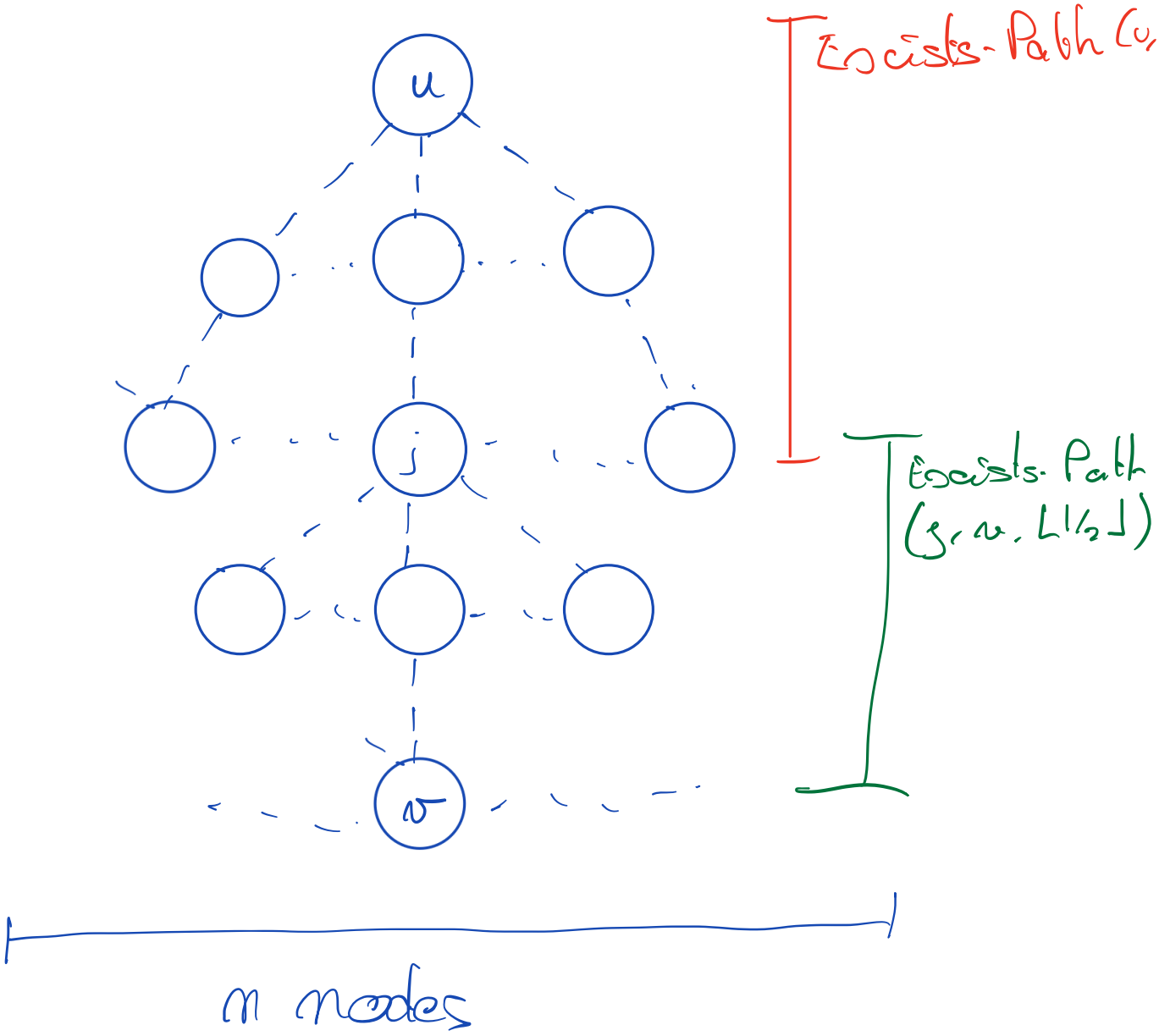
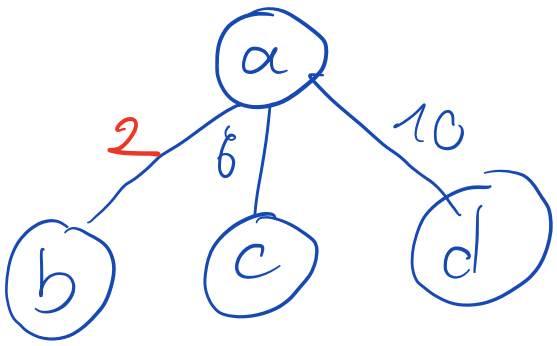
Divide-and-Conquer BFS



$v' = \infty \; 2$

$v = 0$

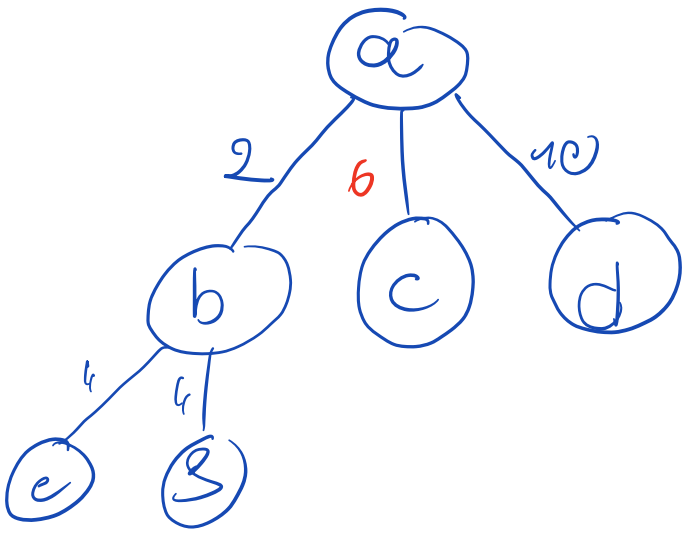


$i = 1$

$v' = \infty \; 6$

$v = 2$

call : $\{(b, 2)\}$

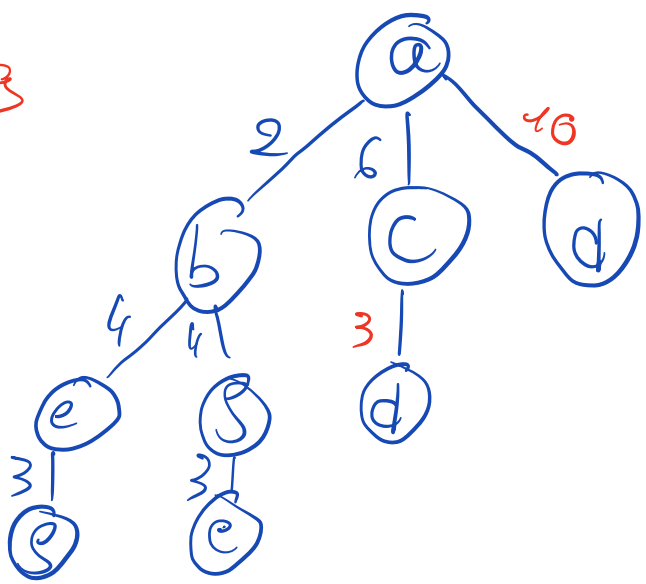


after c, d

$v' = 8 \; 10 \; 9$

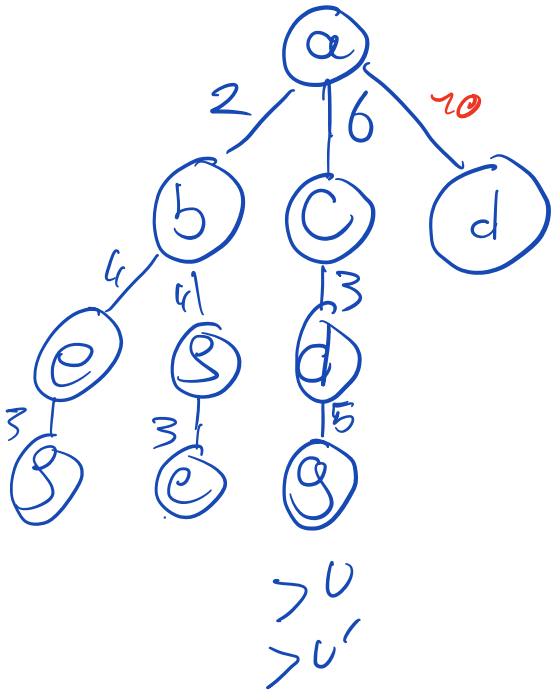
$v = 6$

call : $\{(b, 2)(c, 6)$
$(e, 6)(s, 6)\}$



$U' = \cancel{\infty}\ 10$

$U = 9$



call : (b,2), (c,6)

~~(c,6)~~, ~~(g,6)~~, (d,9)

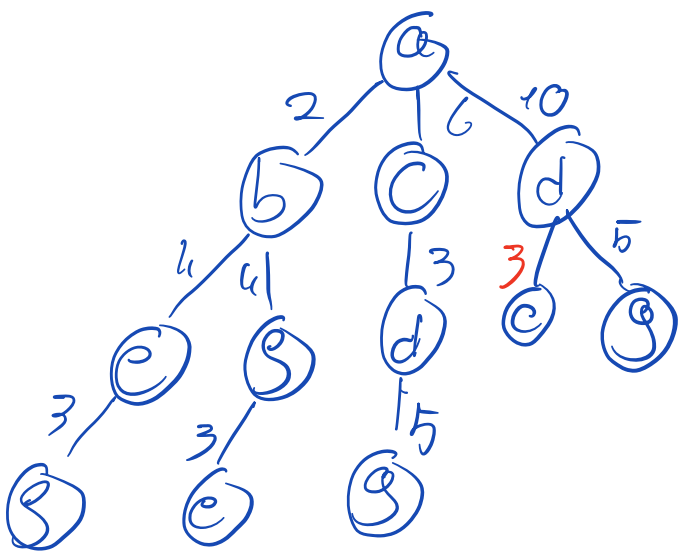
(g,8), (e,9)

$> U$

$> U'$

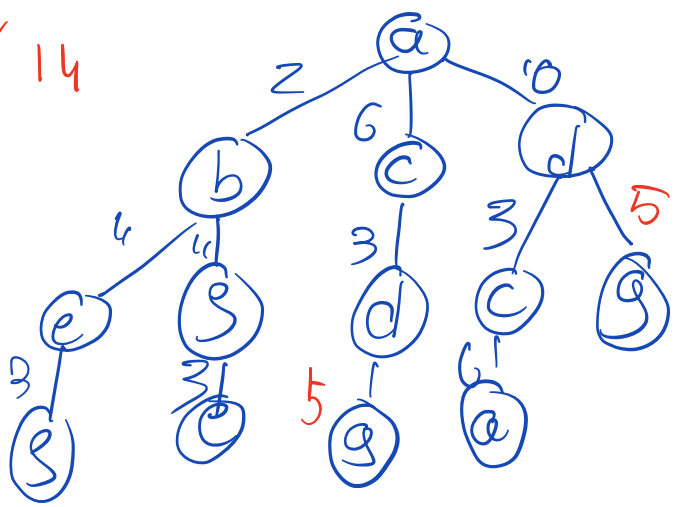
$U' = \cancel{9}\ 13$

$U = 10$



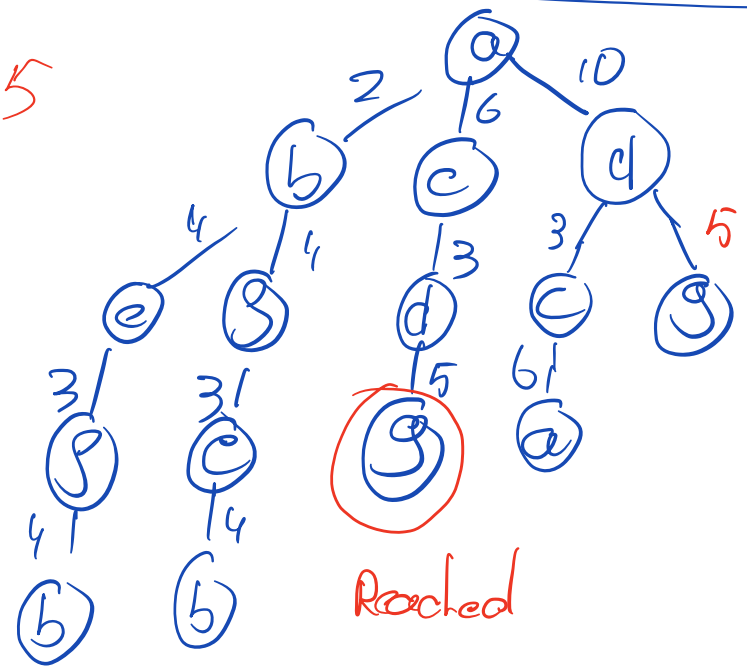
$U' = \cancel{\infty}\ \cancel{15}\ 14$

$U = 13$



U' = ~~∞~~ 15

U = 14



Depth-first Iterative deepening

Theorem: Depth-first iterative deepening is asymptotically optimal among brute-force tree searches in terms of time, space, and length of solution.

Proof: (Optimality) DFID generates all nodes at a given depth d before expanding to a greater depth. It always find a shortest path to the goal. Hence, it is optimal in terms of solution. ∎

Proof (time):

The nodes at depth d are generated once during the final iteration of the search.

The nodes at depth $d-1$ are generated twice, once during the final iteration at depth d, and once during the penultimate iteration at depth $d-1$.

Similarly, the nodes at depth $d-2$ are generated thee times, iterations d, $d-1$, $d-2$.

Thus the total number of nodes generated by DFID to depth d is:

$$b^d + 2b^{d-1} + 3b^{d-2} + \ldots + db$$

Factoring out b^d gives:

$$b^d \left(1 + 2b^{-1} + 3b^{-2} + \ldots + db^{1-d}\right)$$

letting $x = \frac{1}{b}$,

$$b^d \left(1 + 2x^1 + 3x^2 + \ldots + dx^{d-1}\right)$$

This is less than the infinite serie

$$b^d(1 + 2x^1 + 3x^2 + \dots),$$

which converges to

$$b^d(1-x)^{-2} \quad \text{for } \text{abs}(x) < 1.$$

Since $(1-x)^{-2}$ or $(1-\frac{1}{b})^{-2}$ is a constant that is independent of d, if $b > 1$ then the running time of depth-first iterative-deepening is $O(b^d)$.

The optimality can be showed with an adversary argument.

The number of nodes at depth d is b^d.
Assume that there is an optimal algorithm that examine less than b^d nodes. Then, there is at least one node not examine by this algorithm. Since we have no additional information, an adversary could place the goal at this node, hence the algorithm would fail.

Hence any Brute-force algorithm must take at least cb^d time, for some constant c. ■

Proof (Space): Since at any point DFID is engaged in a depth-first search, it need only store a stack of nodes which represents the branch of the tree it is expanding.

Since it finds an optimal solution, the maximum depth of this stack is d, and hence the maximum amount of space is $O(d)$.