Position Heaps

Problem

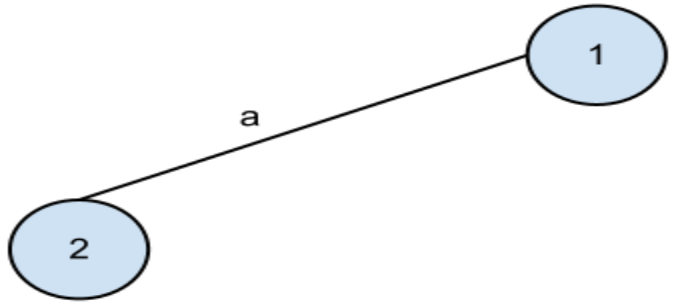
We want to search for all occurrences of pattern P in text T.

m = length of P
n = length of T
k = # occurrences of P in T

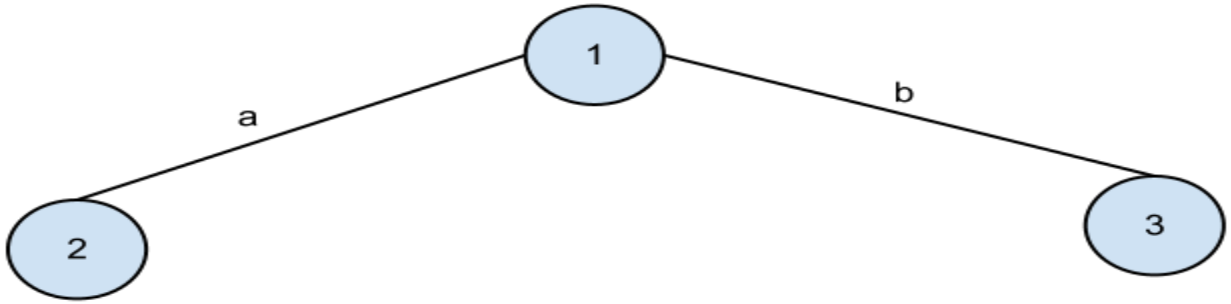
We are allowed to preprocess T.

13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b

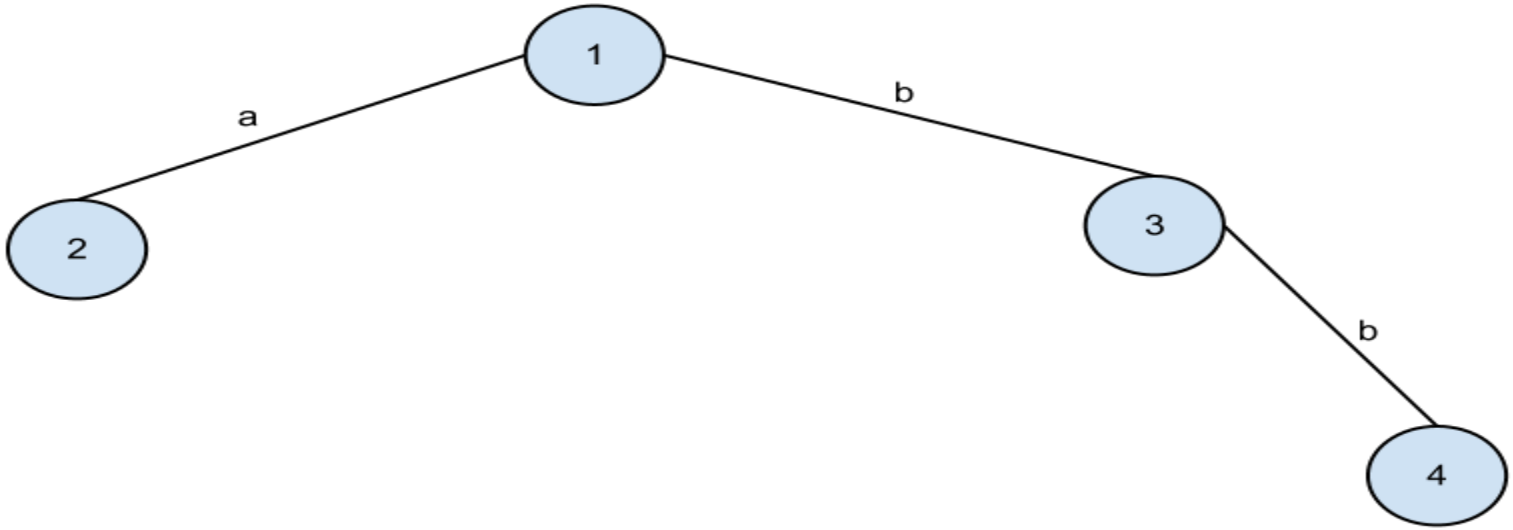
13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



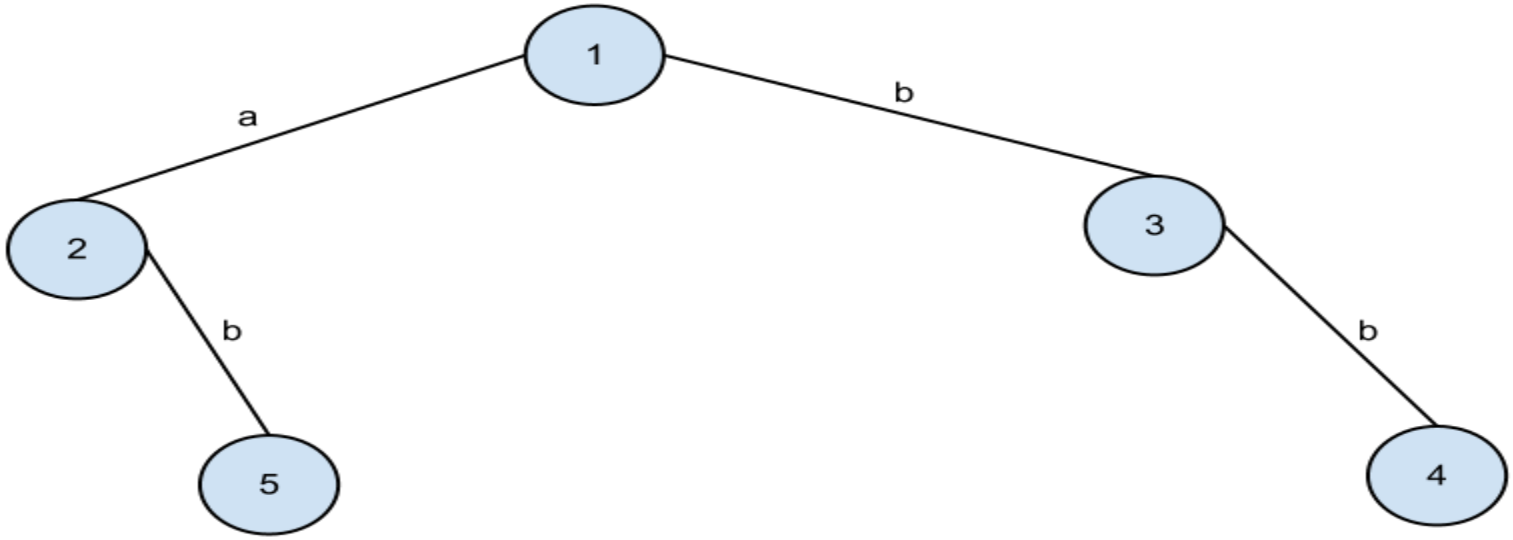
13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



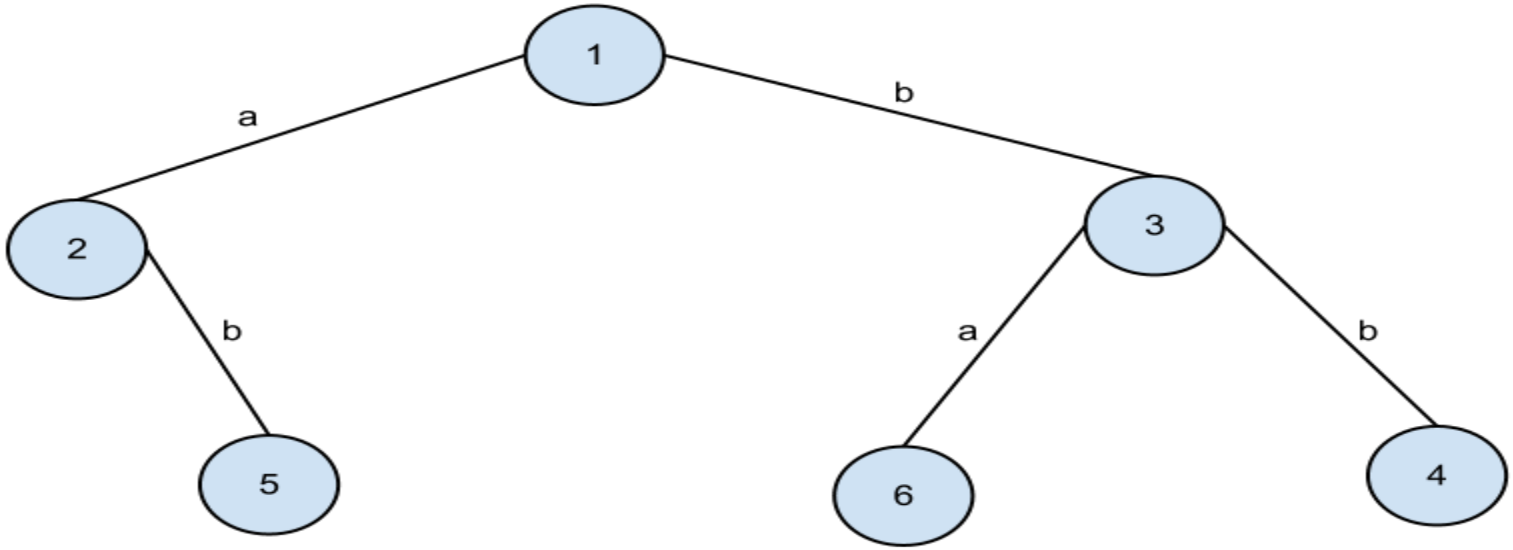
13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



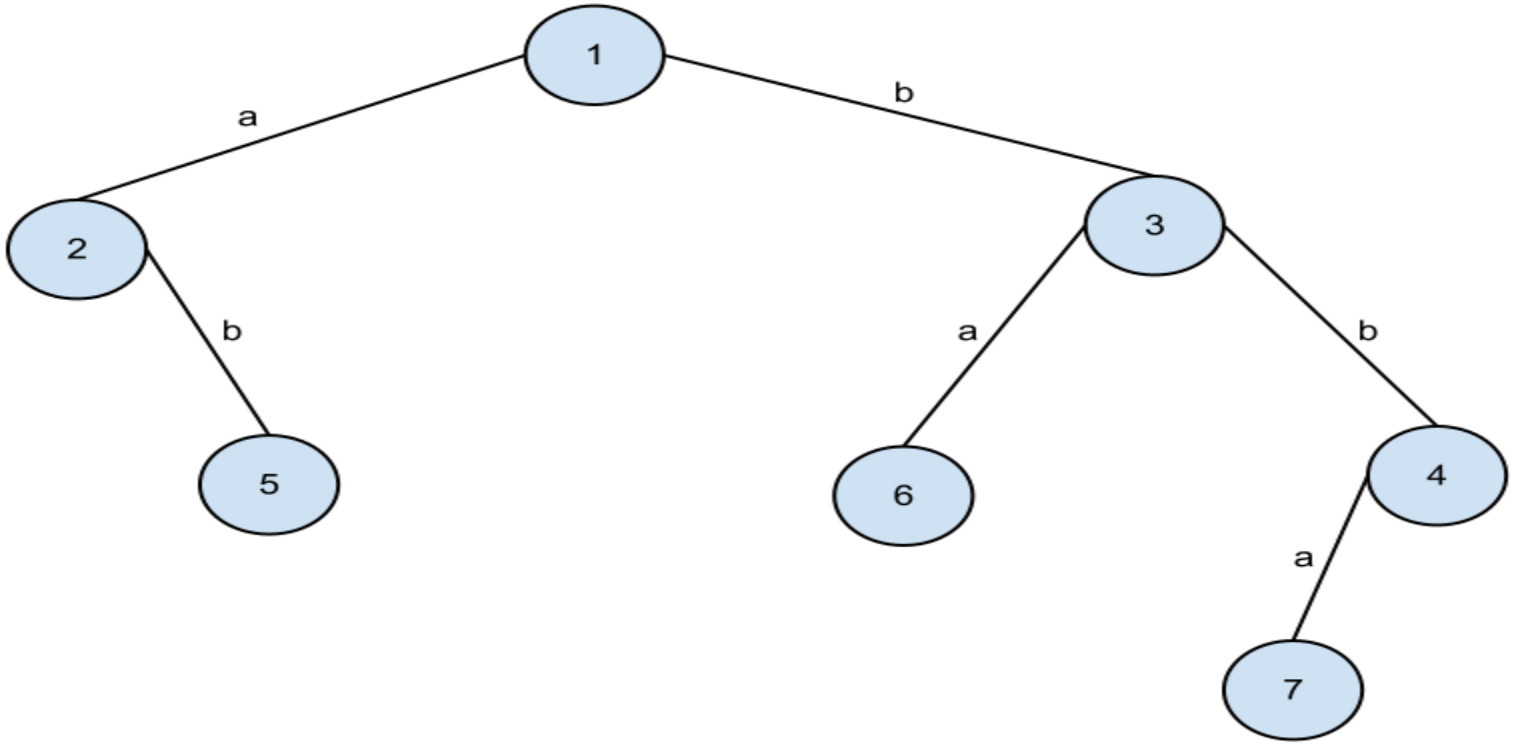
13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



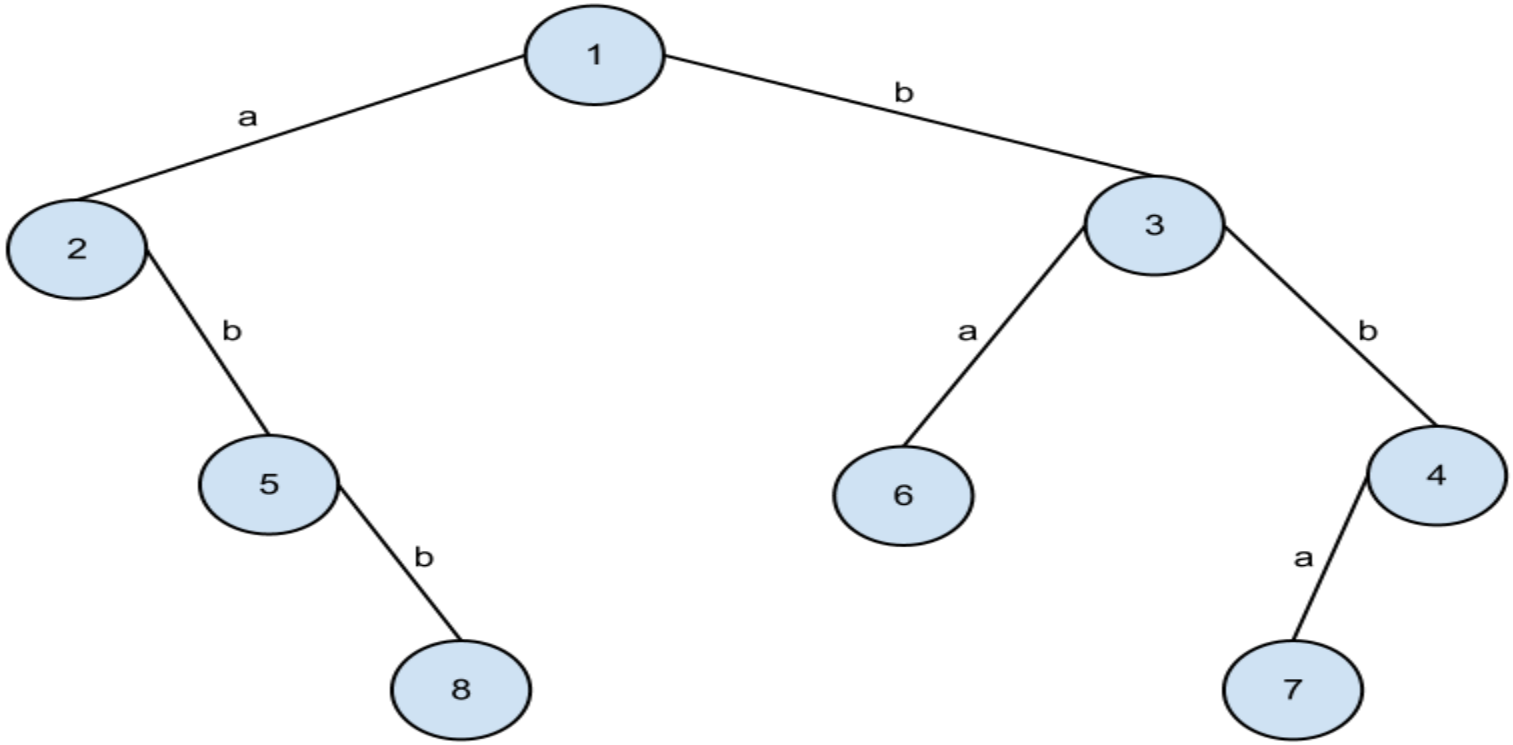
13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



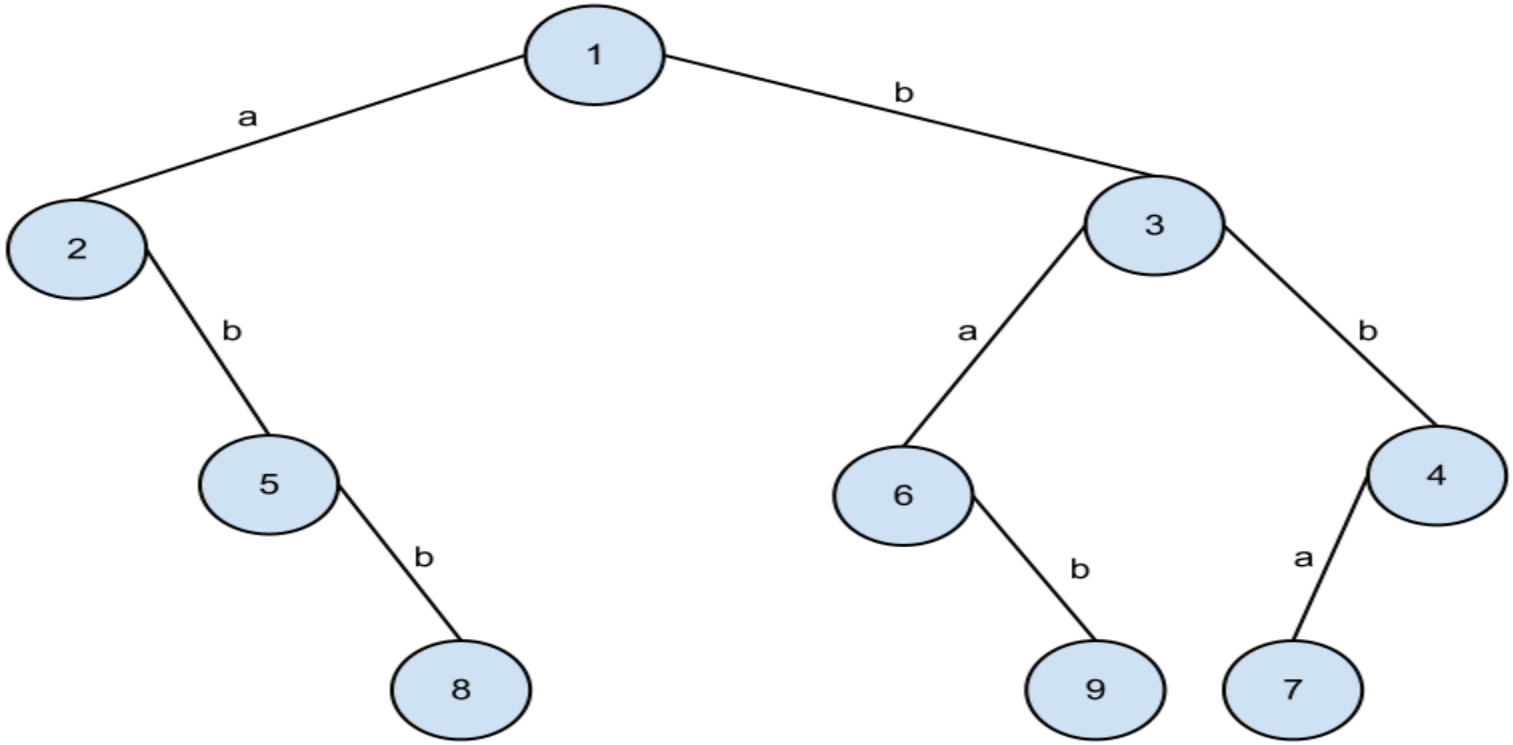
13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



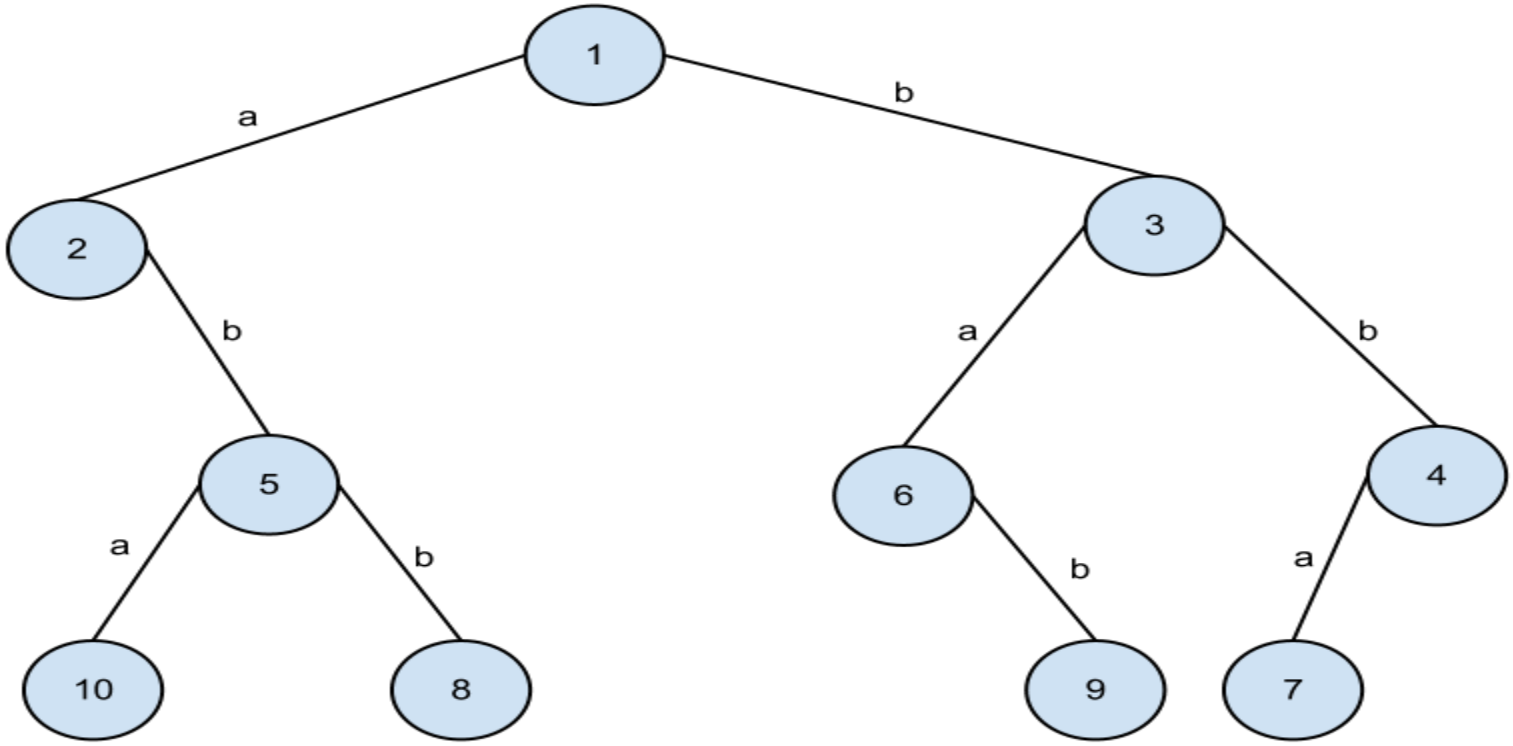
13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



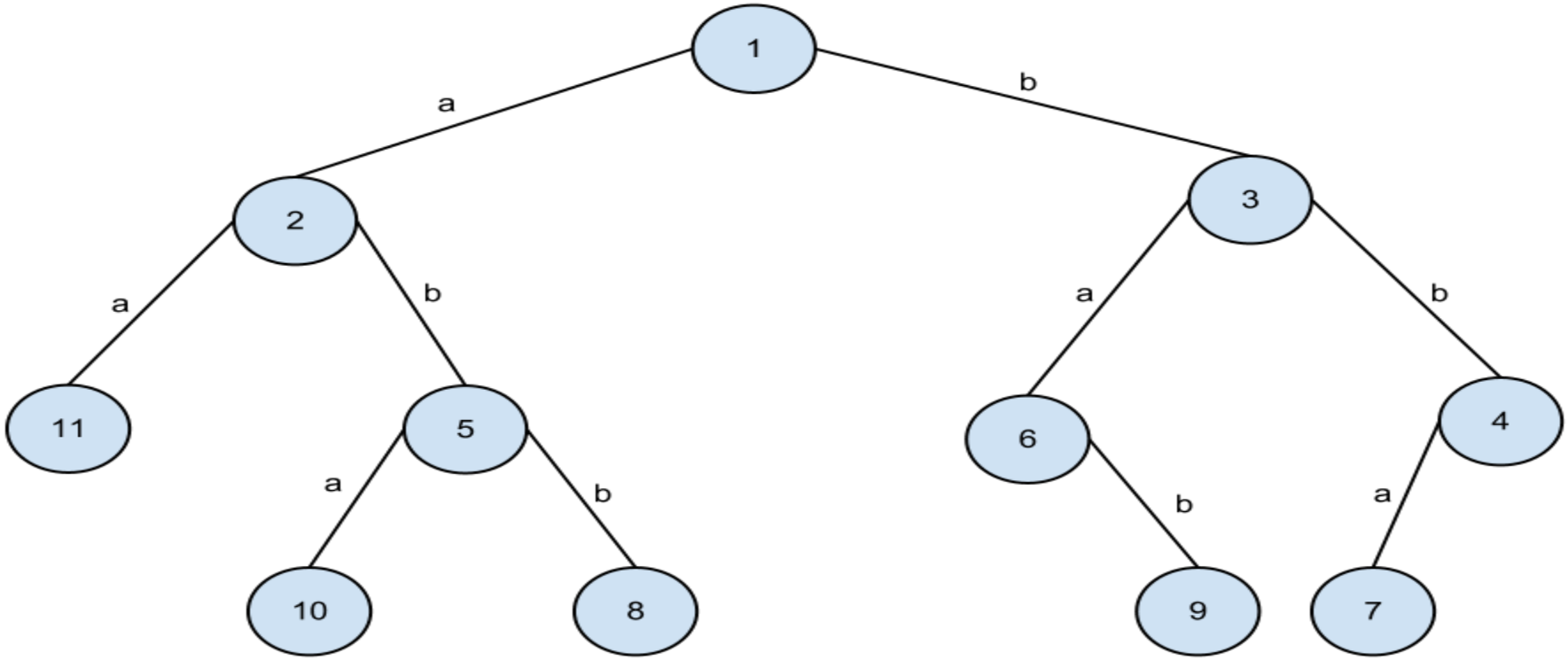
13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



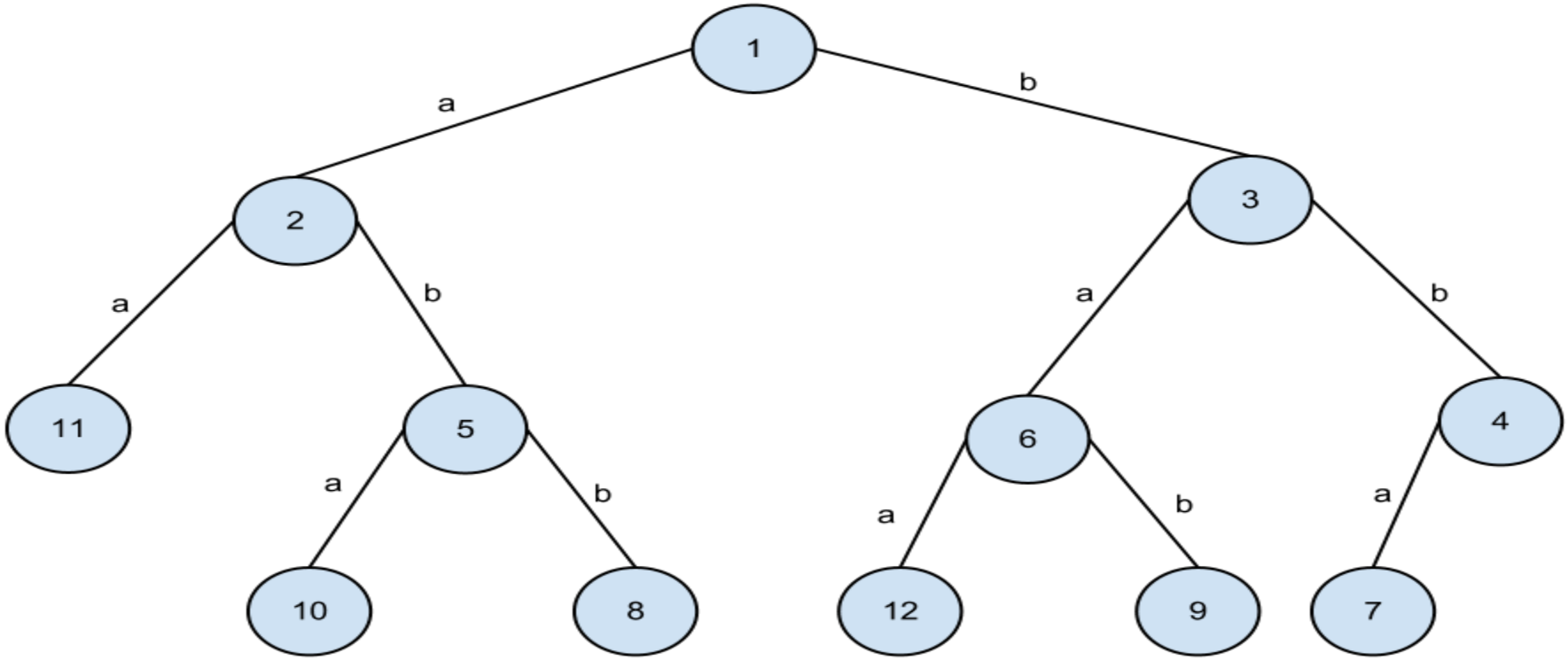
13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



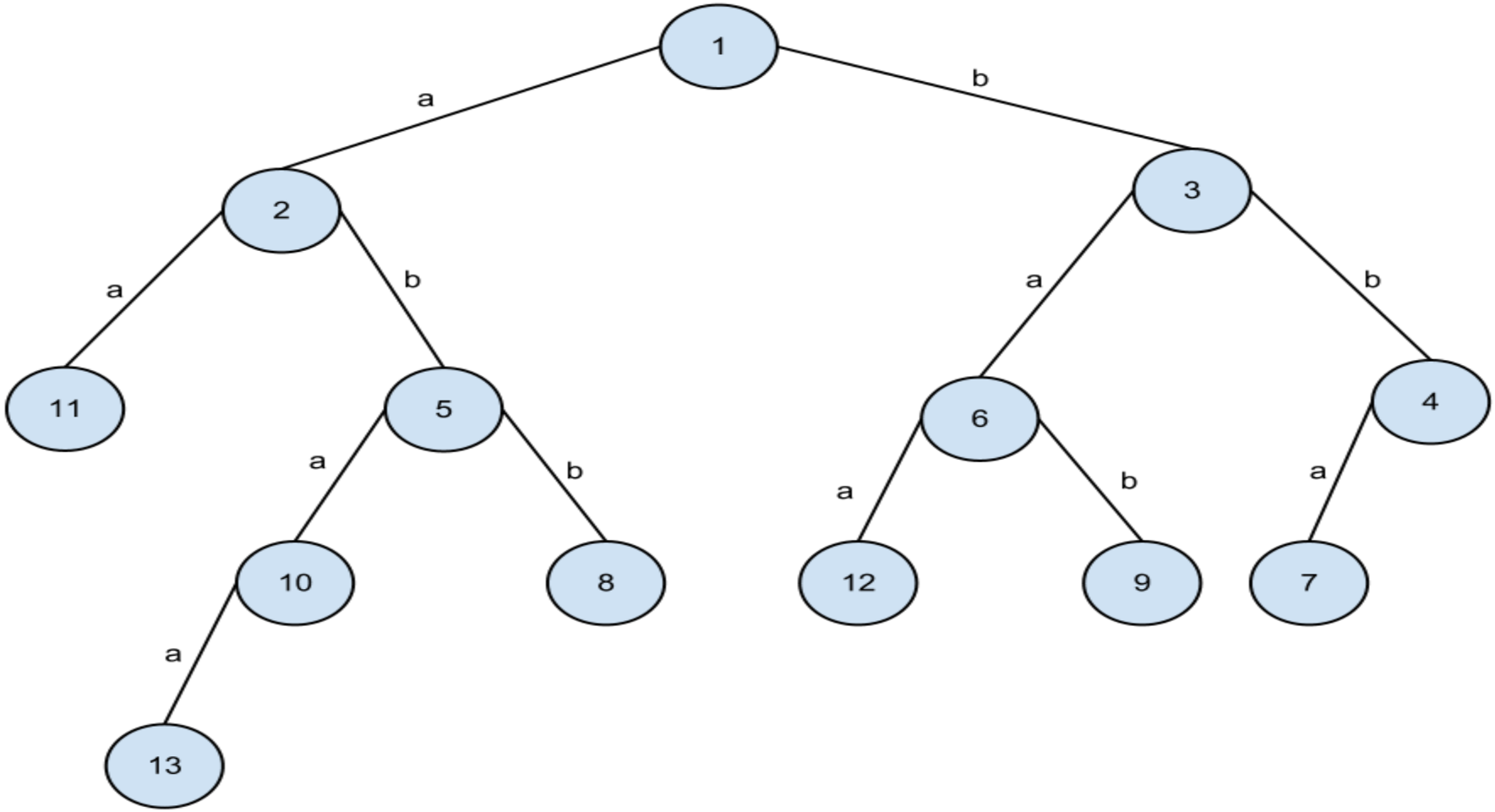
13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



What is the height of a position heap of text T?

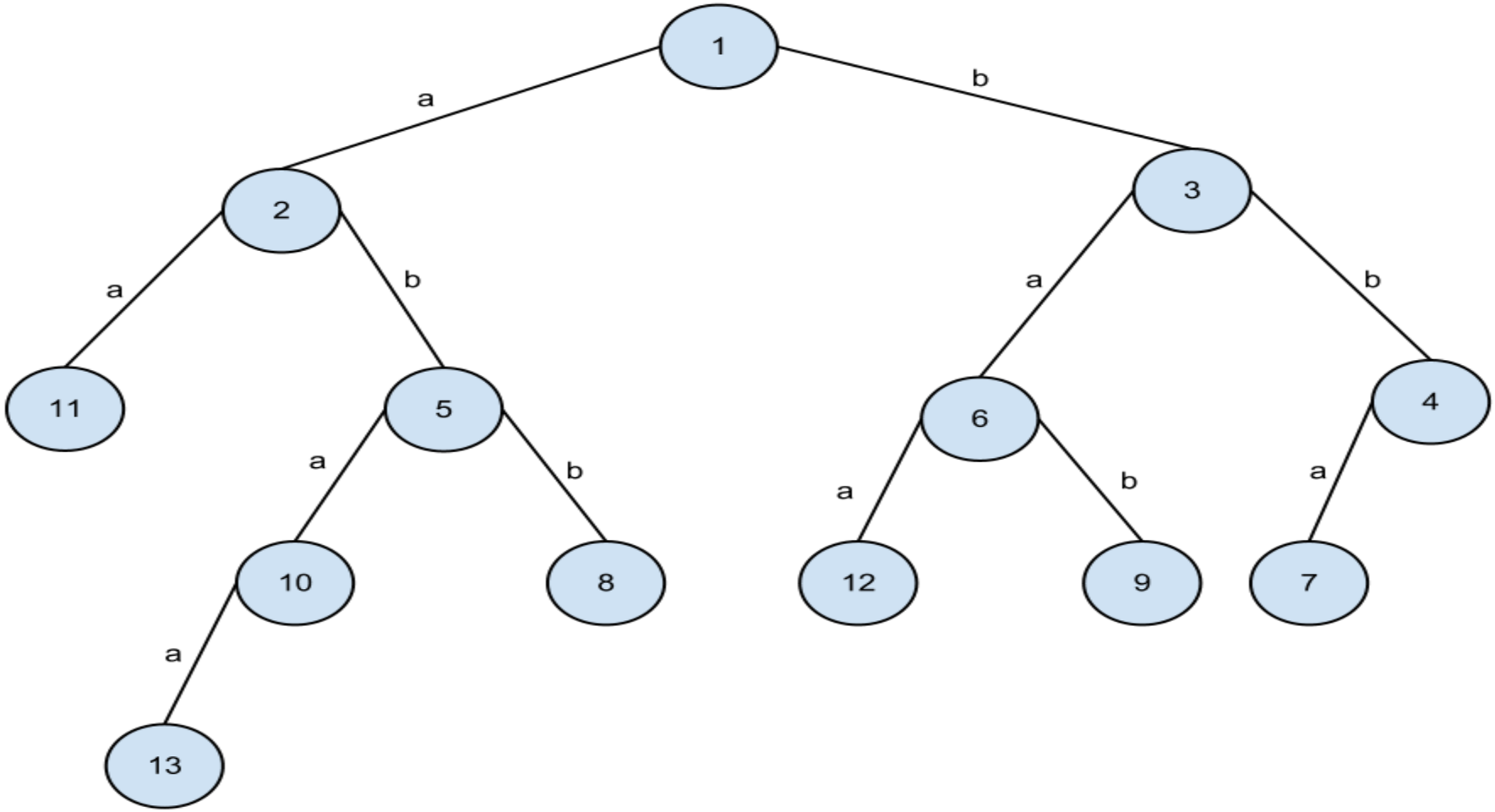
Definition: Let h(T) be the length of the longest substring X of T that is repeated at least |X| times.

Then height is at most 2h(T).

Naive Searching

P = ab

13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



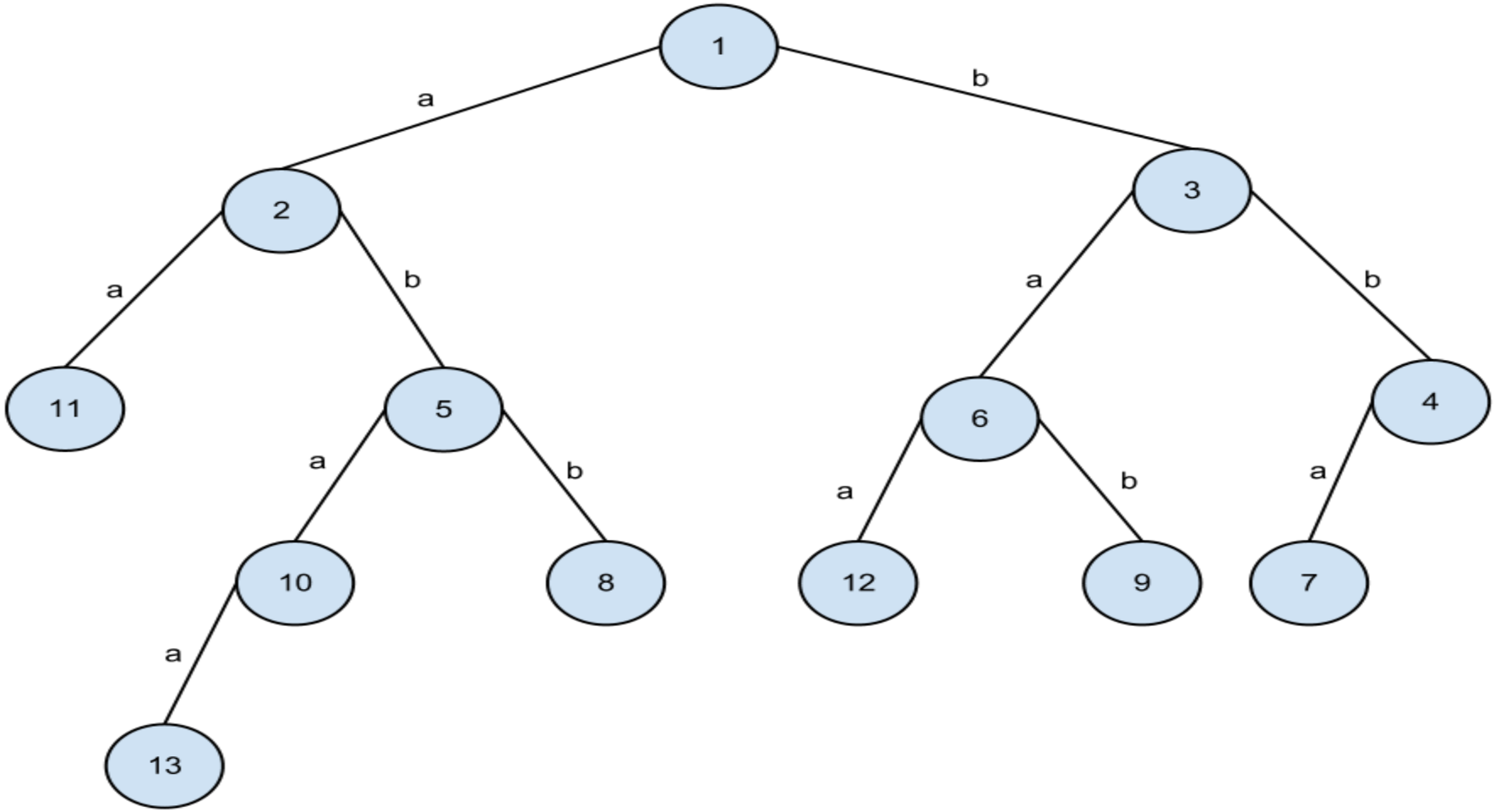
Simple, Elegant, Cool

How come nobody invented it before?

Because

For every complex problem there is an answer that is clear, simple, and wrong.
H. L. Mencken

13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b

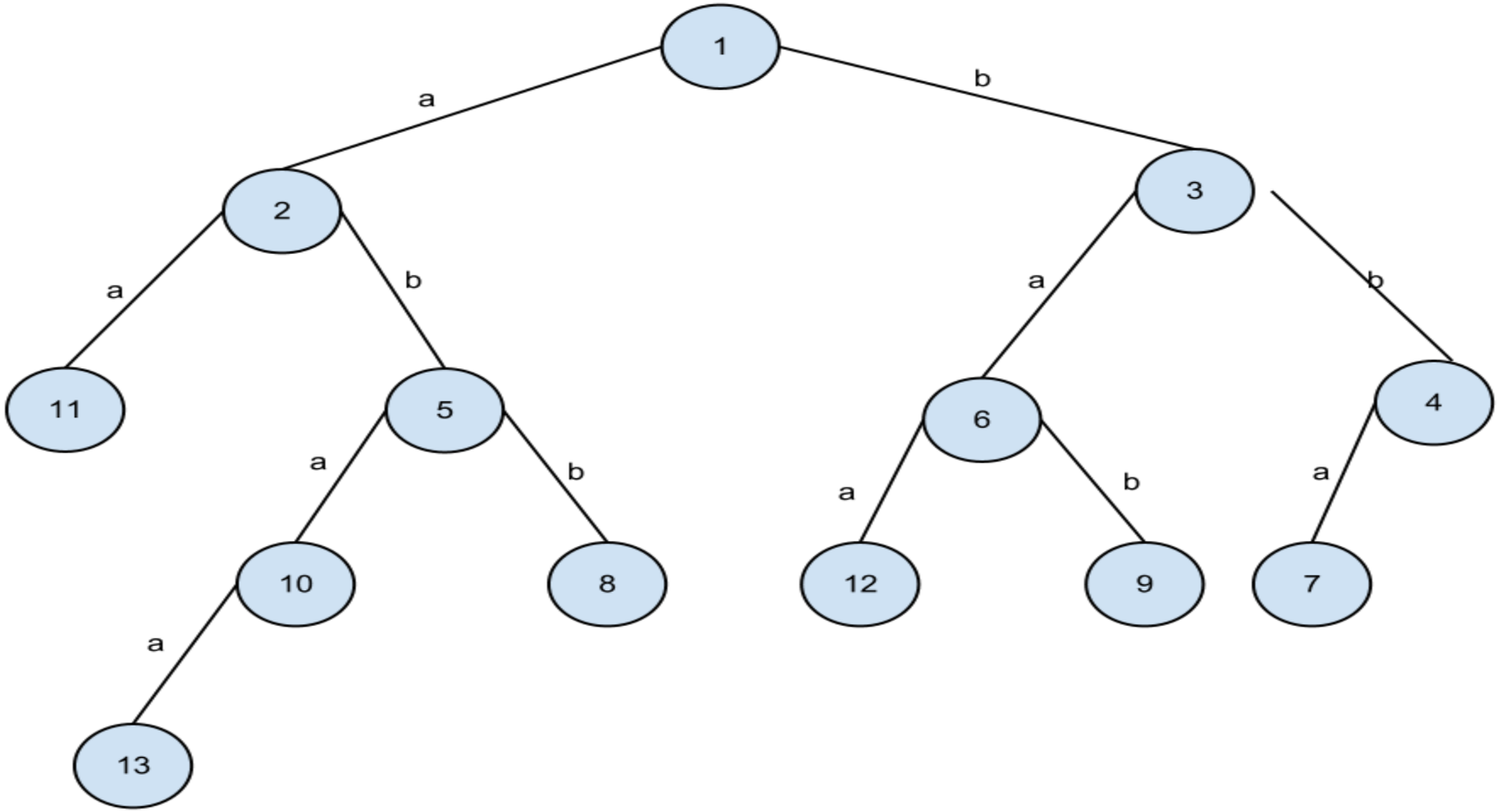


So some ancestors are also occurrences and that gives us O(m*m + k) bound for searching.

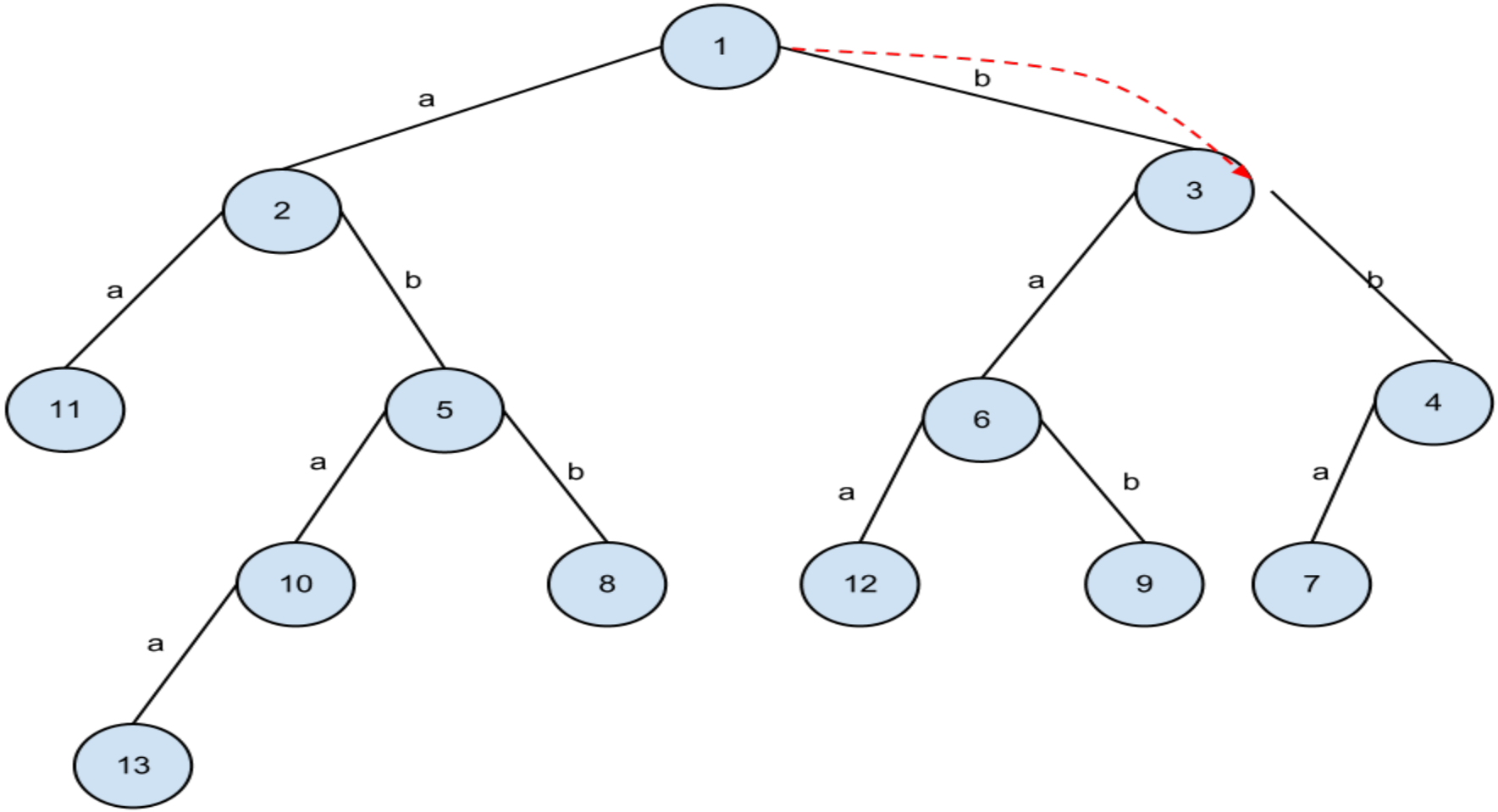
Can we do better?

YES WE CAN!

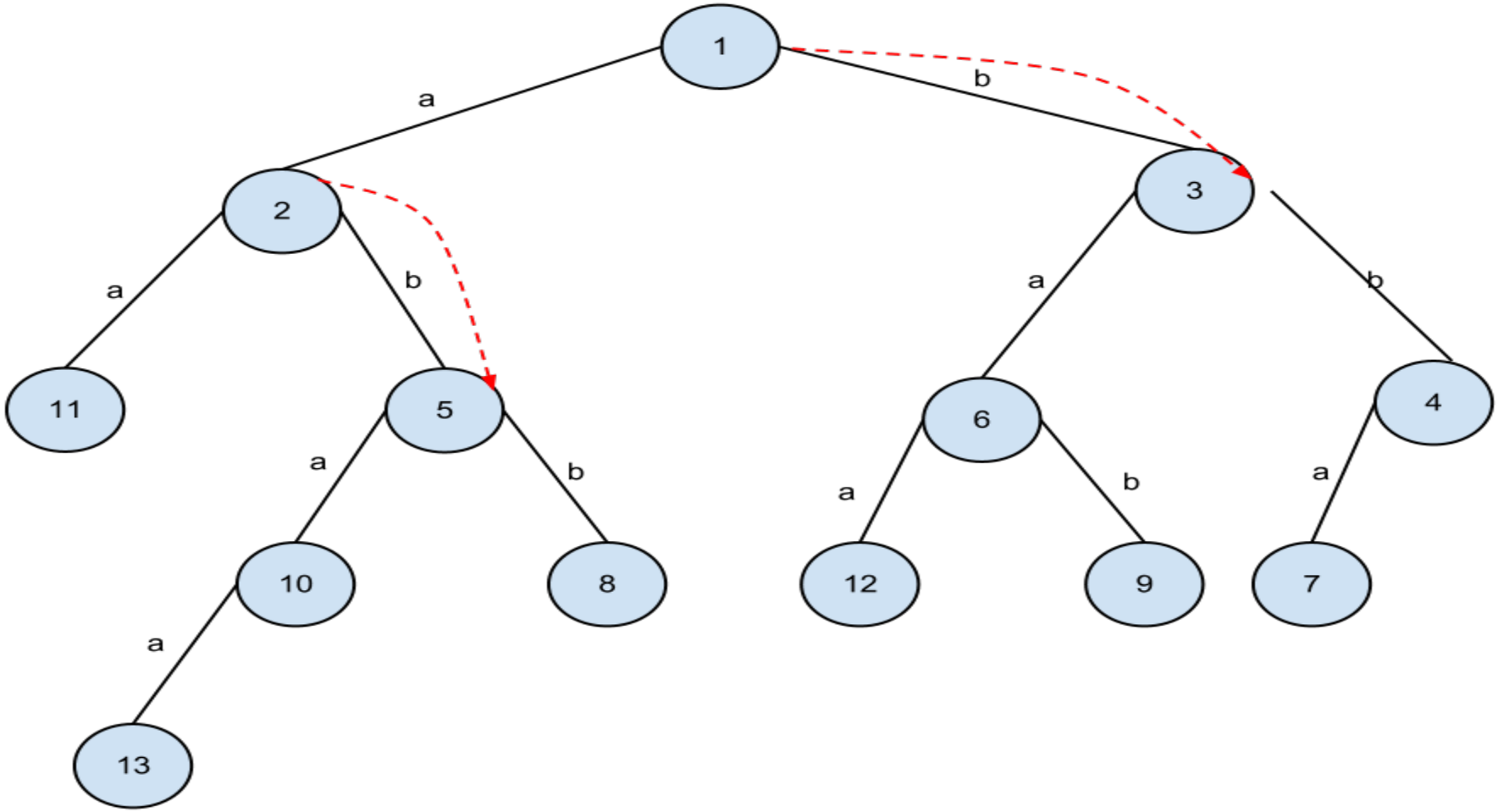
13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



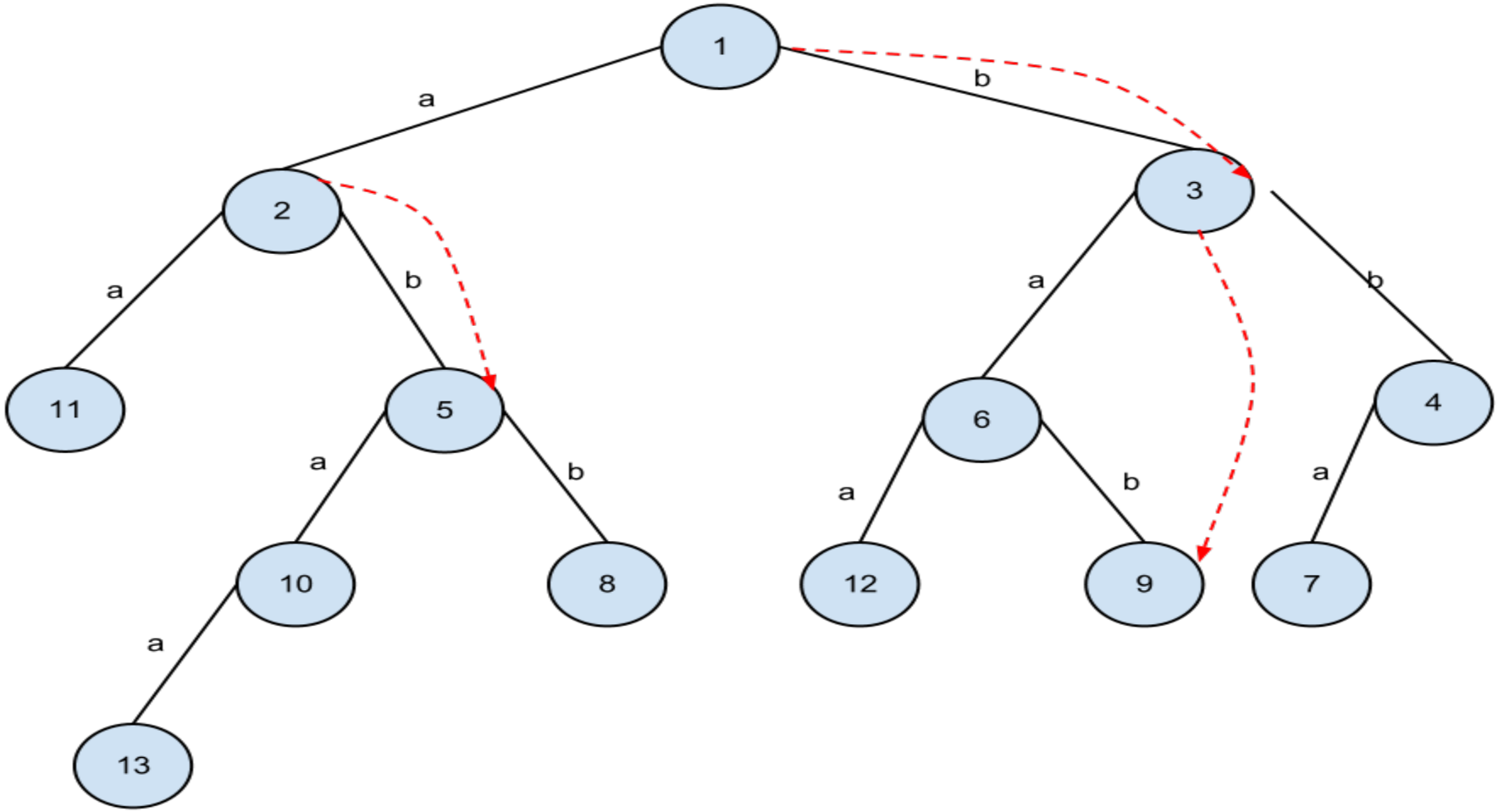
13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



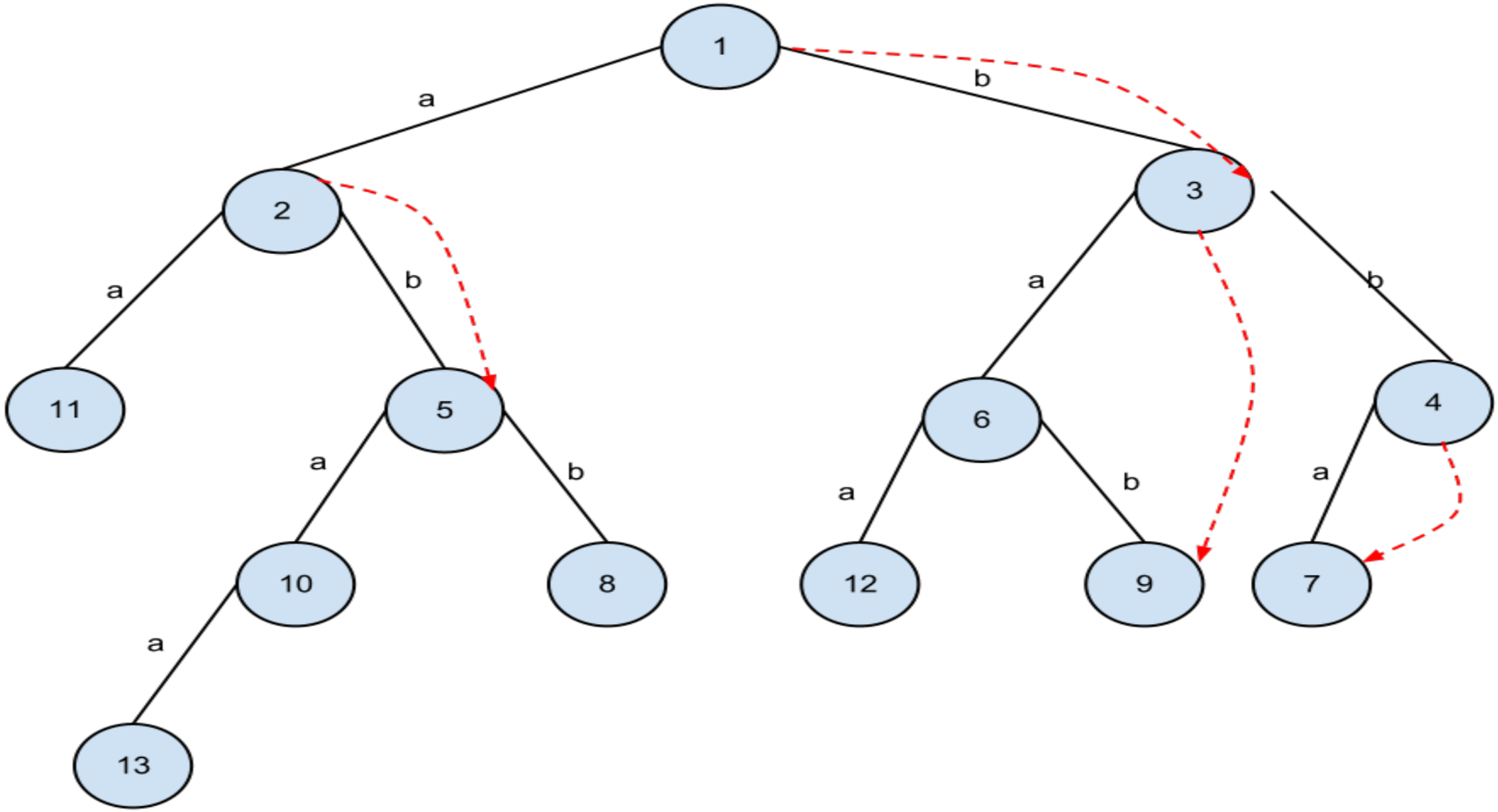
13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



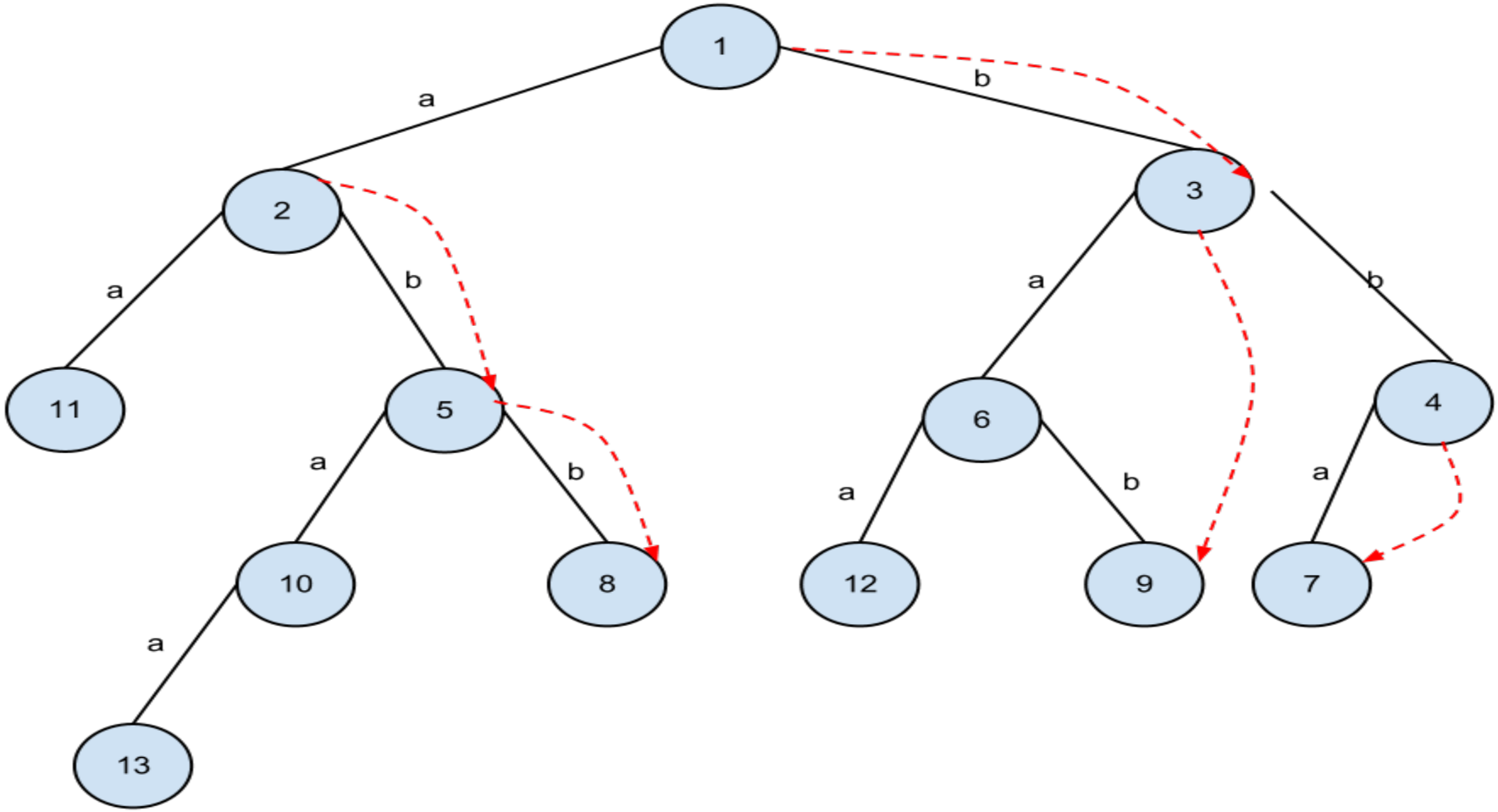
13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



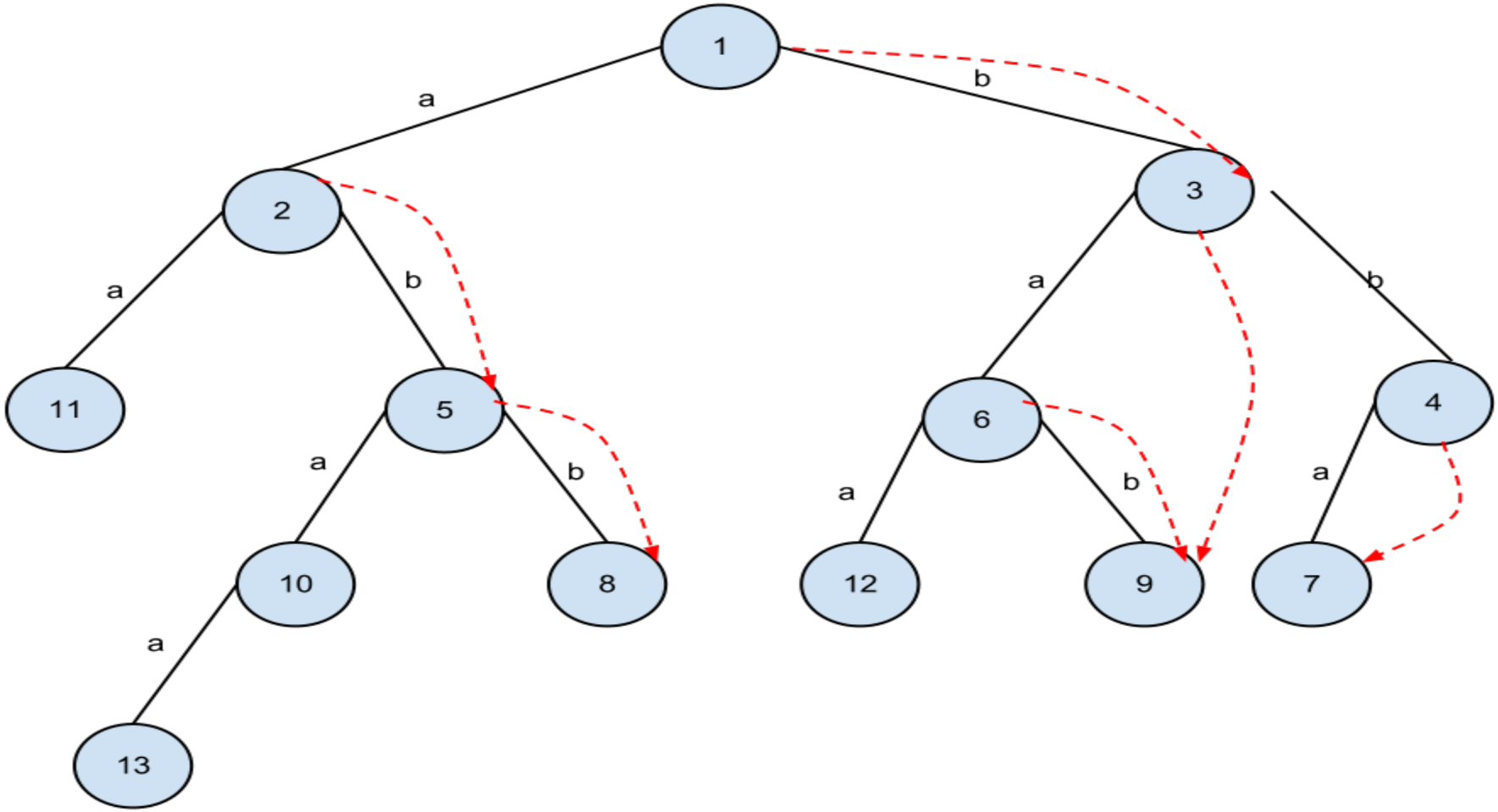
13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b

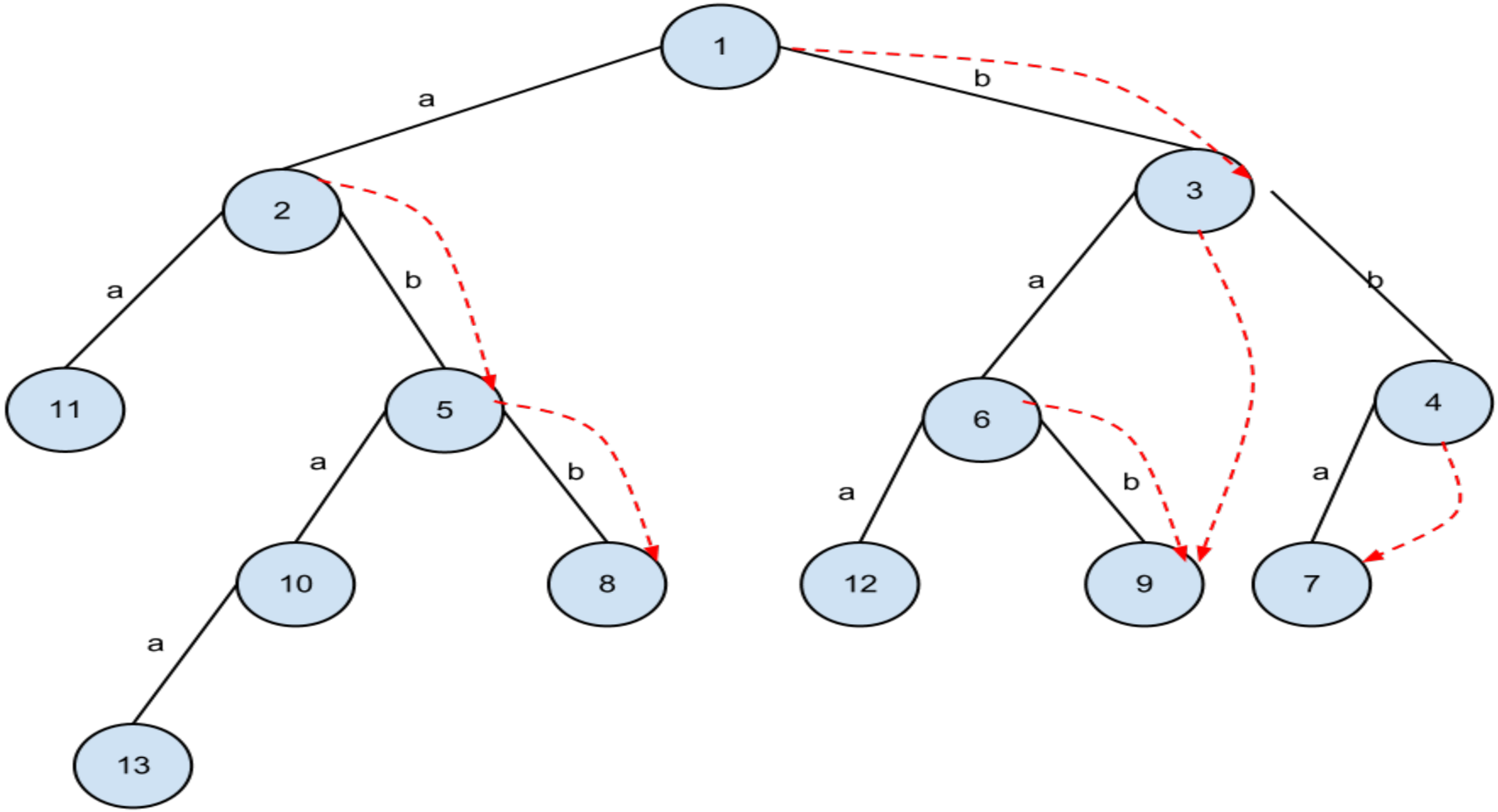


13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b



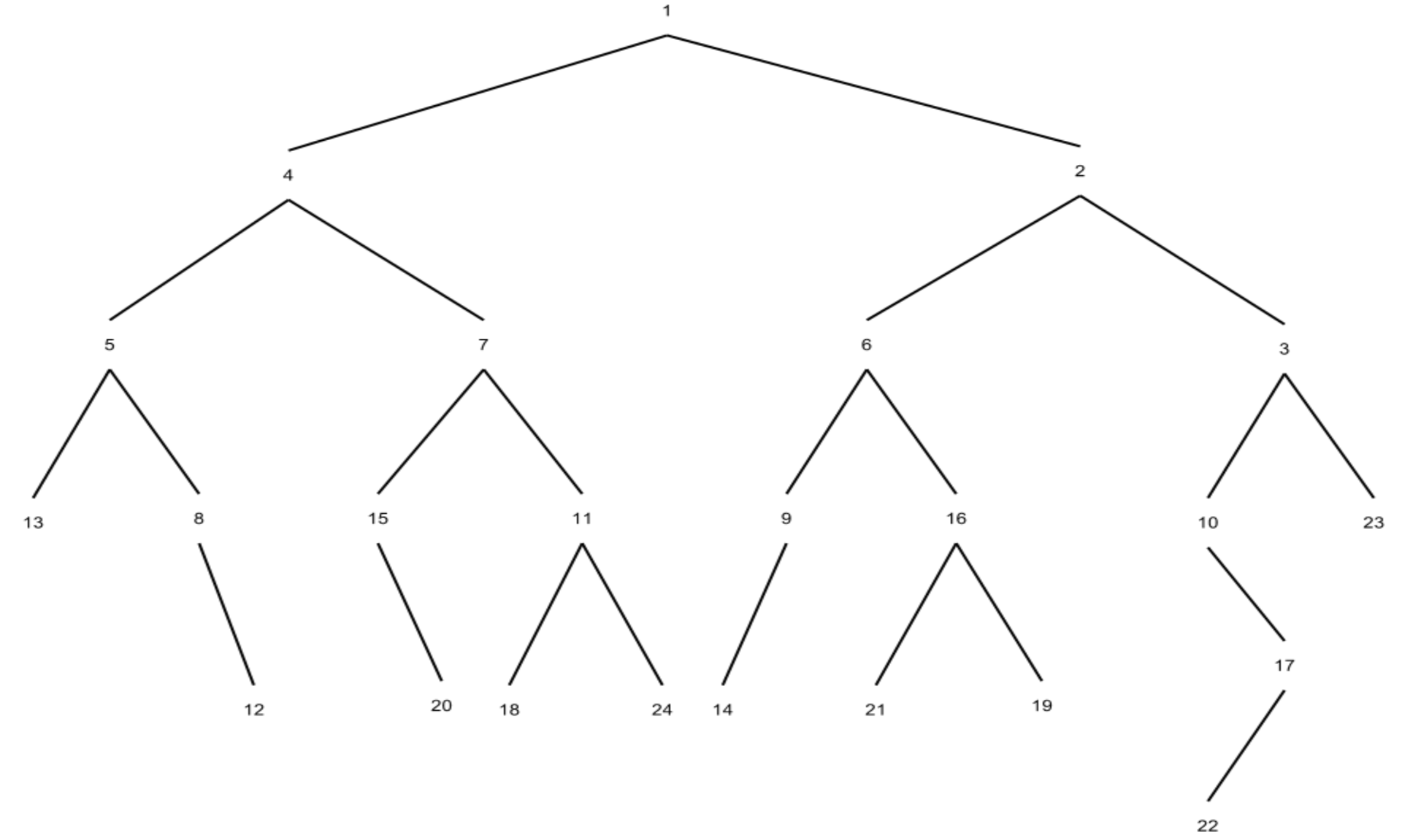
Those red arrows are called Maximal Reach Pointers.

13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	a	a	b	a	b	b	a	b	b	a	b

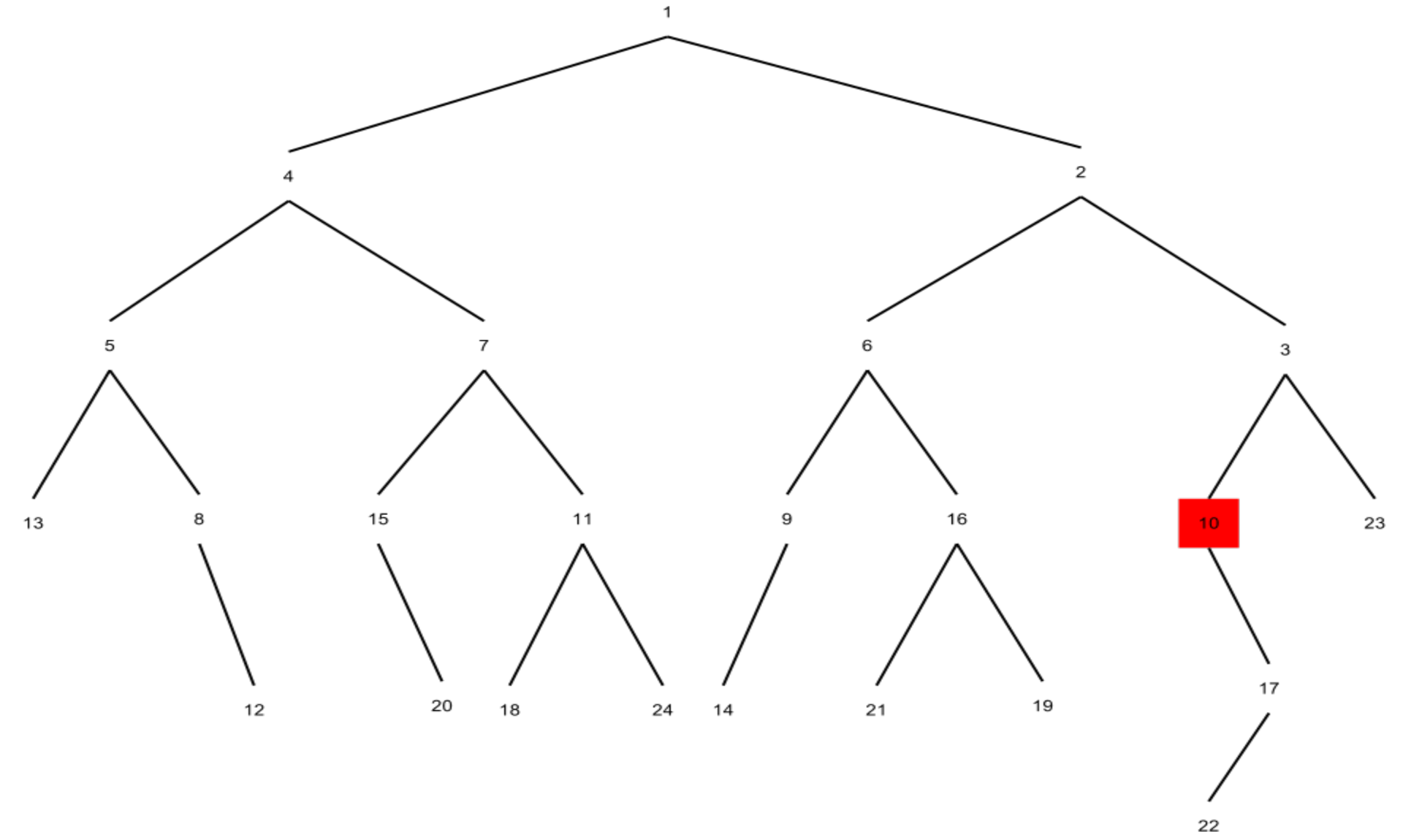


Editing T

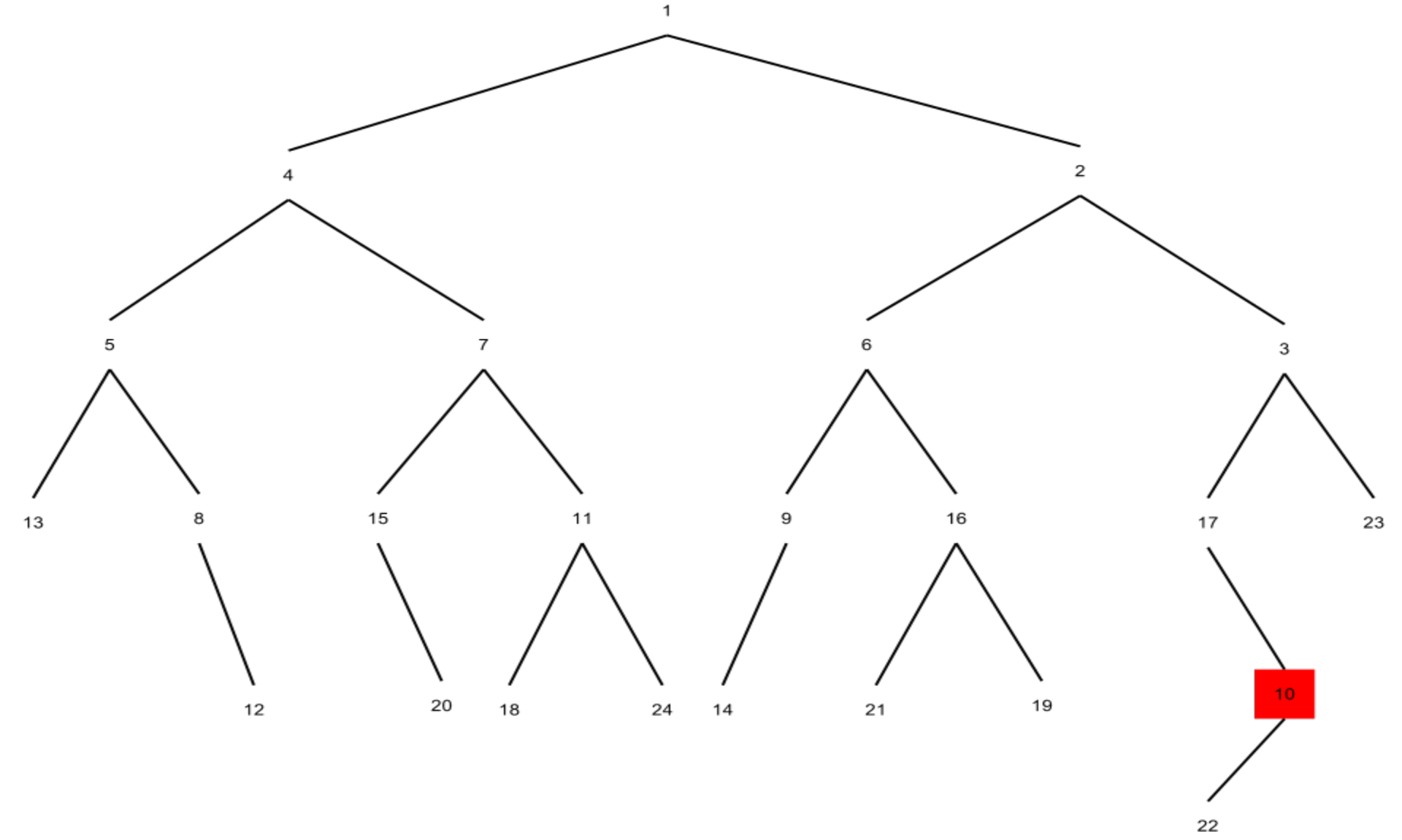
24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	b	a	a	b	a	a	b	b	a



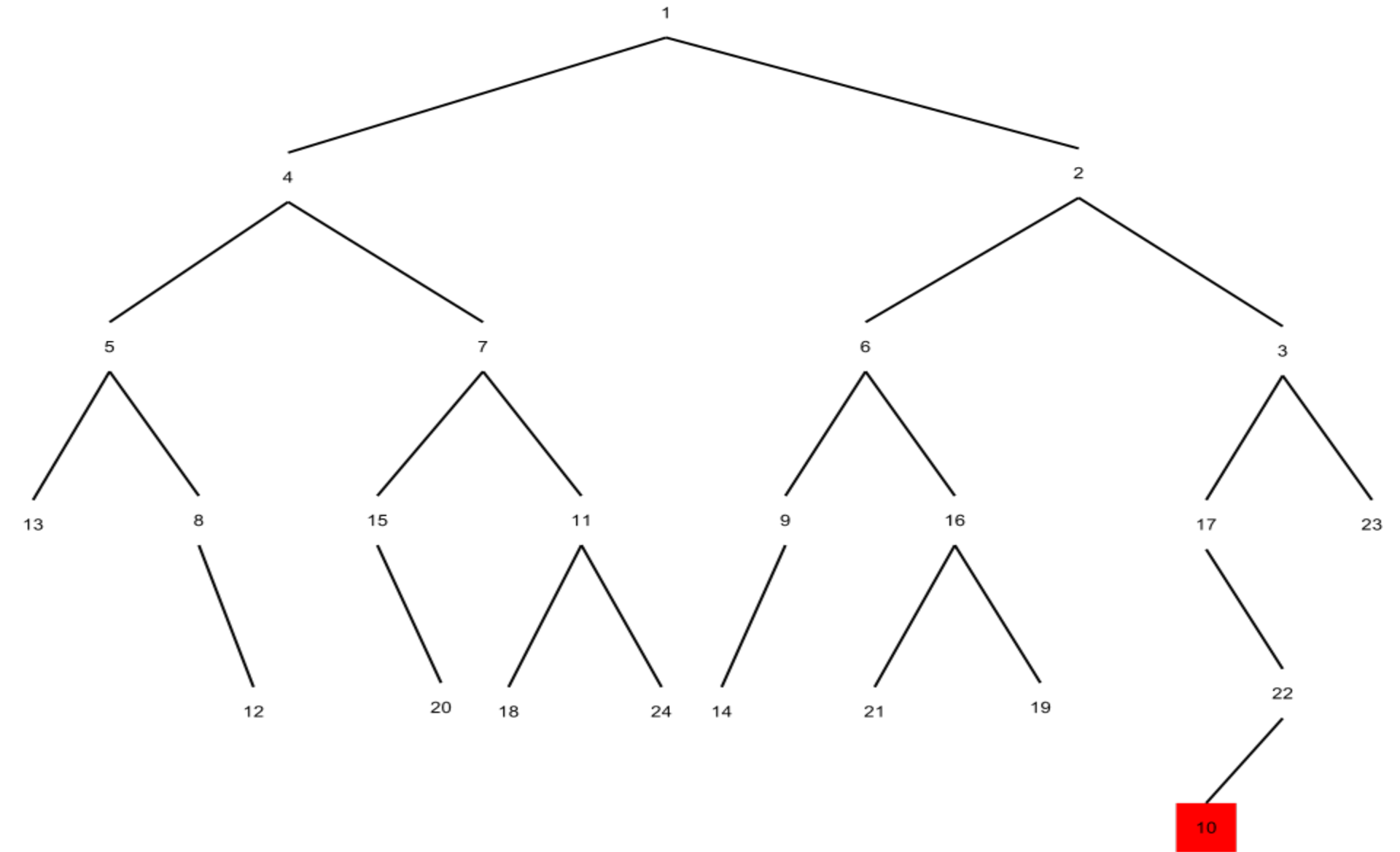
24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	b	a	a	b	a	a	b	b	a
														x									



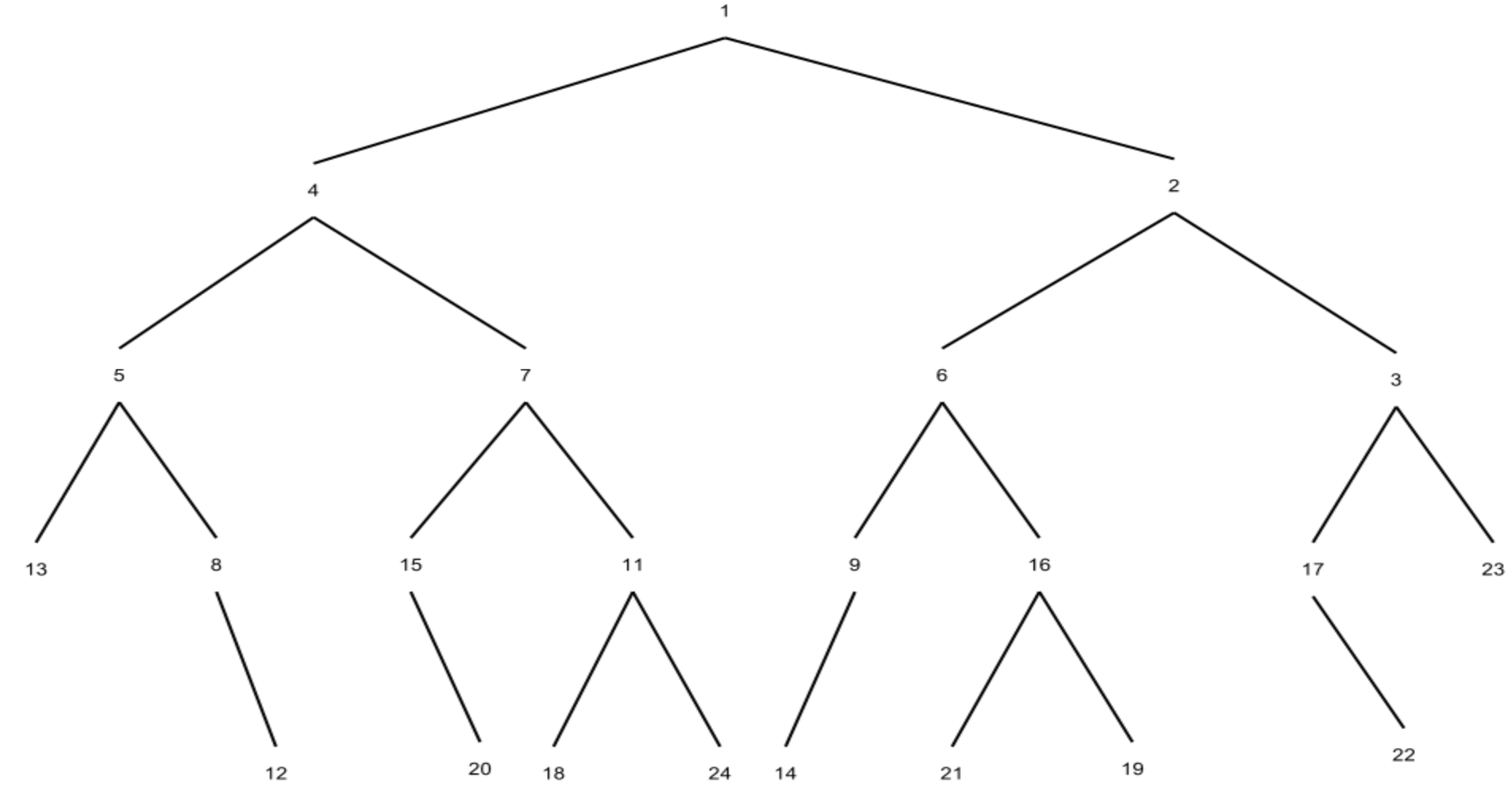
24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	b	a	a	b	a	a	b	b	a
														x									



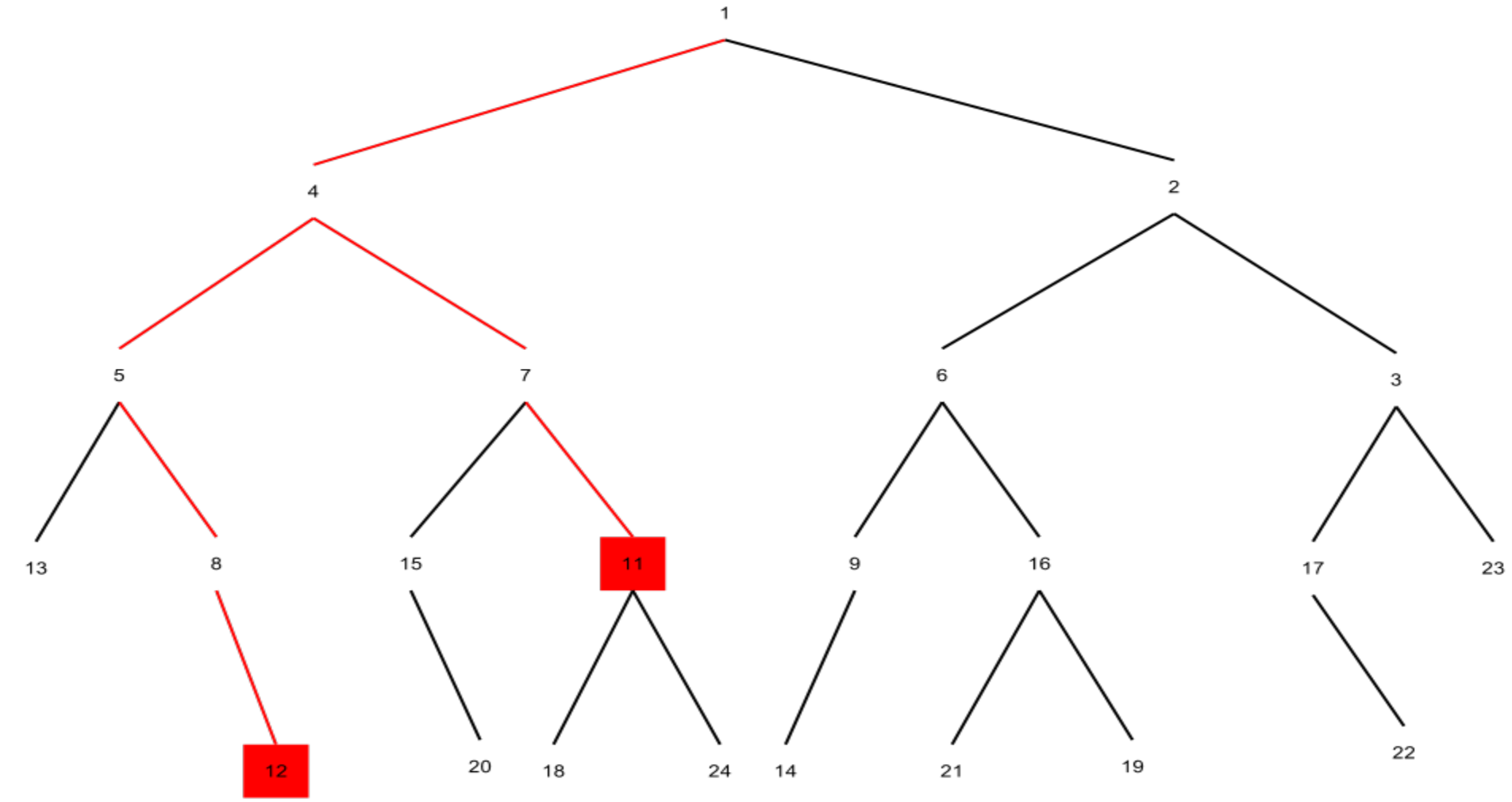
24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	b	a	a	b	a	a	b	b	a
														x									



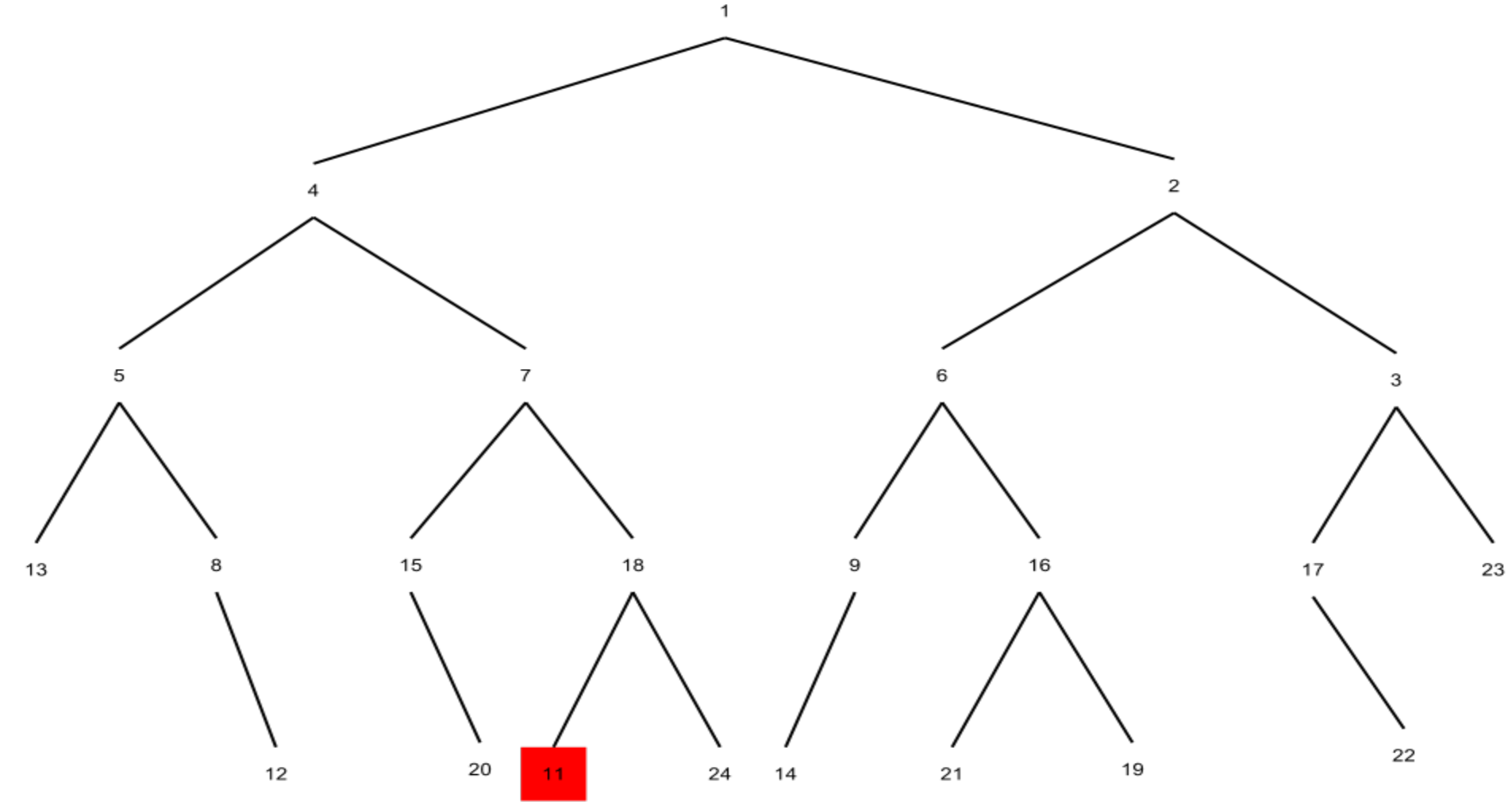
24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	a	a	b	a	a	b	b	a



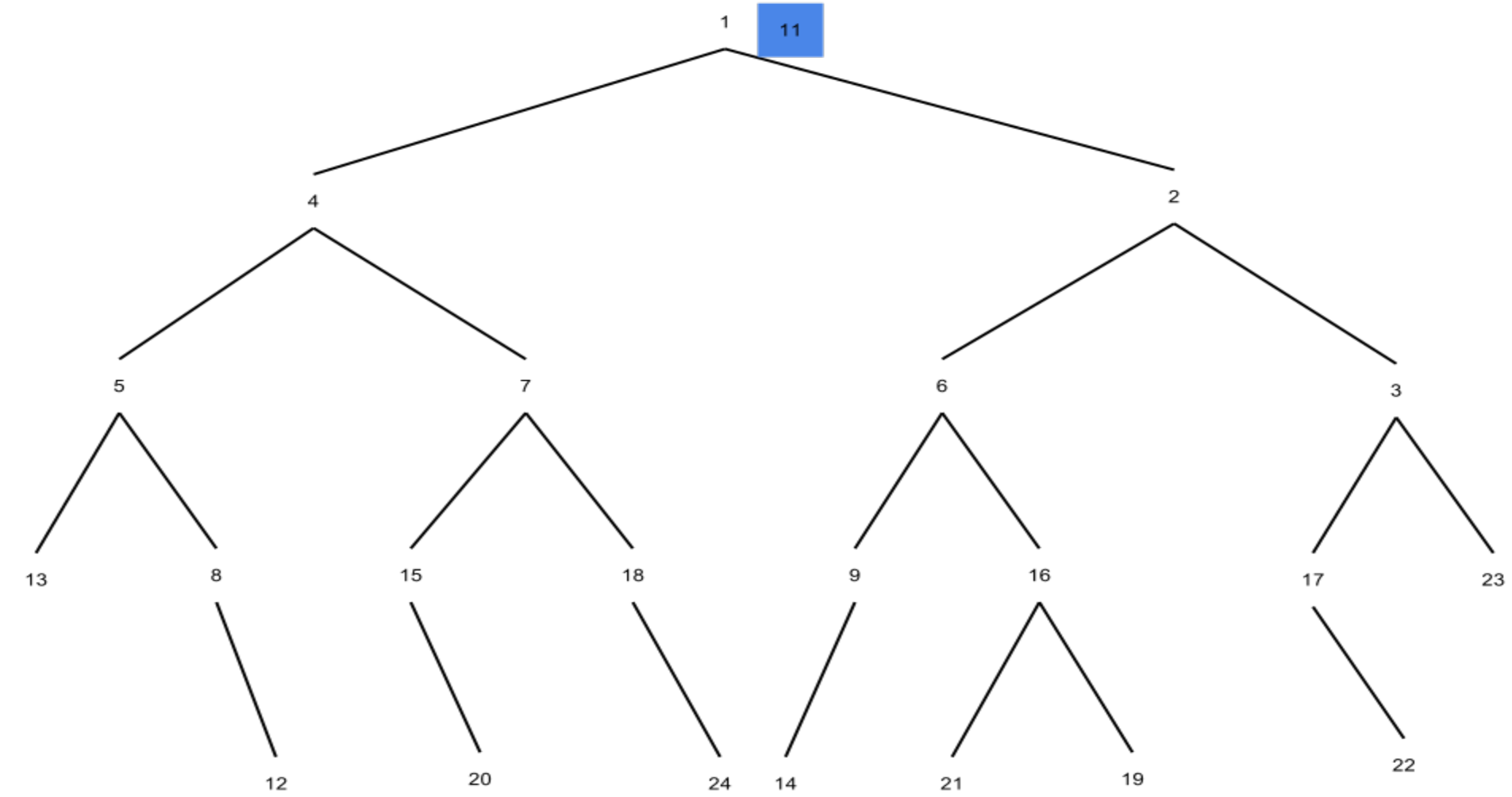
24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	a	a	b	a	a	b	b	a



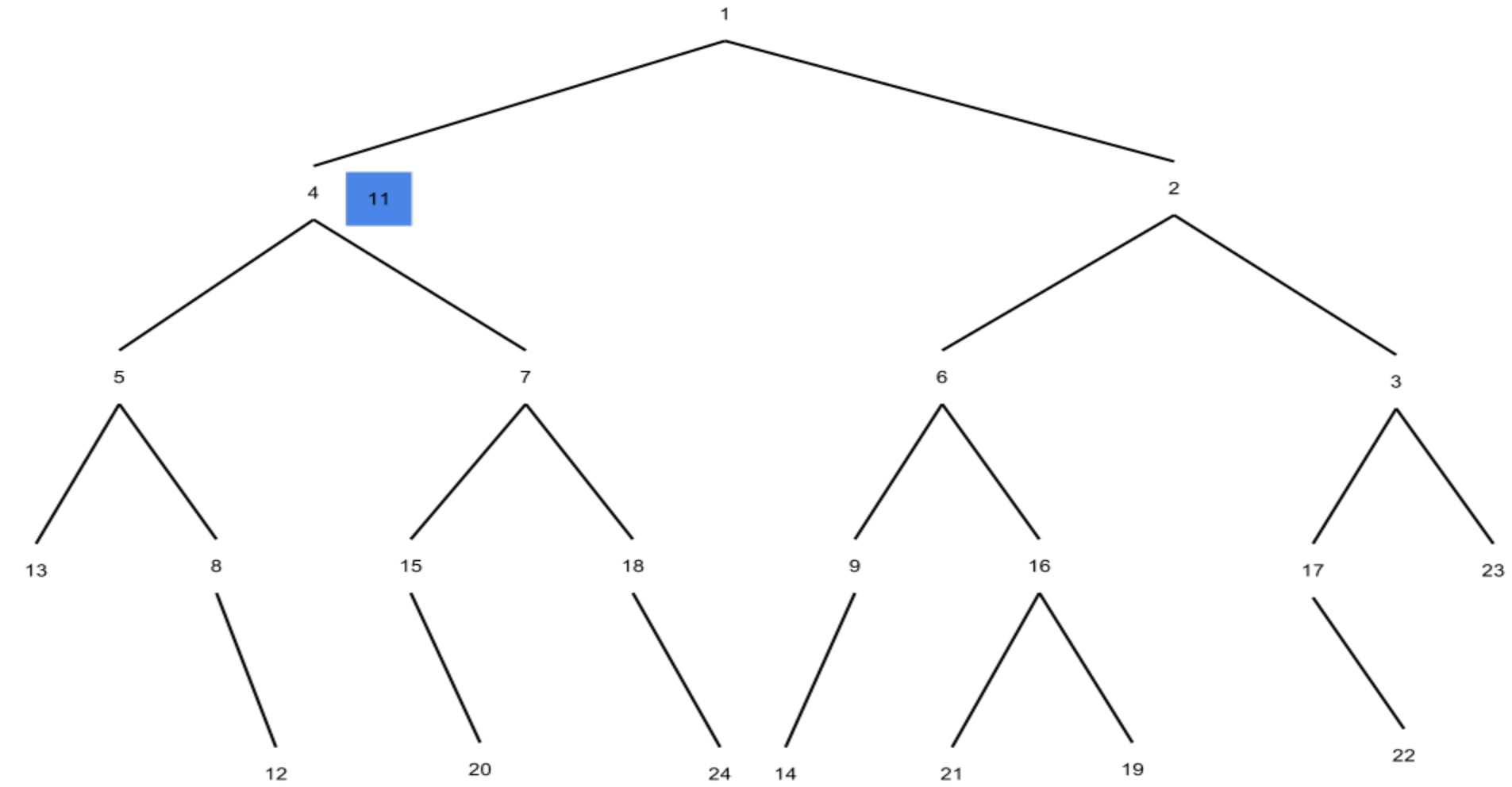
24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	a	a	b	a	a	b	b	a



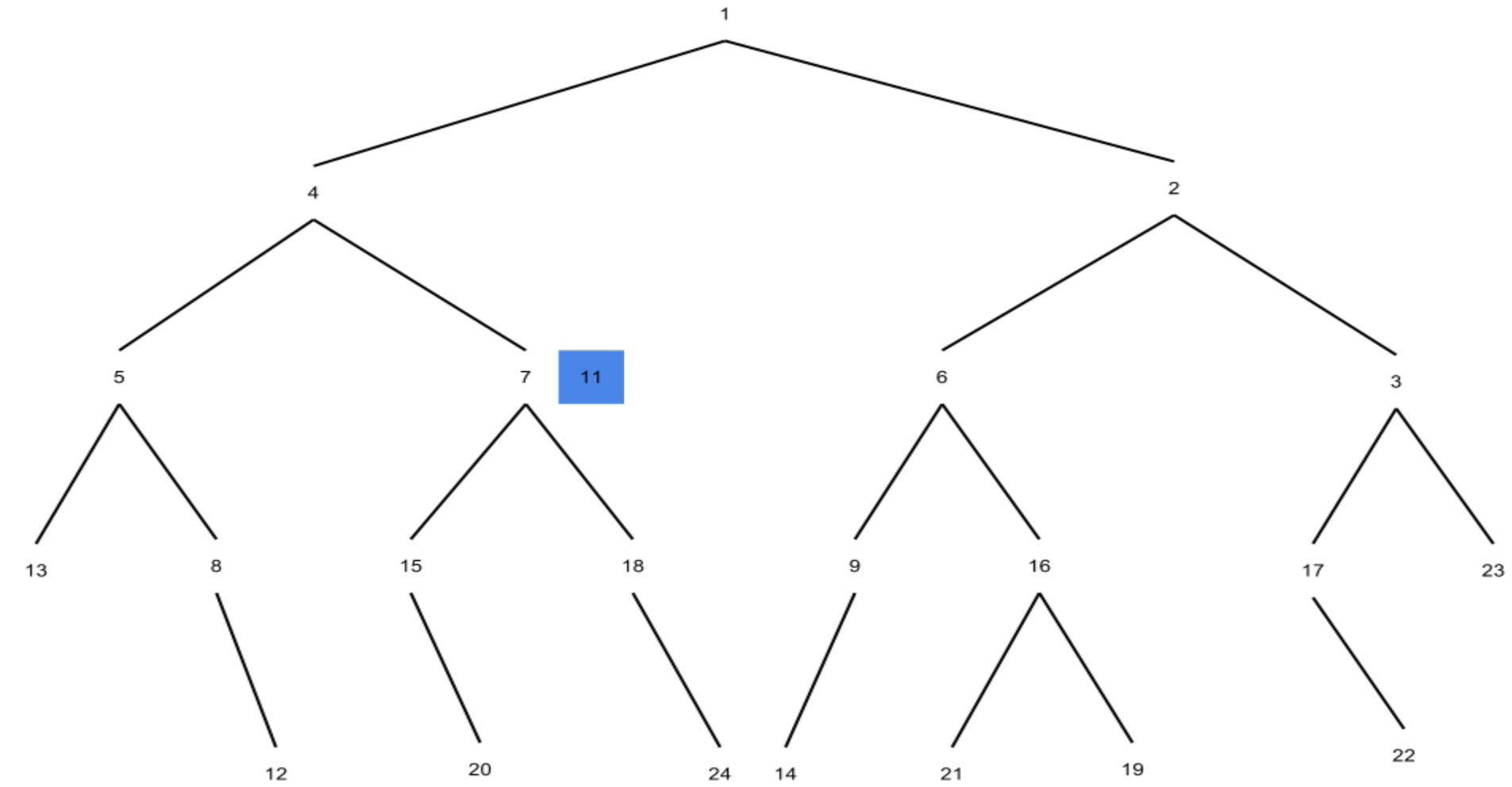
24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	a	a	b	a	a	b	b	a



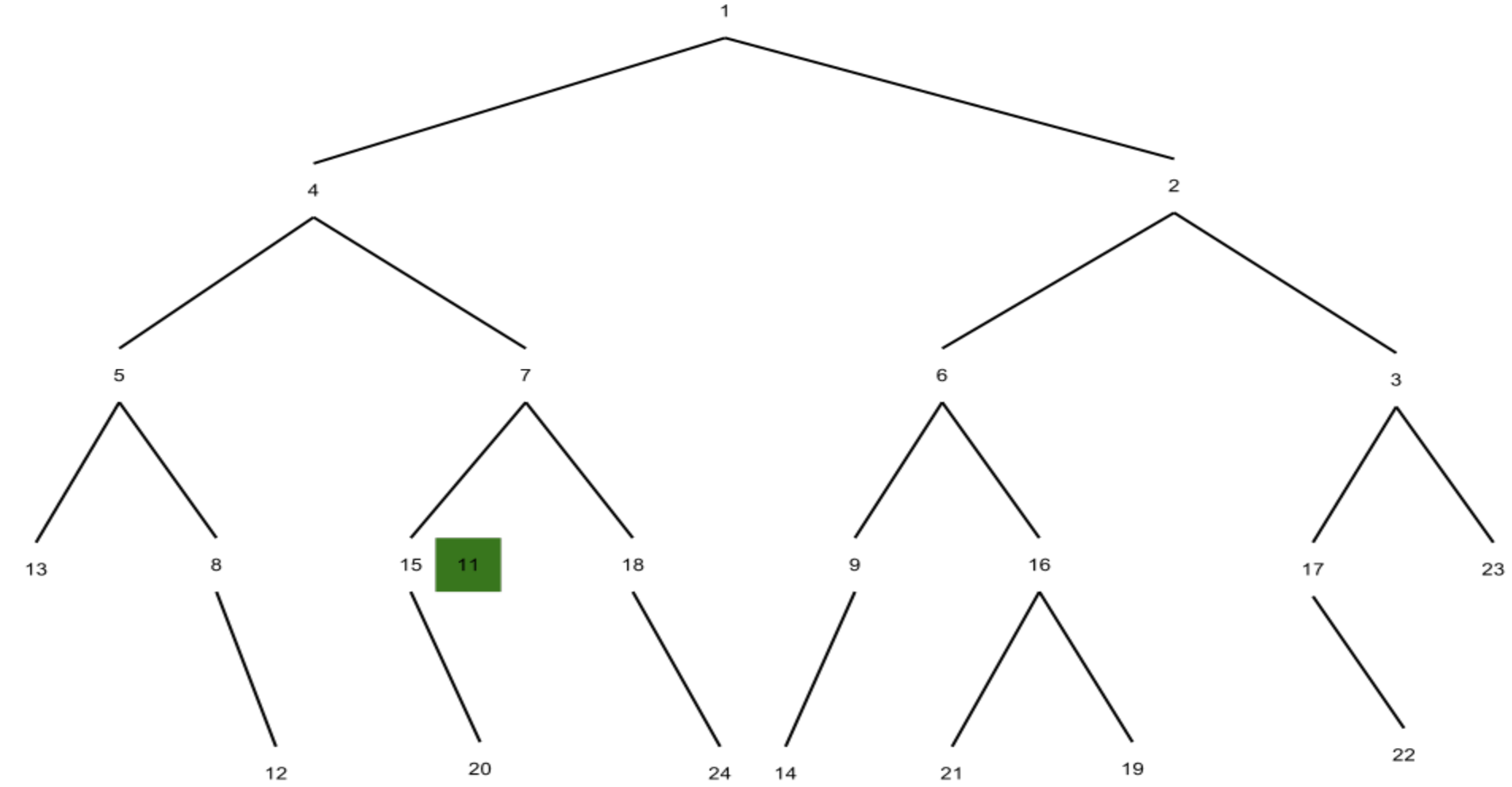
24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	a	a	b	a	a	b	b	a



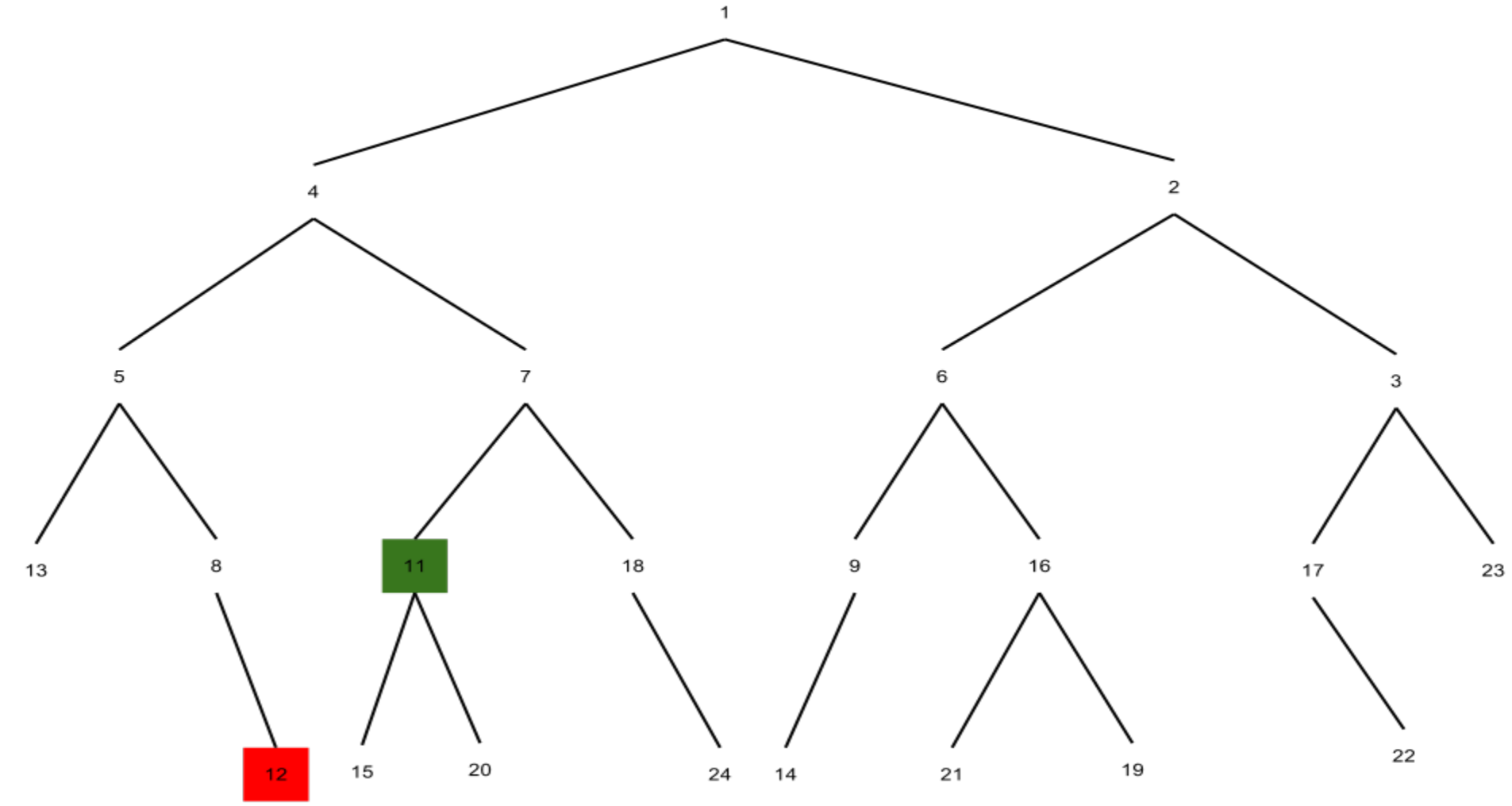
24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	a	a	b	a	a	b	b	a



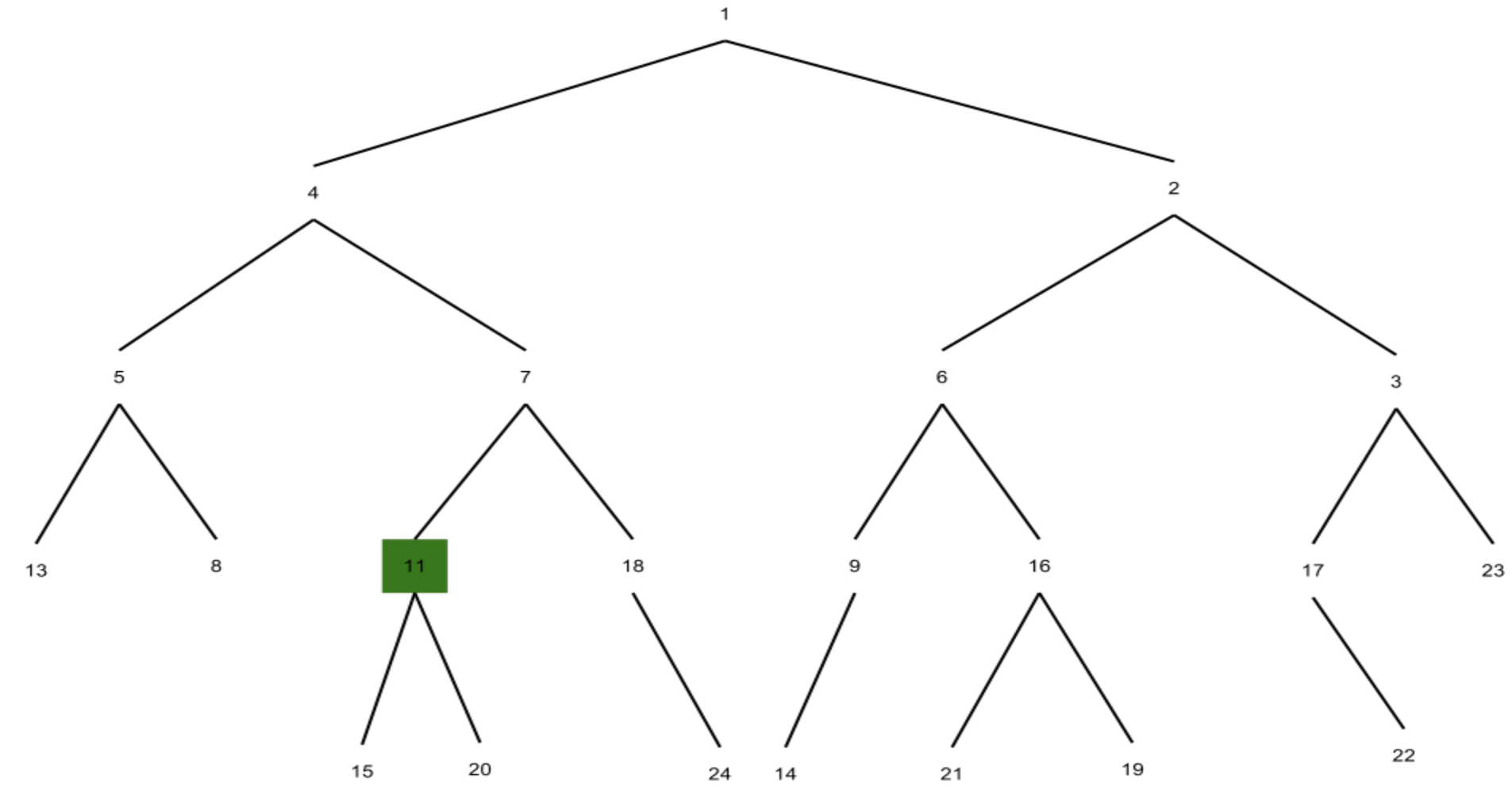
24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	a	a	b	a	a	b	b	a



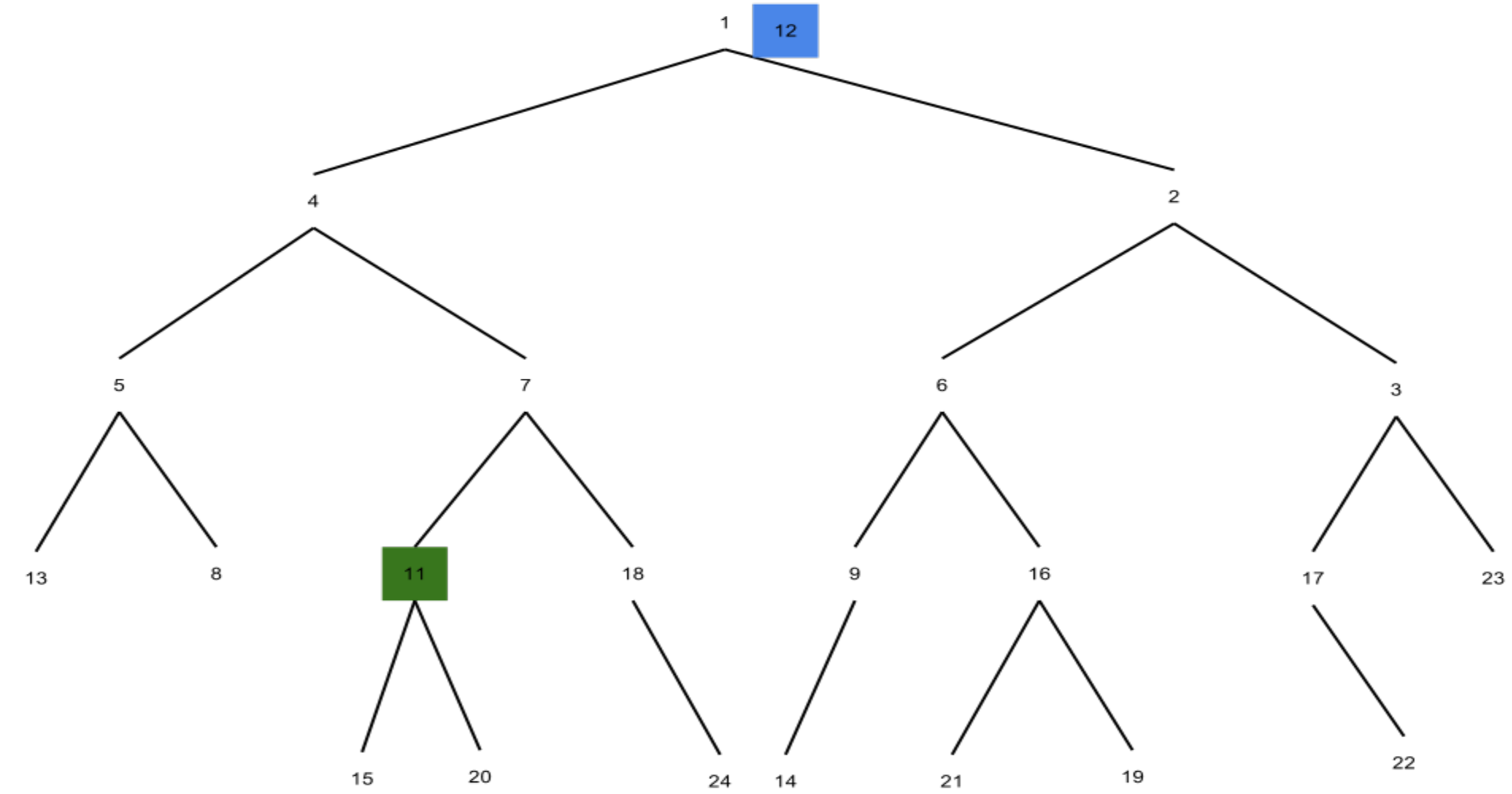
24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	a	a	b	a	a	b	b	a



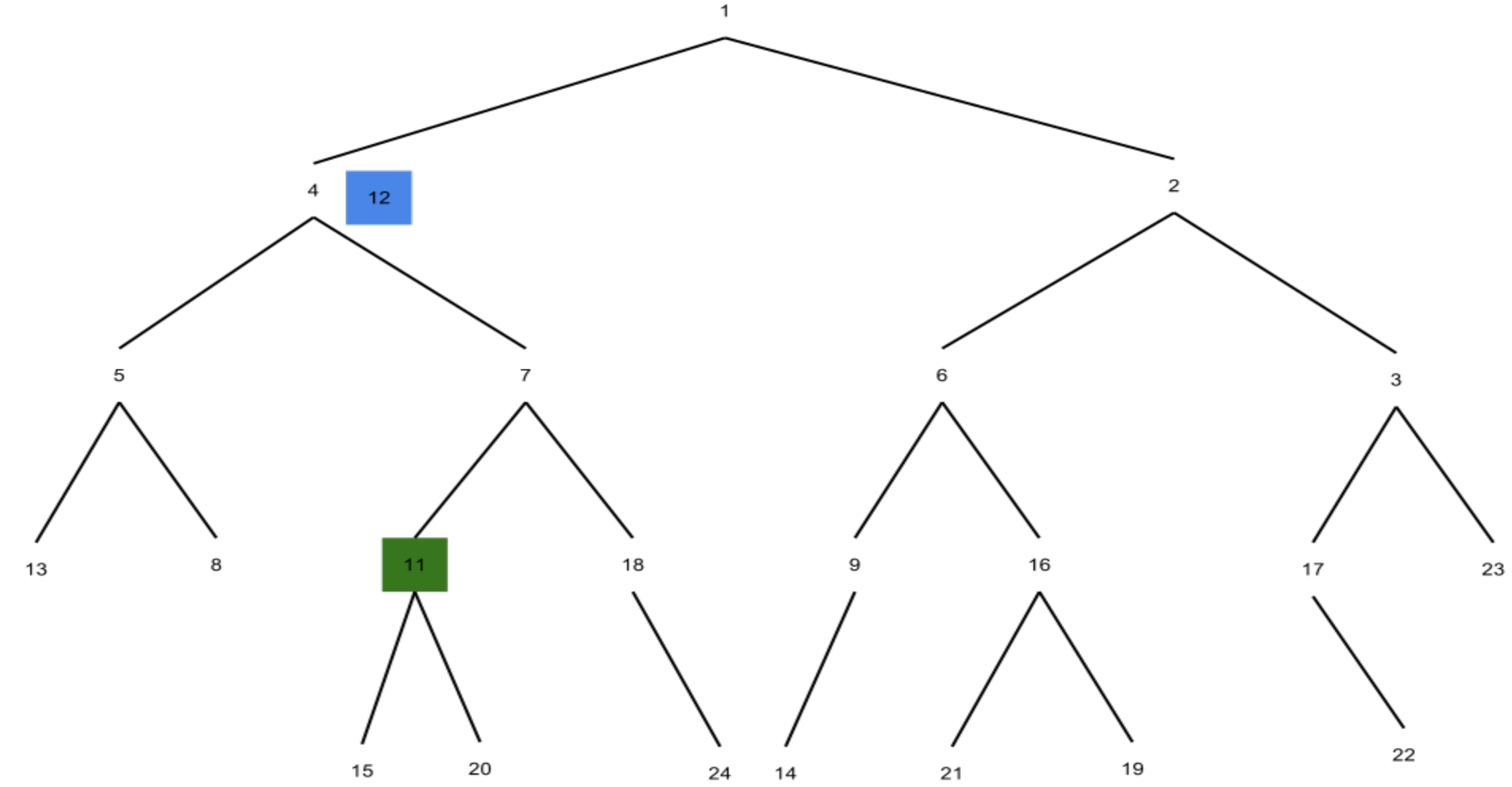
24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	a	a	b	a	a	b	b	a



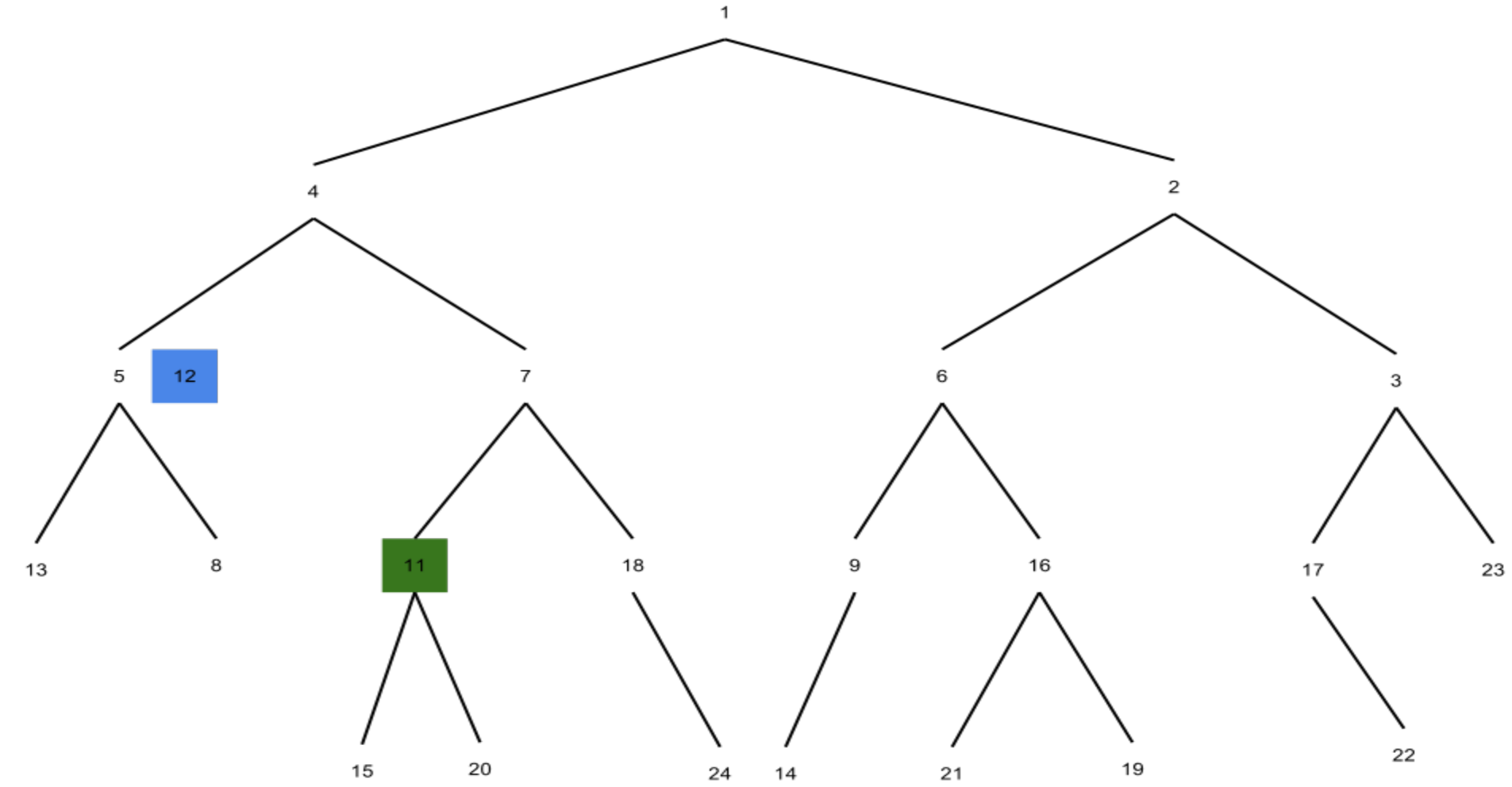
24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	a	a	b	a	a	b	b	a



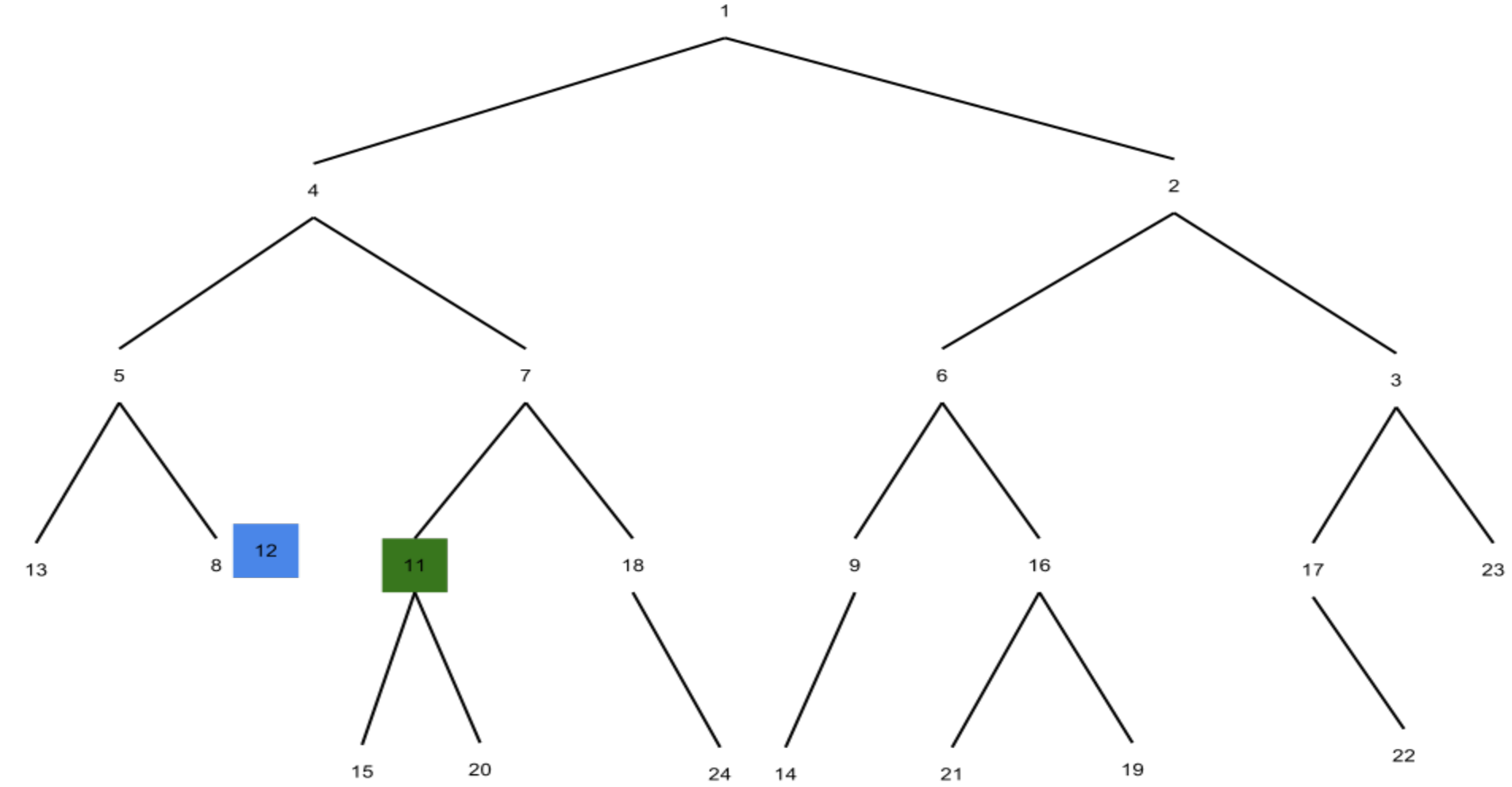
24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	a	a	b	a	a	b	b	a



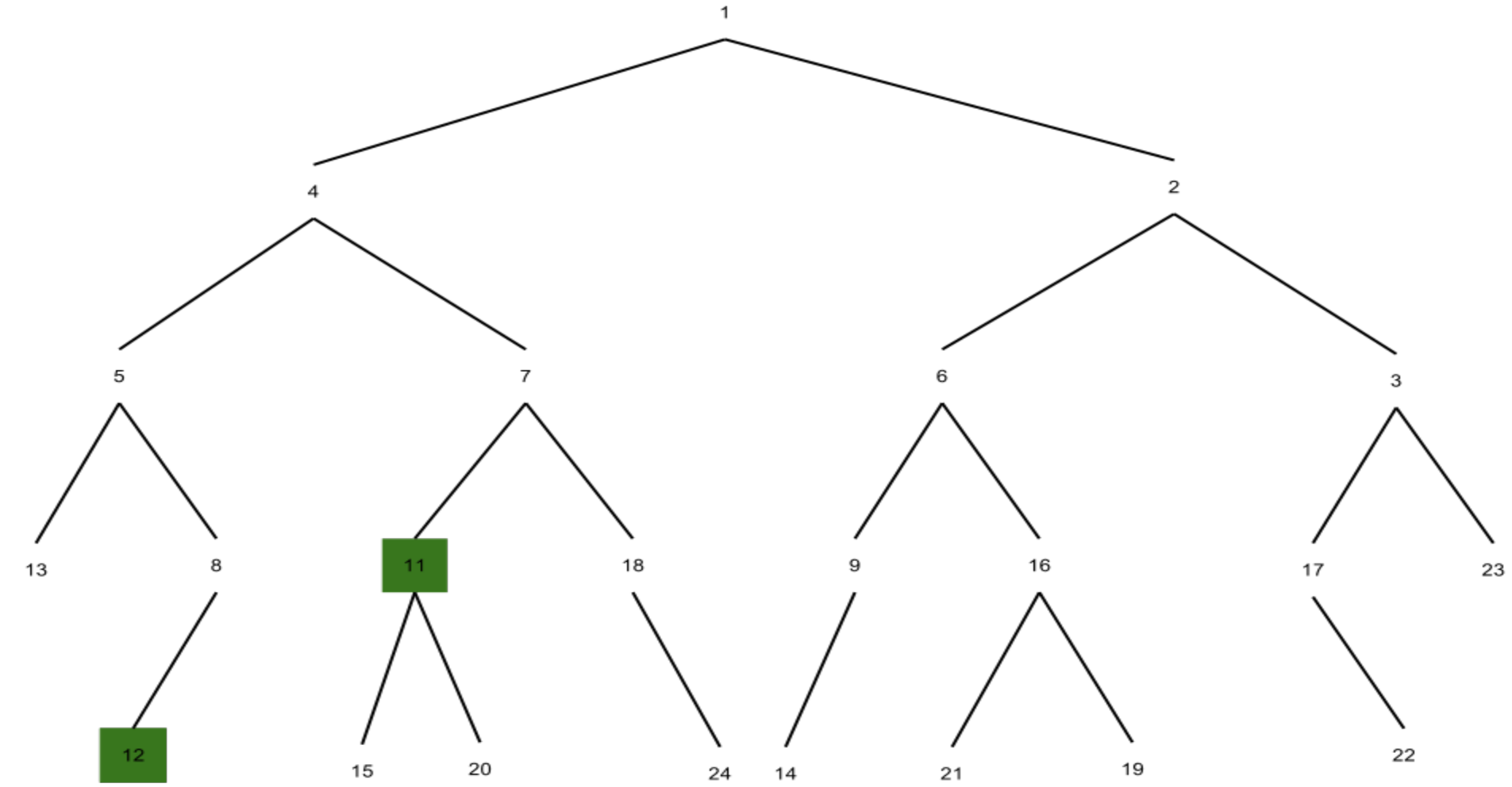
24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	a	a	b	a	a	b	b	a



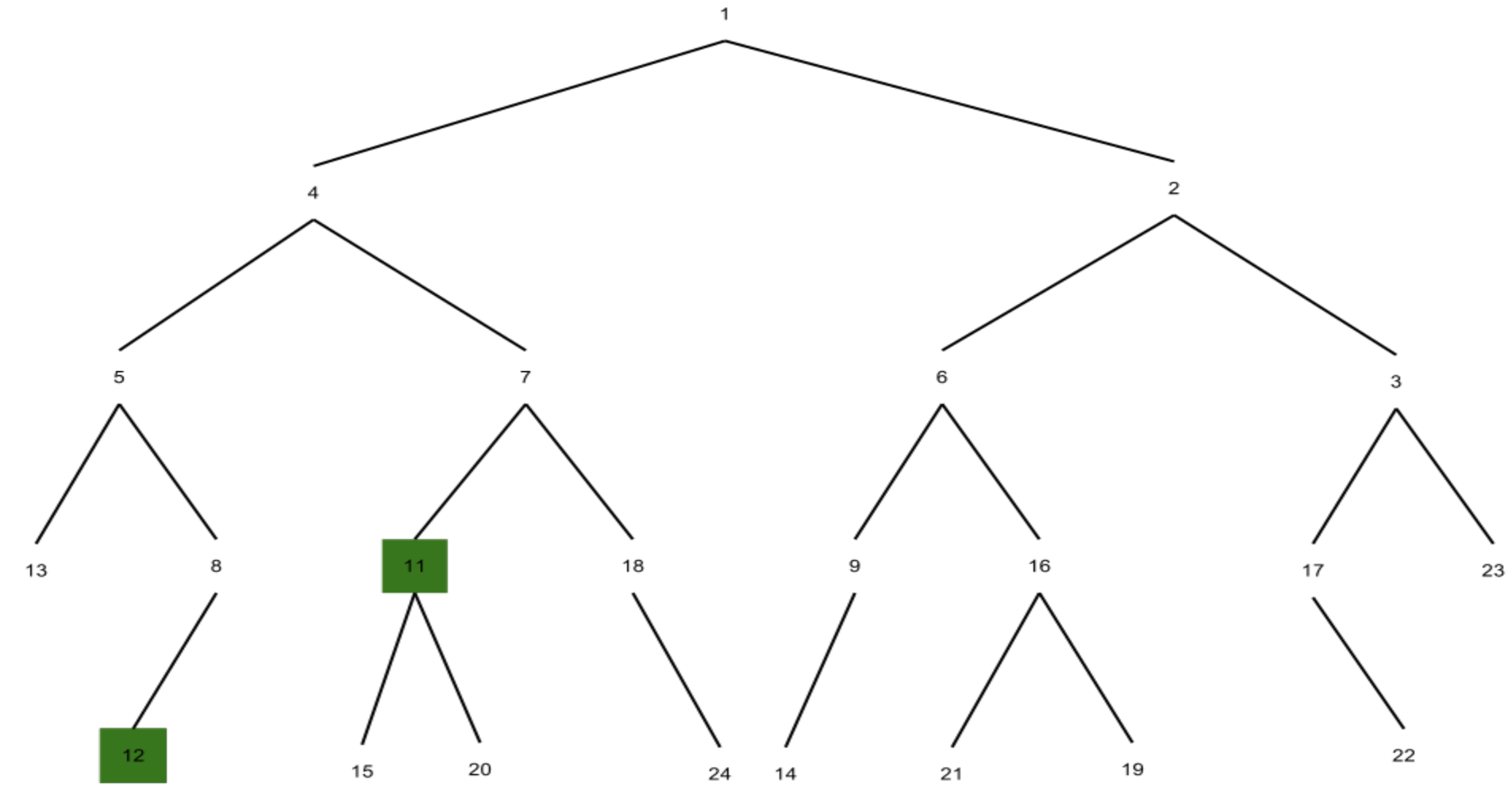
24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	a	a	b	a	a	b	b	a



24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	a	a	b	a	a	b	b	a



24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1
a	b	b	b	a	b	a	b	b	a	b	a	a	a	b	a	a	b	a	a	b	b	a

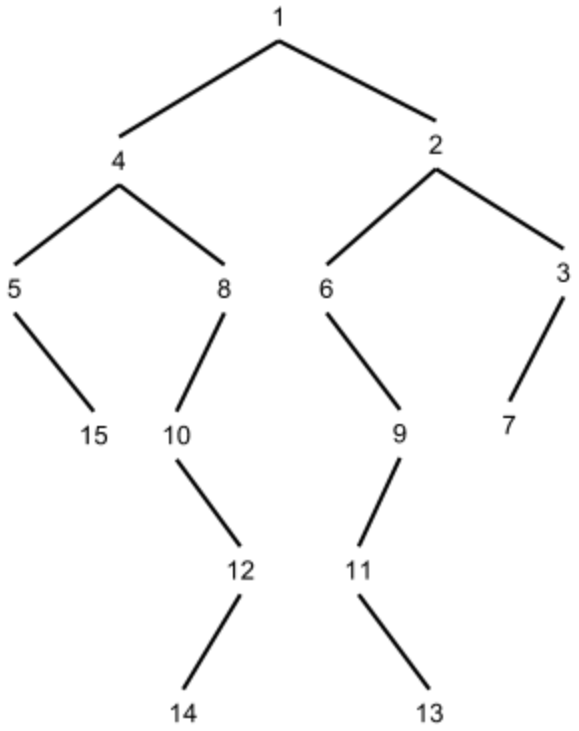


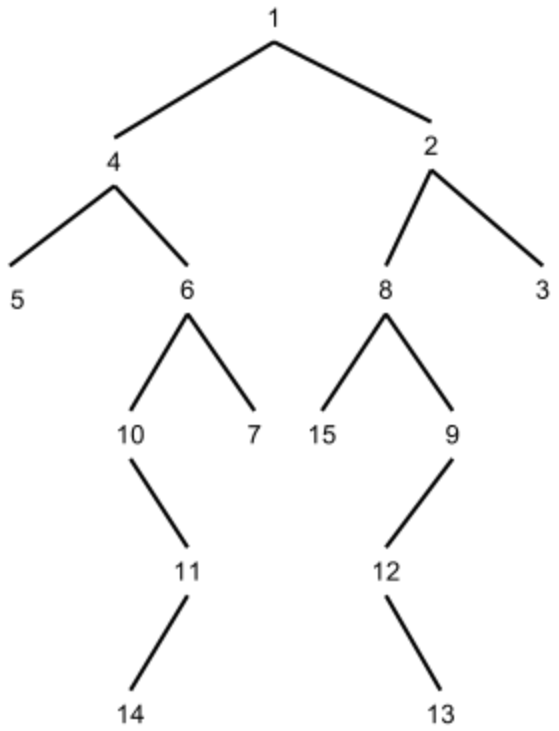
But construction took too long.

Dual Heaps

Definition: Let the **dual** D(T) of the position heap H(T) be the trie where for each node X of H(T), the **reverse** X^R of X is node of D(T).

Example of D(T)



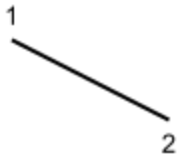


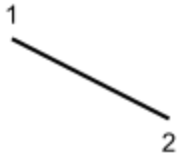
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a

We can build H(T) and D(T) in O(n)

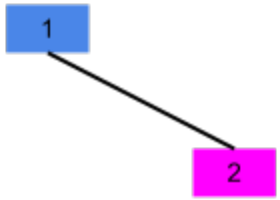
1

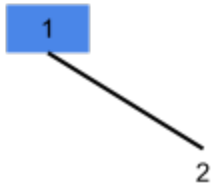
														x
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



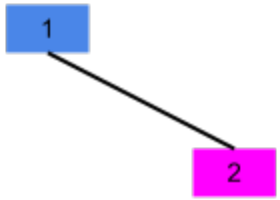


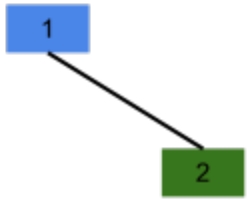
													x	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



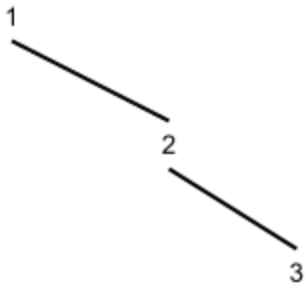


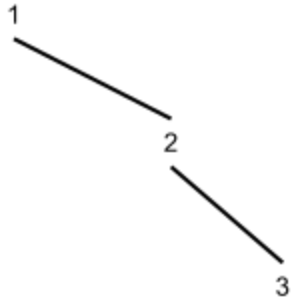
													x	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



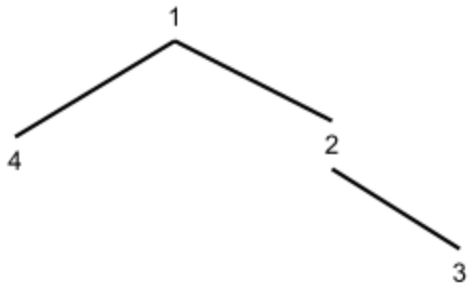


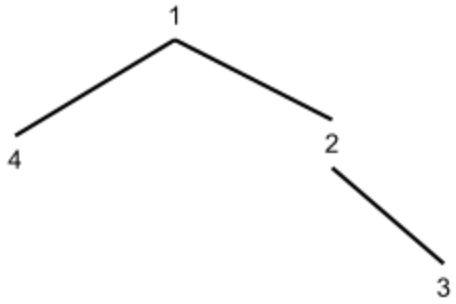
													x	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



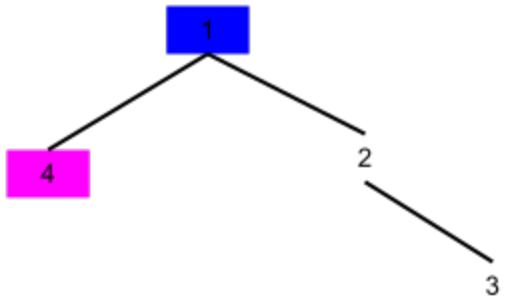


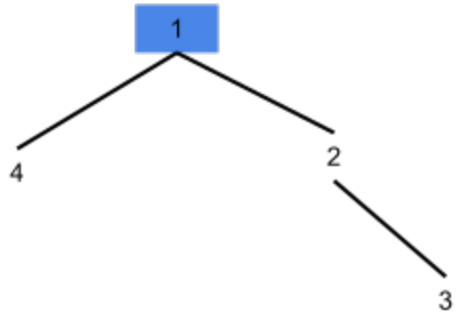
												x	x	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



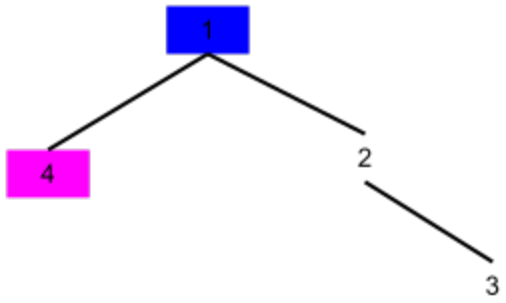


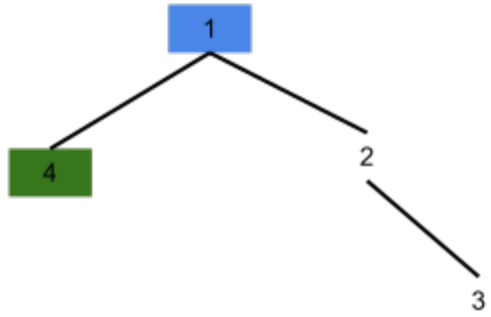
											x			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



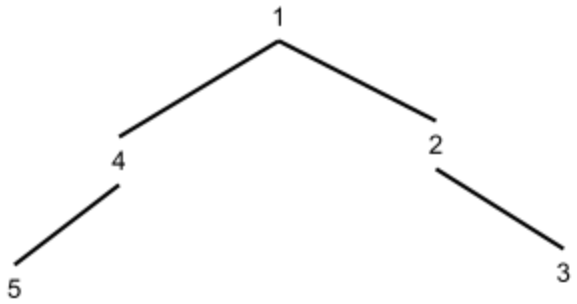


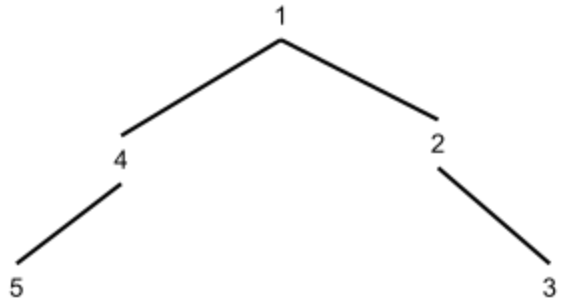
											x			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



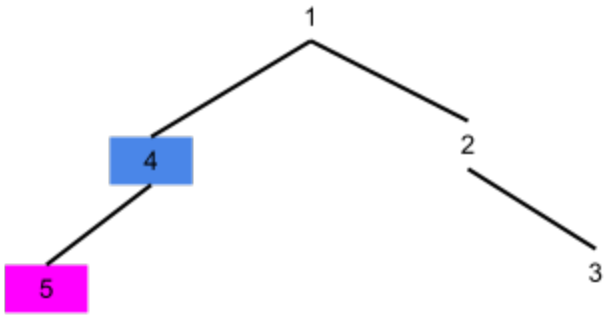


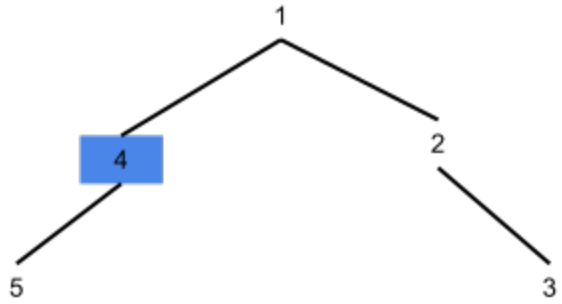
											x			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



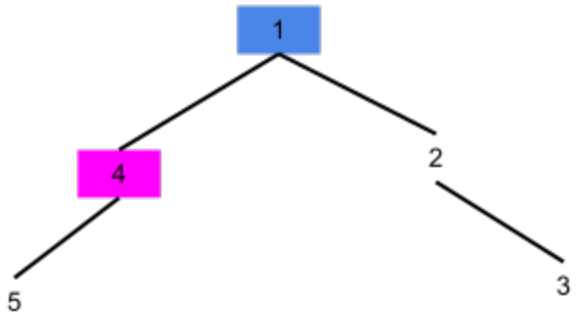


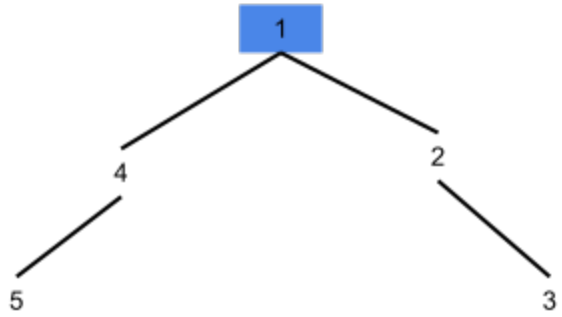
										x	x			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



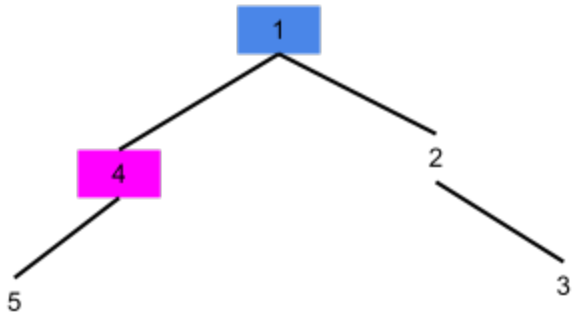


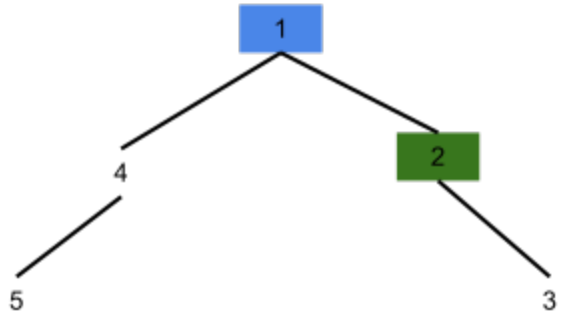
										x	x			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



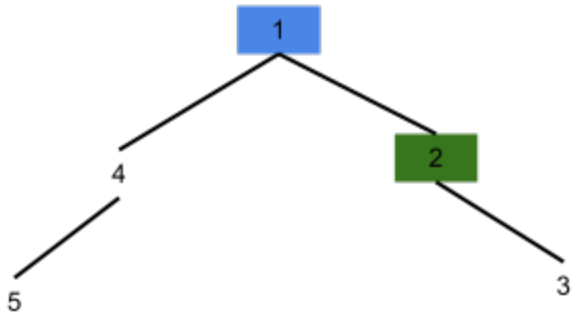


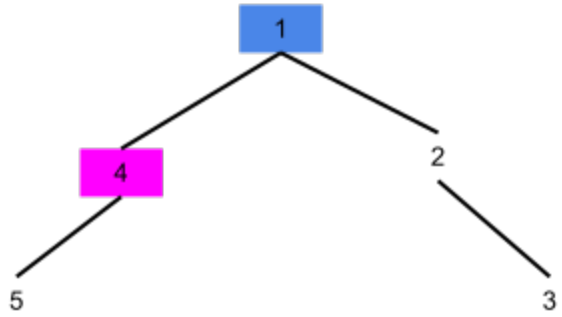
										x				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



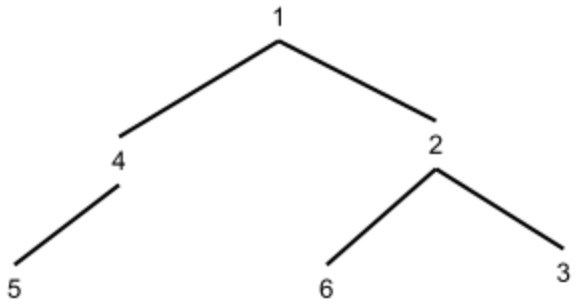


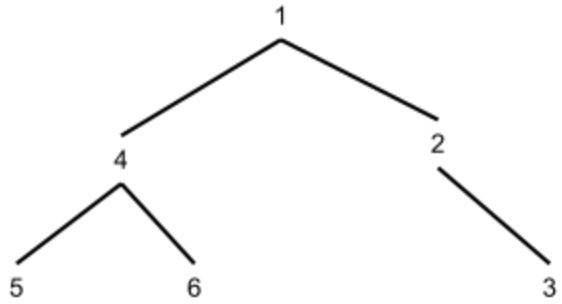
										x				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



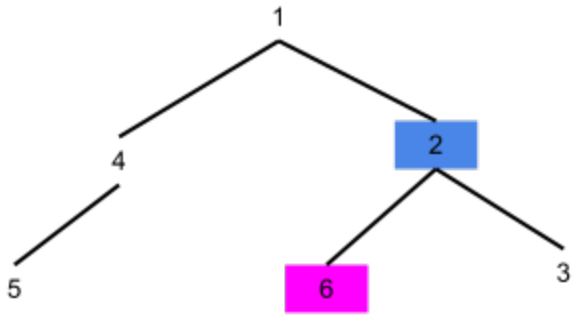


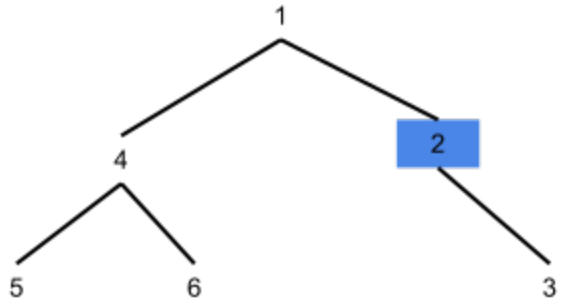
										x				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



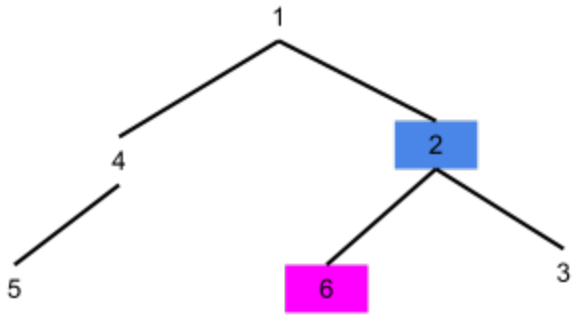


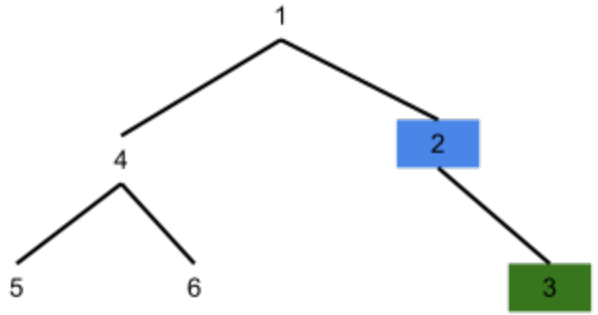
									x	x				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



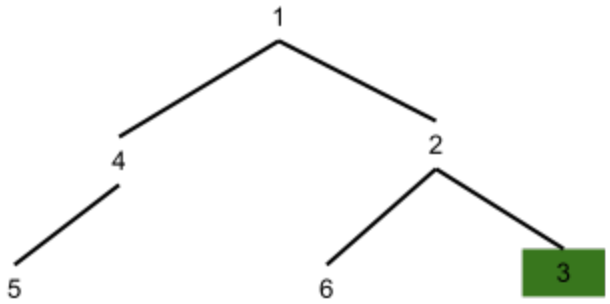


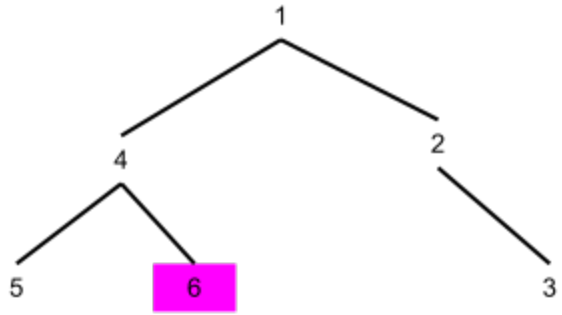
									x	x				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



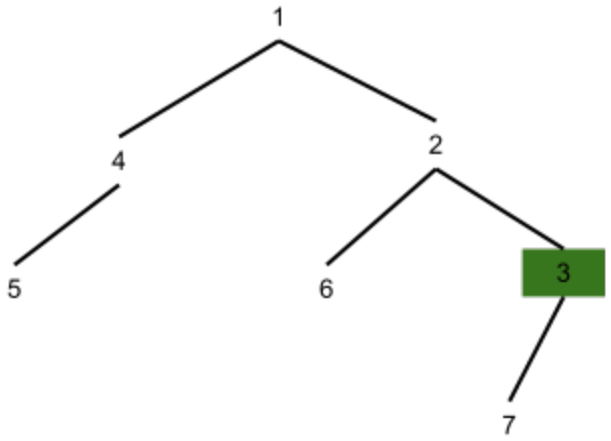


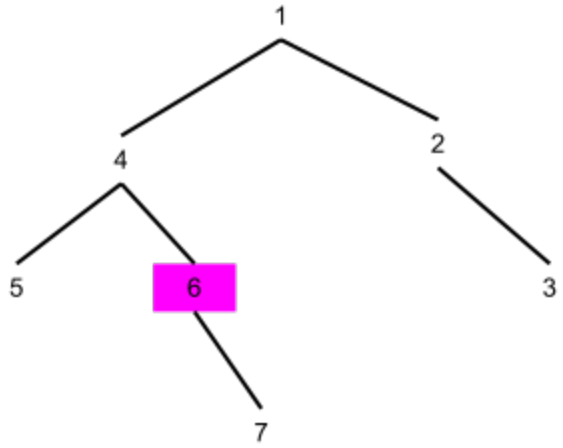
									x	x				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



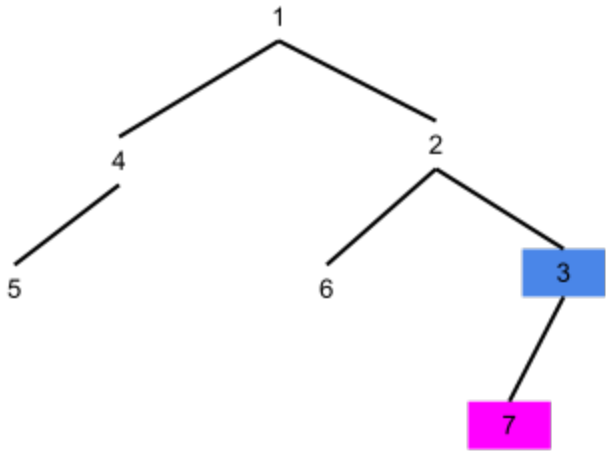


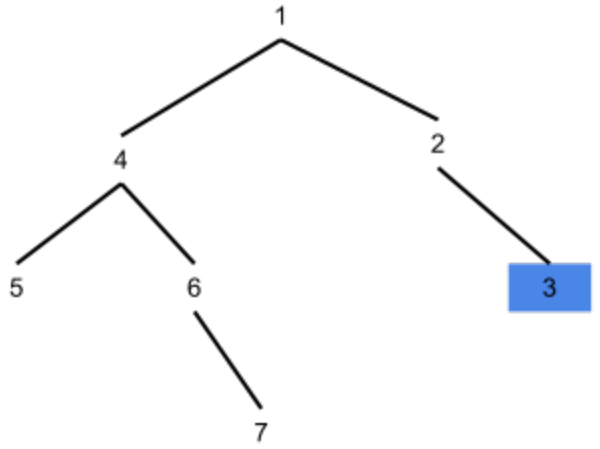
									x	x				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



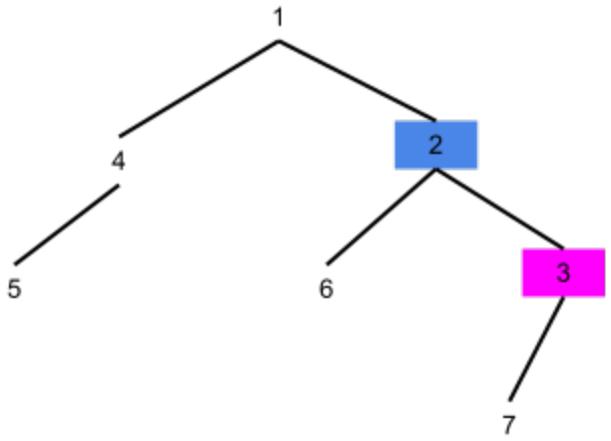


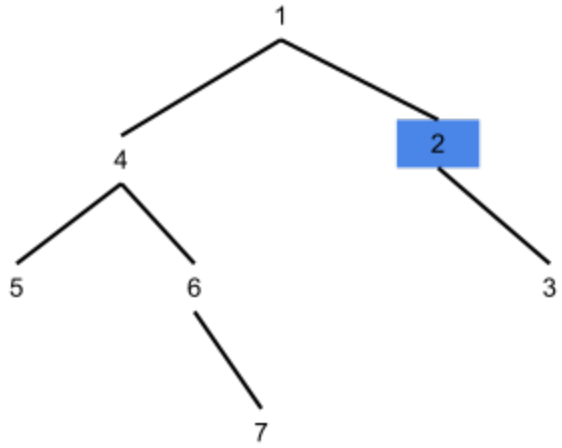
								x	x	x				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



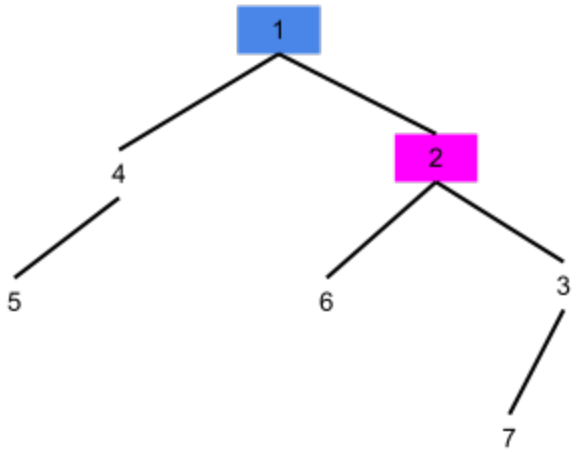


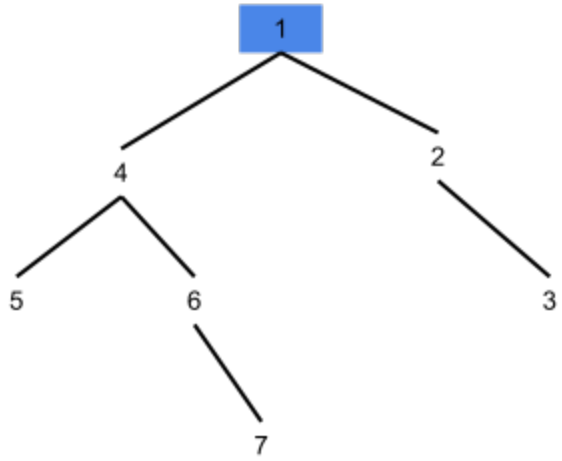
								x	x	x				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



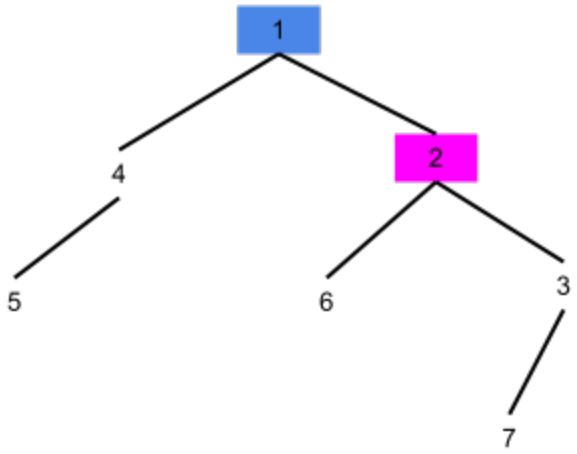


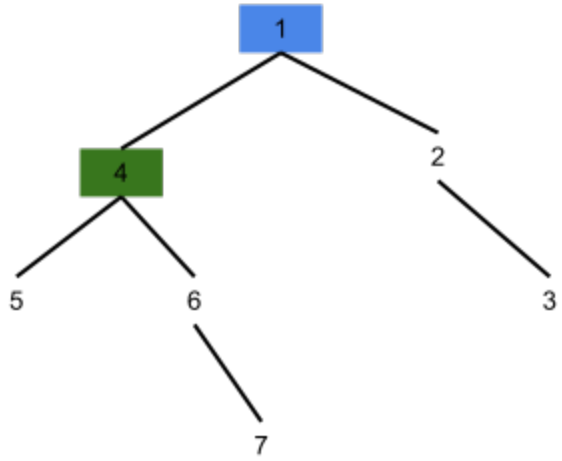
								x	x					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



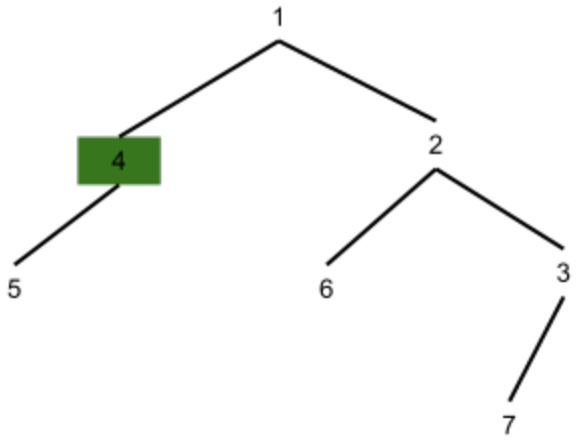


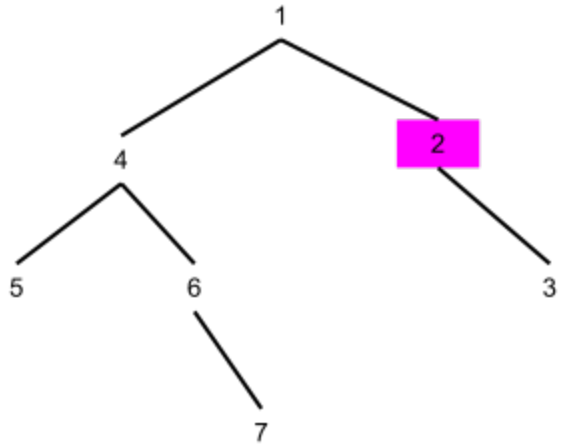
								x						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



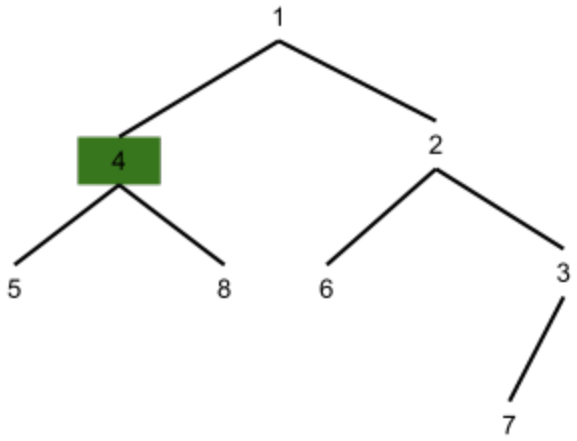


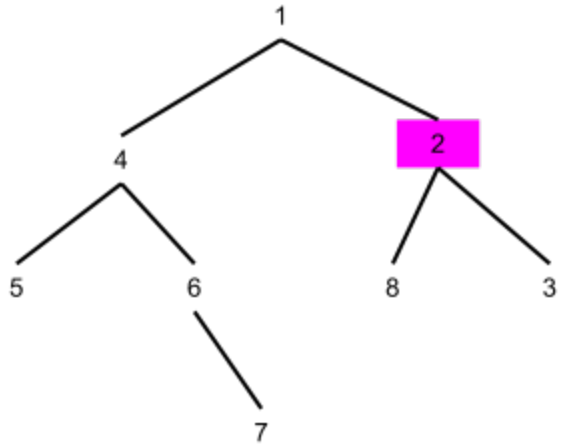
								x						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



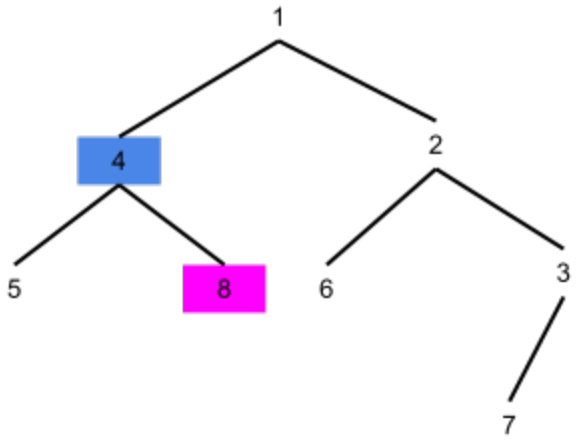


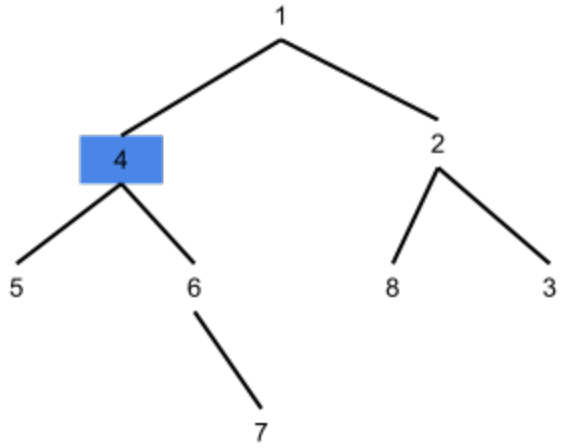
								x						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



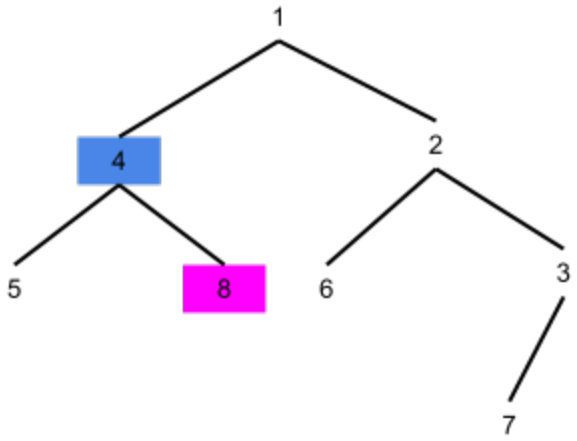


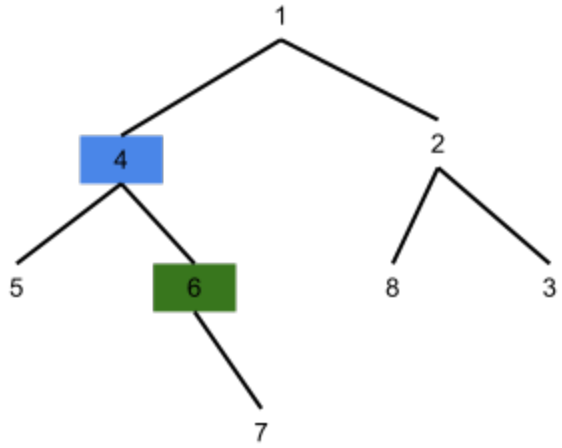
							x	x						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



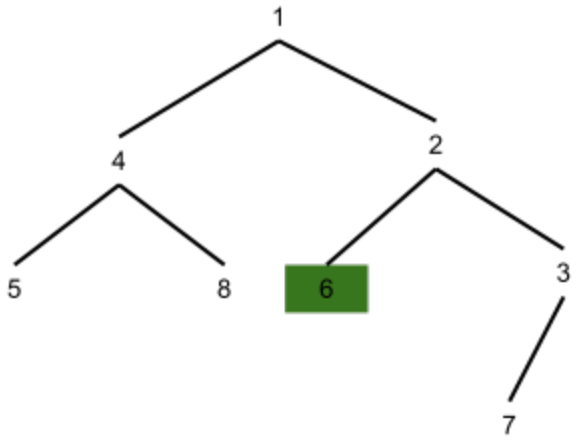


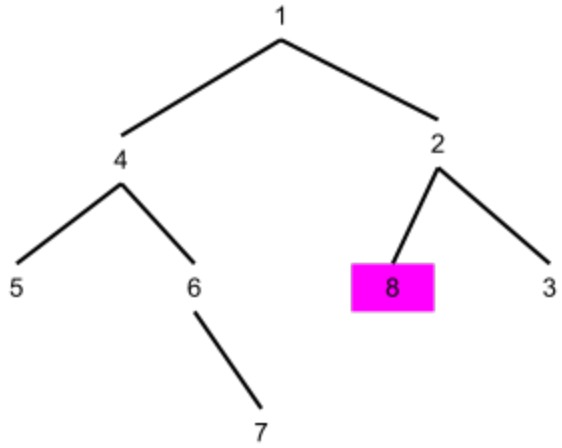
							x	x						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



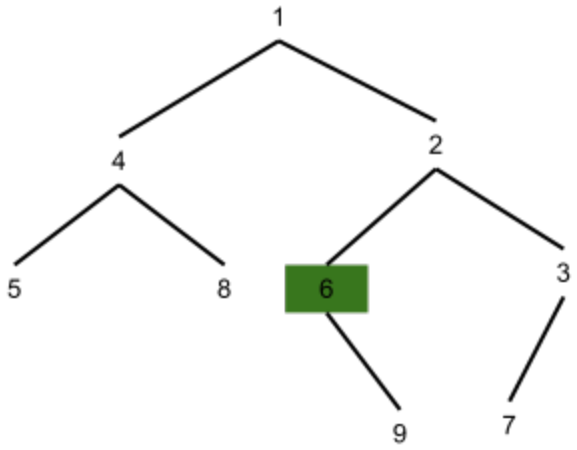


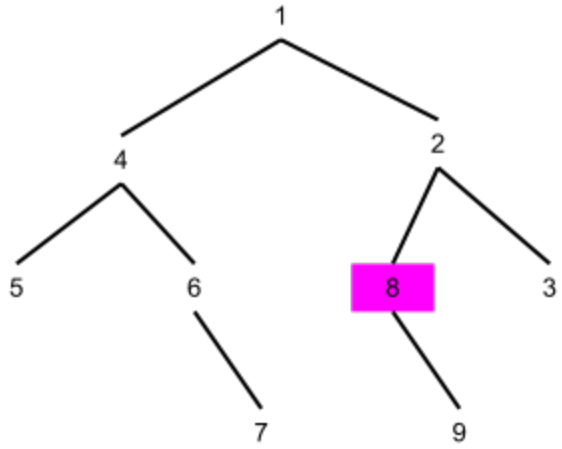
							x	x						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



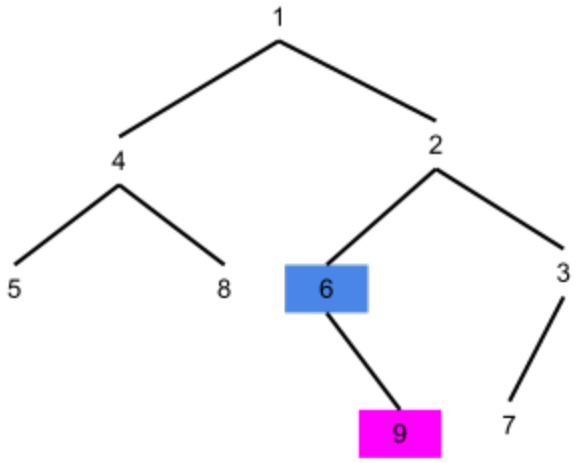


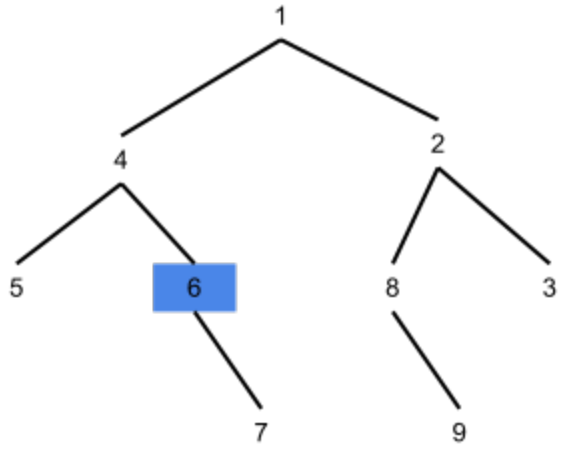
							x	x						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



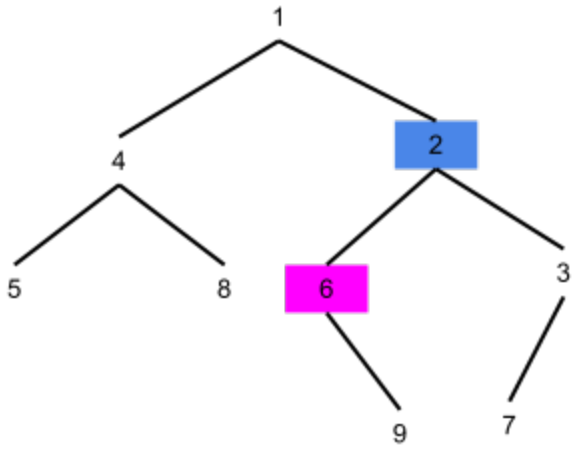


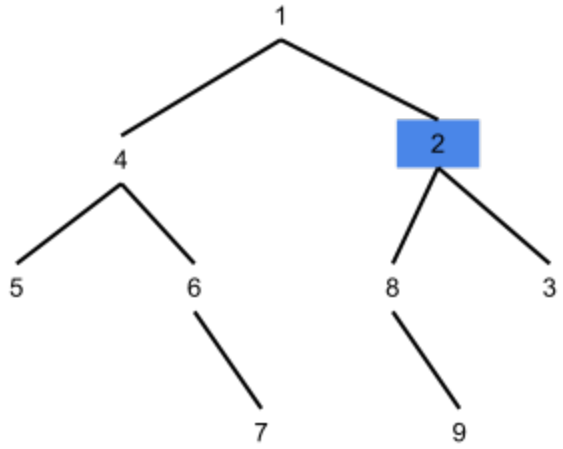
						x	x	x						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



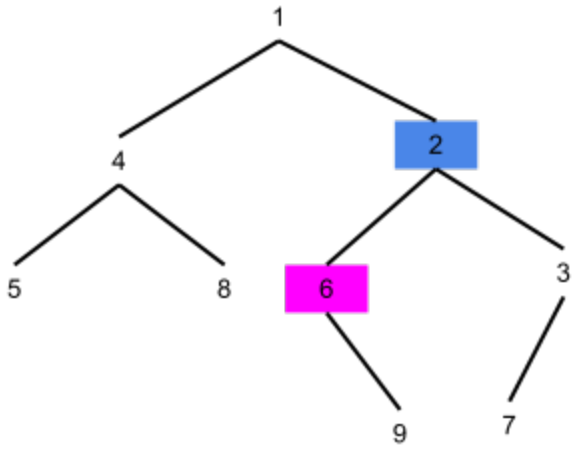


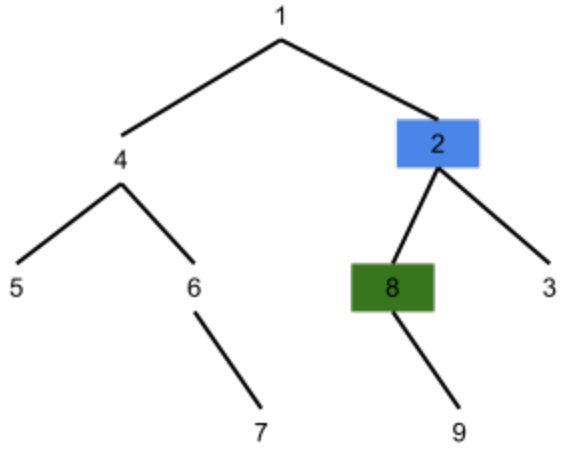
						x	x	x						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



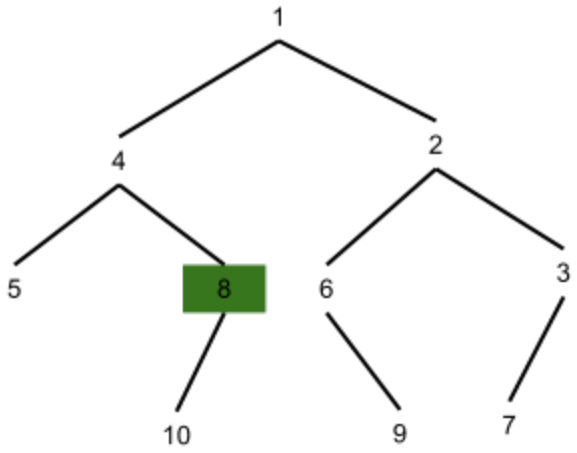


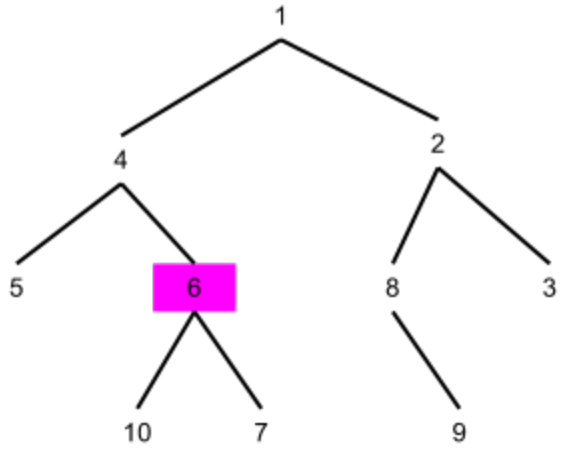
						x	x							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



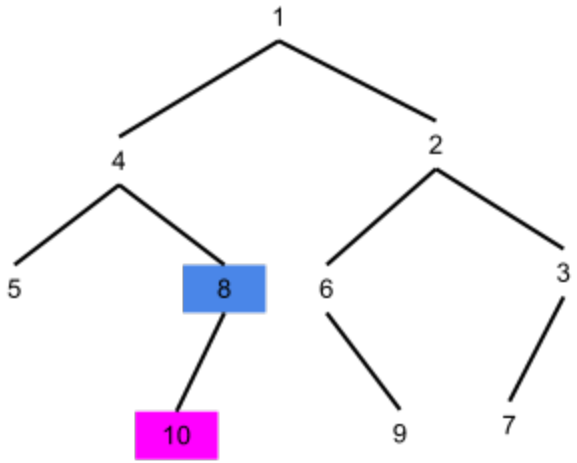


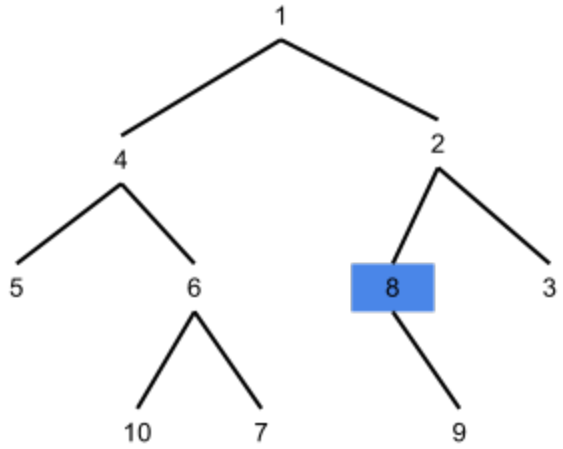
						x	x							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



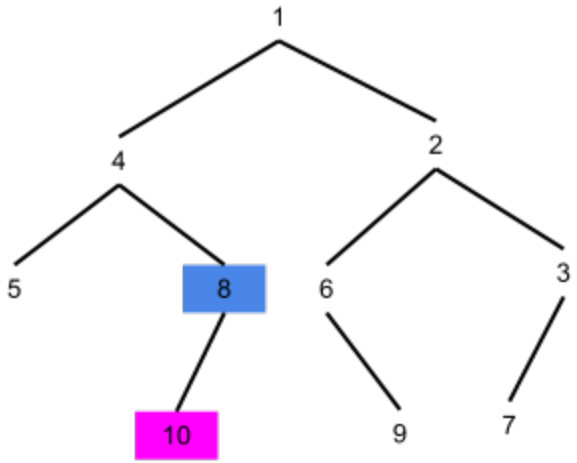


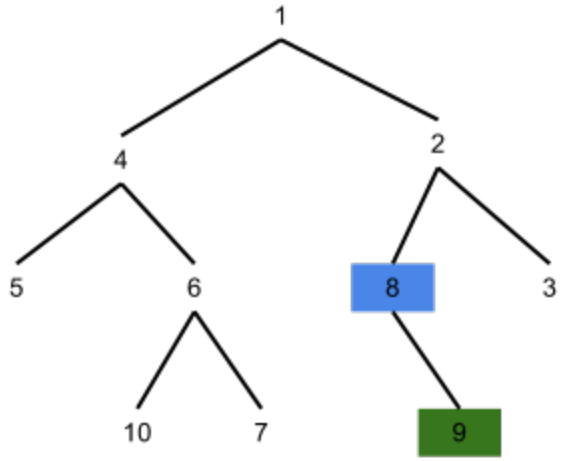
					x	x	x							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



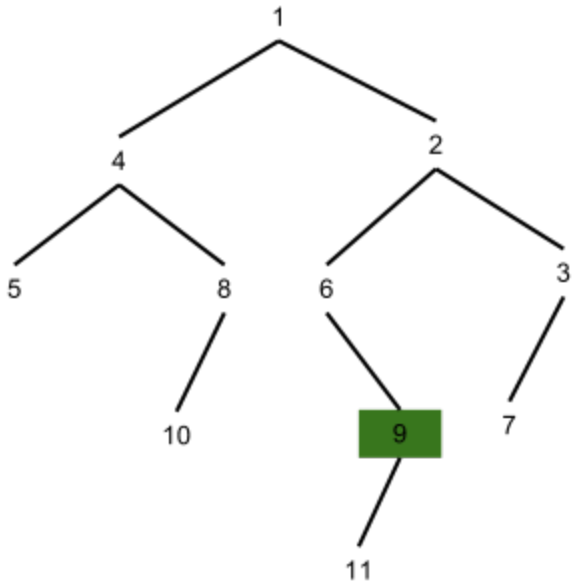


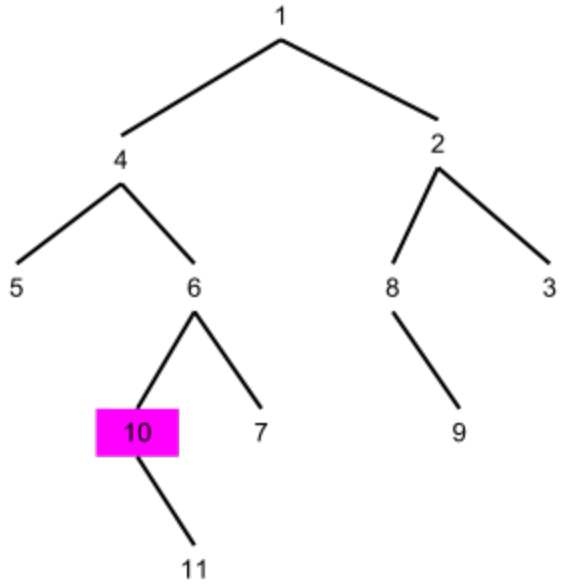
					x	x	x							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



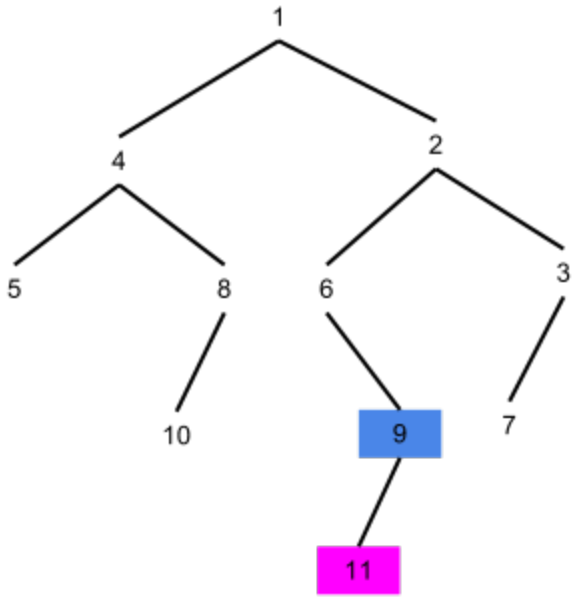


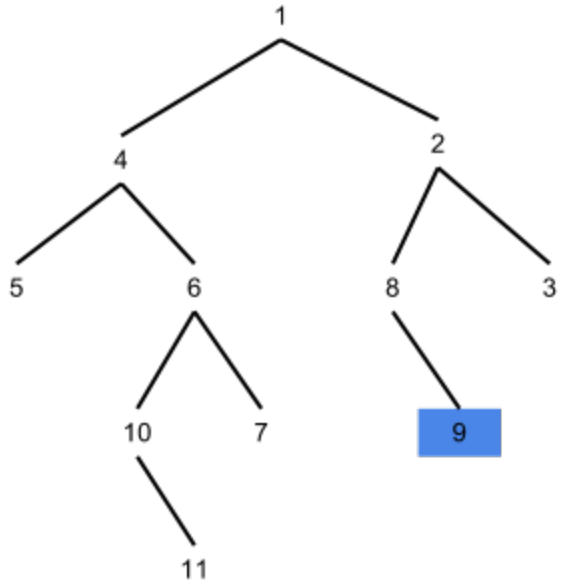
					x	x	x							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



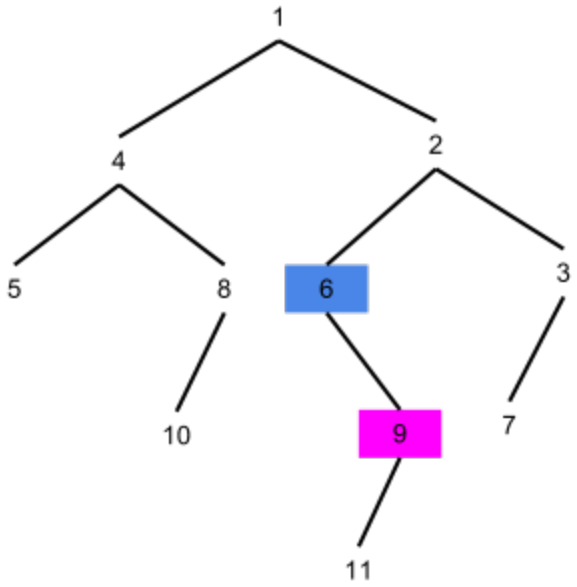


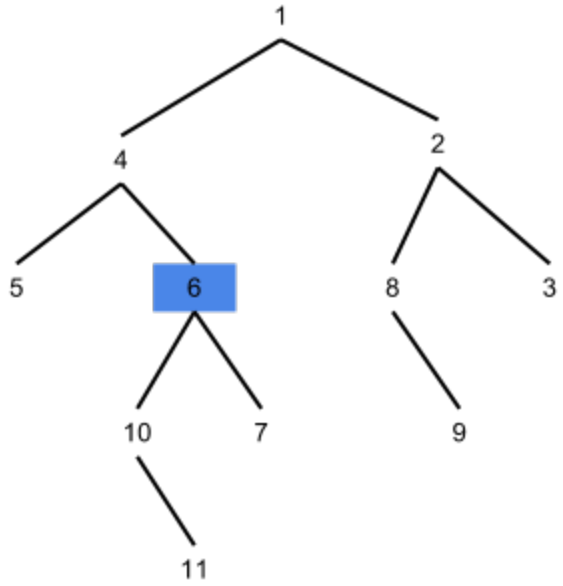
				x	x	x	x							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



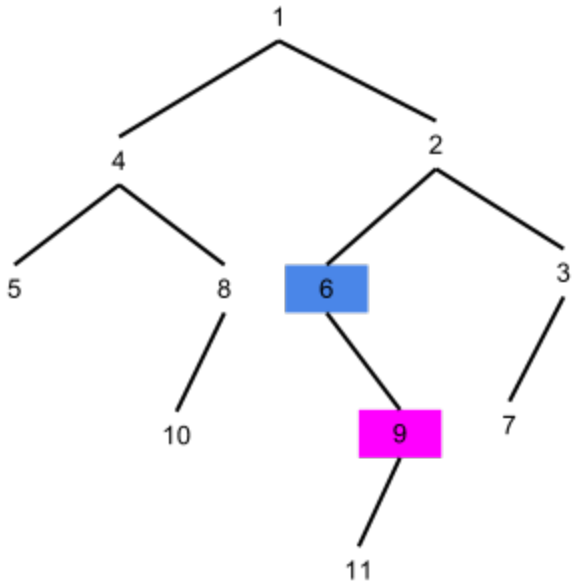


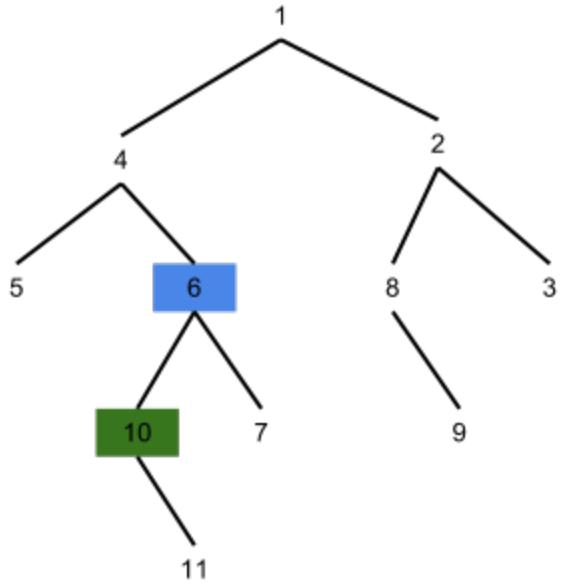
				x	x	x	x							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



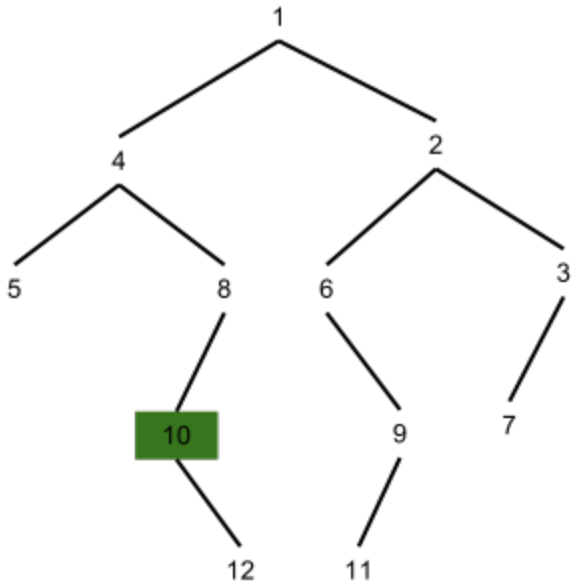


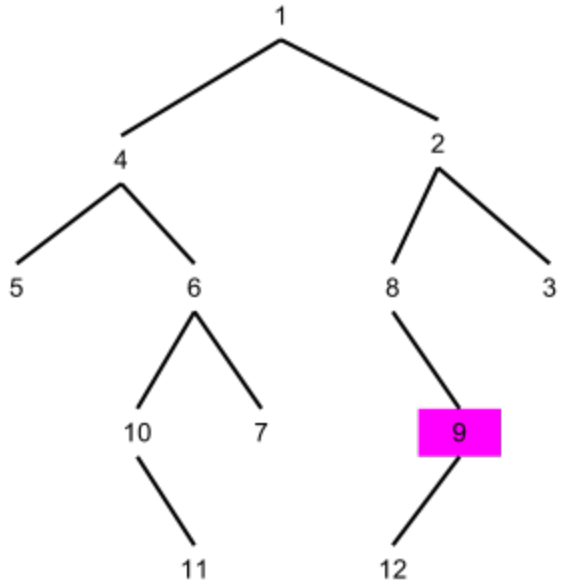
				x	x	x								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



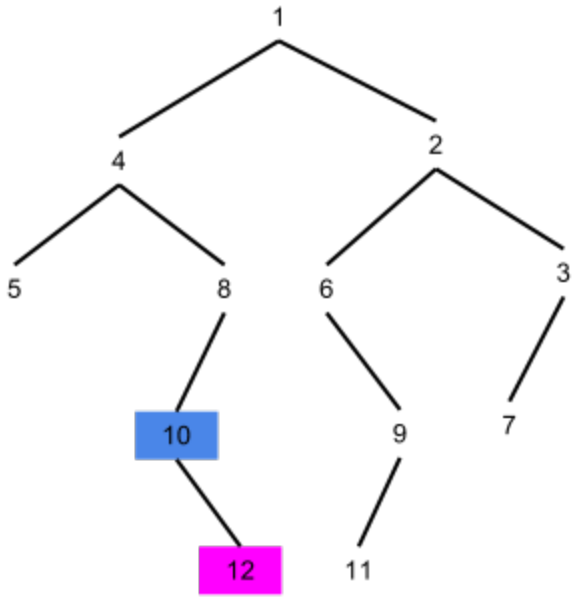


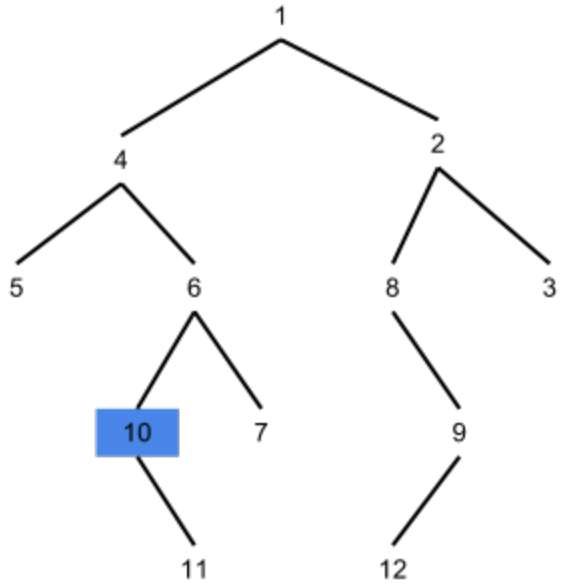
				x	x	x								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



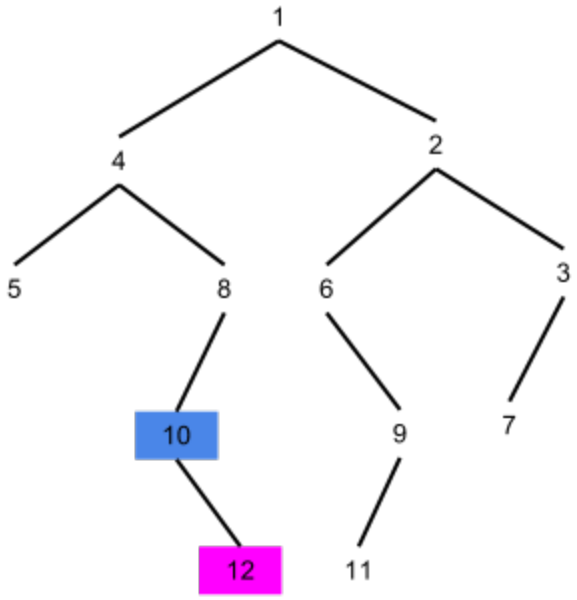


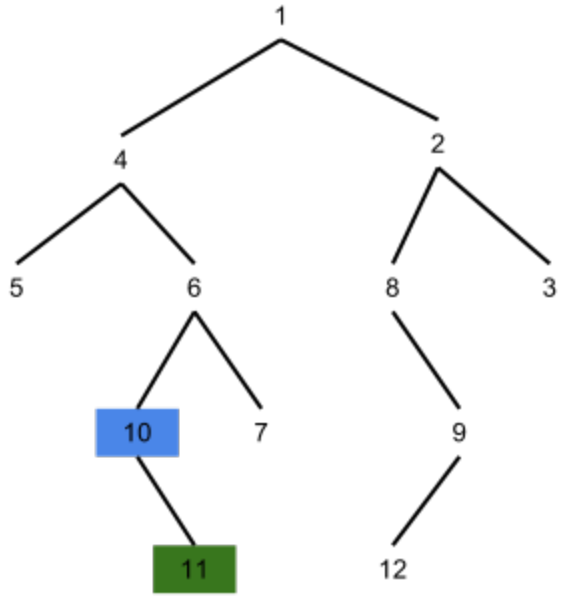
			x	x	x	x								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



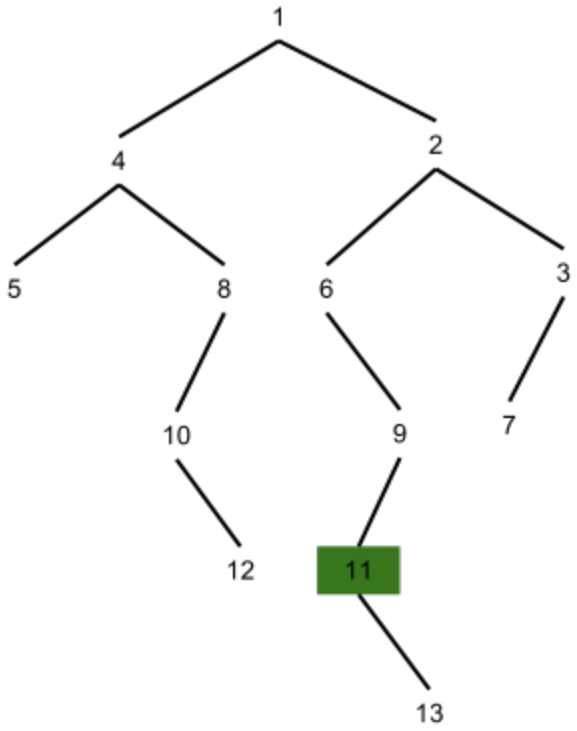


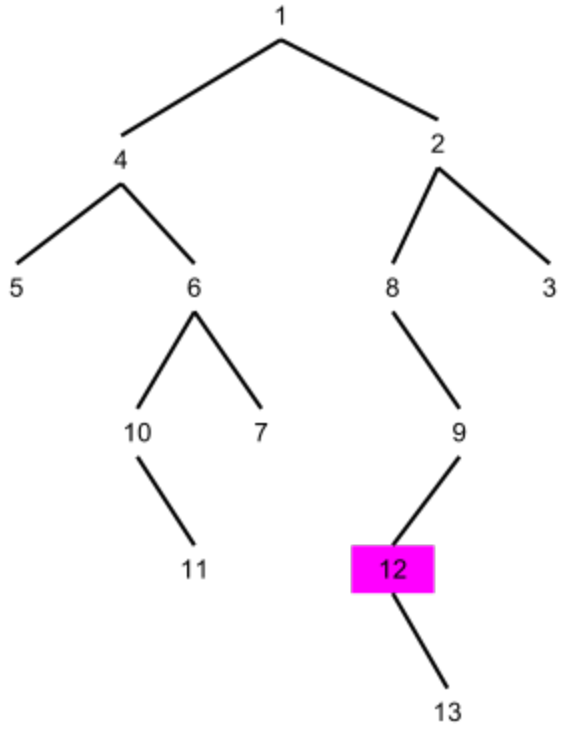
			x	x	x	x								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



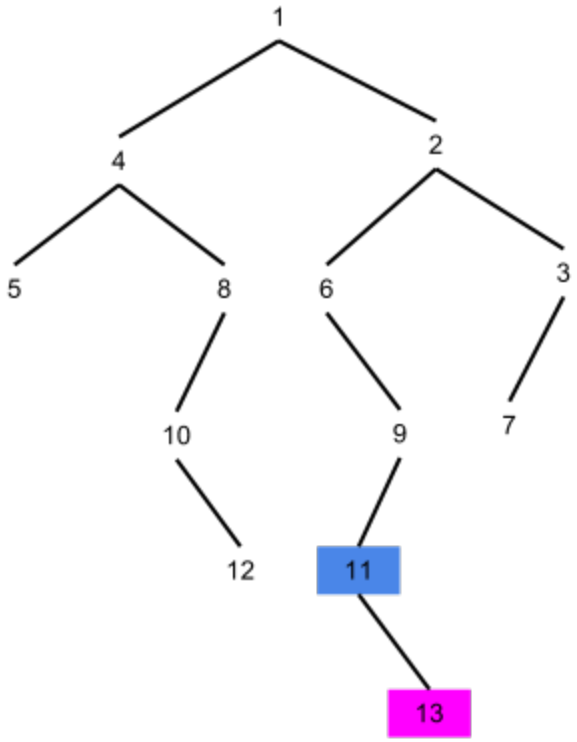


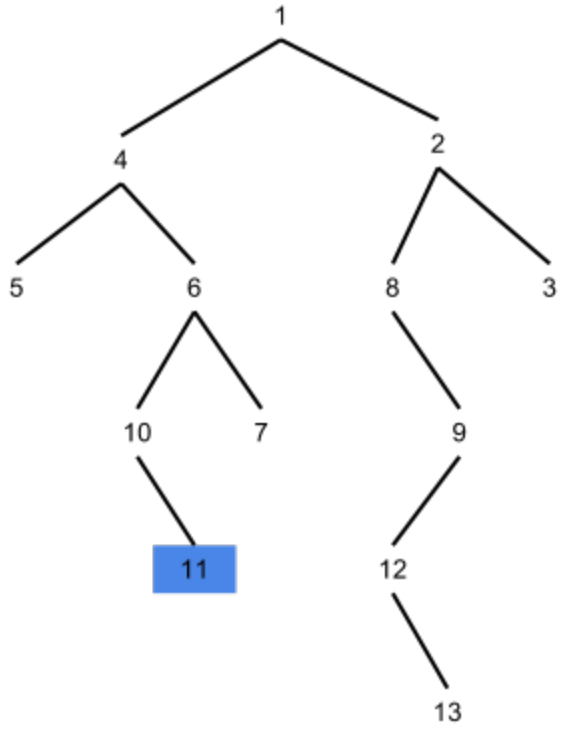
			x	x	x	x								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



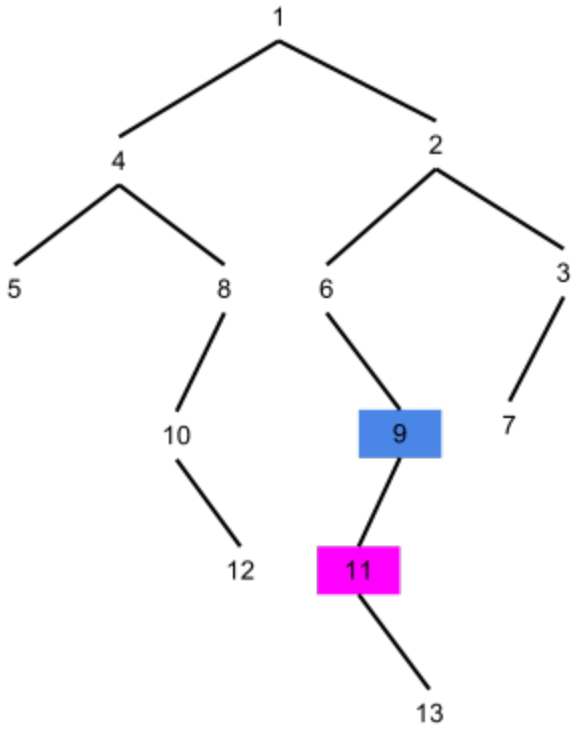


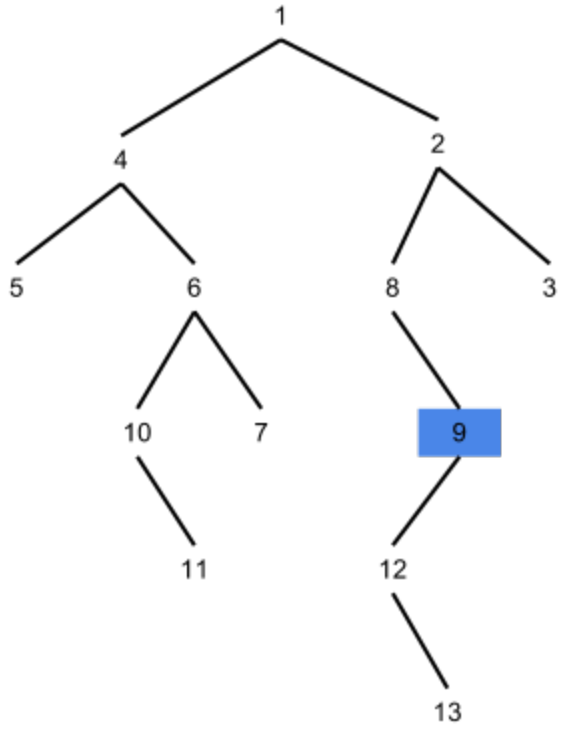
		x	x	x	x	x								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



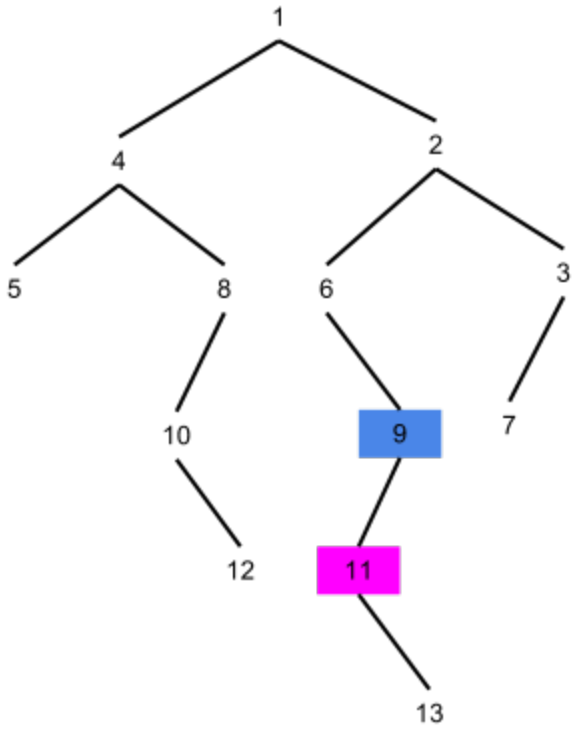


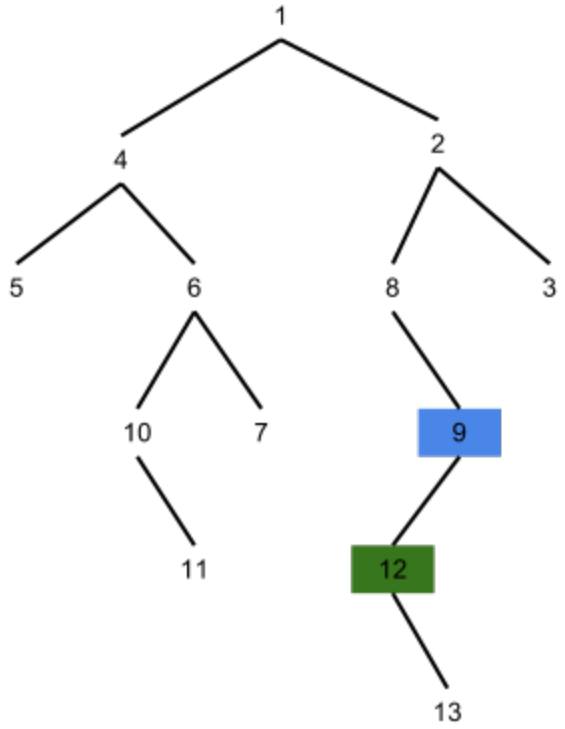
		x	x	x	x	x								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



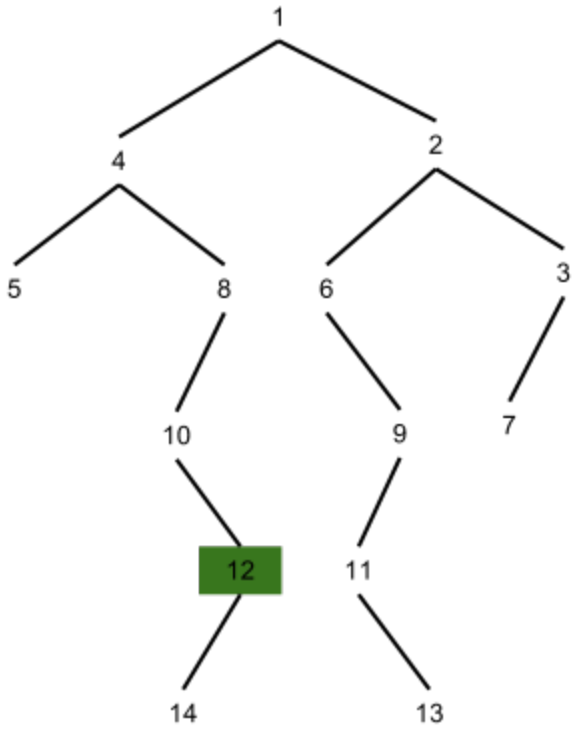


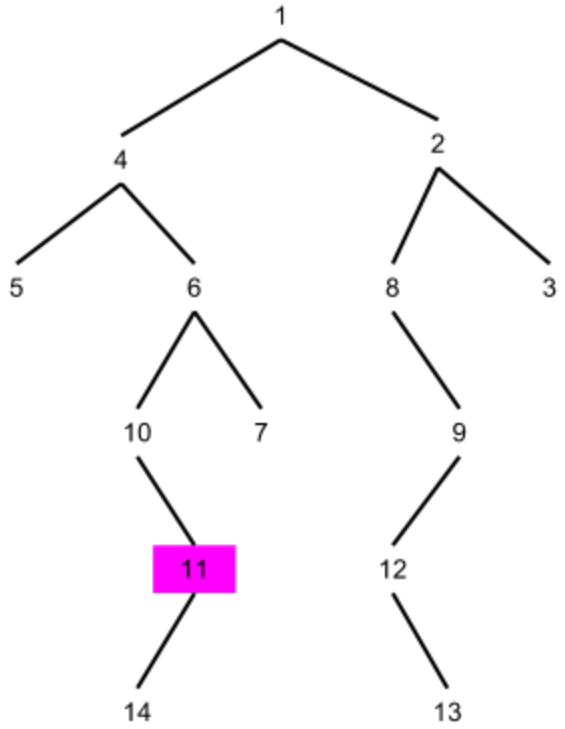
		x	x	x	x									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



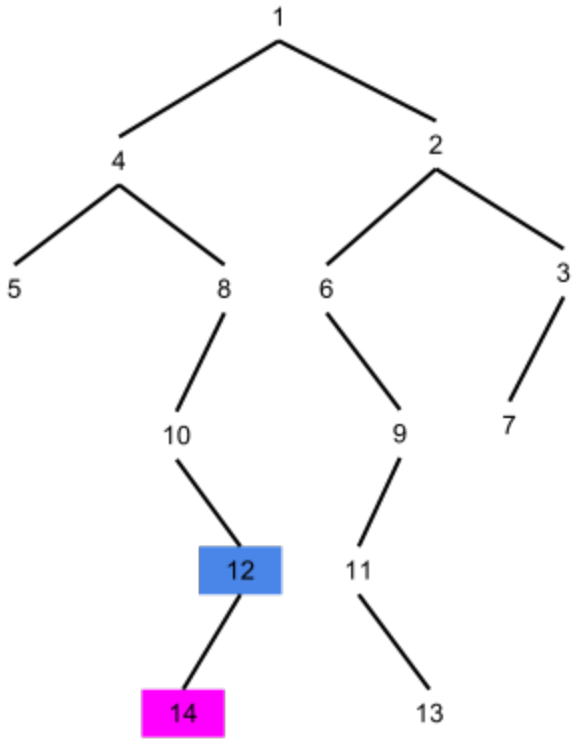


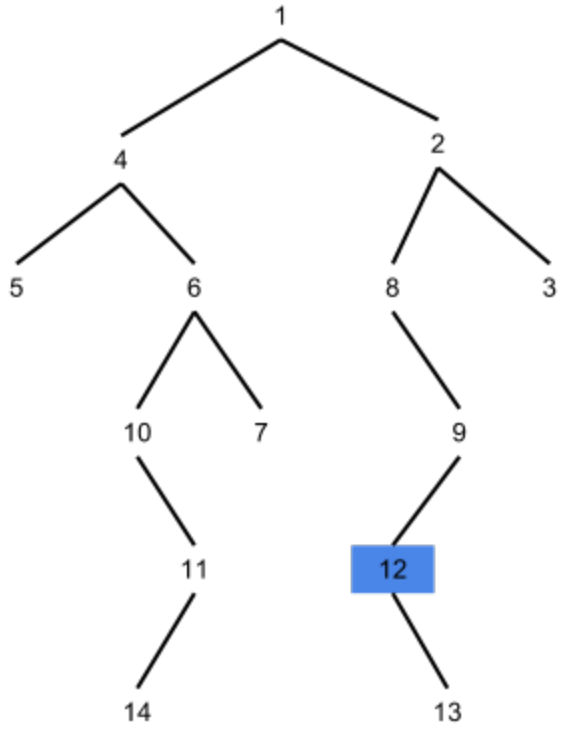
		x	x	x	x									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



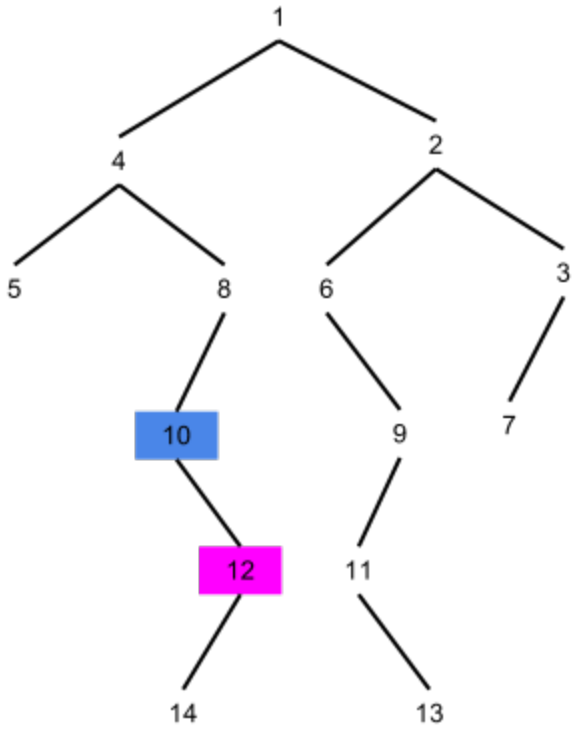


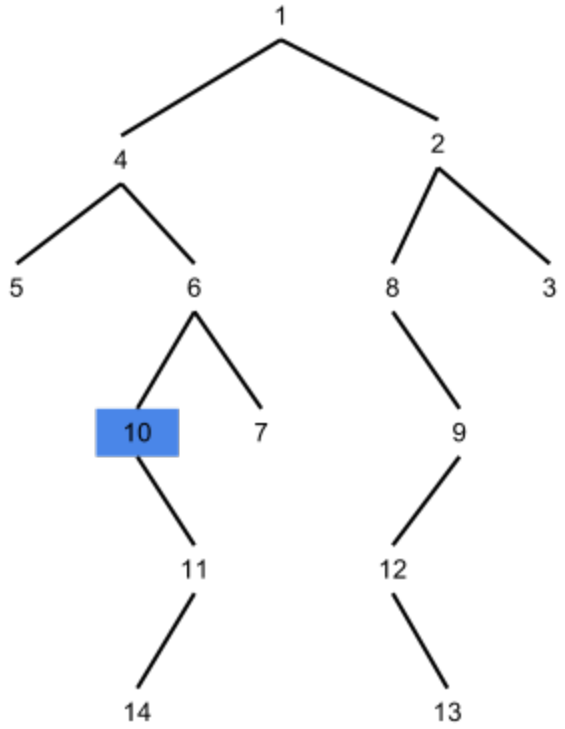
	x	x	x	x	x									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



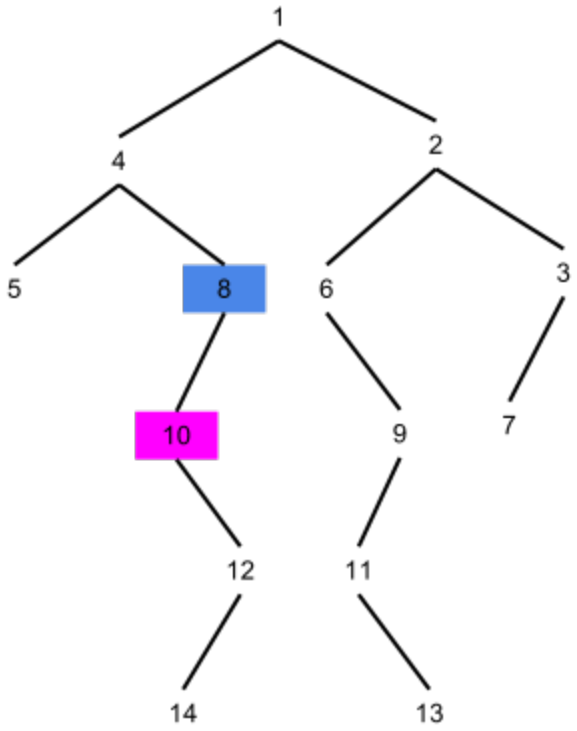


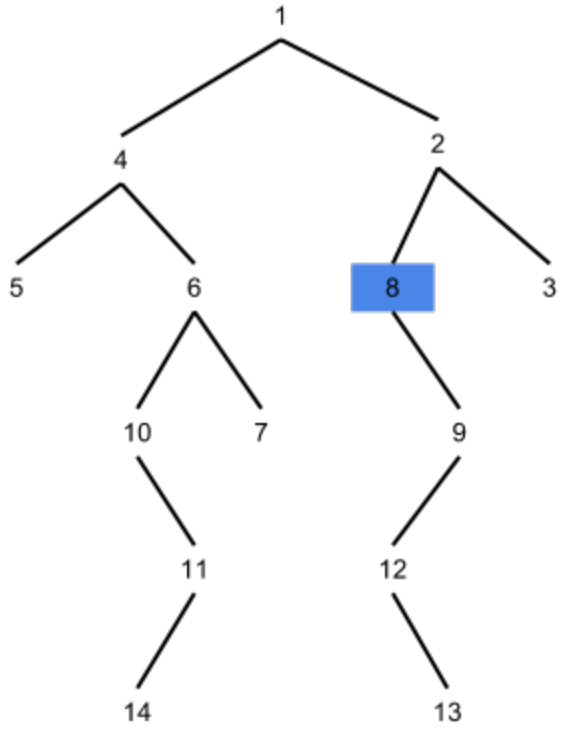
	x	x	x	x	x									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



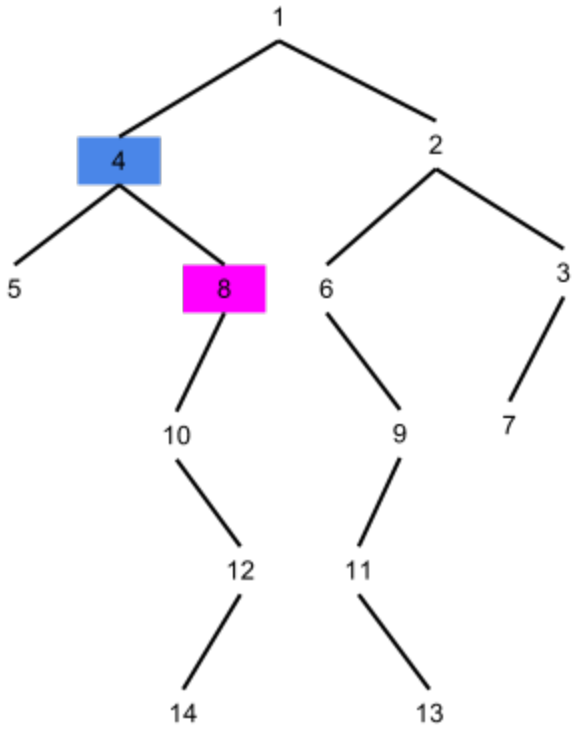


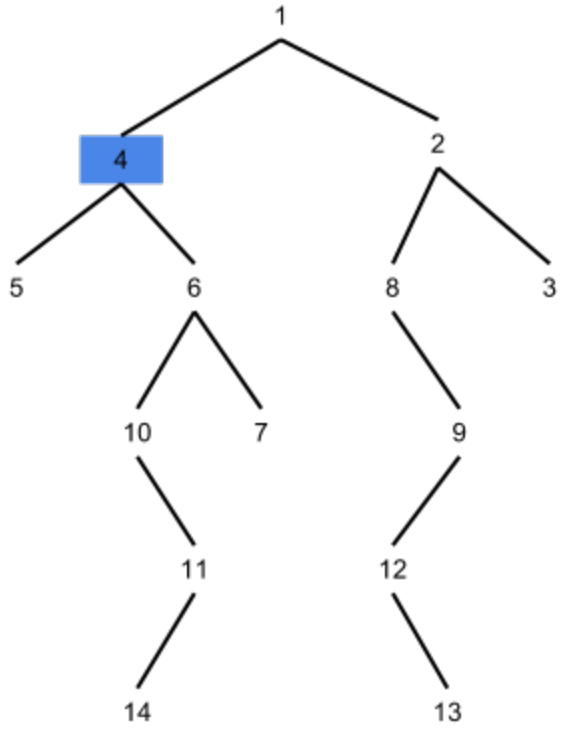
	x	x	x	x										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



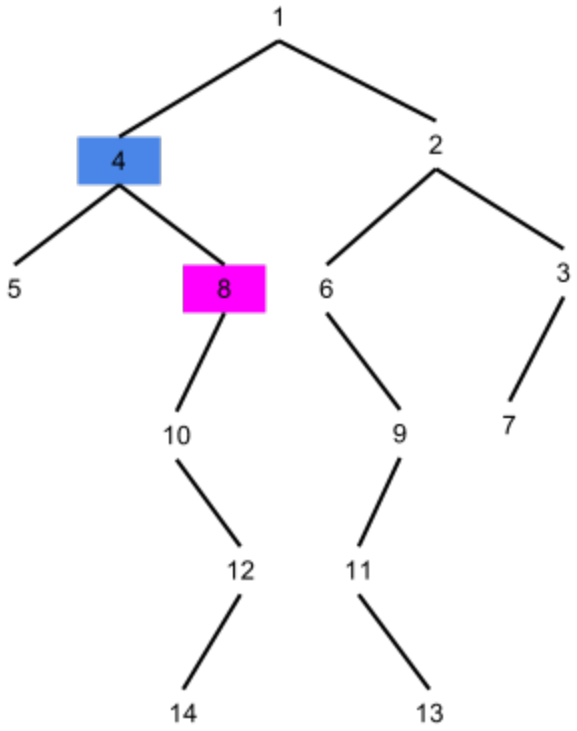


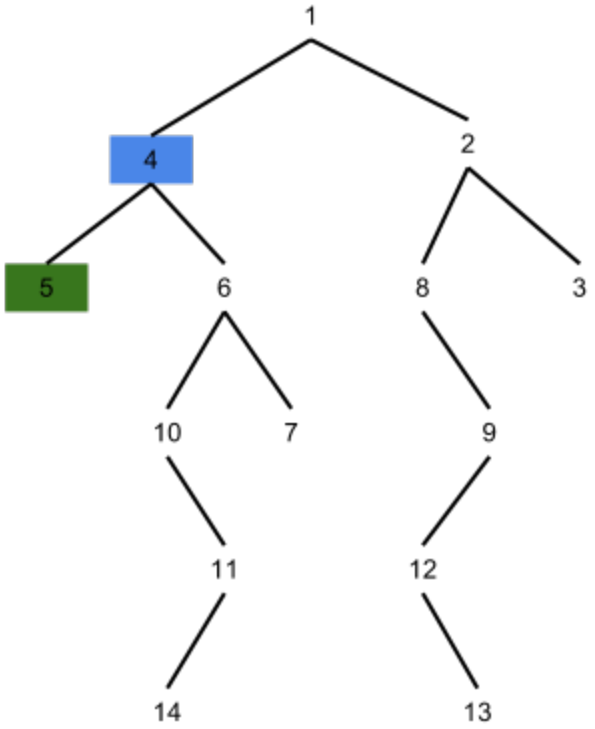
	x	x	x											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a



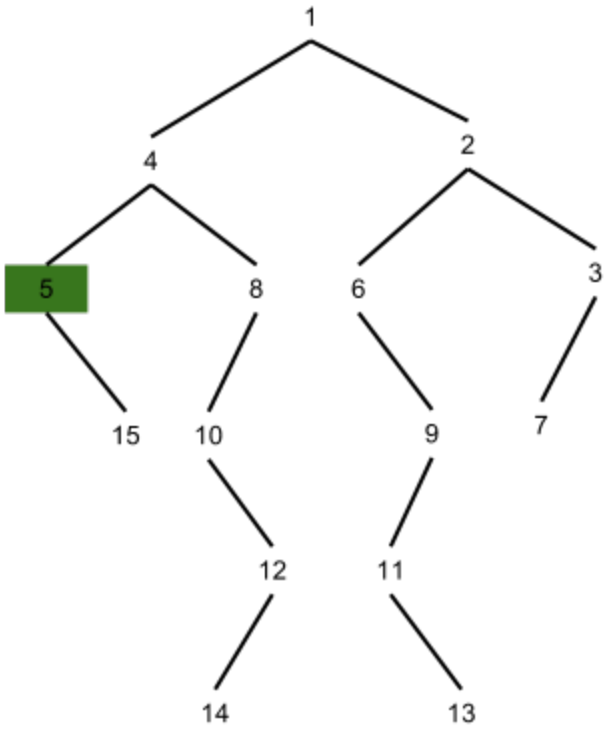


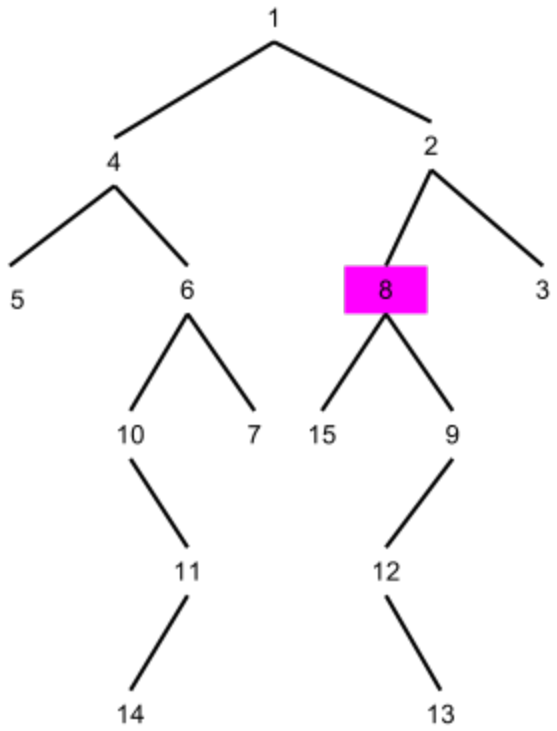
	x	x												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a





	x	x												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a





x	x	x												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
a	a	b	a	b	a	b	a	b	b	a	a	b	b	a

Augmenting H(T) in O(n).

Conclusions

O(m + k) for searching

O(n) for building position heaps.

O(1) amortized for insertions

O((h(T) + b)h(T)log(n)) for DELETE

O((h(T') + b)h(T')log(n)) for INSERT

References

Position heaps: A simple and dynamic text indexing data structure

Andrzej Ehrenfeucht,
Ross M. McConnell,
Nissa Osheim,
Sung-Whan Woo

Thank you!