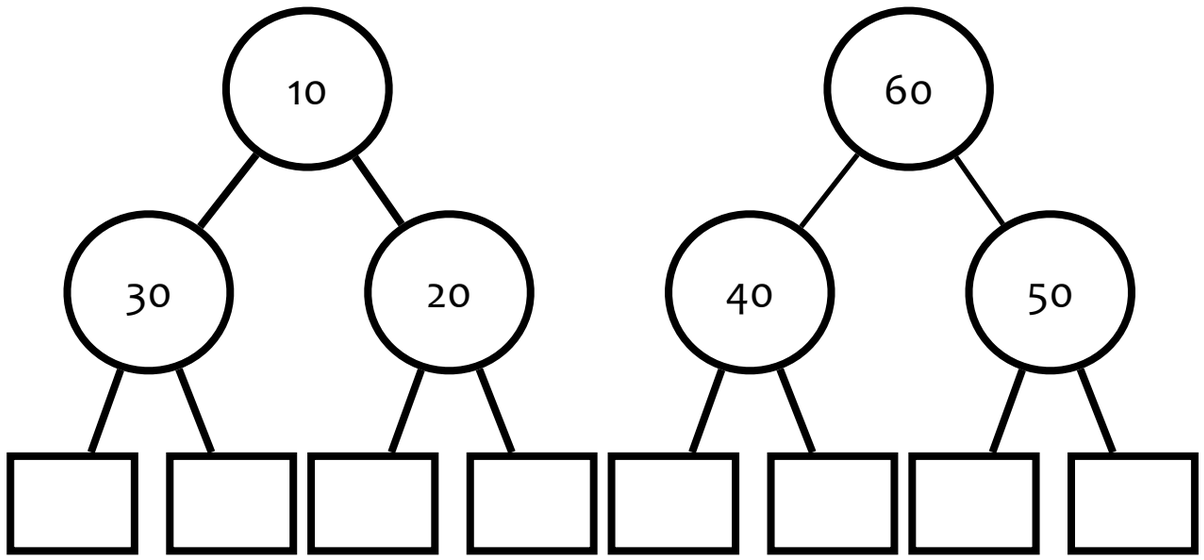
150 70 30 10 | 20 | 60 40 50 | 140 100 80 90 130 110 120



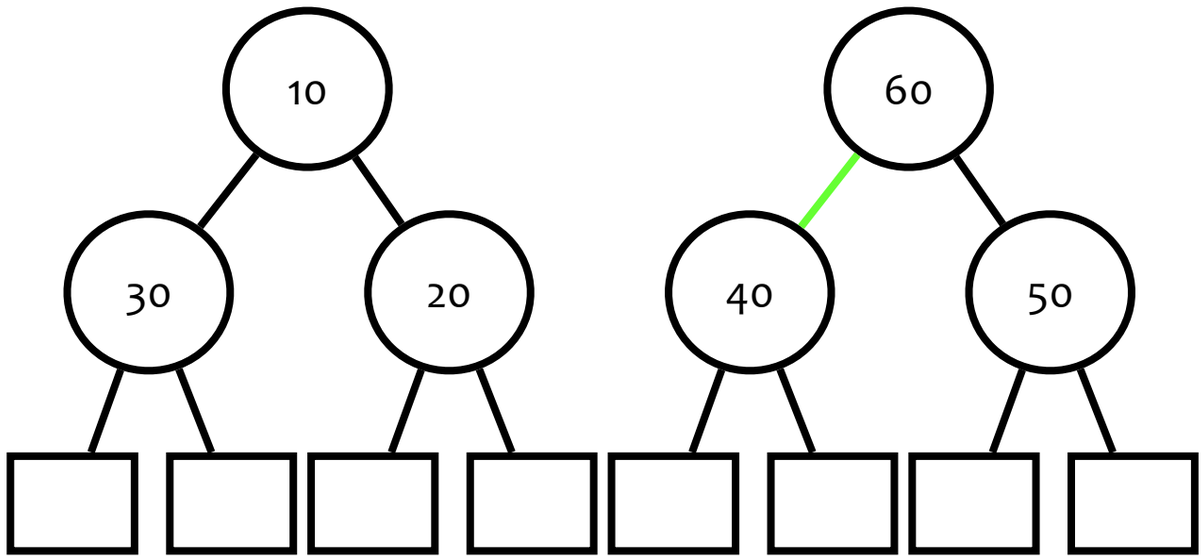
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



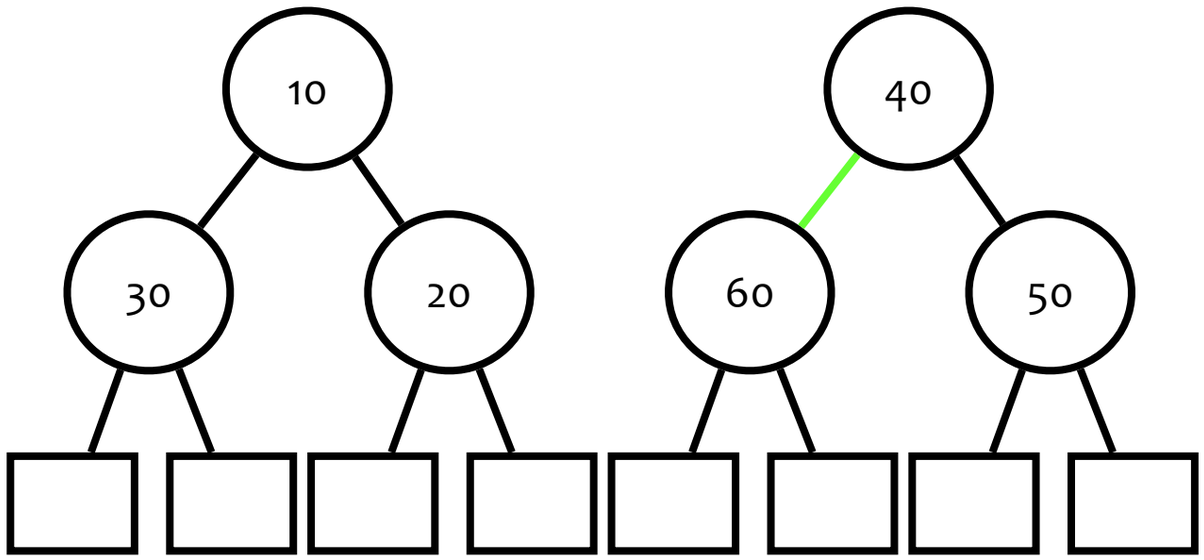
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



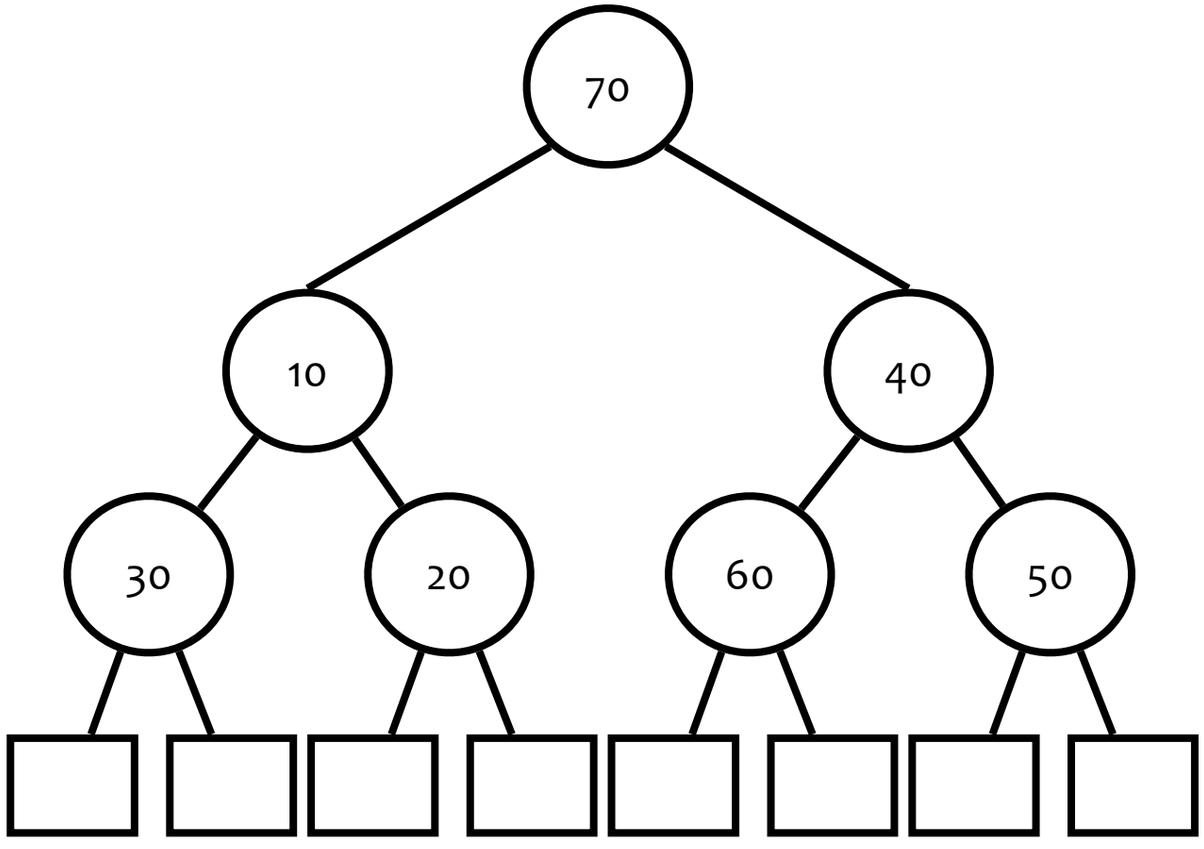
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



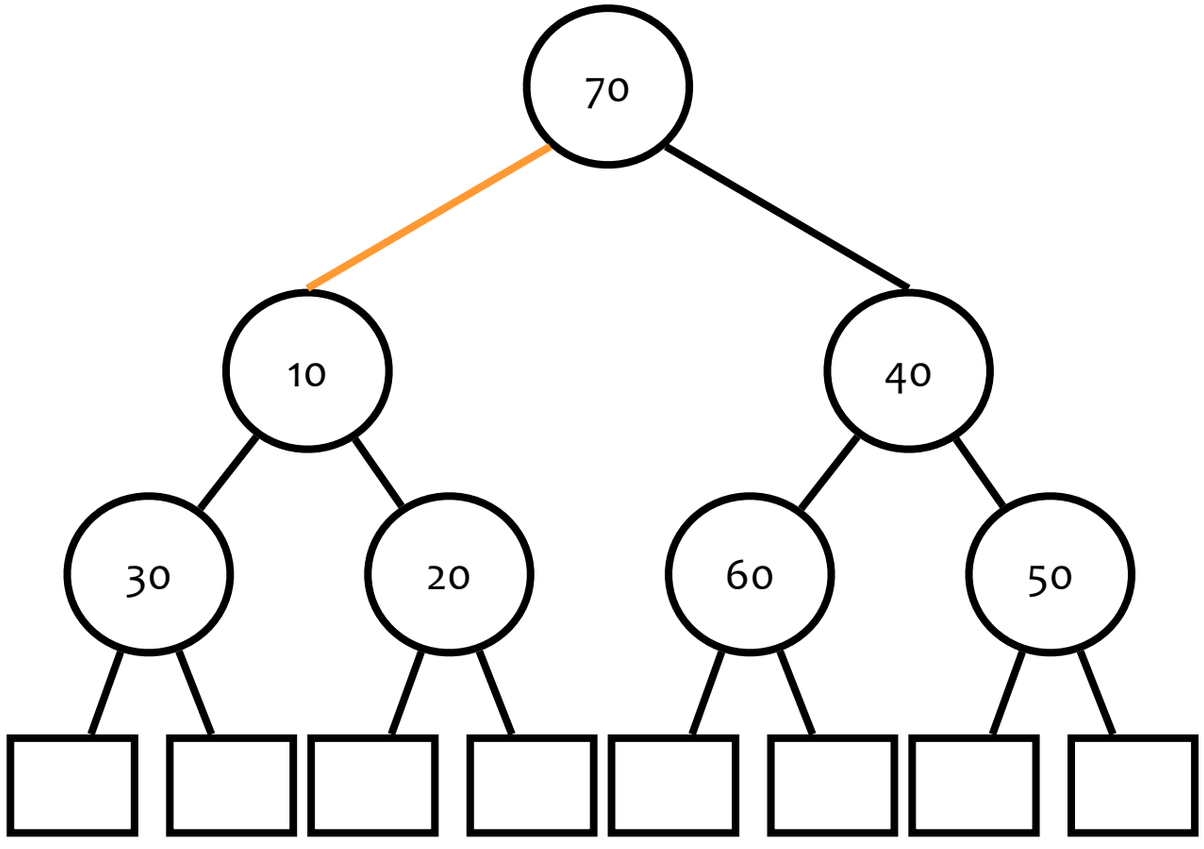
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



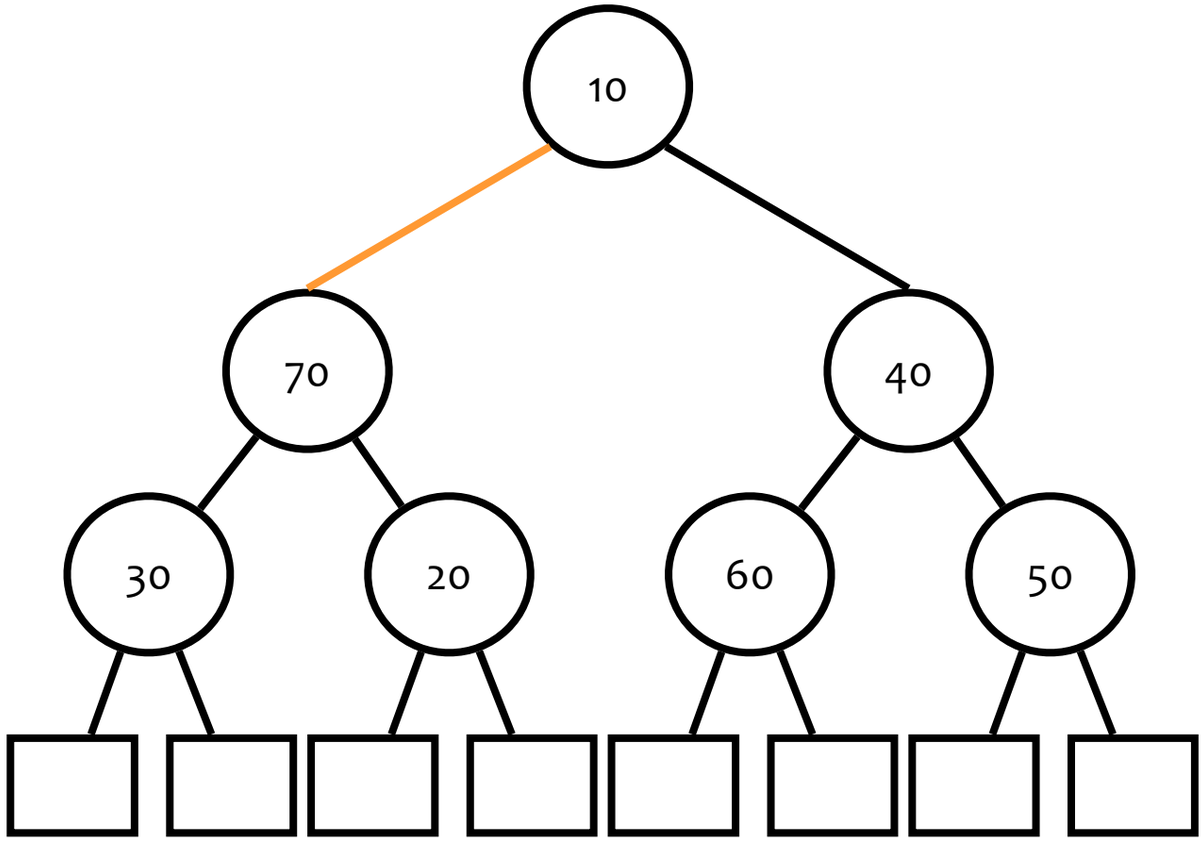
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



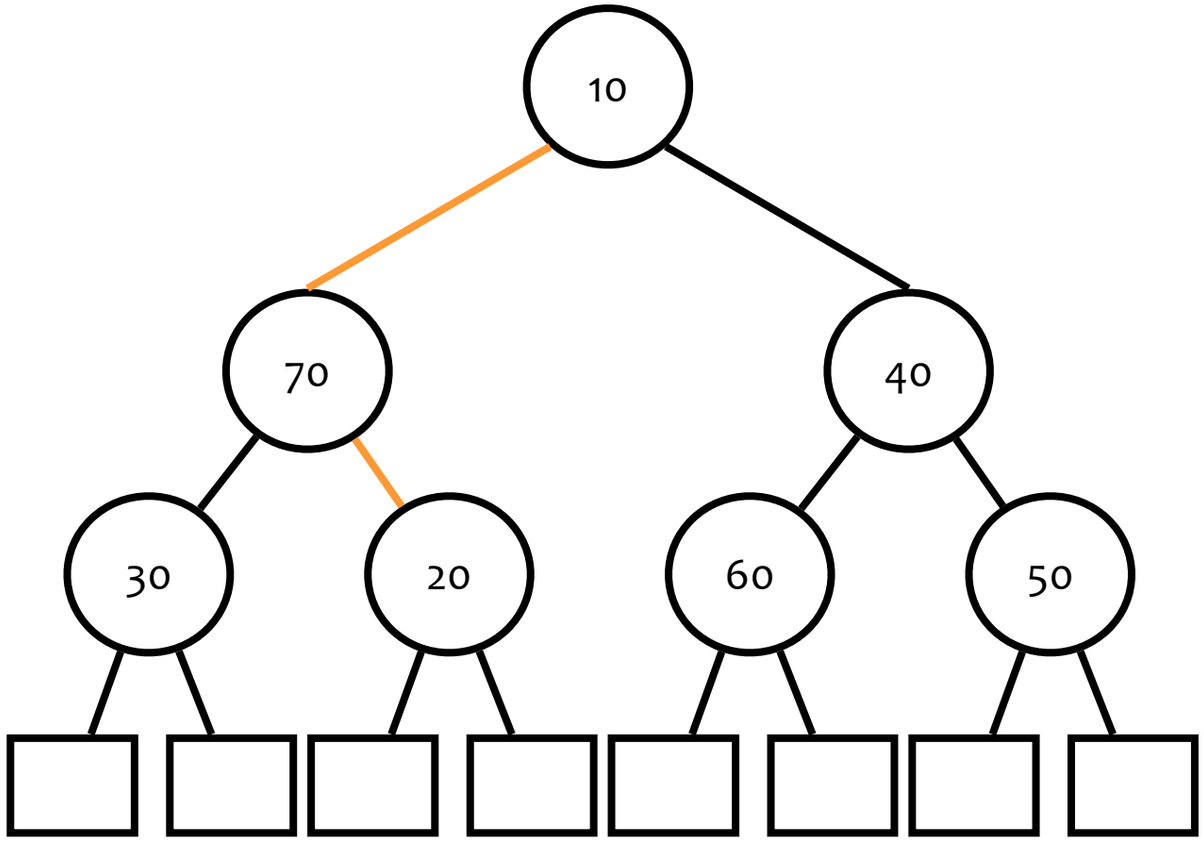
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



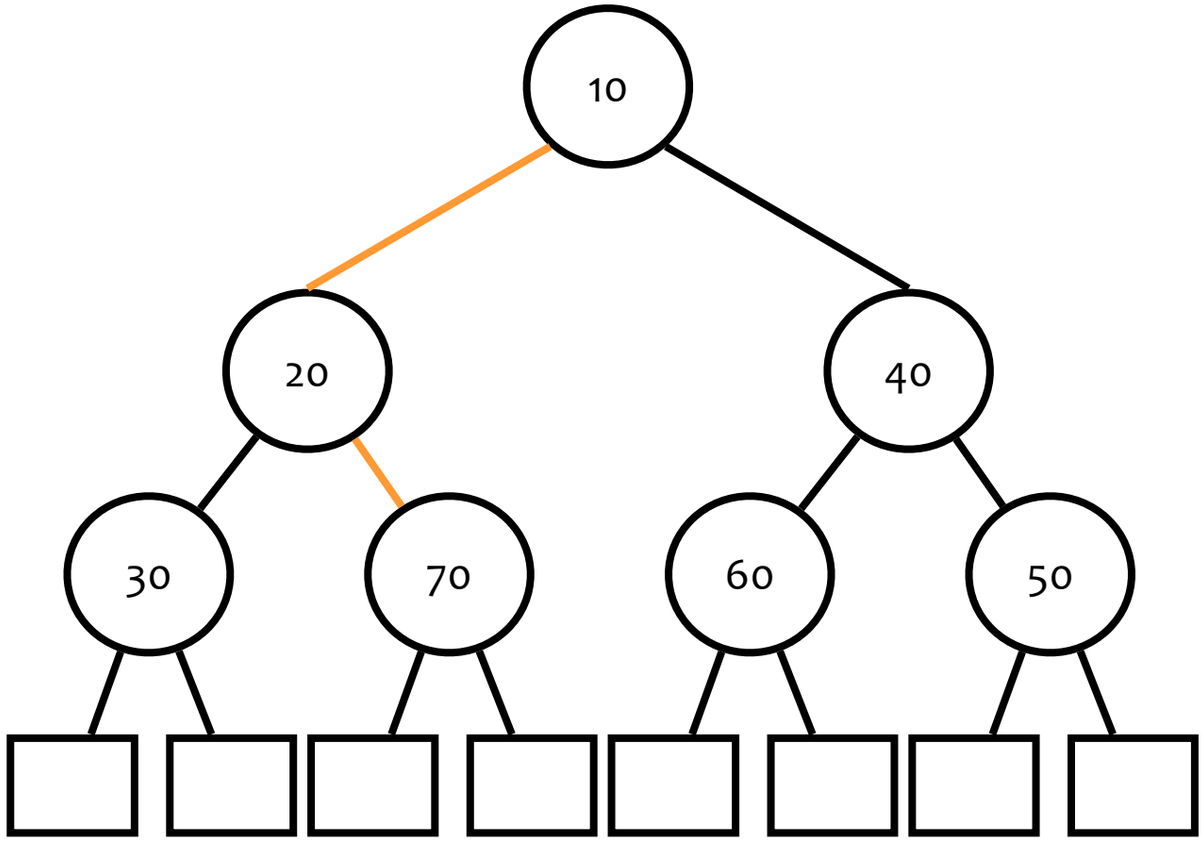
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



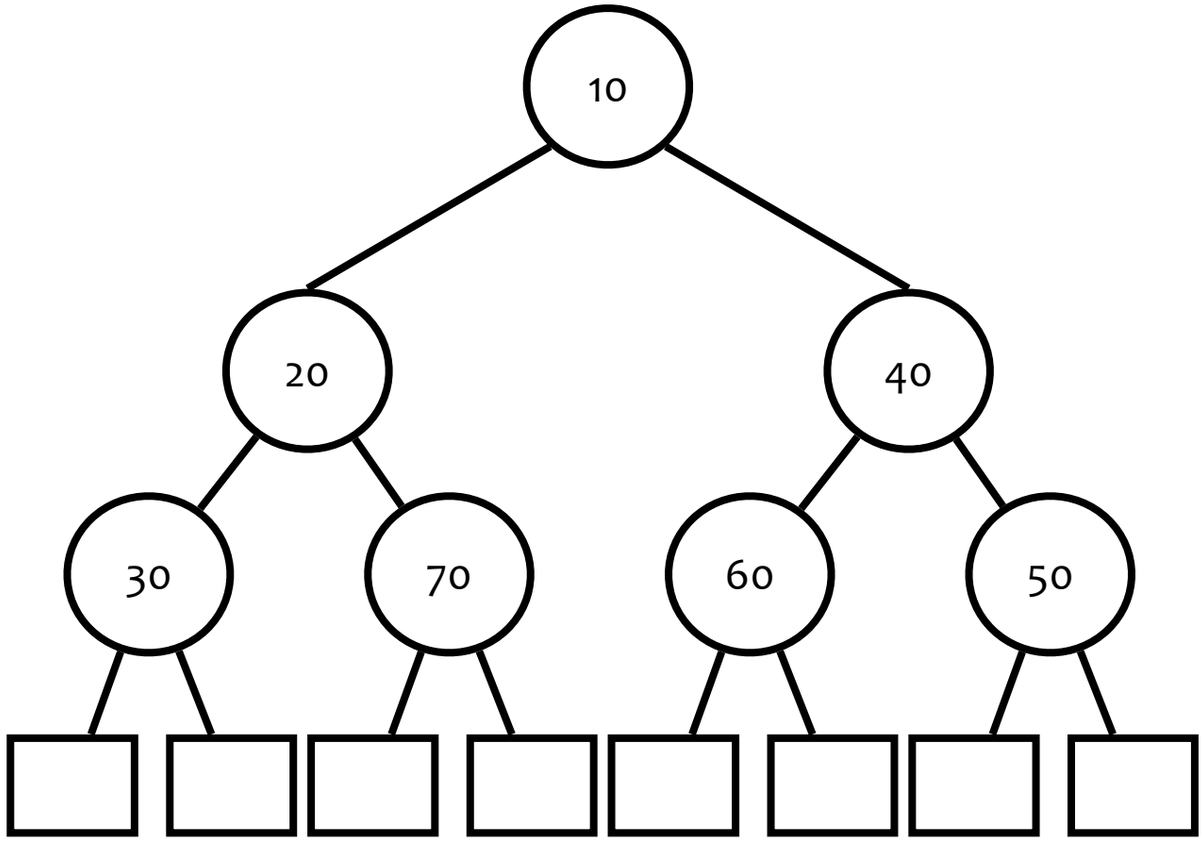
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



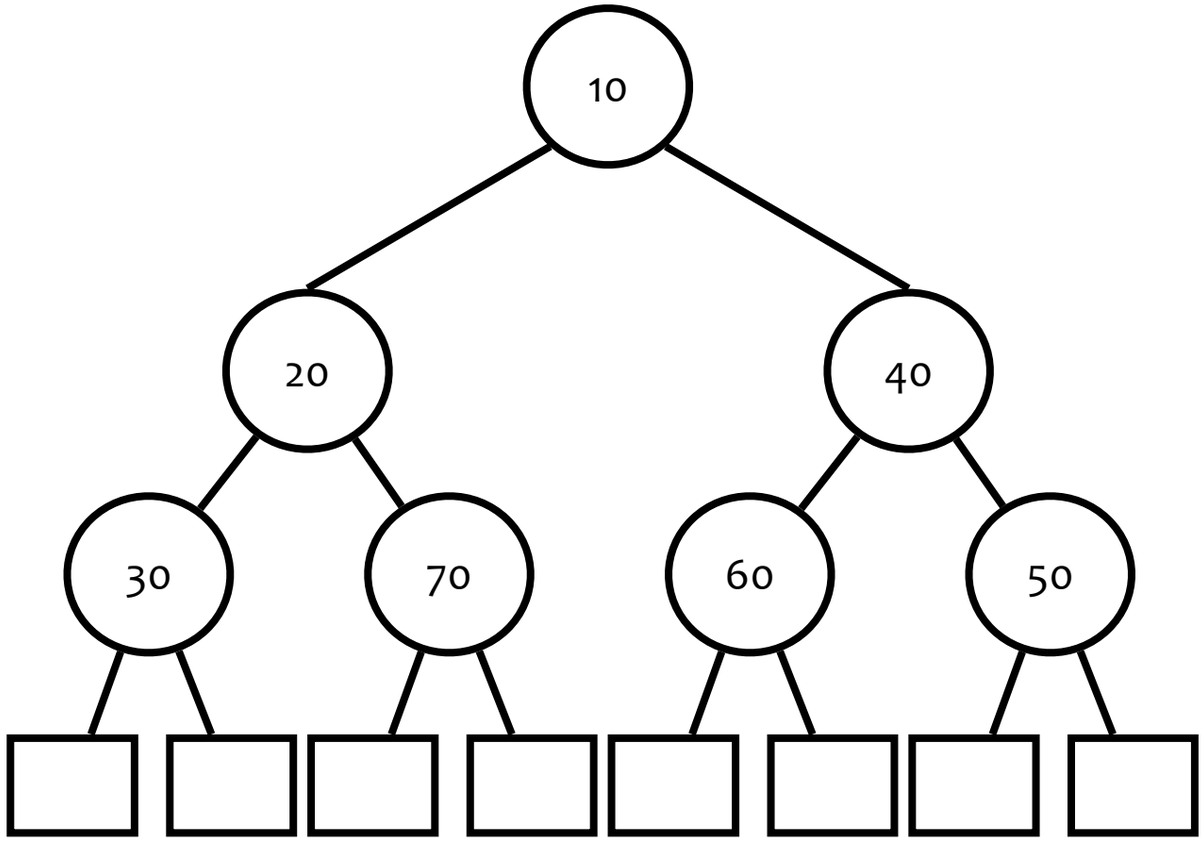
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



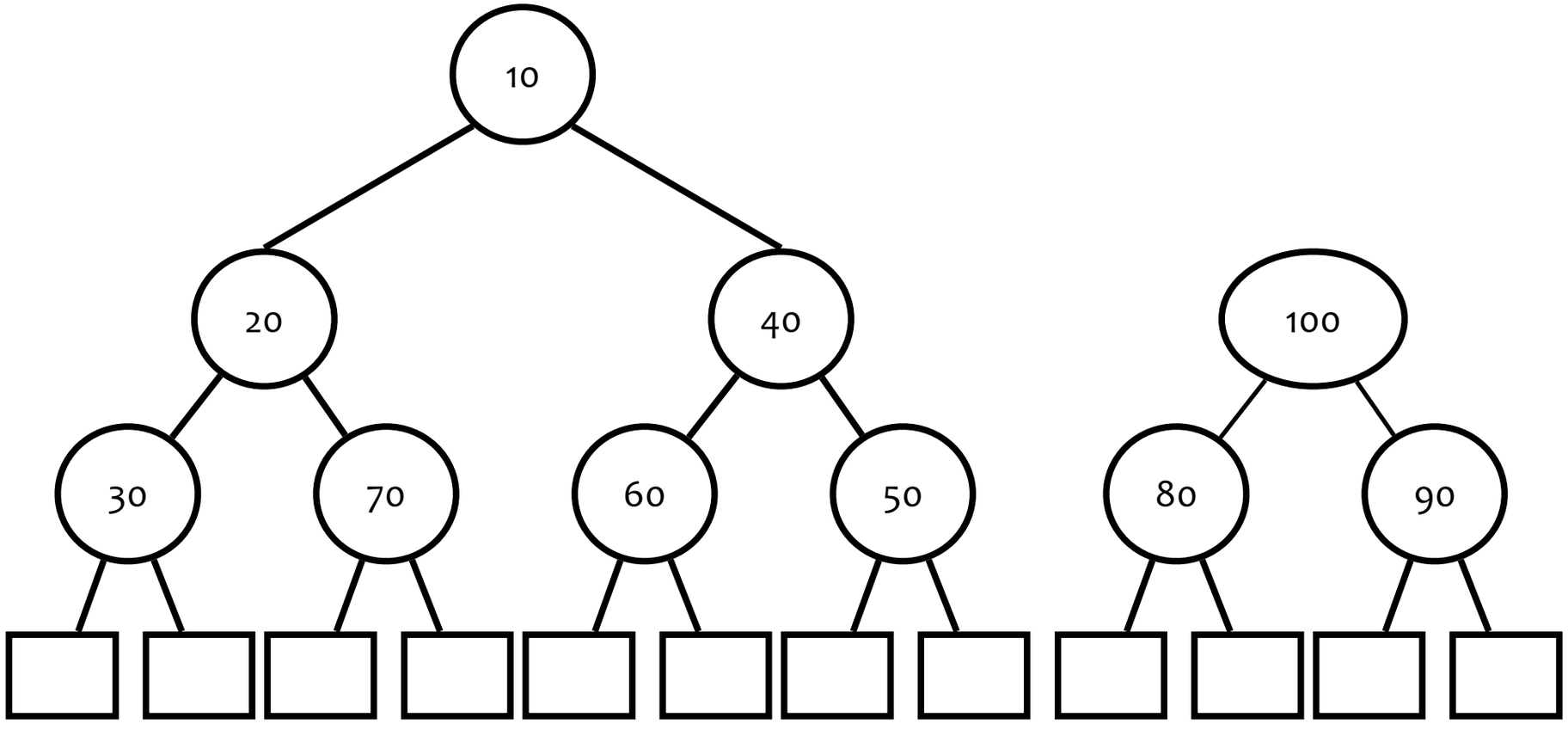
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



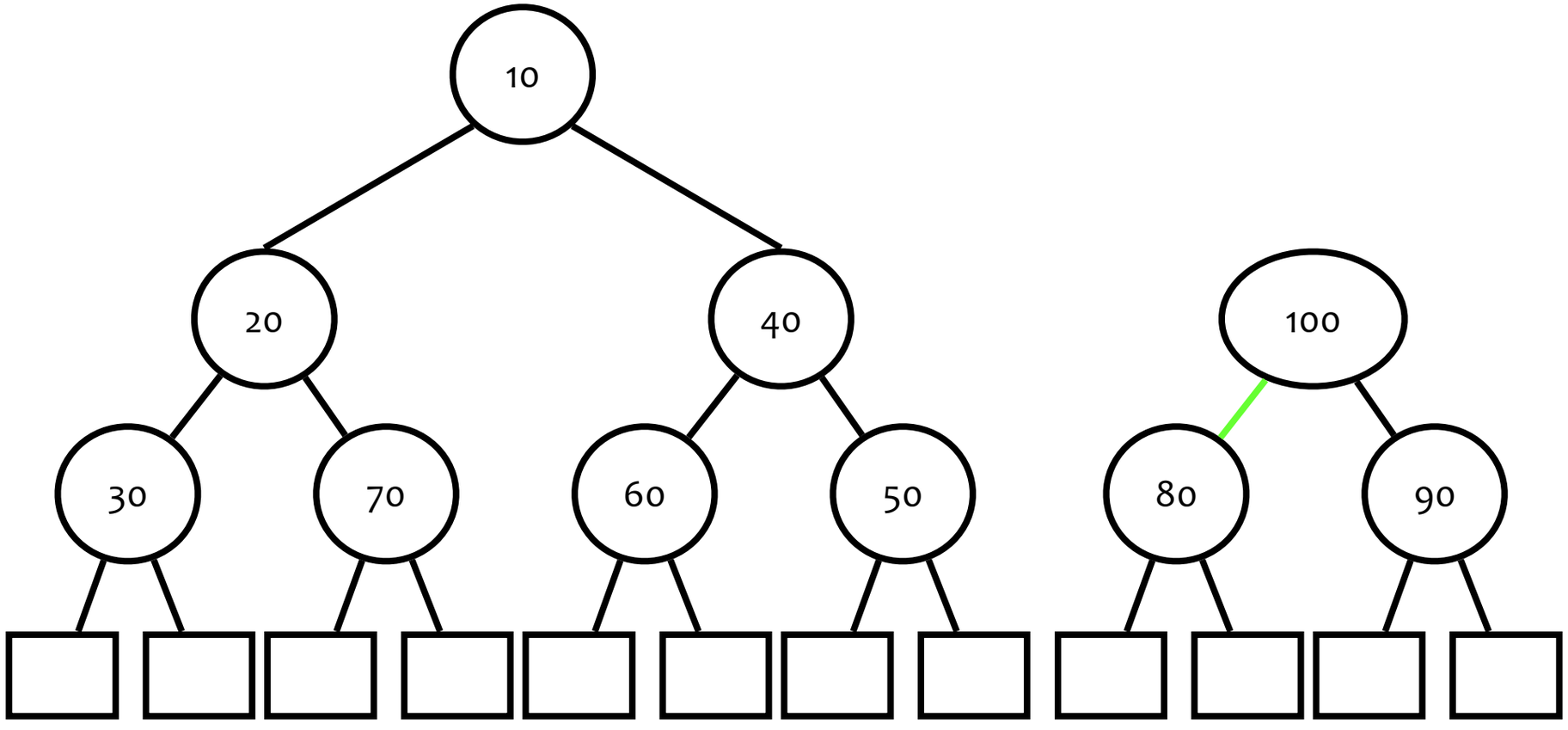
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



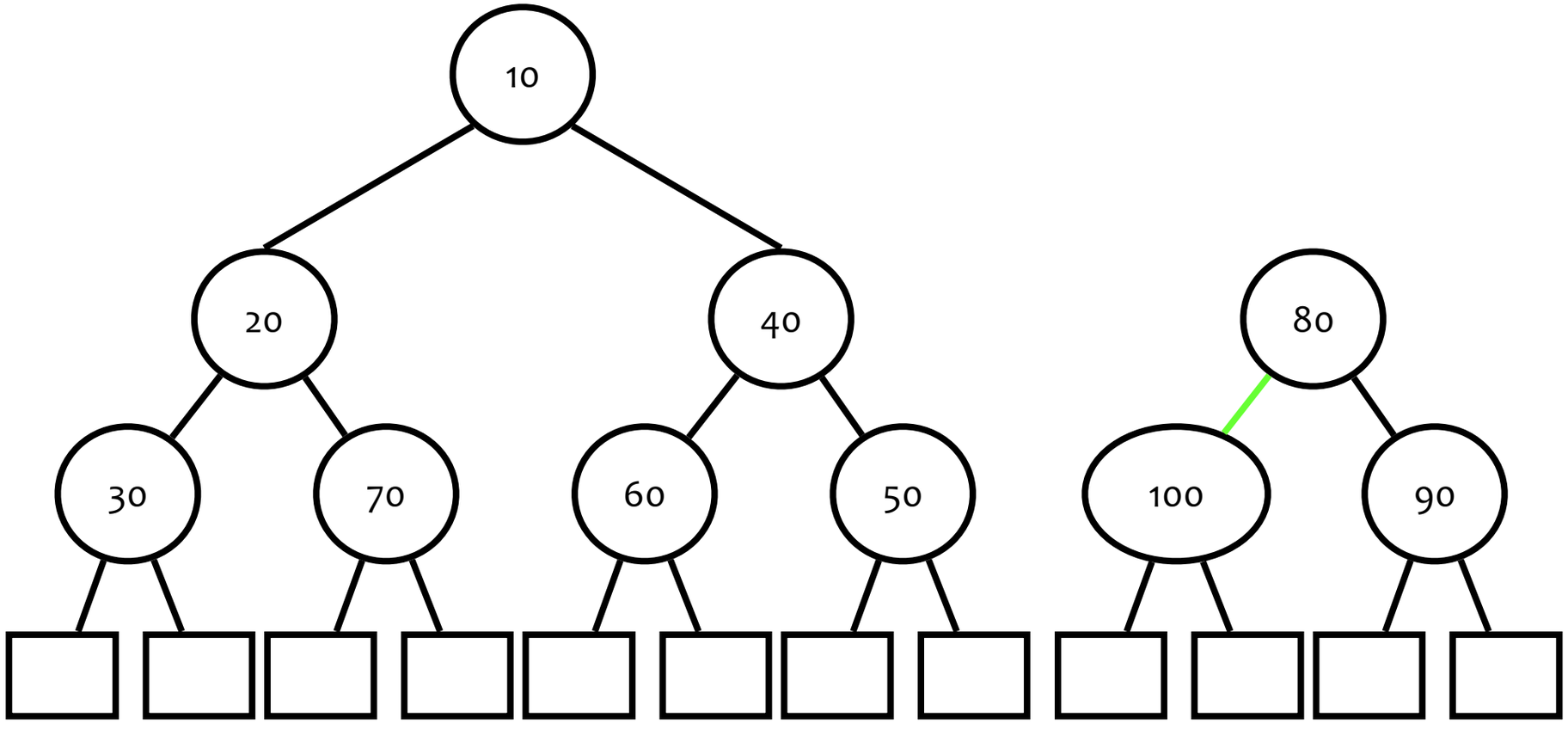
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



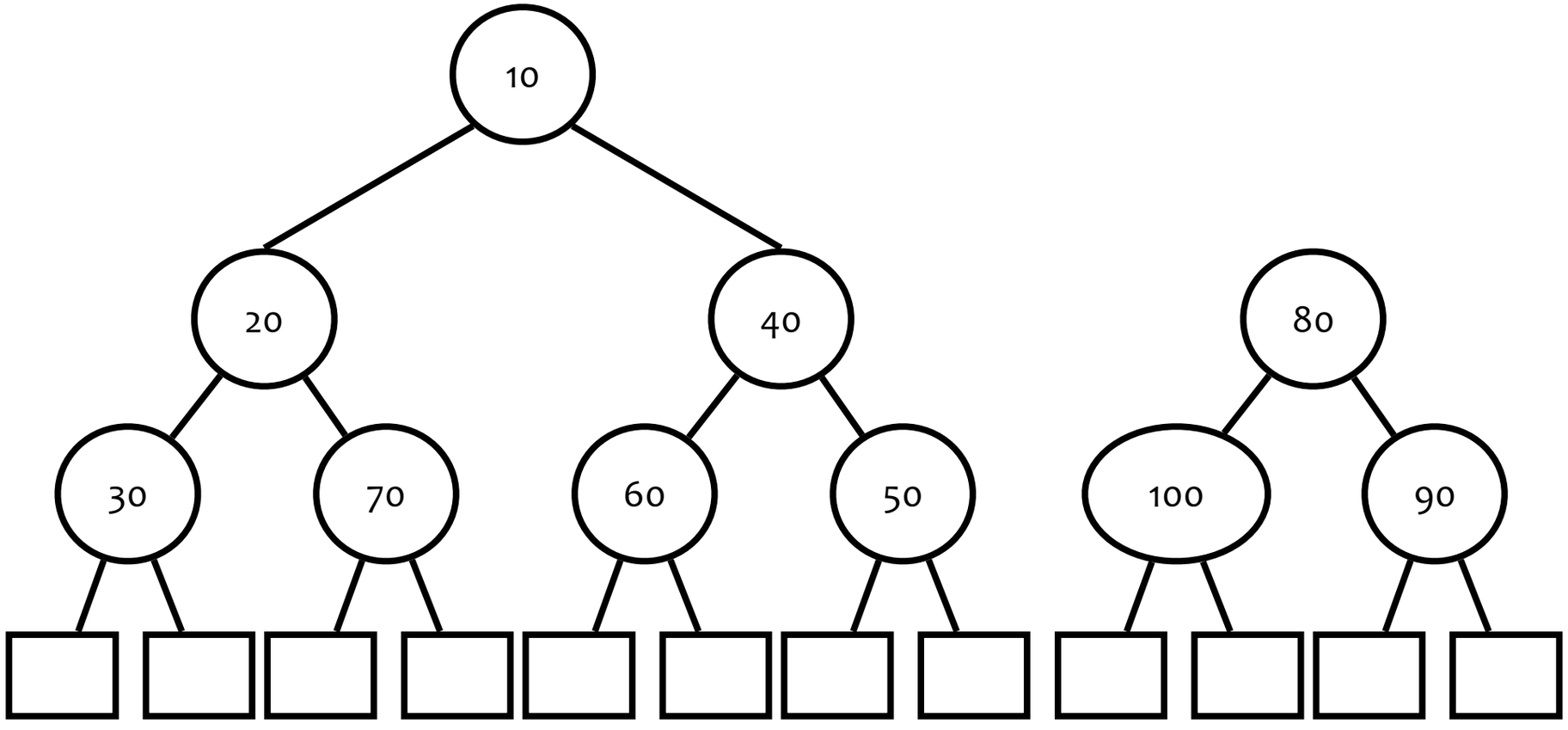
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



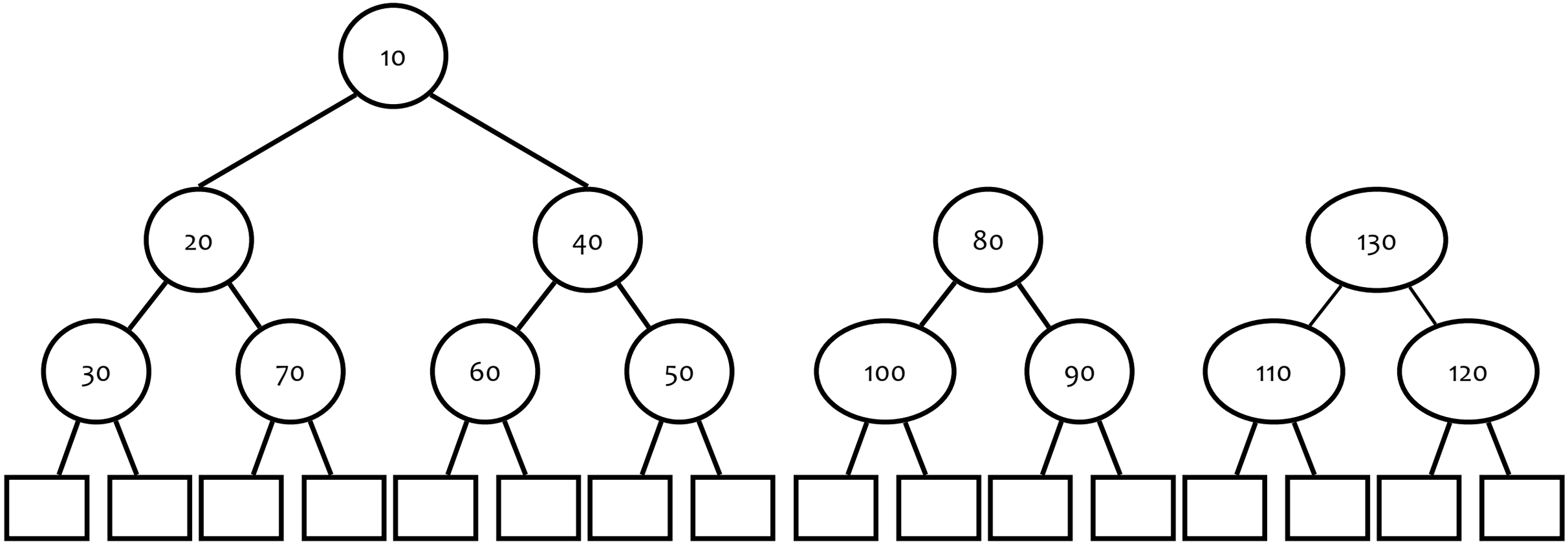
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



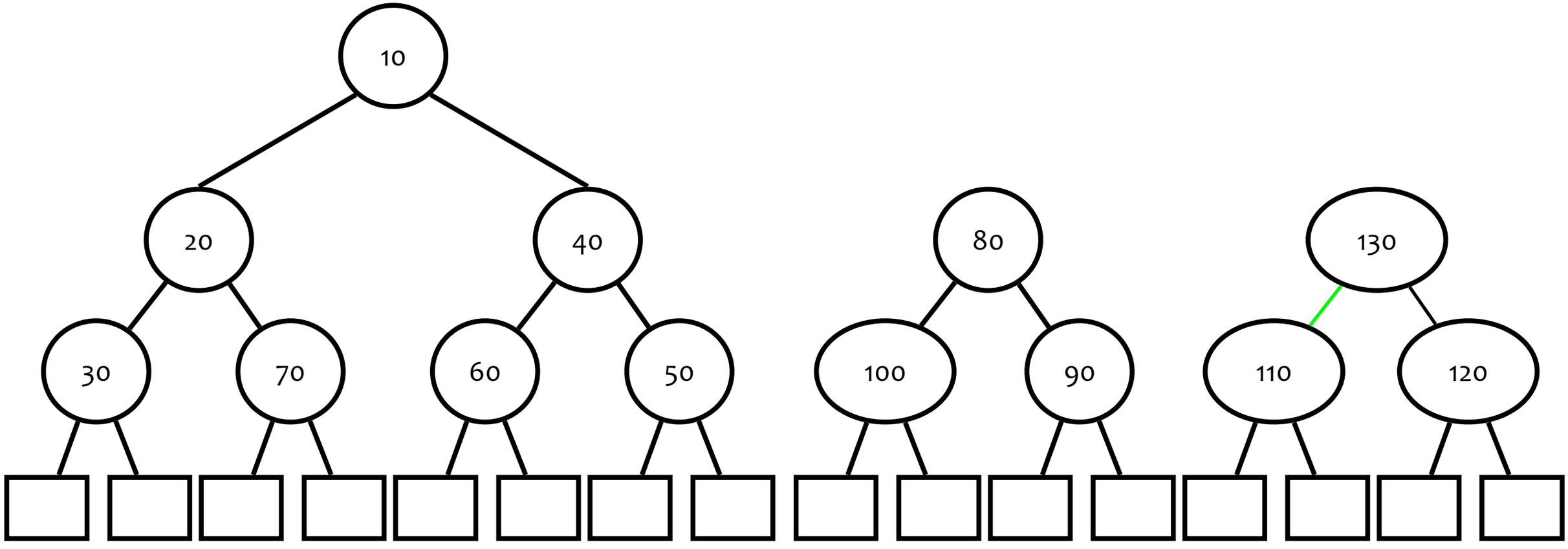
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



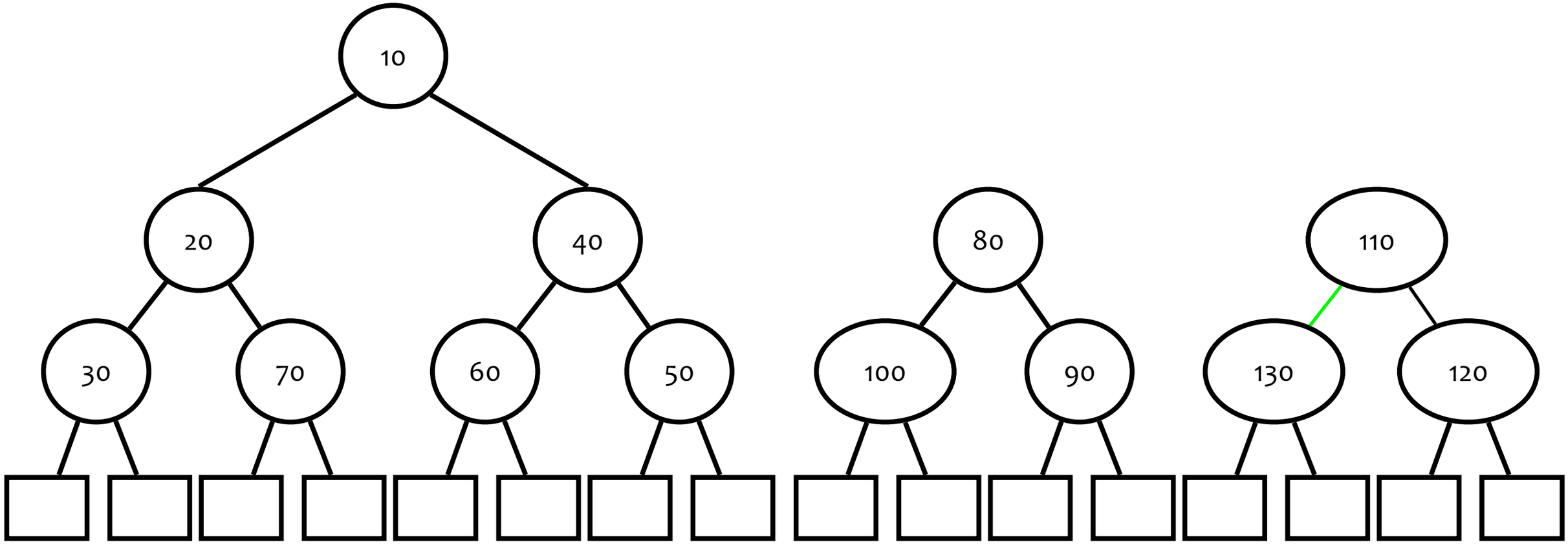
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



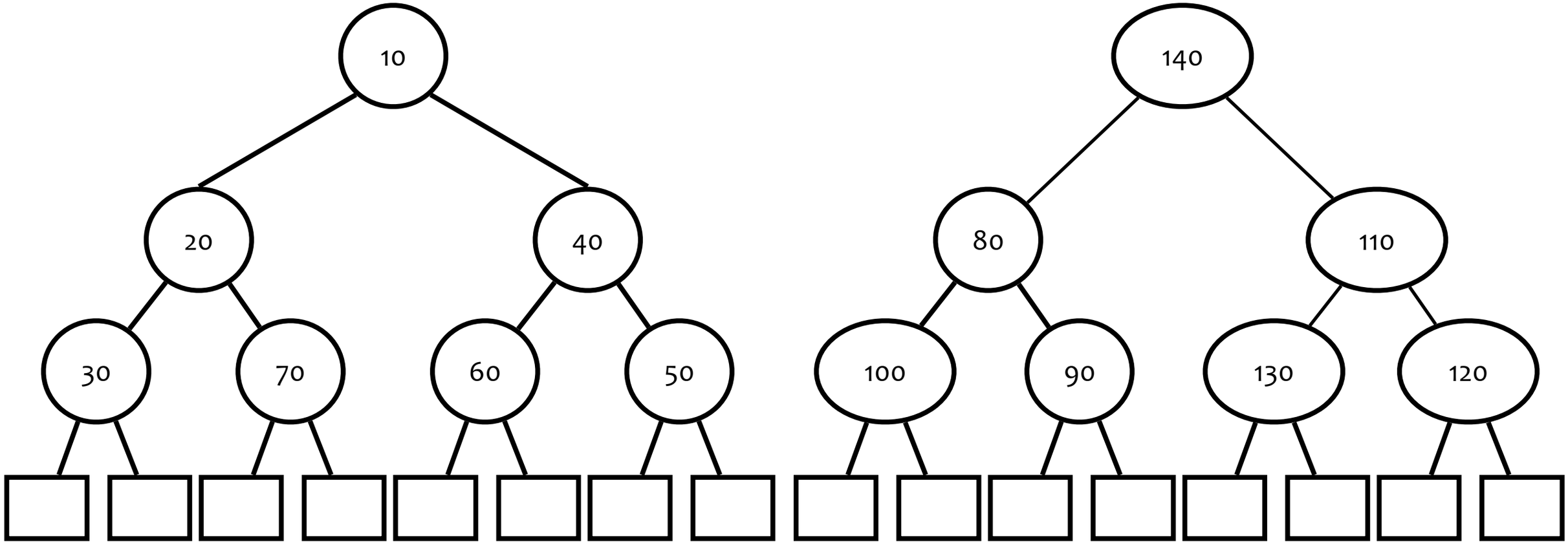
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



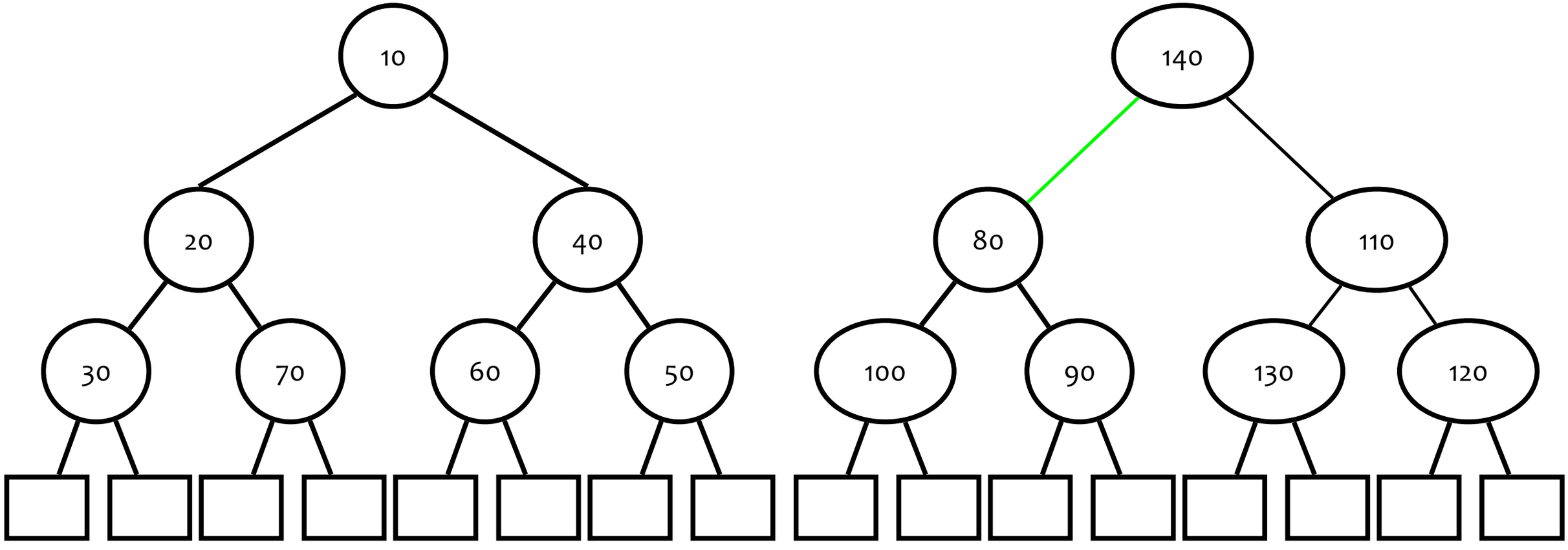
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



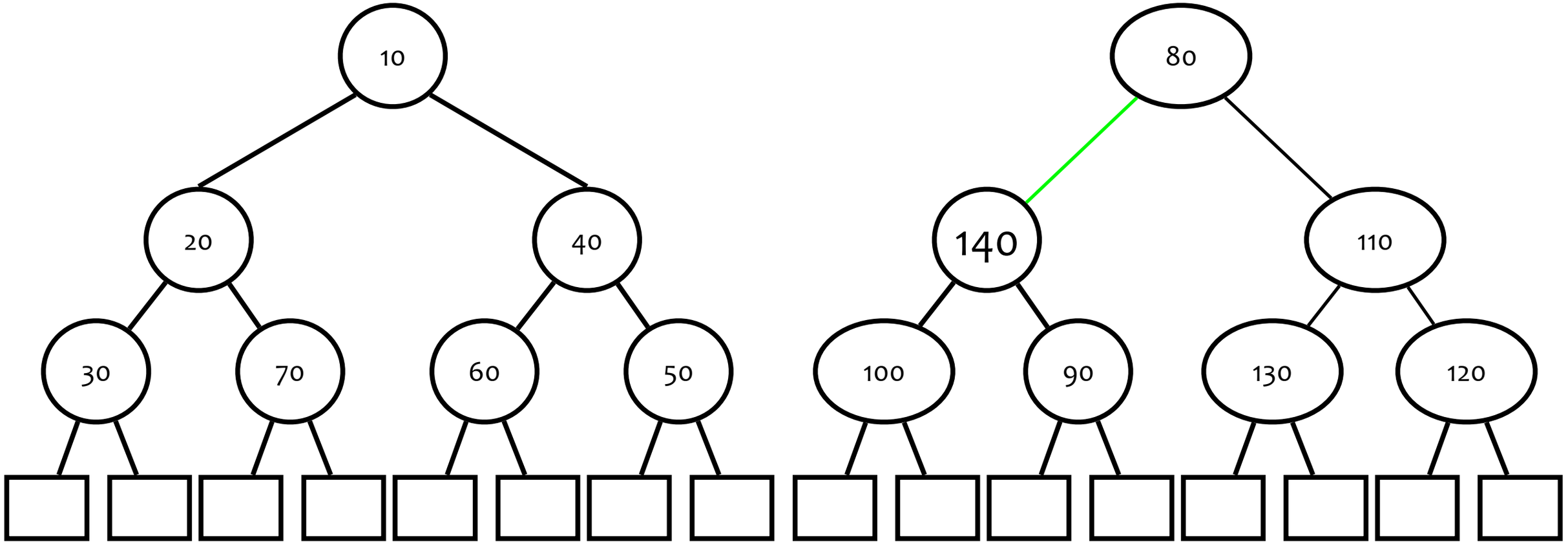
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



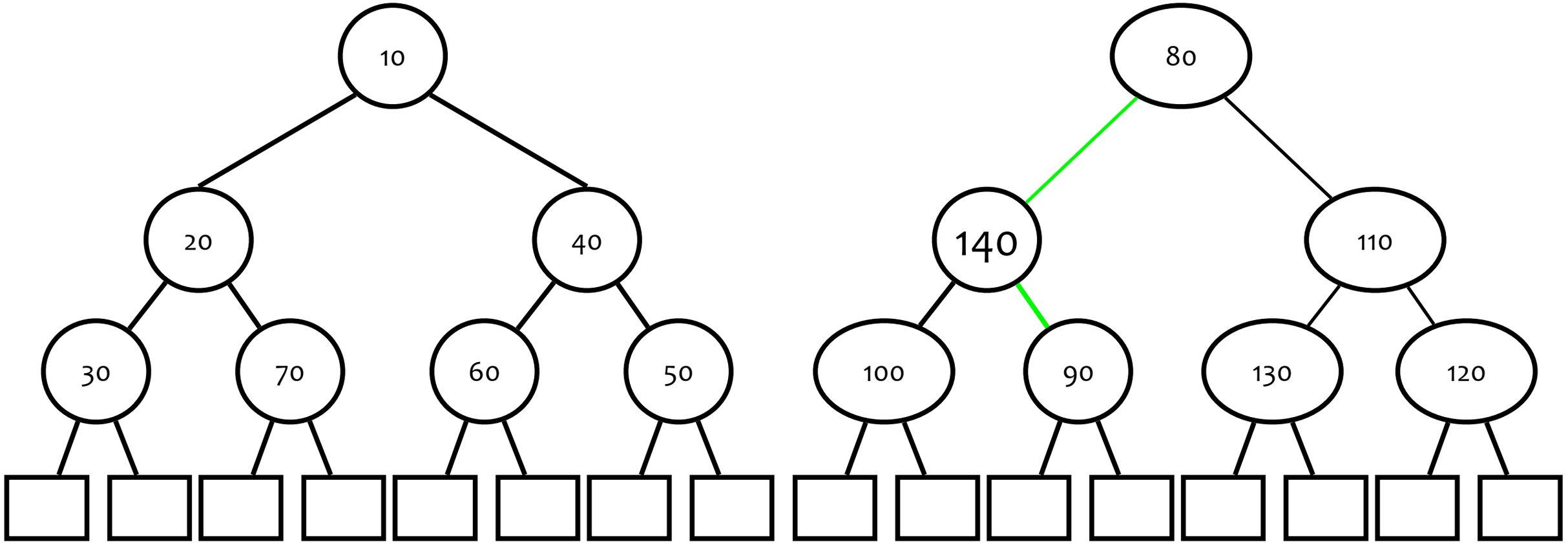
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



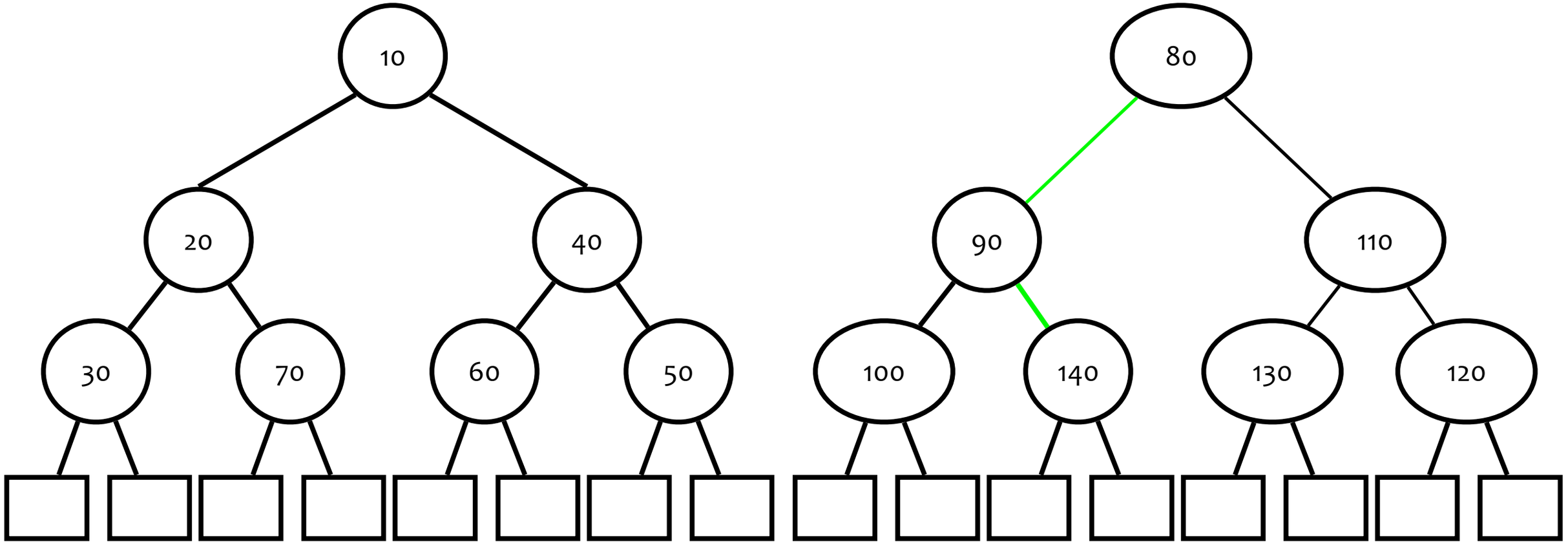
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



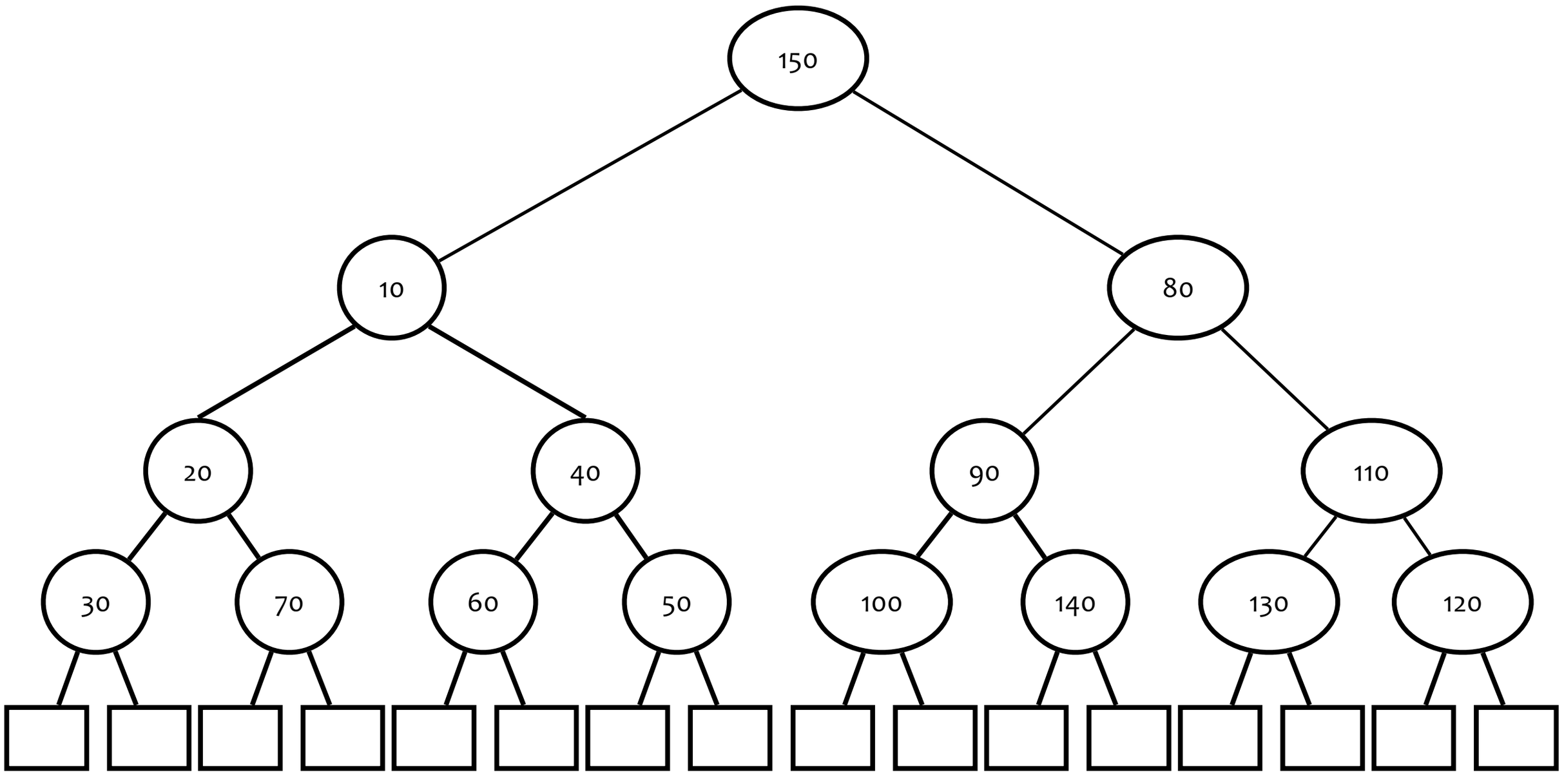
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



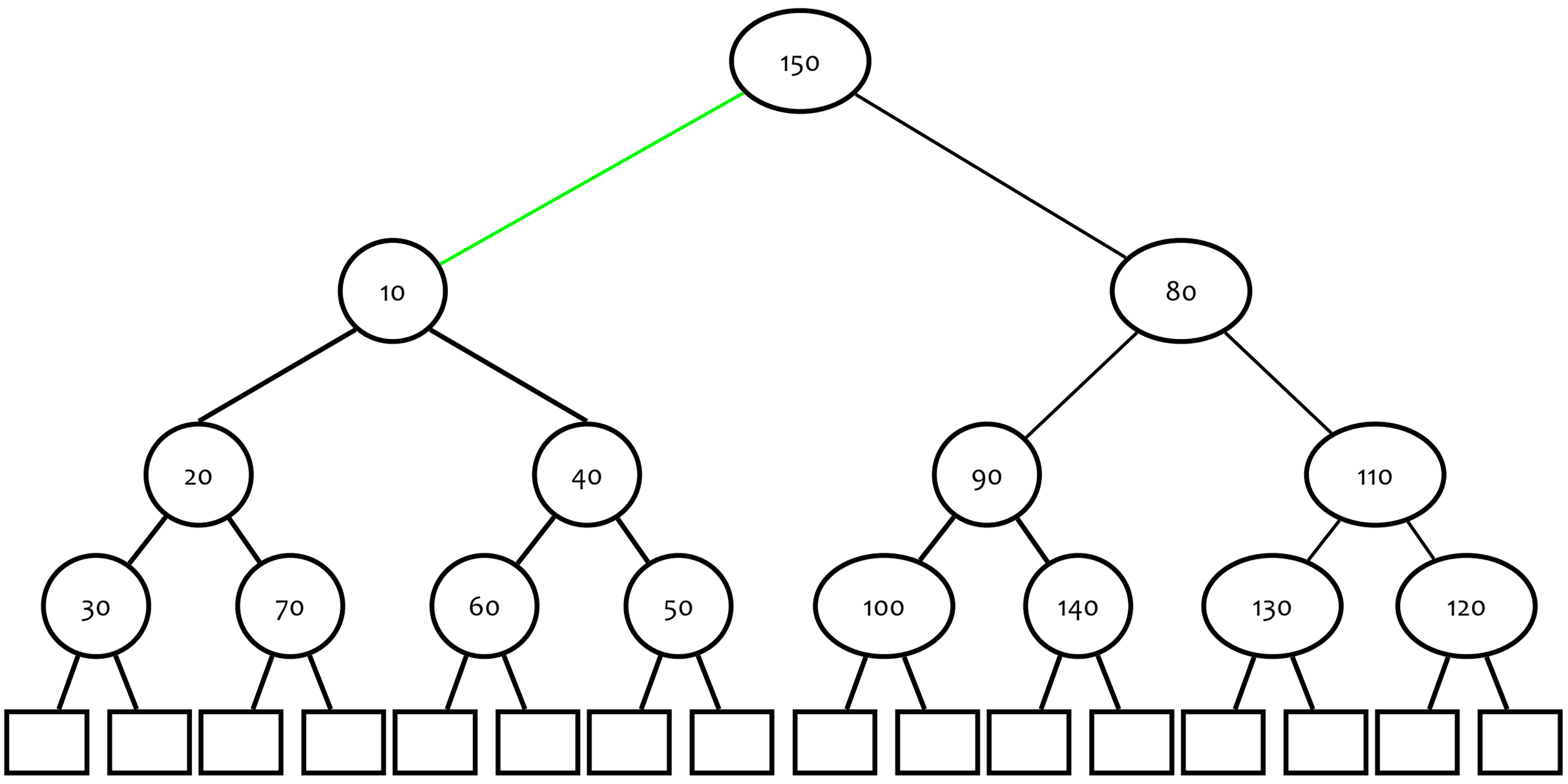
150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



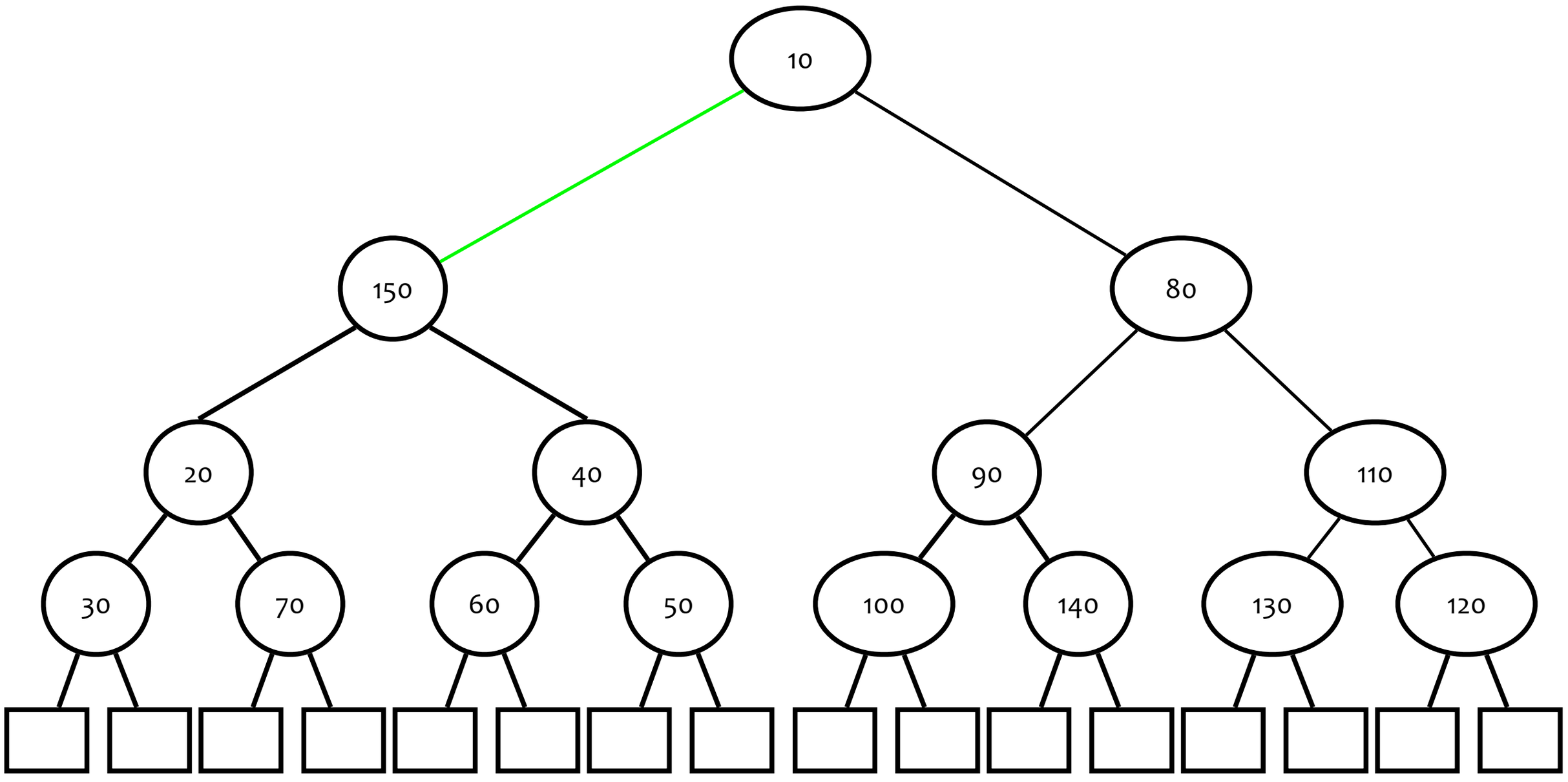
150	70	30	10	20	60	40	50	140	100	80	90	130	110	120



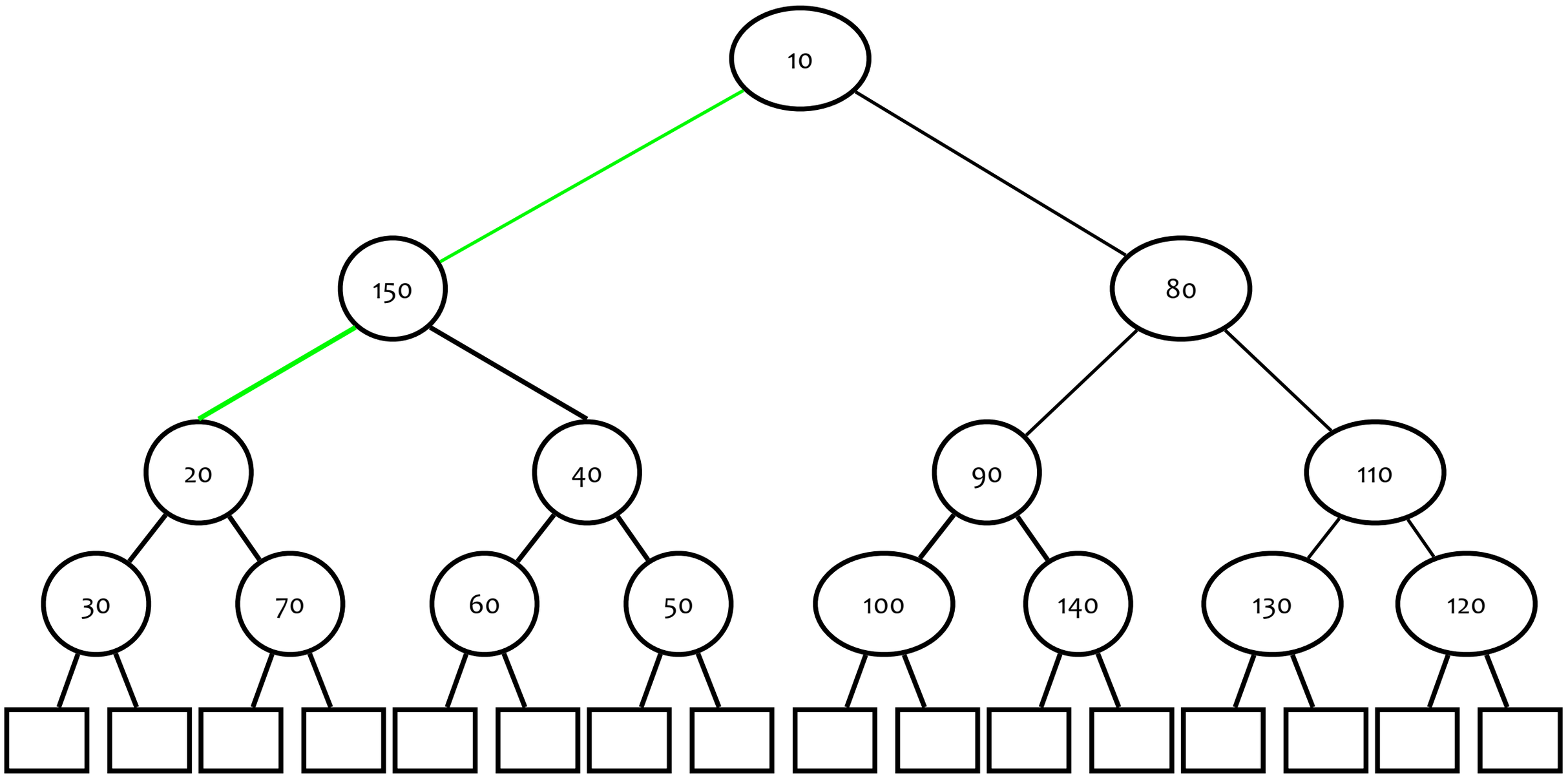
150 | 70 | 30 | 10 | 20 | 60 | 40 | 50 | 140 | 100 | 80 | 90 | 130 | 110 | 120



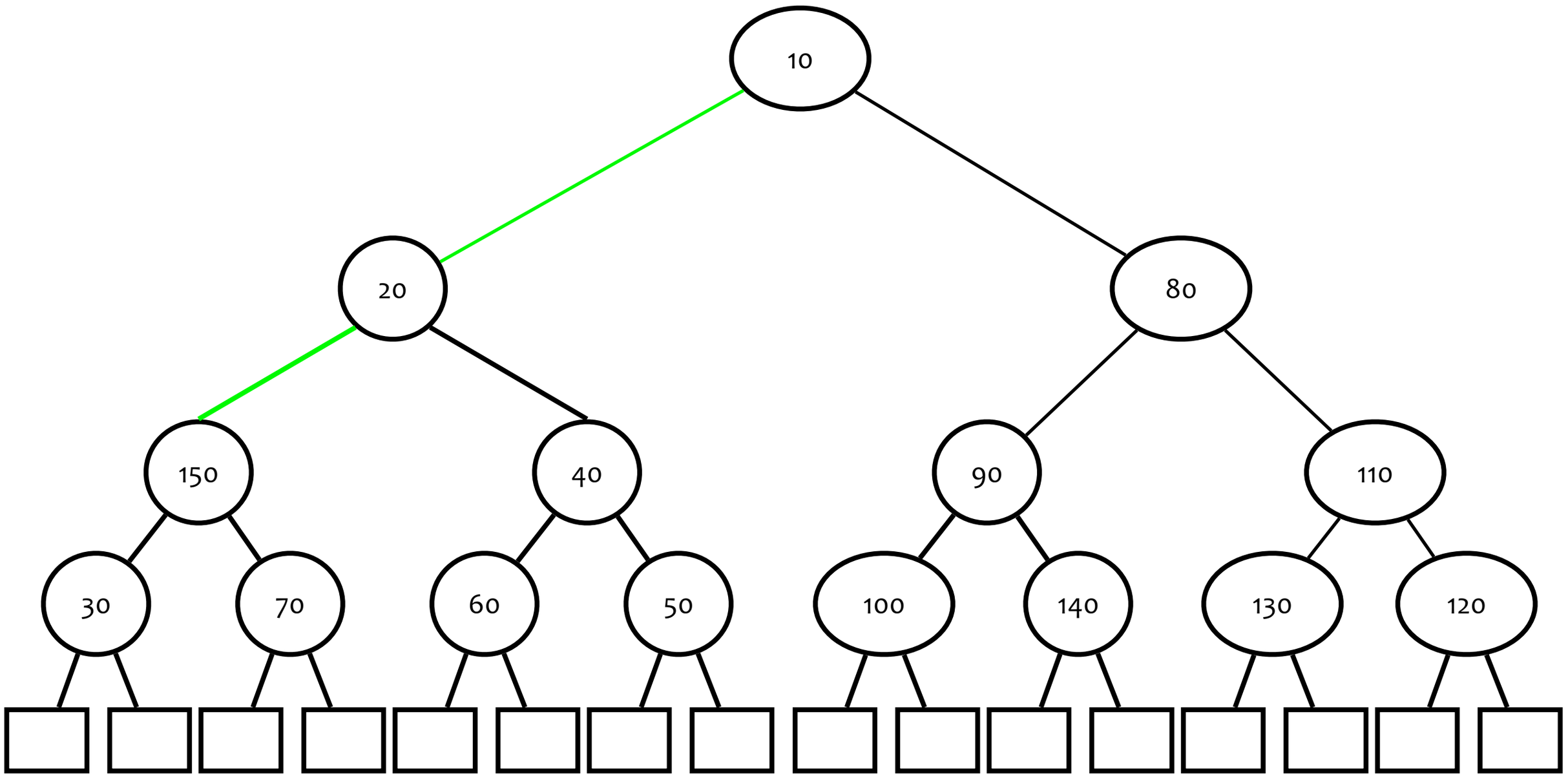
150	70	30	10	20	60	40	50	140	100	80	90	130	110	120



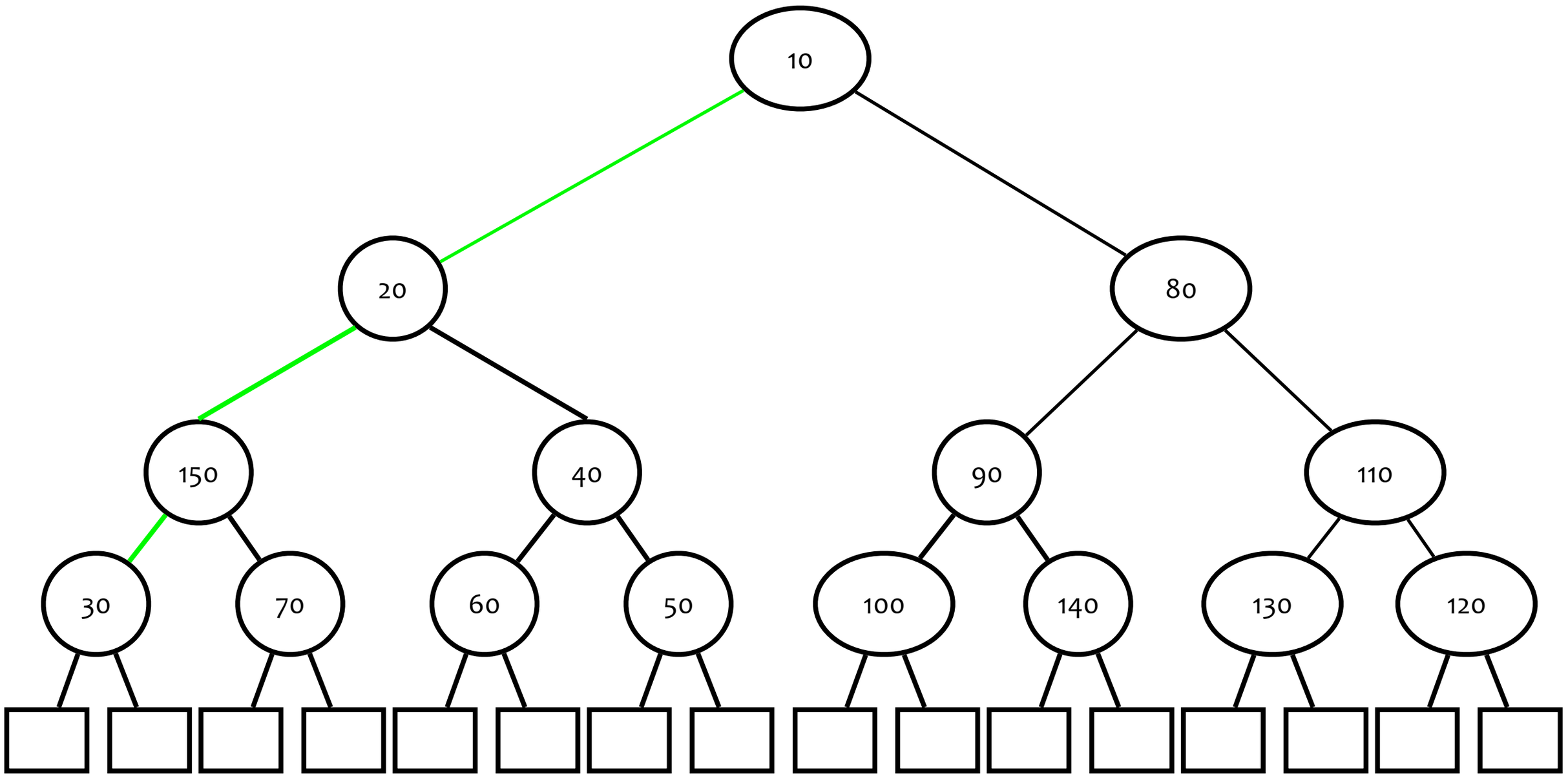
150	70	30	10	20	60	40	50	140	100	80	90	130	110	120



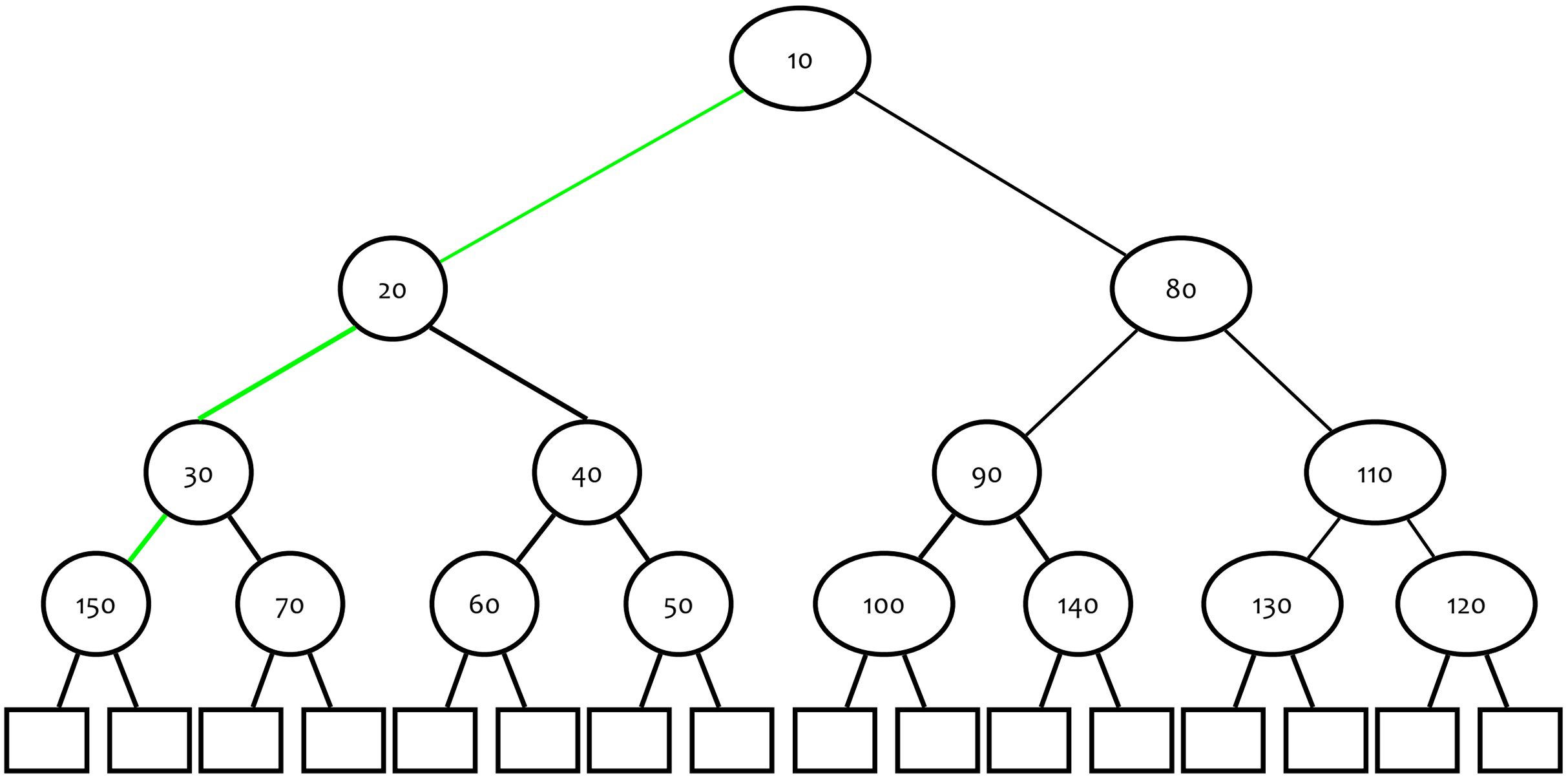
150 | 70 | 30 | 10 | 20 | 60 | 40 | 50 | 140 | 100 | 80 | 90 | 130 | 110 | 120



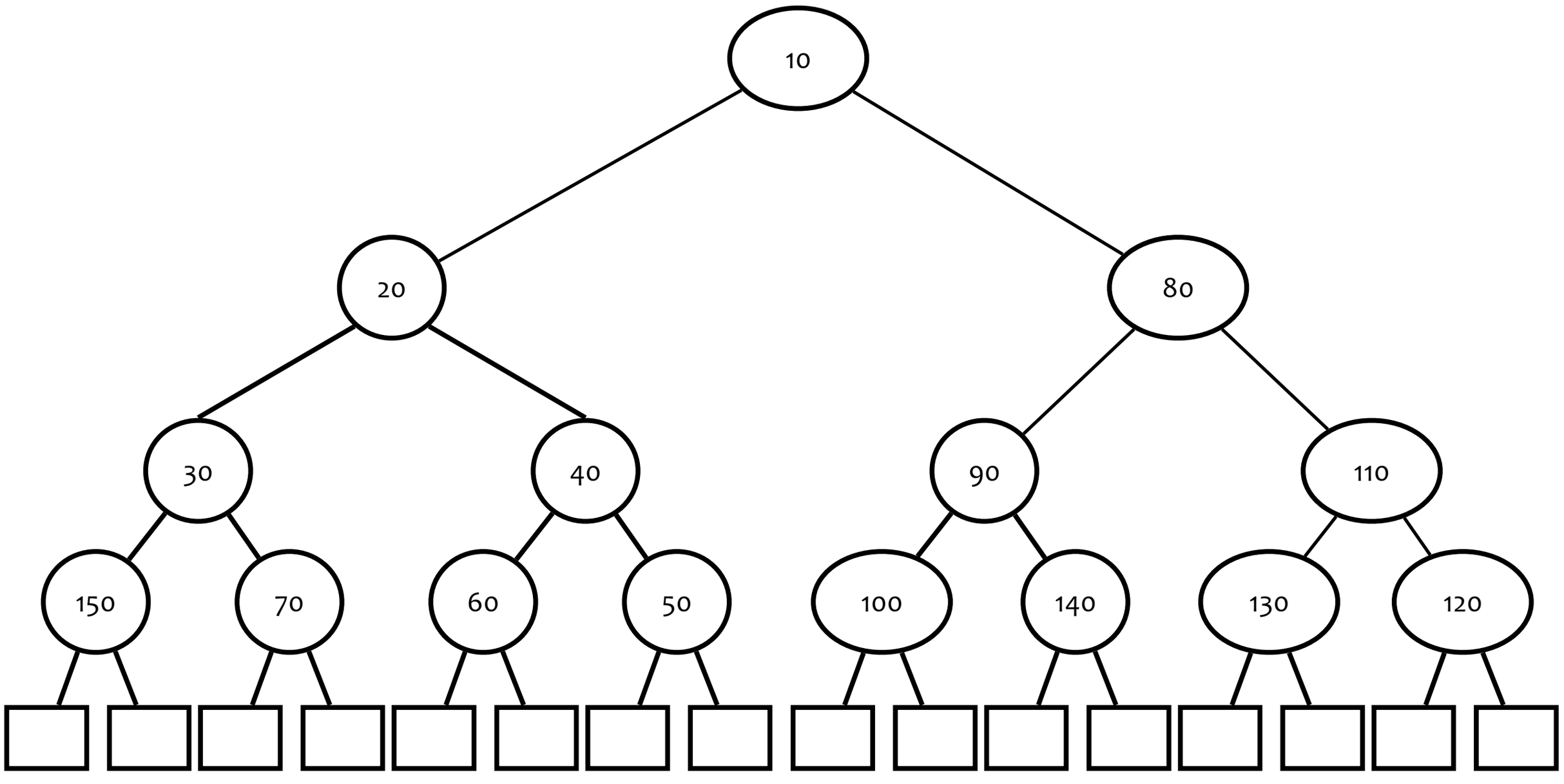
150	70	30	10	20	60	40	50	140	100	80	90	130	110	120



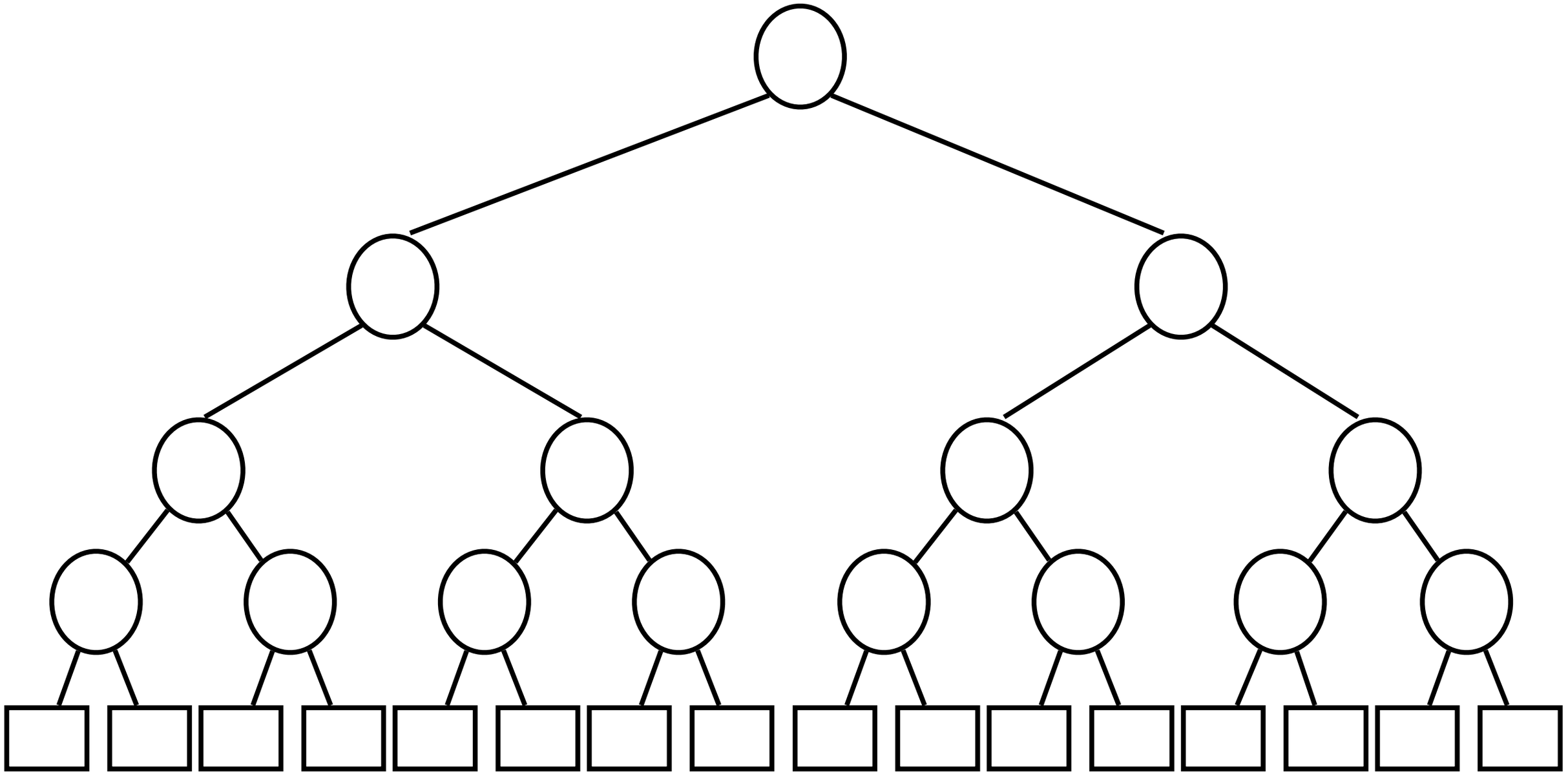
150	70	30	10	20	60	40	50	140	100	80	90	130	110	120



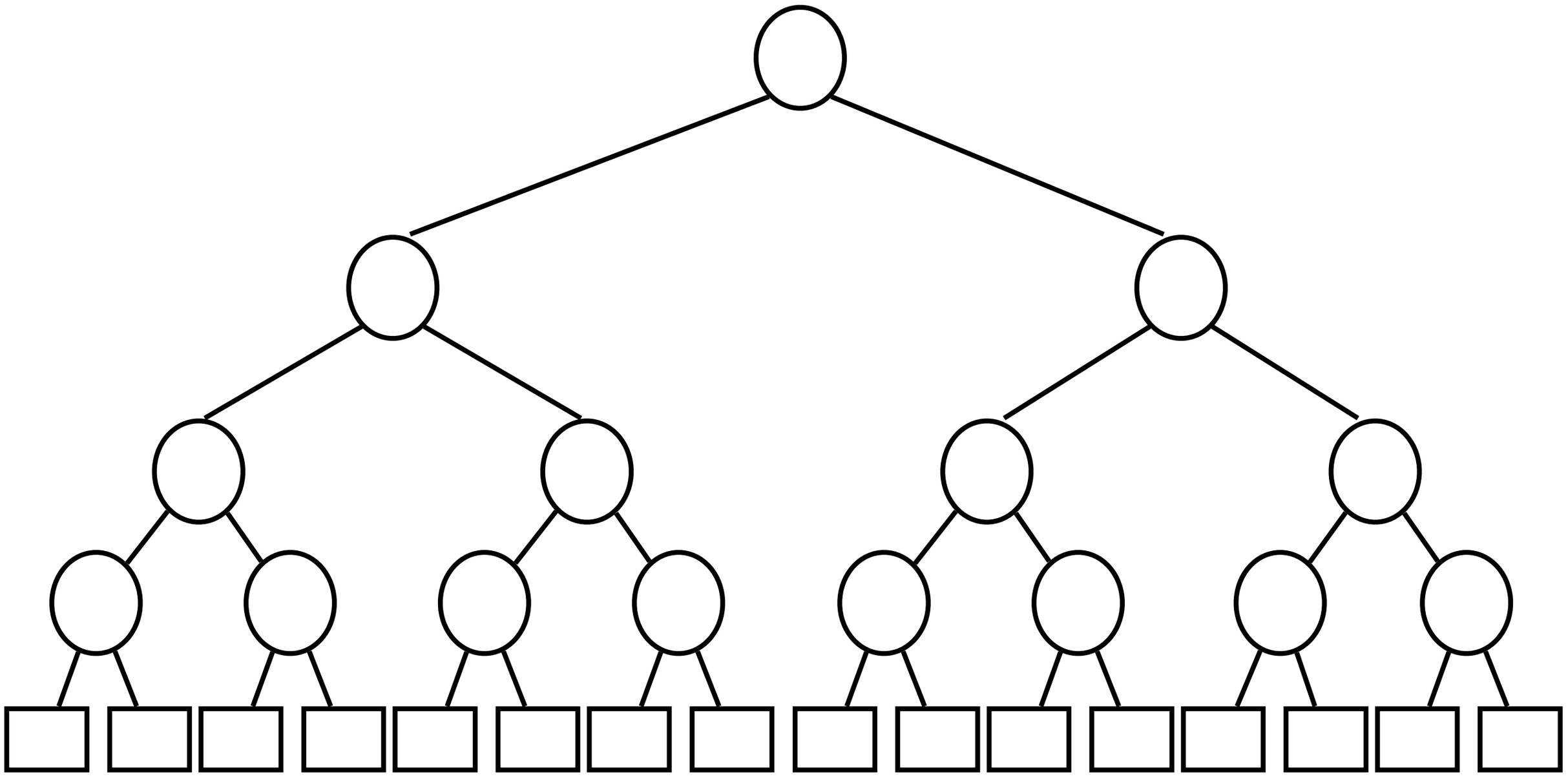
150	70	30	10	20	60	40	50	140	100	80	90	130	110	120



Did we really insert all n elements in O(n) time??



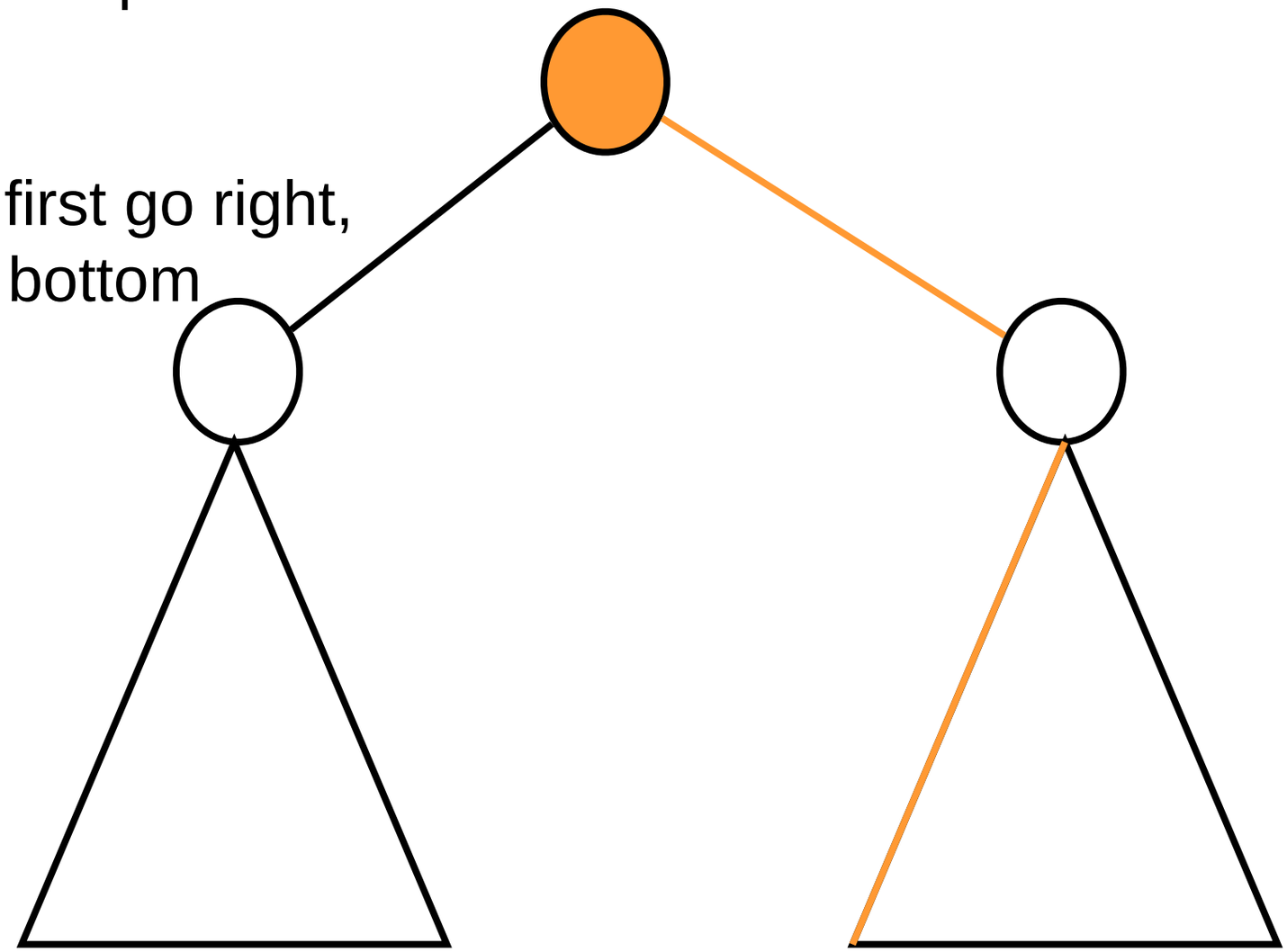
We show: max# bubble down ops < # heap edges



Proof idea

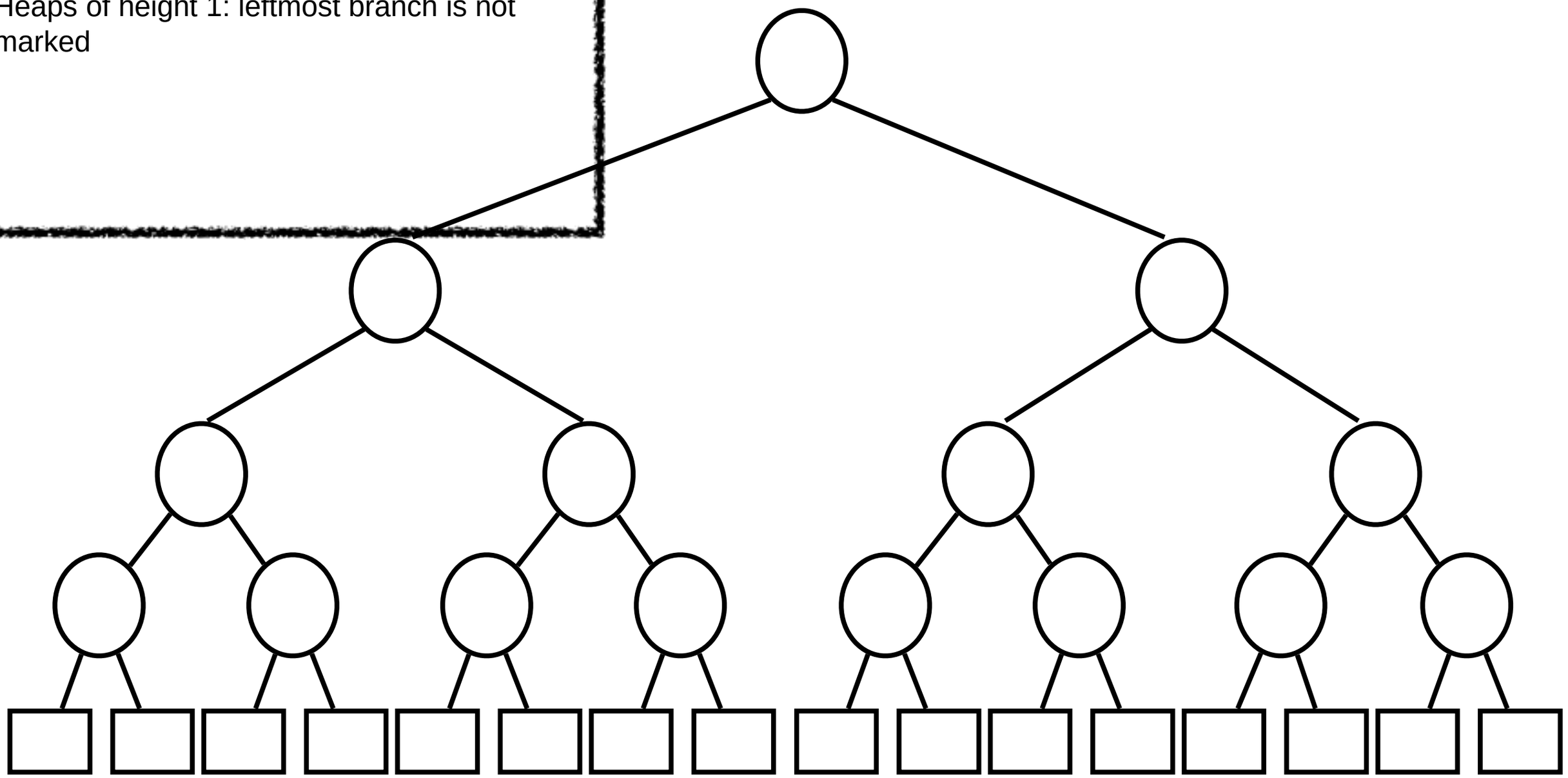
For each new node joining two heaps: mark path of maximum number of bubble-down operations

For marking: first go right, then left until bottom

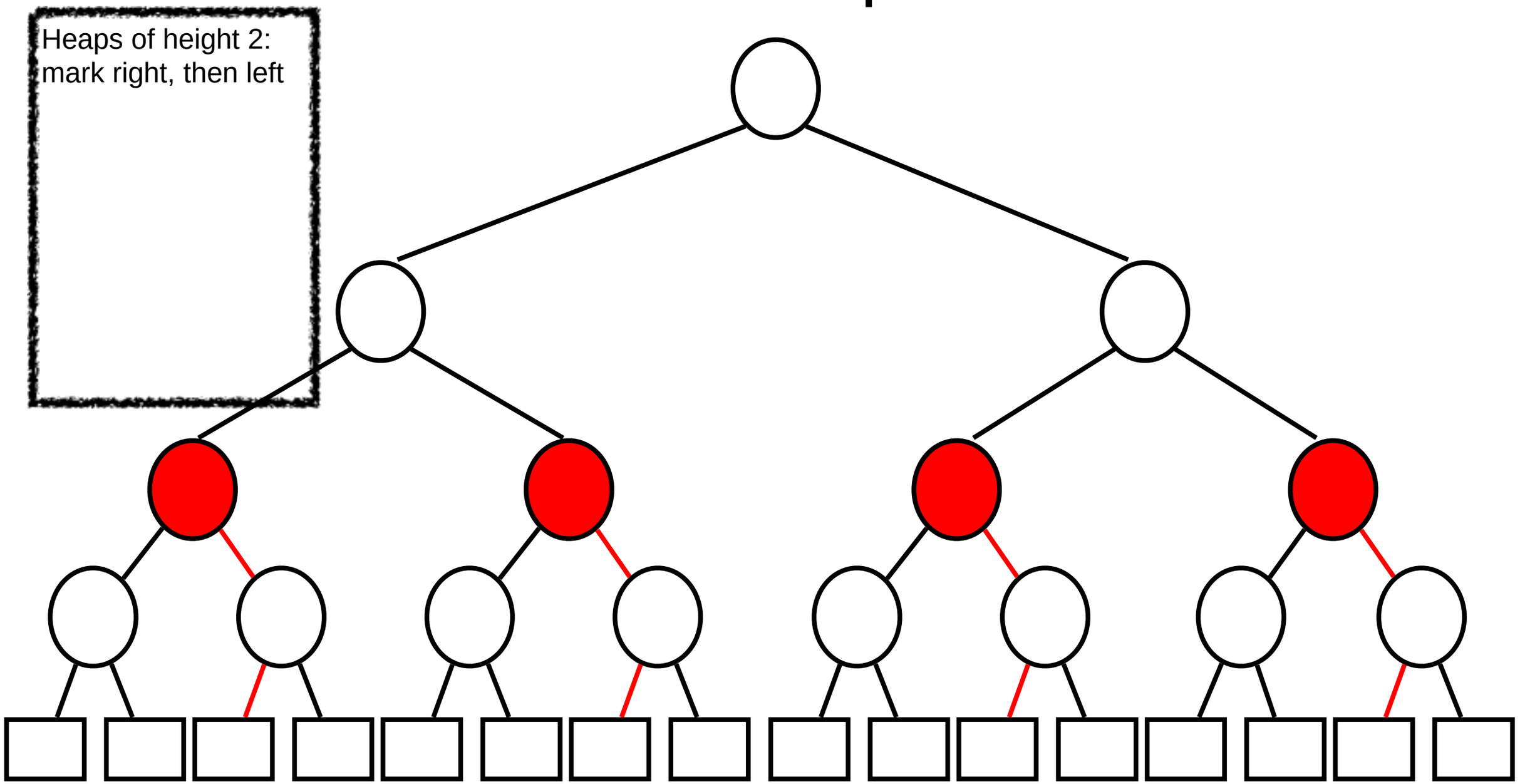


For each new node joining two heaps: mark path with maximum number of bubble-down operations

Heaps of height 1: leftmost branch is not marked



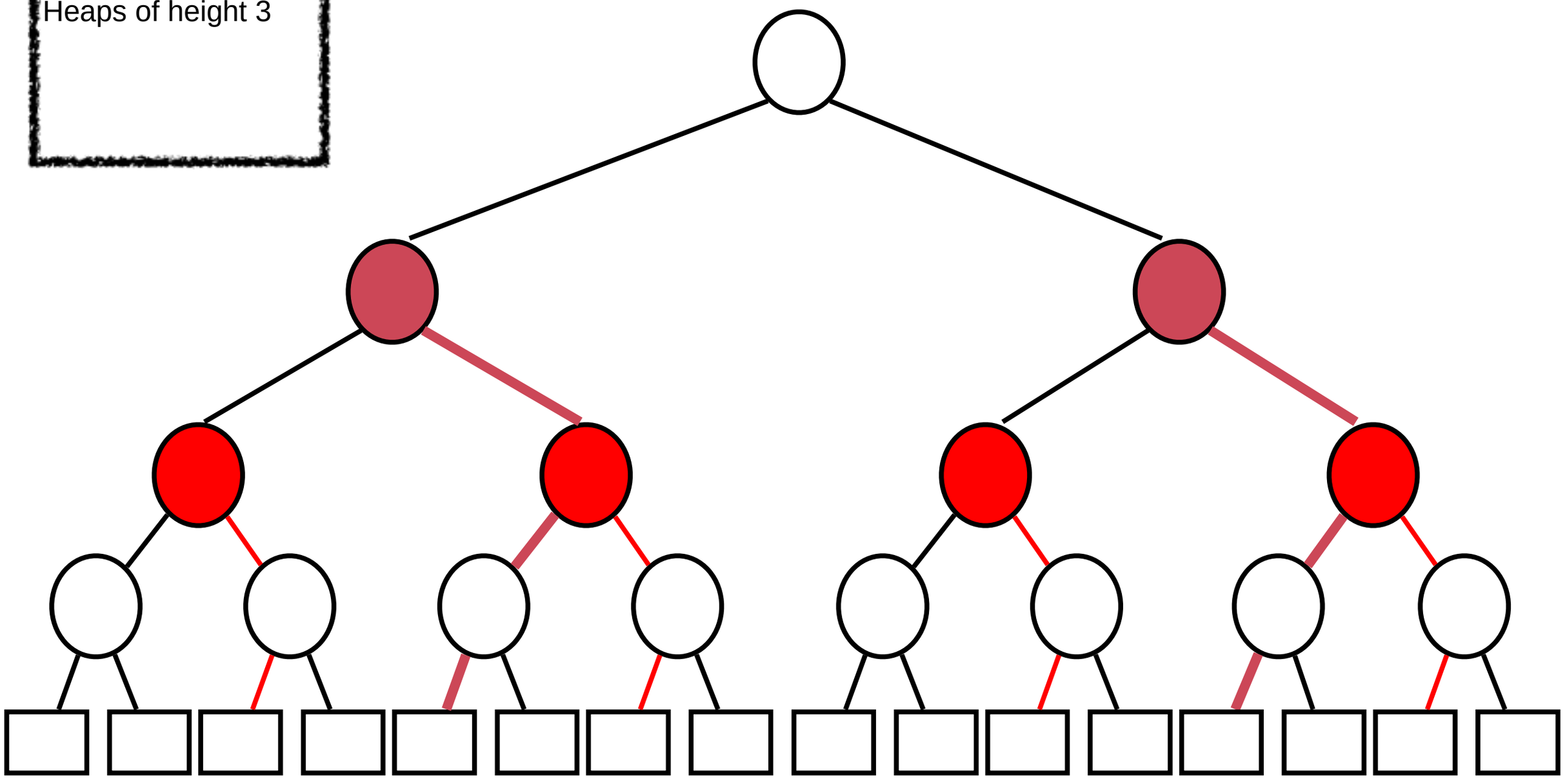
For each new node joining two heaps: mark path with maximum number of bubble-down operations



For each height-2 heap, leftmost branch not marked

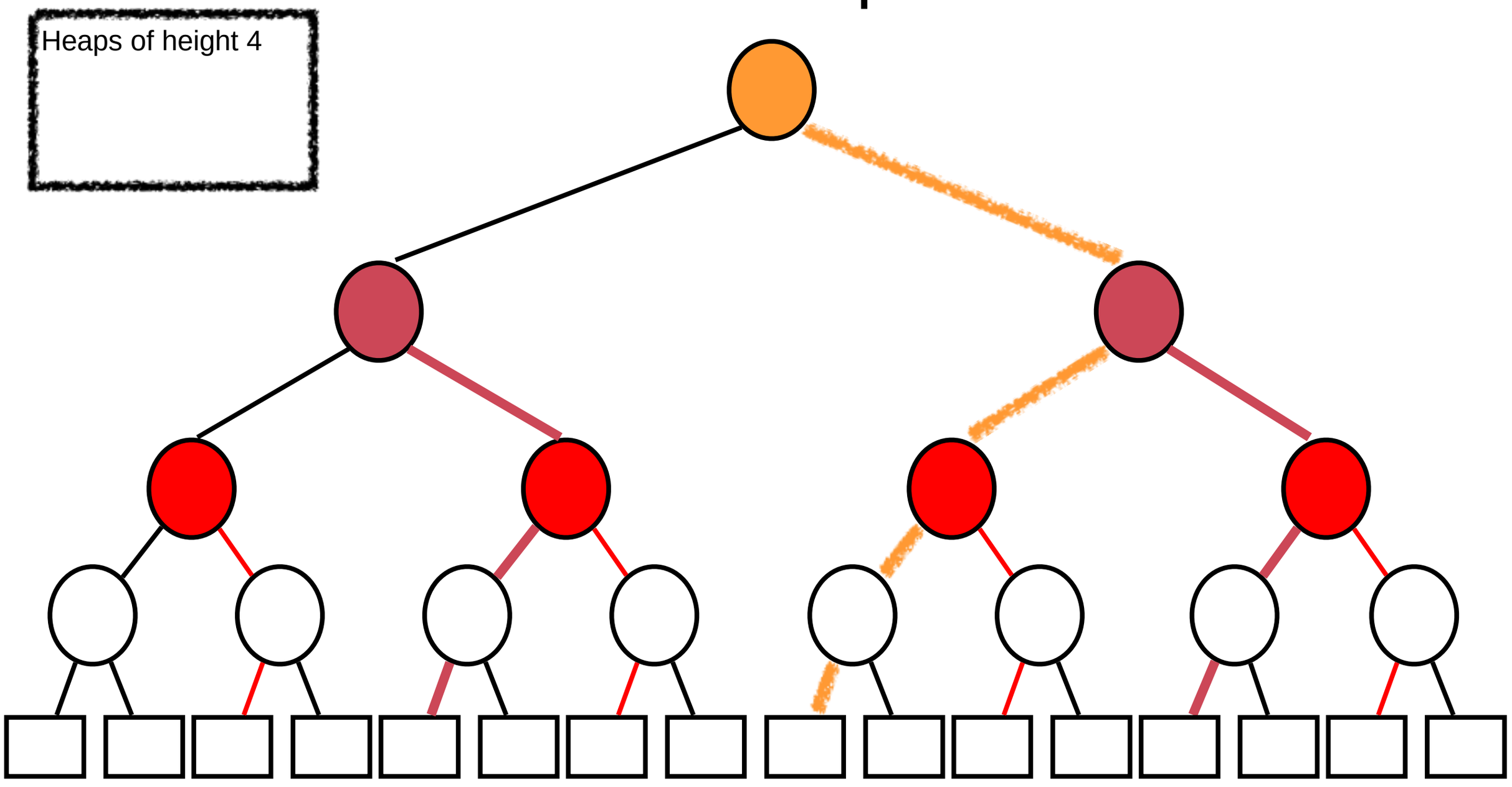
For each new node joining two heaps: mark path with maximum number of bubble-down operations

Heaps of height 3



For each height-3 heap, leftmost branch not marked

For each new node joining two heaps: mark path with maximum number of bubble-down operations



For height-4 heap, leftmost branch not marked

Inductive argument: marking procedure will never mark all edges in heap, since the leftmost branch is never marked

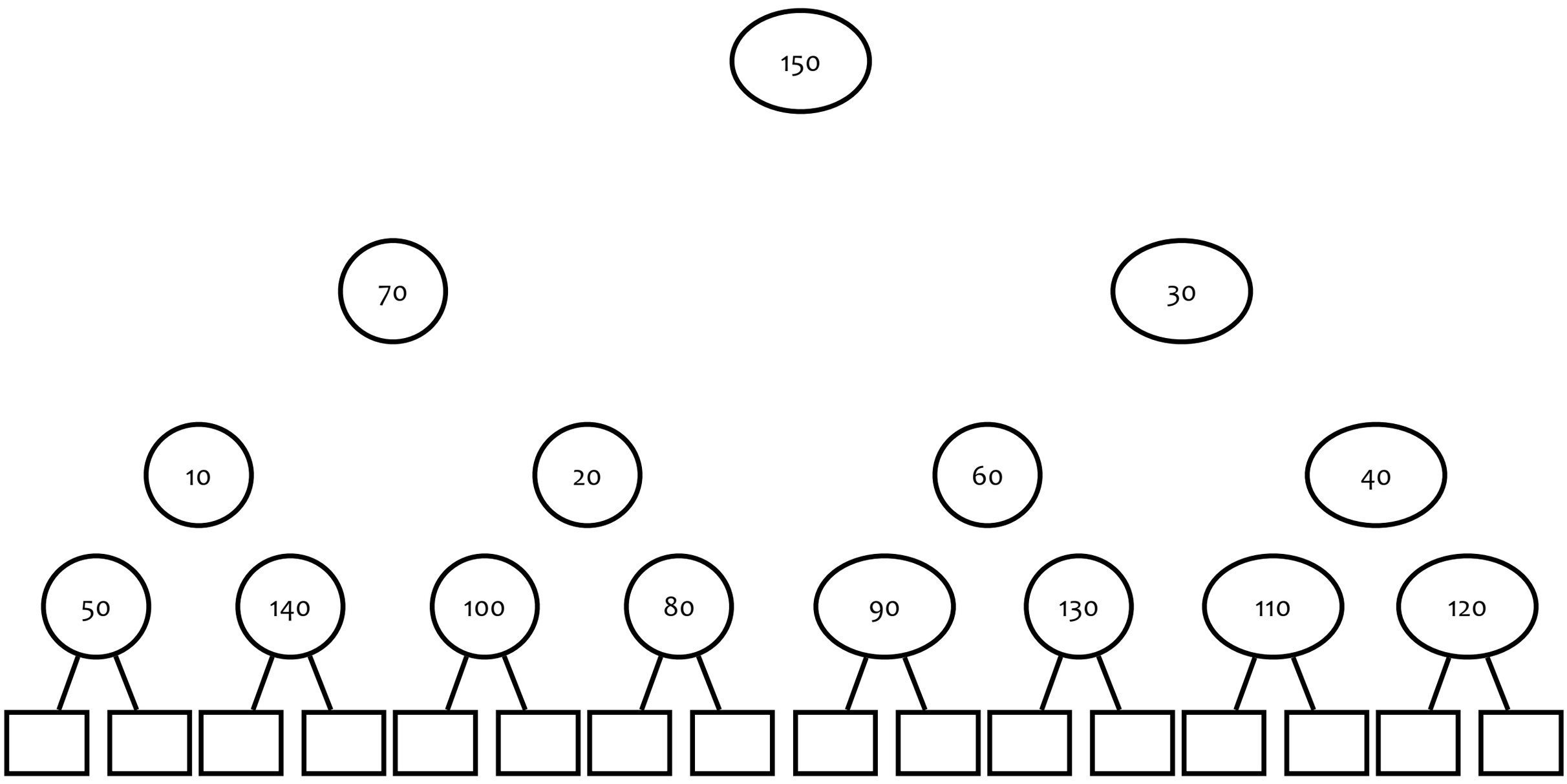
- Note: leftmost branch in height-h heap: not marked
- When joining 2 heaps of height h to heap of height $h + 1$: new edges to be marked are
  - edge joining new node and right heap of height h, and
  - edges on left path in the right heap of height h
- We conclude: leftmost branch in height $(h+1)$ heap is not marked

Build Heap In-place

Algorithm buildHeap(A,n):
    for $i \leftarrow \lfloor n/2 \rfloor$ **to** 1 **do**
        downHeap(A,i)

Algorithm downHeap(A,i):
    $l \leftarrow 2i$
    $r \leftarrow 2i + 1$
    if $l \leq n \wedge A[l] < A[i]$ **then**
        $min \leftarrow l$
    else
        $min \leftarrow i$
    if $r \leq n \wedge A[r] < A[min]$ **then**
        $min \leftarrow r$
    if $i \neq min$ **then**
        swap(i,min)
        downHeap(A,min)

150 70 30 10 20 60 40 50 140 100 80 90 130 110 120



10 20 30 50 80 60 40 70 140 100 150 90 130 110 120