CPSC 322
Introduction to Artificial Intelligence

October 15, 2004

Generic graph search algorithm

Given a set of start nodes, a set of goal nodes,
and a graph (i.e., the nodes and arcs):

make a "list" of the start nodes - let's call it the "frontier"

```
repeat
  if no nodes on the frontier then terminate with failure
  choose one node from the frontier and remove it
  if the chosen node matches the goal node
    then terminate with success
    else put next nodes (neighbors) on frontier
end repeat
```

Datalog graph search algorithm

```
search(F0) <- choose(Node,F0,F1) ^
              is_goal(Node).

search(F0) <- choose(Node,F0,F1) ^
              neighbors(Node,NN) ^
              add_to_frontier(NN,F1,F2) ^
              search(F2).
```

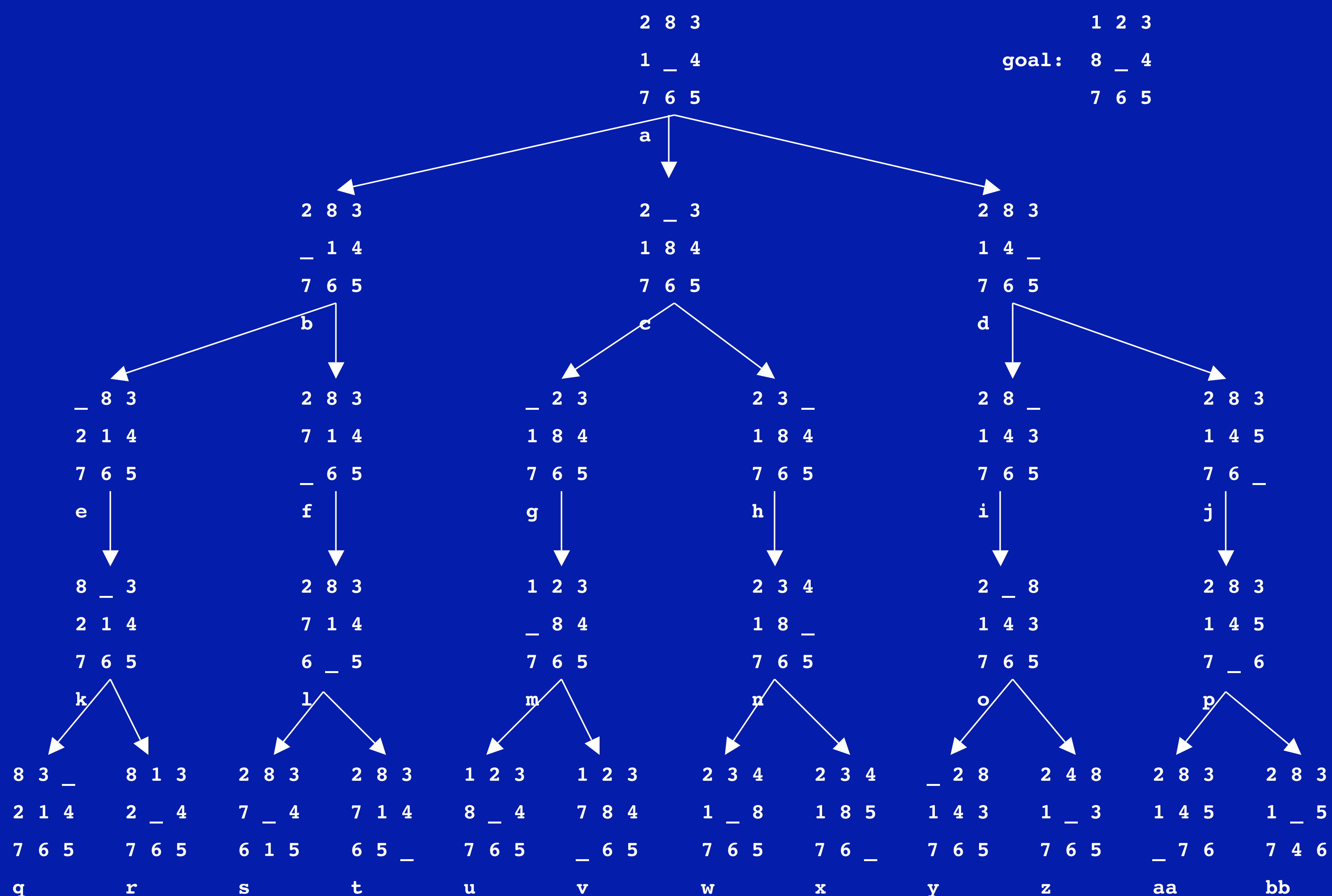
CILOG graph search

```
neighbors([2,8,3,1,0,4,7,6,5],[[2,8,3,0,1,4,7,6,5],[2,0,3,1,8,4,7,6,5],
  [2,8,3,1,4,0,7,6,5]]).
neighbors([2,8,3,0,1,4,7,6,5],[[0,8,3,2,1,4,7,6,5],[2,8,3,7,1,4,0,6,5]]).
neighbors([2,0,3,1,8,4,7,6,5],[[0,2,3,1,8,4,7,6,5],[2,3,0,1,8,4,7,6,5]]).
neighbors([2,8,3,1,4,0,7,6,5],[[2,8,0,1,4,3,7,6,5],[2,8,3,1,4,5,7,6,0]]).
neighbors([0,8,3,2,1,4,7,6,5],[[8,0,3,2,1,4,7,6,5]]).
   :
   :
neighbors([2,3,4,1,0,8,7,6,5],[]).
neighbors([2,3,4,1,8,5,7,6,0],[]).
neighbors([0,2,8,1,4,3,7,6,5],[]).
neighbors([2,4,8,1,0,3,7,6,5],[]).
neighbors([2,8,3,1,4,5,0,7,6],[]).
neighbors([2,8,3,1,0,5,7,4,6],[]).



is_goal([1,2,3,8,0,4,7,6,5]).
```

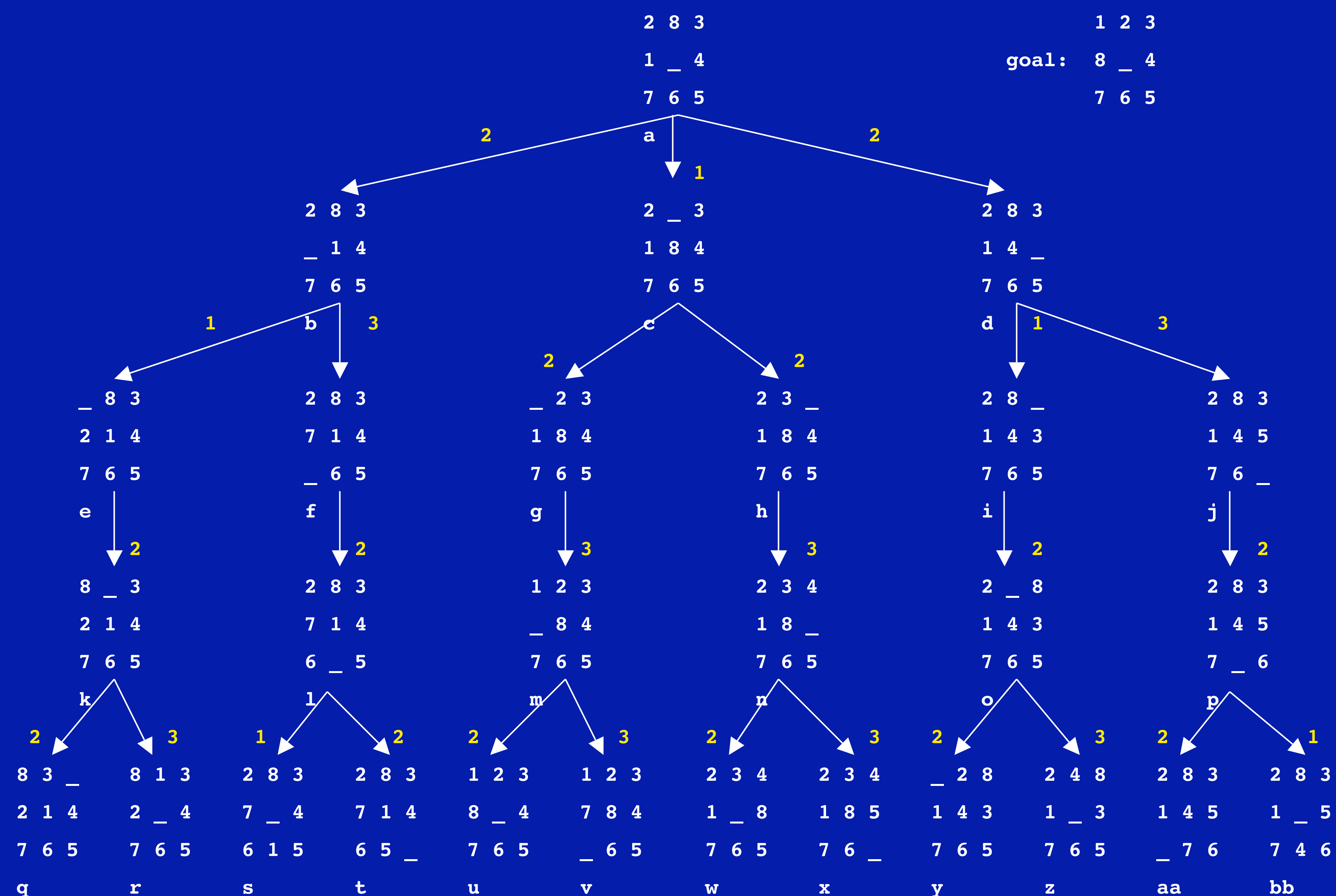
CILOG graph search

```
search(F0) <- choose(Node,F0,F1) &
              neighbors(Node,NN) &
              add_to_frontier(NN,F1,F2) &
              search(F2).

search(F0) <- choose(Node,F0,F1) & is_goal(Node).

/* choose(N,Flist0,Flist1) <- append([N],Flist1,Flist0). */

choose(N,[N|Flist],Flist).

add_to_frontier(Nodelist,Flist1,Flist2) <- append(Nodelist,Flist1,Flist2).
```

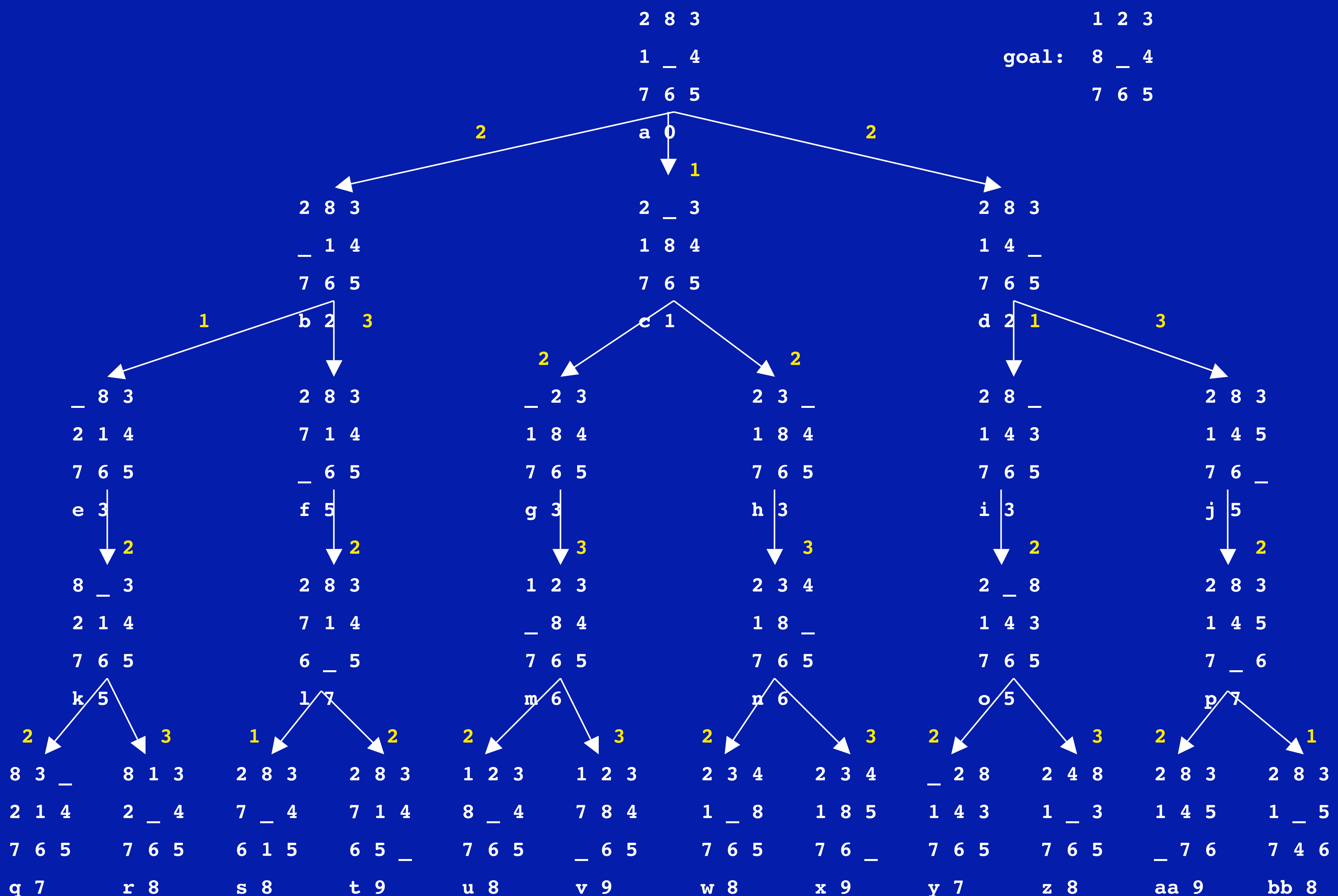
What if the arcs have different costs?



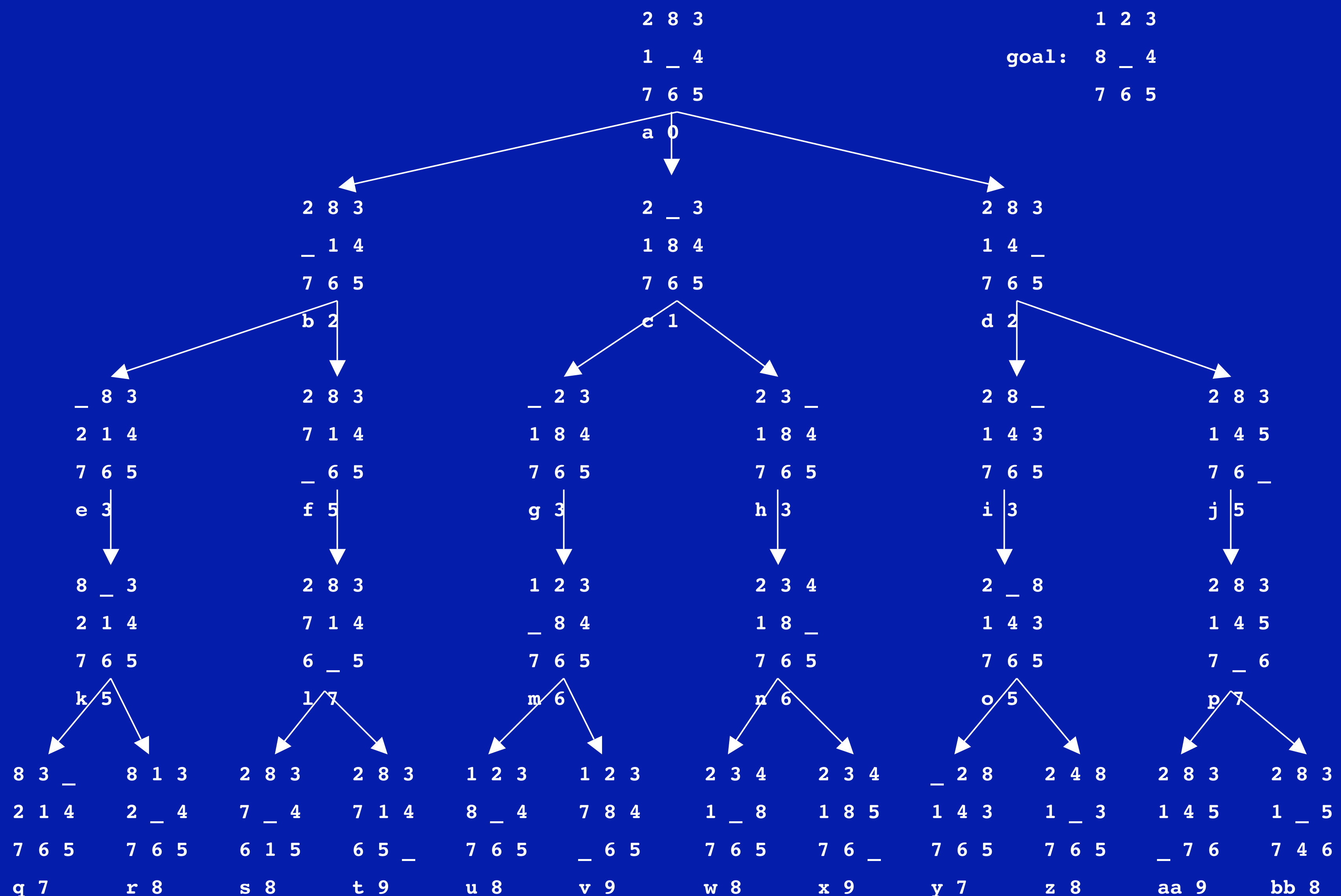
What if the arcs have different costs?



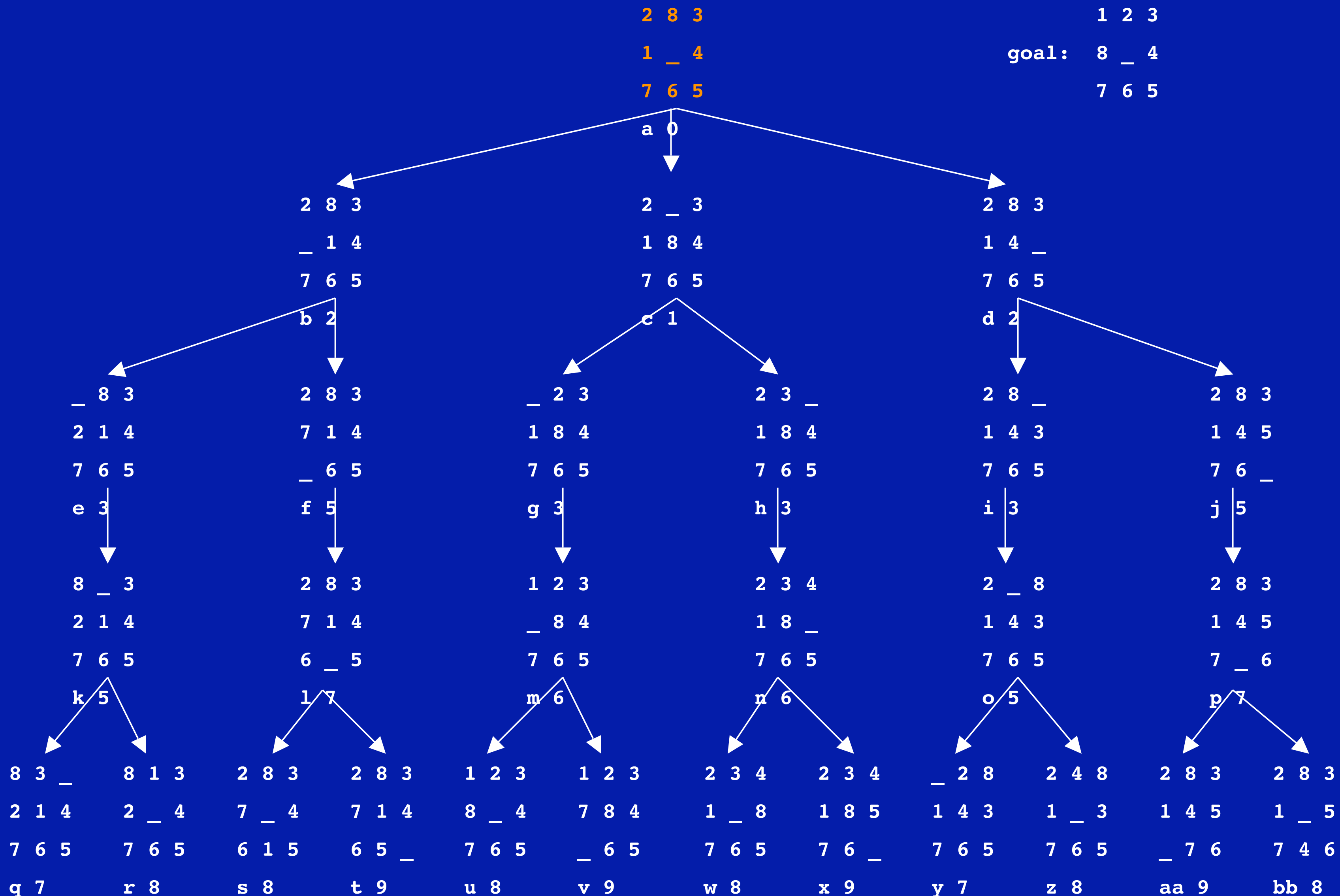
What if the arcs have different costs?



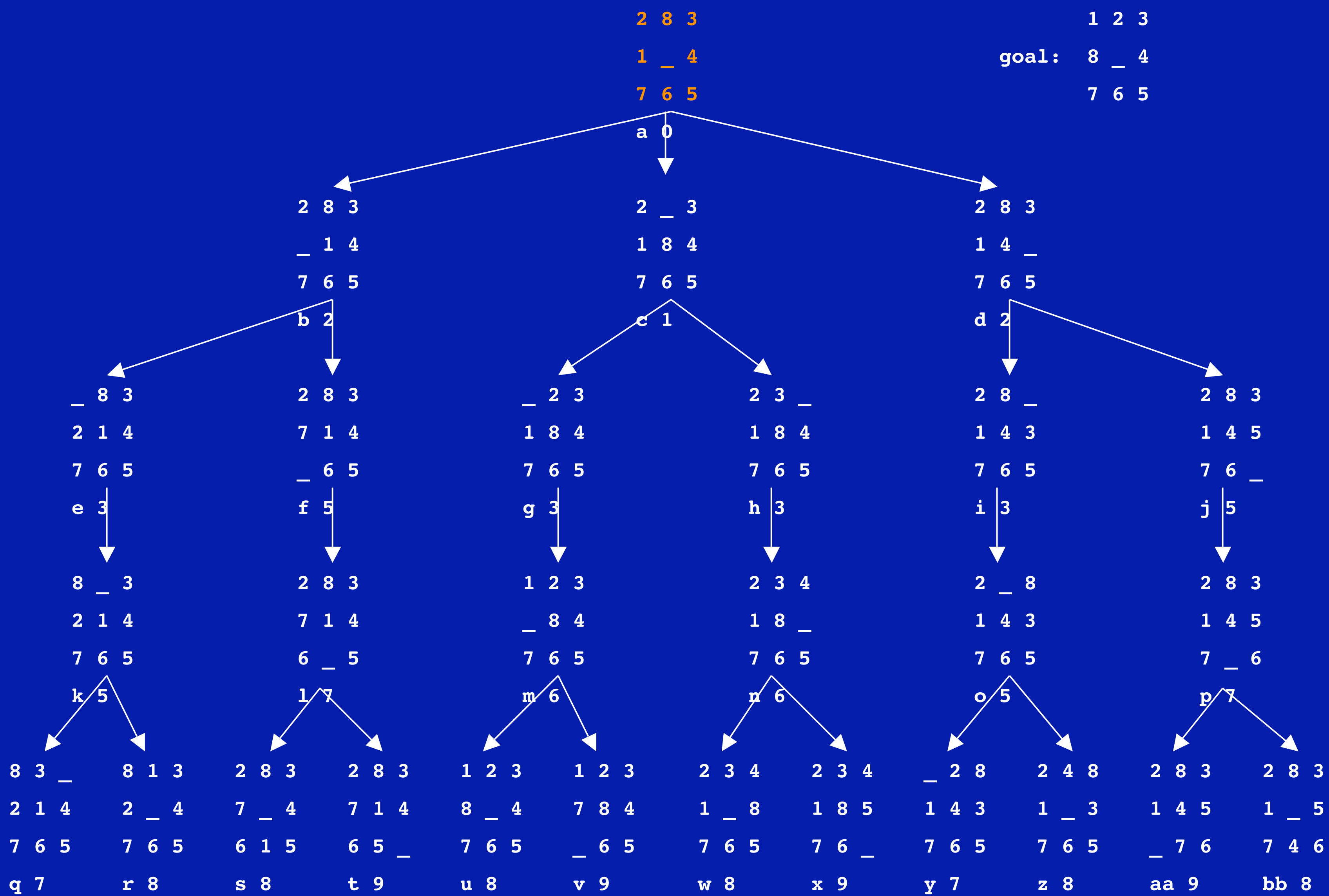
What if the arcs have different costs?



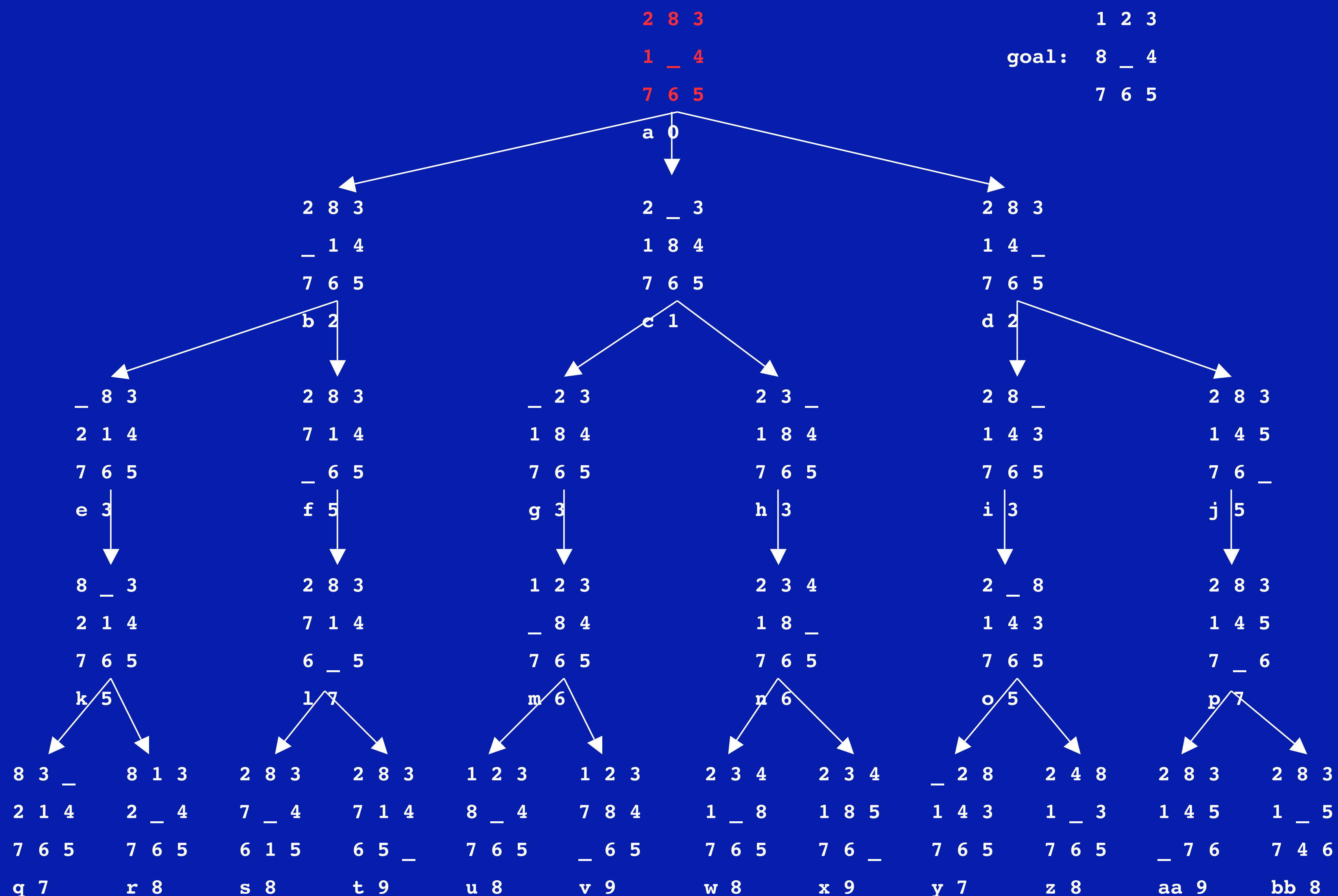
frontier: [a 0]


2 8 3 / 1 _ 4 / 7 6 5
a 0
goal: 1 2 3 / 8 _ 4 / 7 6 5
2 8 3 / _ 1 4 / 7 6 5
b 2
2 _ 3 / 1 8 4 / 7 6 5
c 1
2 8 3 / 1 4 _ / 7 6 5
d 2
_ 8 3 / 2 1 4 / 7 6 5
e 3
2 8 3 / 7 1 4 / _ 6 5
f 5
_ 2 3 / 1 8 4 / 7 6 5
g 3
2 3 _ / 1 8 4 / 7 6 5
h 3
2 8 _ / 1 4 3 / 7 6 5
i 3
2 8 3 / 1 4 5 / 7 6 _
j 5
8 _ 3 / 2 1 4 / 7 6 5
k 5
2 8 3 / 7 1 4 / 6 _ 5
l 7
1 2 3 / _ 8 4 / 7 6 5
m 6
2 3 4 / 1 8 _ / 7 6 5
n 6
2 _ 8 / 1 4 3 / 7 6 5
o 5
2 8 3 / 1 4 5 / 7 _ 6
p 7
8 3 _ / 2 1 4 / 7 6 5
q 7
8 1 3 / 2 _ 4 / 7 6 5
r 8
2 8 3 / 7 _ 4 / 6 1 5
s 8
2 8 3 / 7 1 4 / 6 5 _
t 9
1 2 3 / 8 _ 4 / 7 6 5
u 8
1 2 3 / 7 8 4 / _ 6 5
v 9
2 3 4 / 1 _ 8 / 7 6 5
w 8
2 3 4 / 1 8 5 / 7 6 _
x 9
_ 2 8 / 1 4 3 / 7 6 5
y 7
2 4 8 / 1 _ 3 / 7 6 5
z 8
2 8 3 / 1 4 5 / _ 7 6
aa 9
2 8 3 / 1 _ 5 / 7 4 6
bb 8

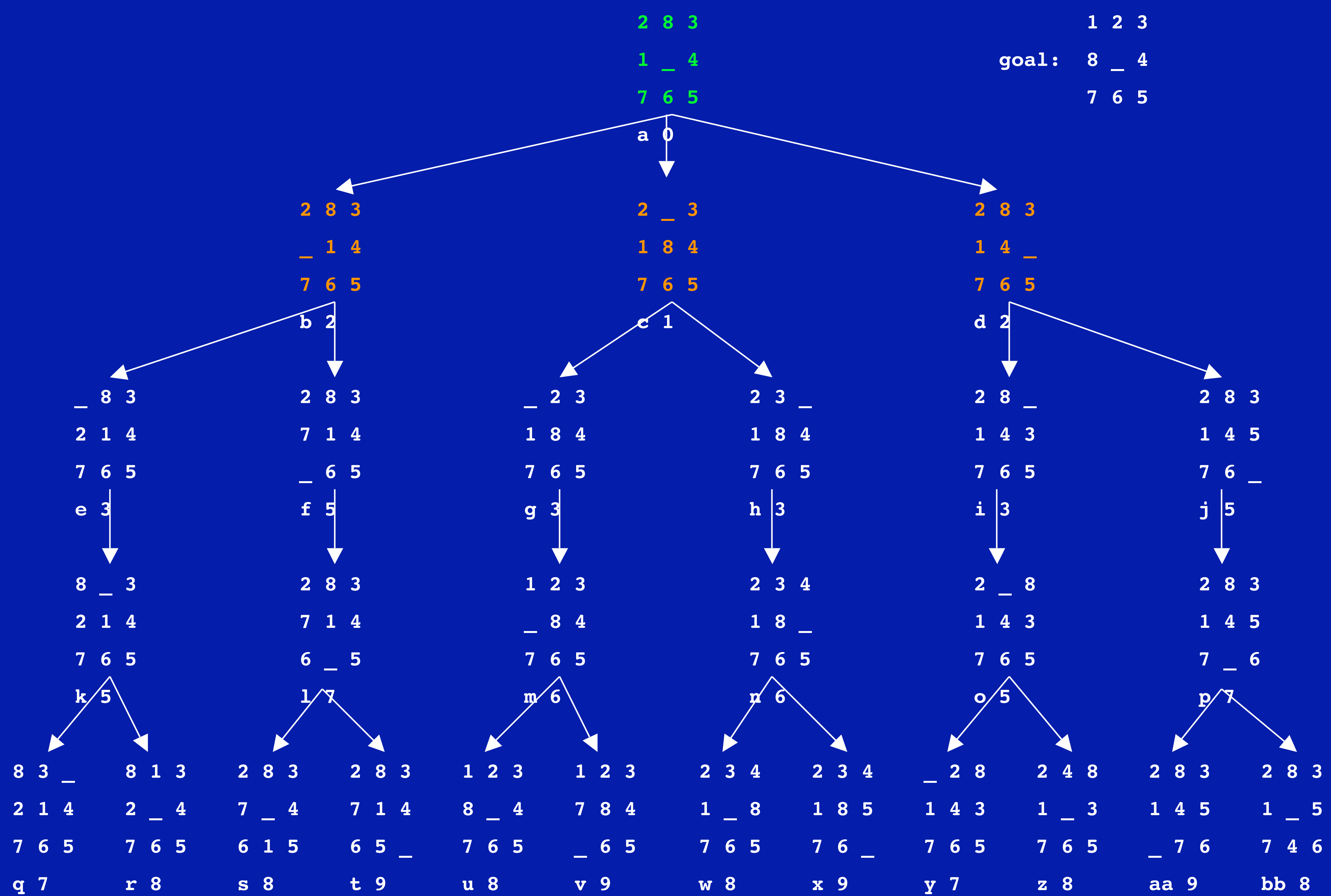
frontier: [a 0] sorted



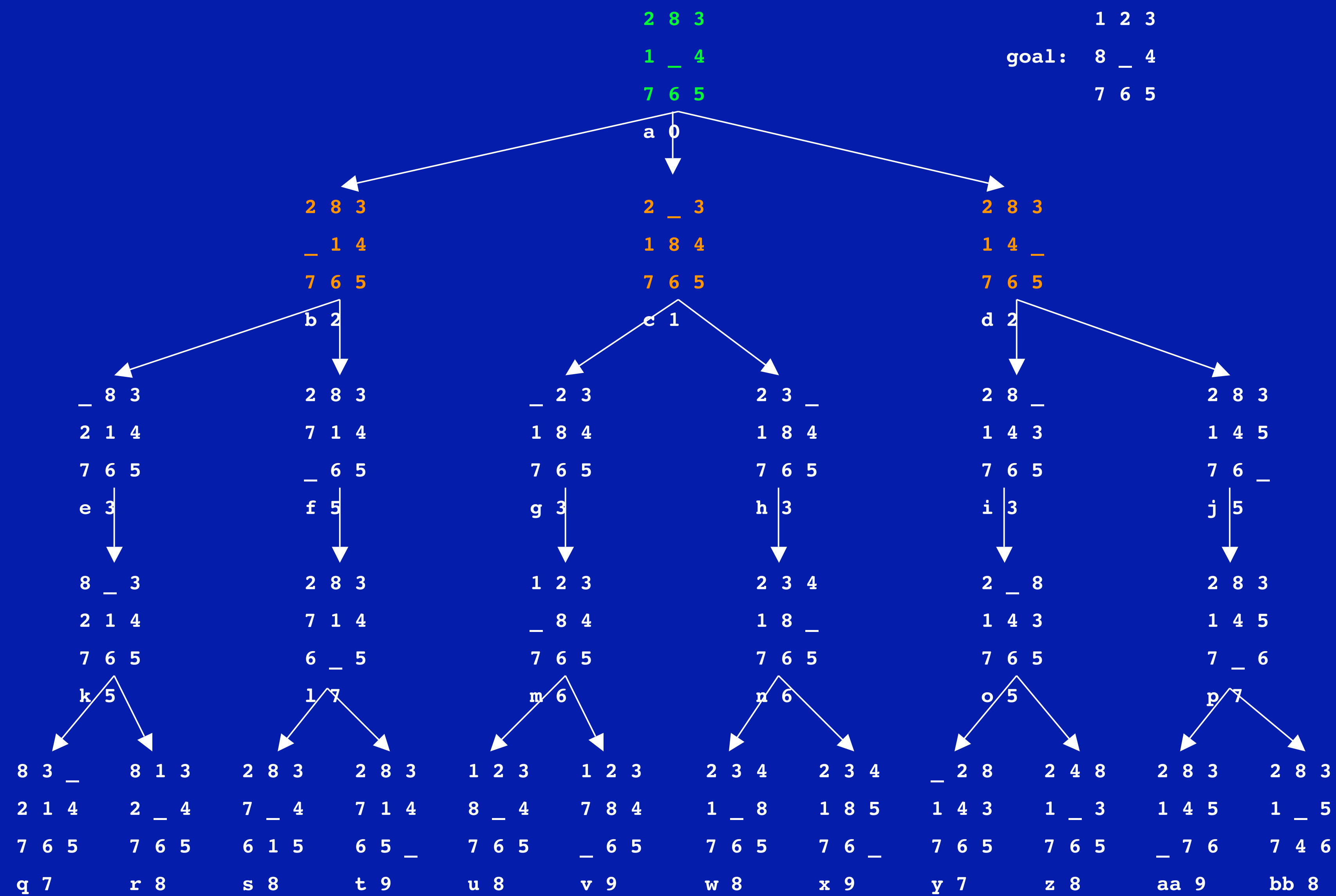
frontier: [ ]


2 8 3
1 _ 4
7 6 5
a 0
goal: 1 2 3
8 _ 4
7 6 5
2 8 3
_ 1 4
7 6 5
b 2
2 _ 3
1 8 4
7 6 5
c 1
2 8 3
1 4 _
7 6 5
d 2
_ 8 3
2 1 4
7 6 5
e 3
2 8 3
7 1 4
_ 6 5
f 5
_ 2 3
1 8 4
7 6 5
g 3
2 3 _
1 8 4
7 6 5
h 3
2 8 _
1 4 3
7 6 5
i 3
2 8 3
1 4 5
7 6 _
j 5
8 _ 3
2 1 4
7 6 5
k 5
2 8 3
7 1 4
6 _ 5
l 7
1 2 3
_ 8 4
7 6 5
m 6
2 3 4
1 8 _
7 6 5
n 6
2 _ 8
1 4 3
7 6 5
o 5
2 8 3
1 4 5
7 _ 6
p 7
8 3 _
2 1 4
7 6 5
q 7
8 1 3
2 _ 4
7 6 5
r 8
2 8 3
7 _ 4
6 1 5
s 8
2 8 3
7 1 4
6 5 _
t 9
1 2 3
8 _ 4
7 6 5
u 8
1 2 3
7 8 4
_ 6 5
v 9
2 3 4
1 _ 8
7 6 5
w 8
2 3 4
1 8 5
7 6 _
x 9
_ 2 8
1 4 3
7 6 5
y 7
2 4 8
1 _ 3
7 6 5
z 8
2 8 3
1 4 5
_ 7 6
aa 9
2 8 3
1 _ 5
7 4 6
bb 8

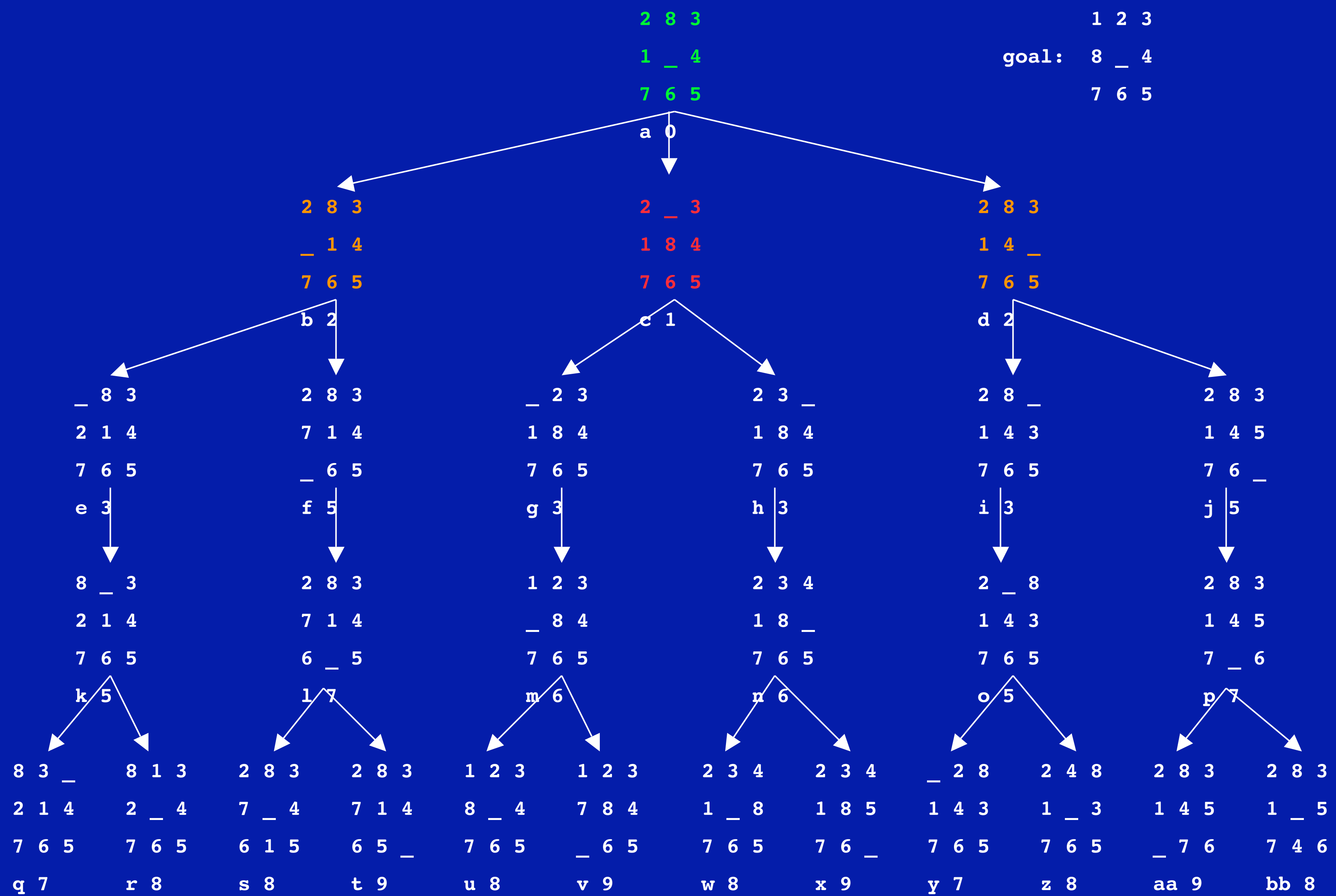
frontier: [b 2,c 1,d 2]



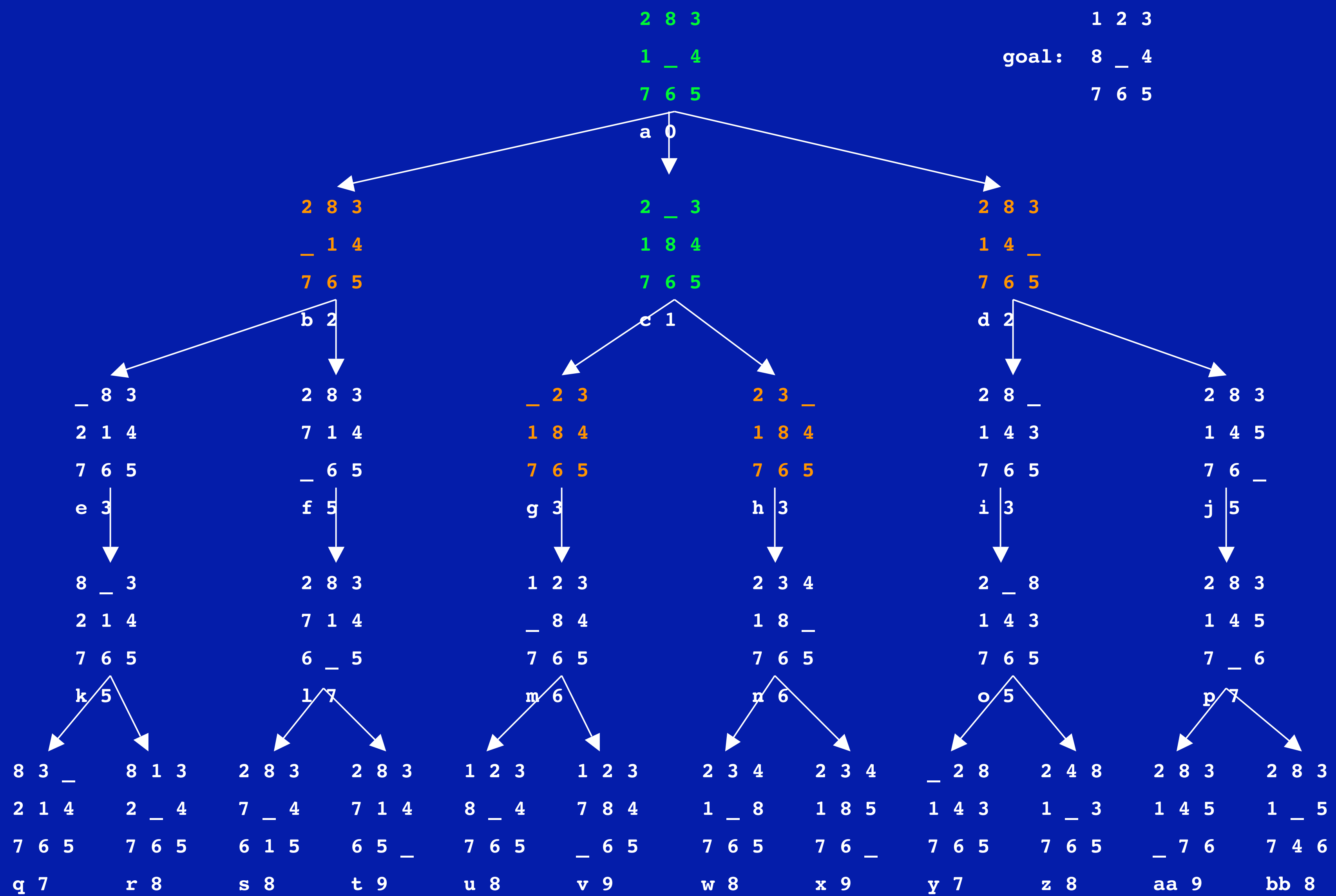
frontier: [c 1,b 2,d 2] sorted



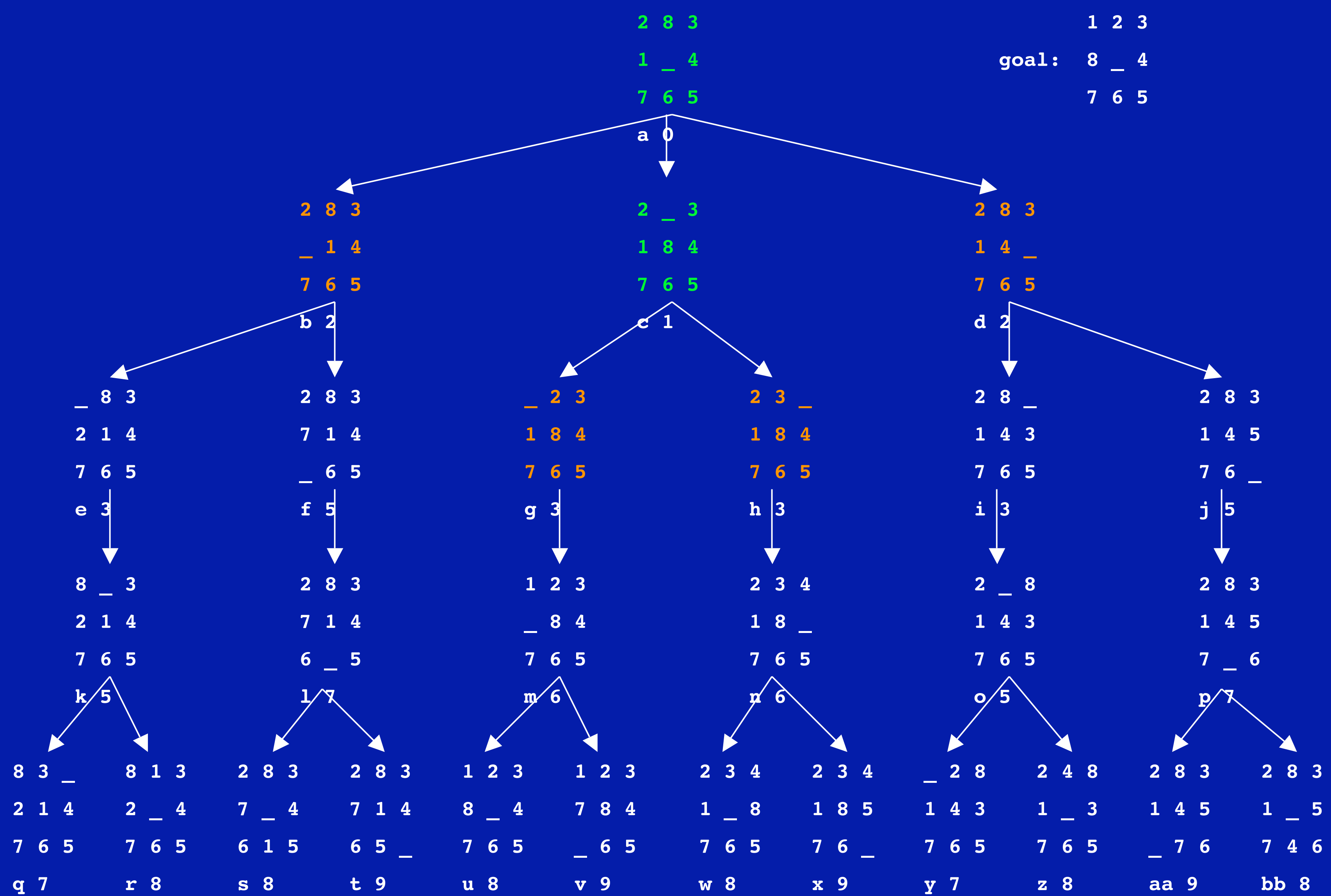
frontier: [b 2,d 2]



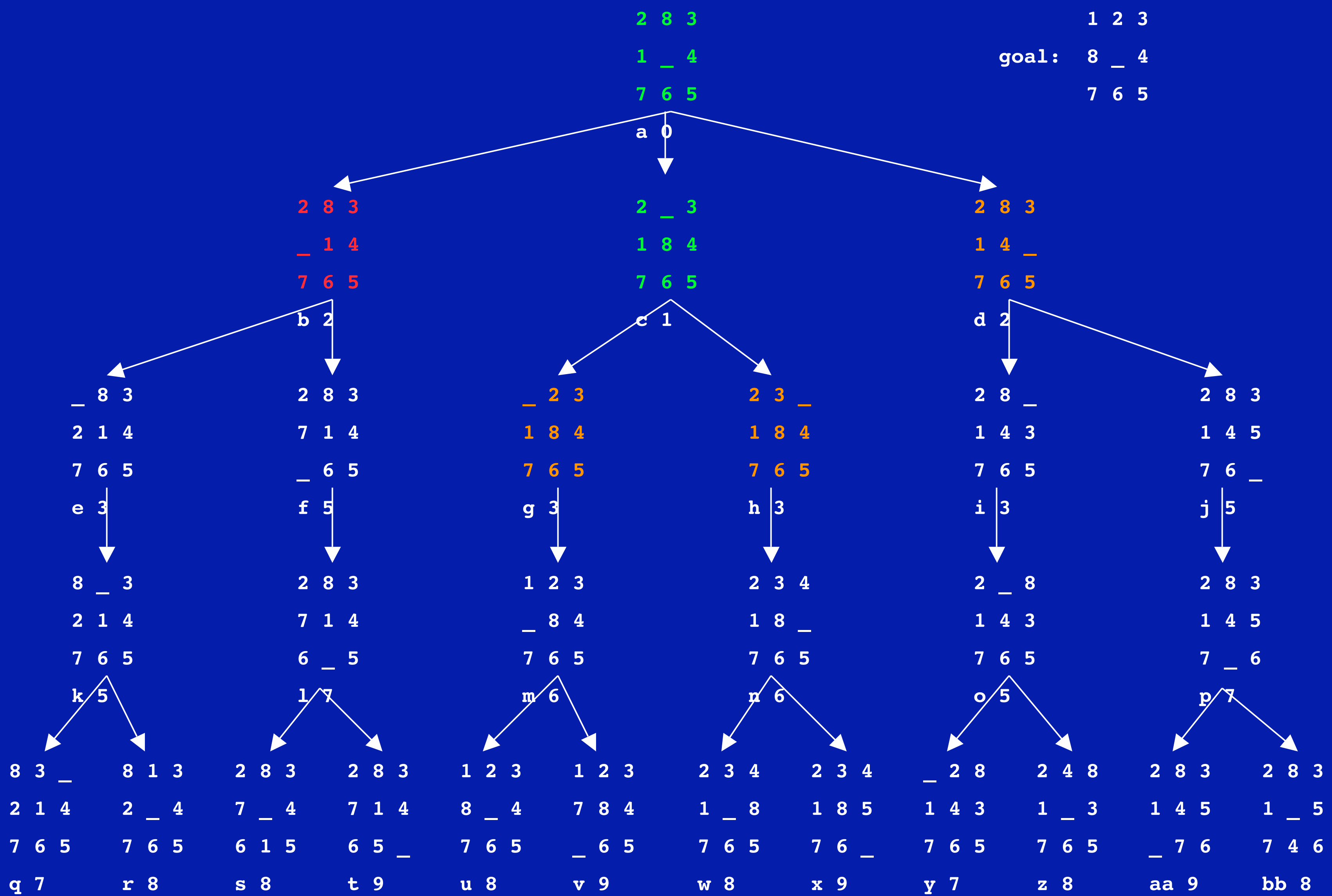
frontier: [g 3,h 3,b 2,d 2]



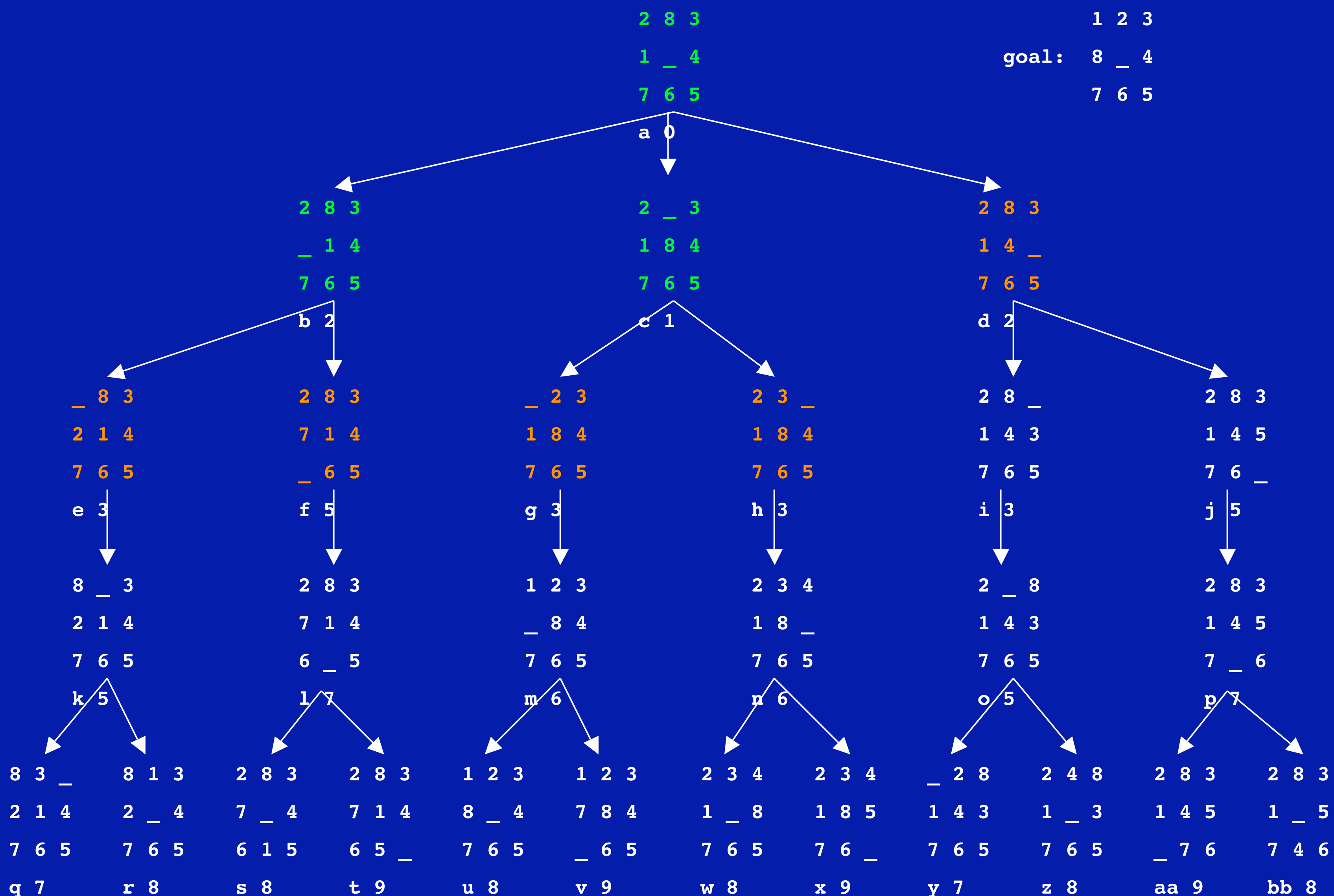
frontier: [b 2,d 2,g 3,h 3] *



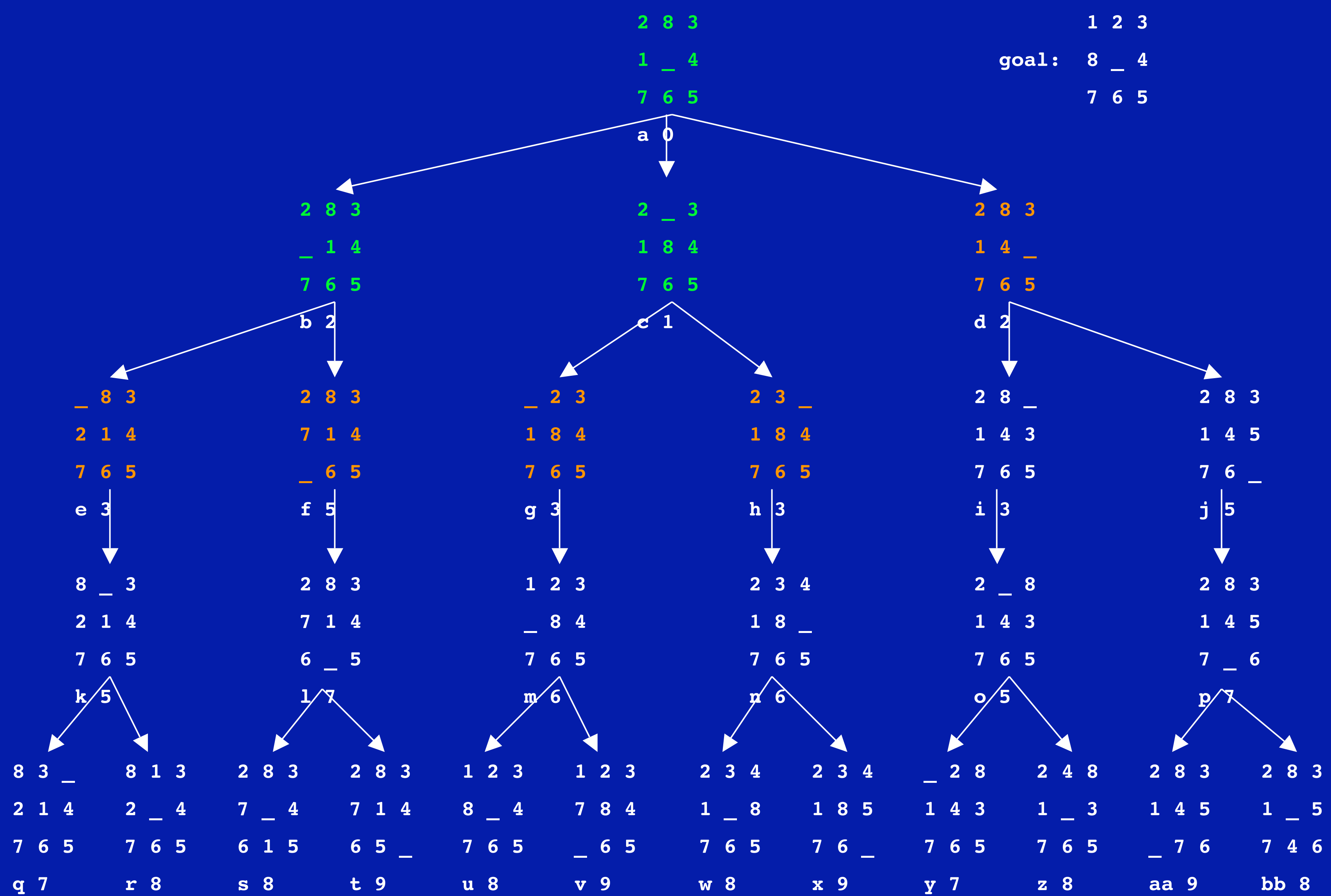
frontier: [d 2,g 3,h 3]



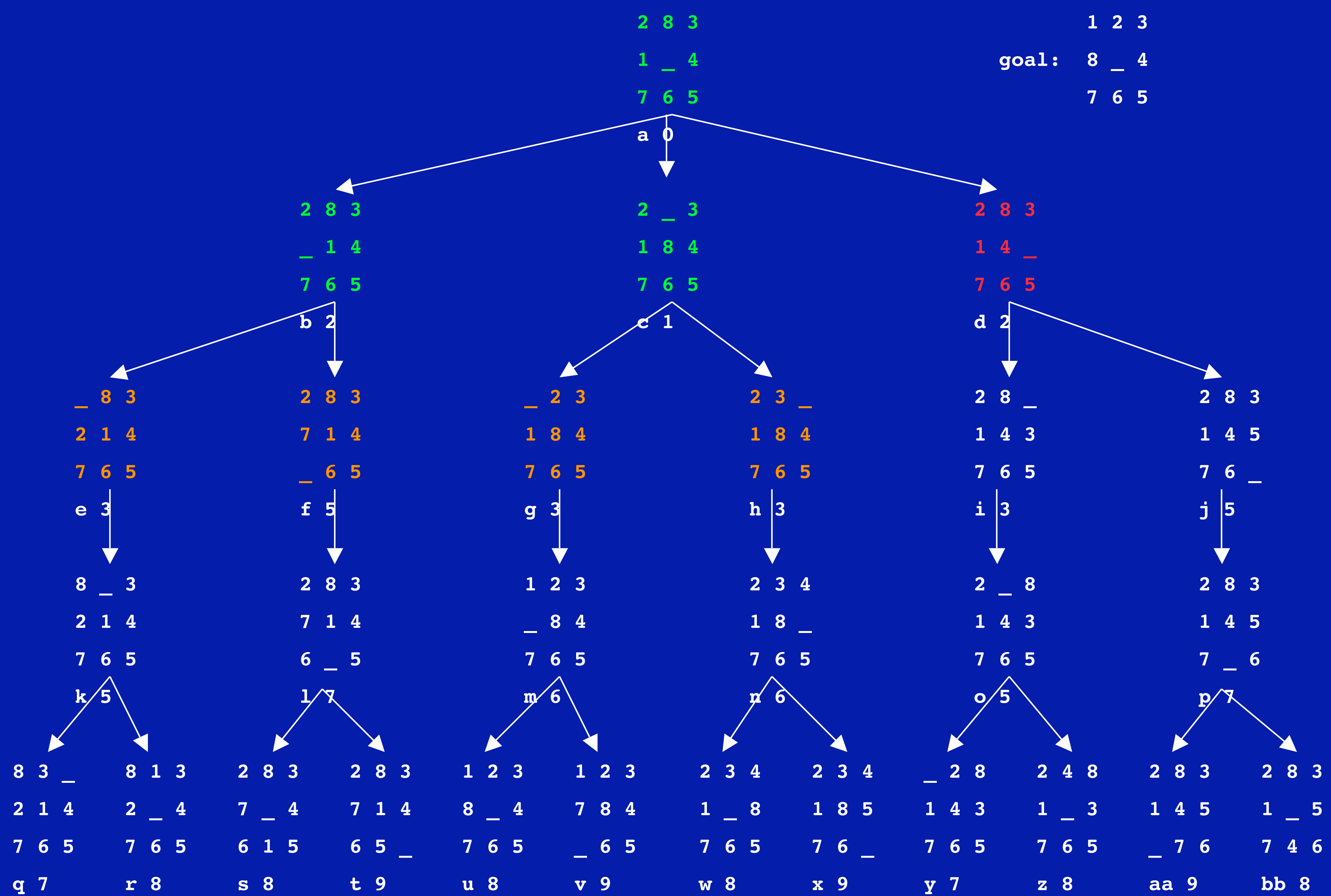
frontier: [e 3,f 5,d 2,g 3,h 3]



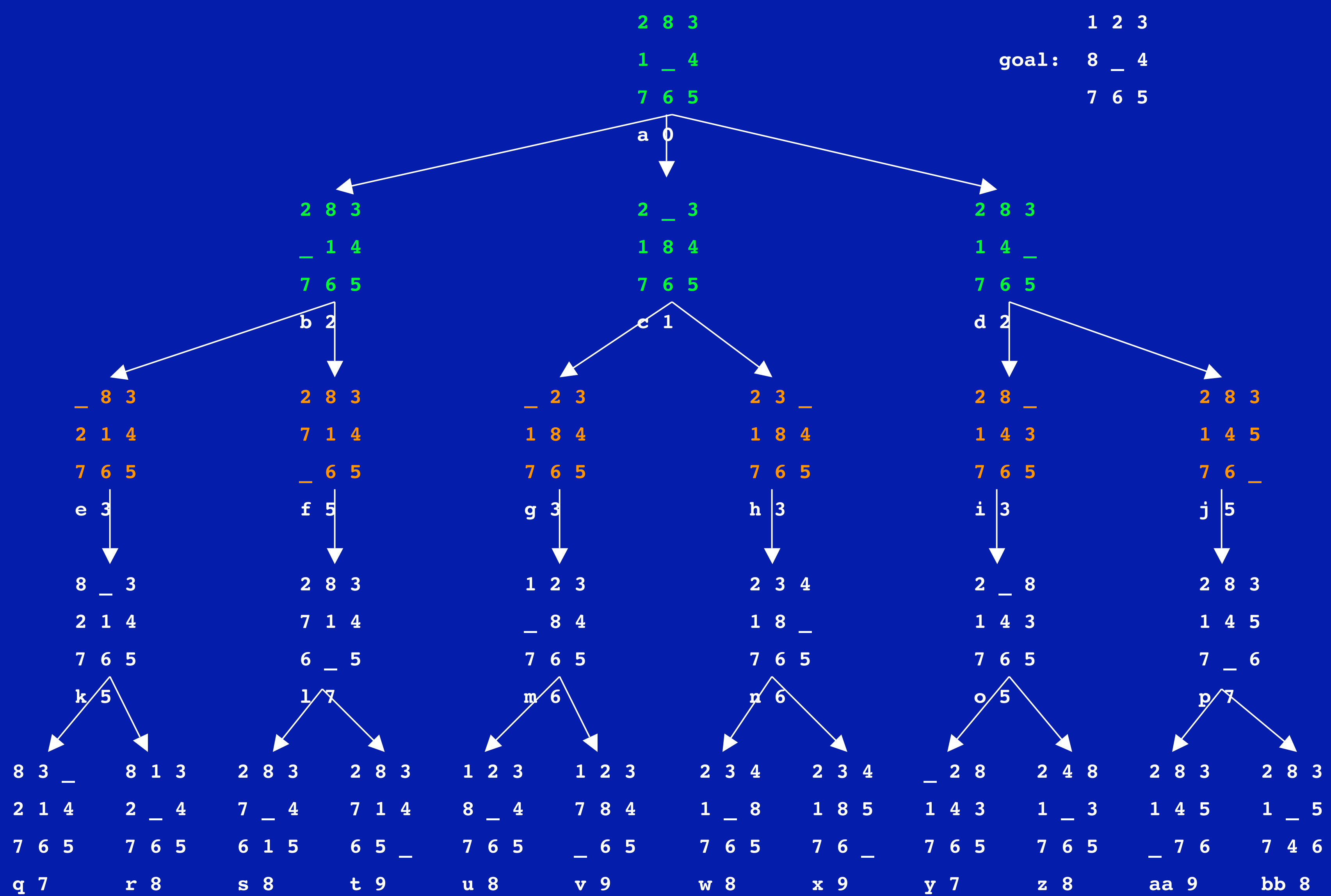
frontier: [d 2,e 3,g 3,h 3,f 5] *



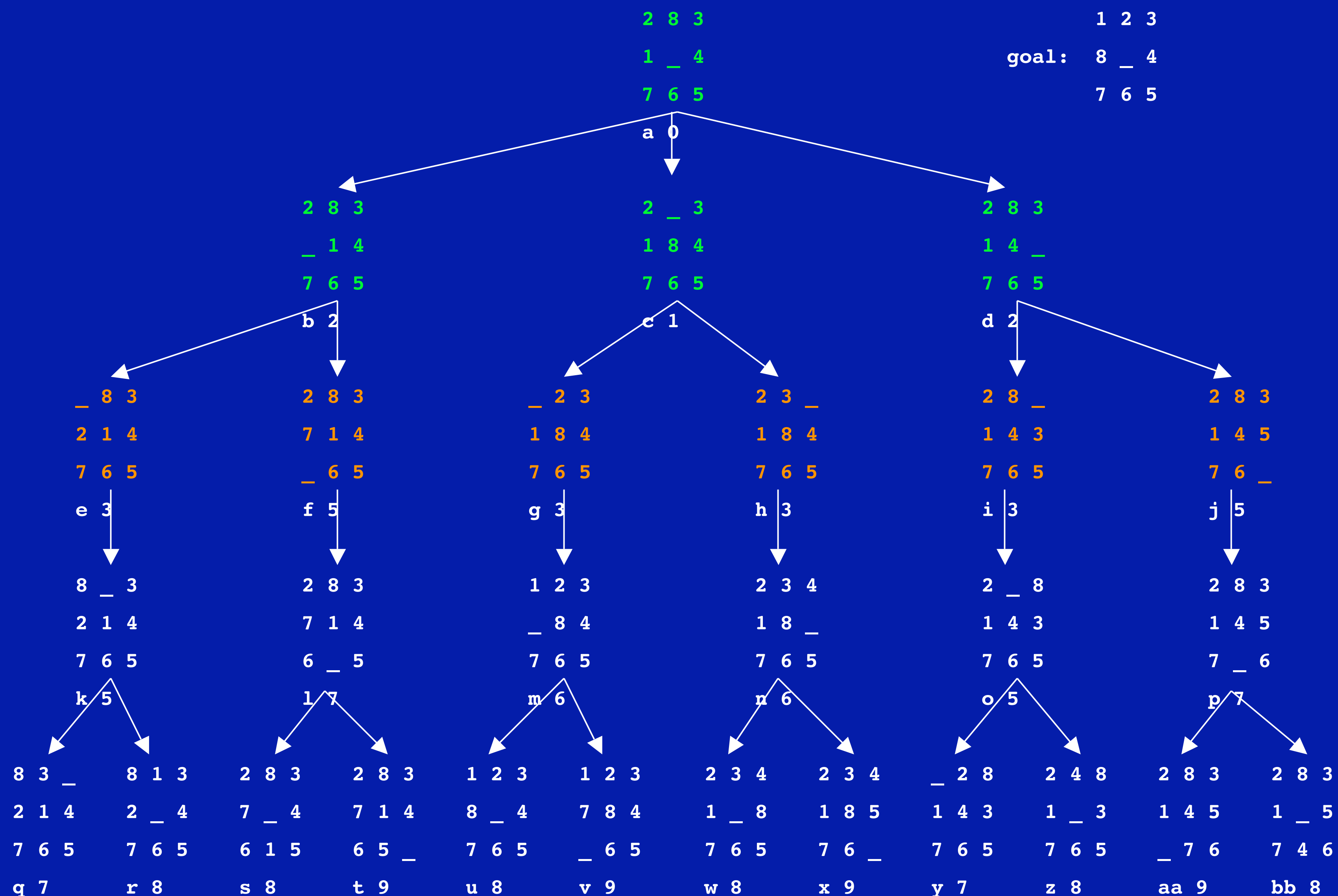
frontier: [e 3,g 3,h 3,f 5]



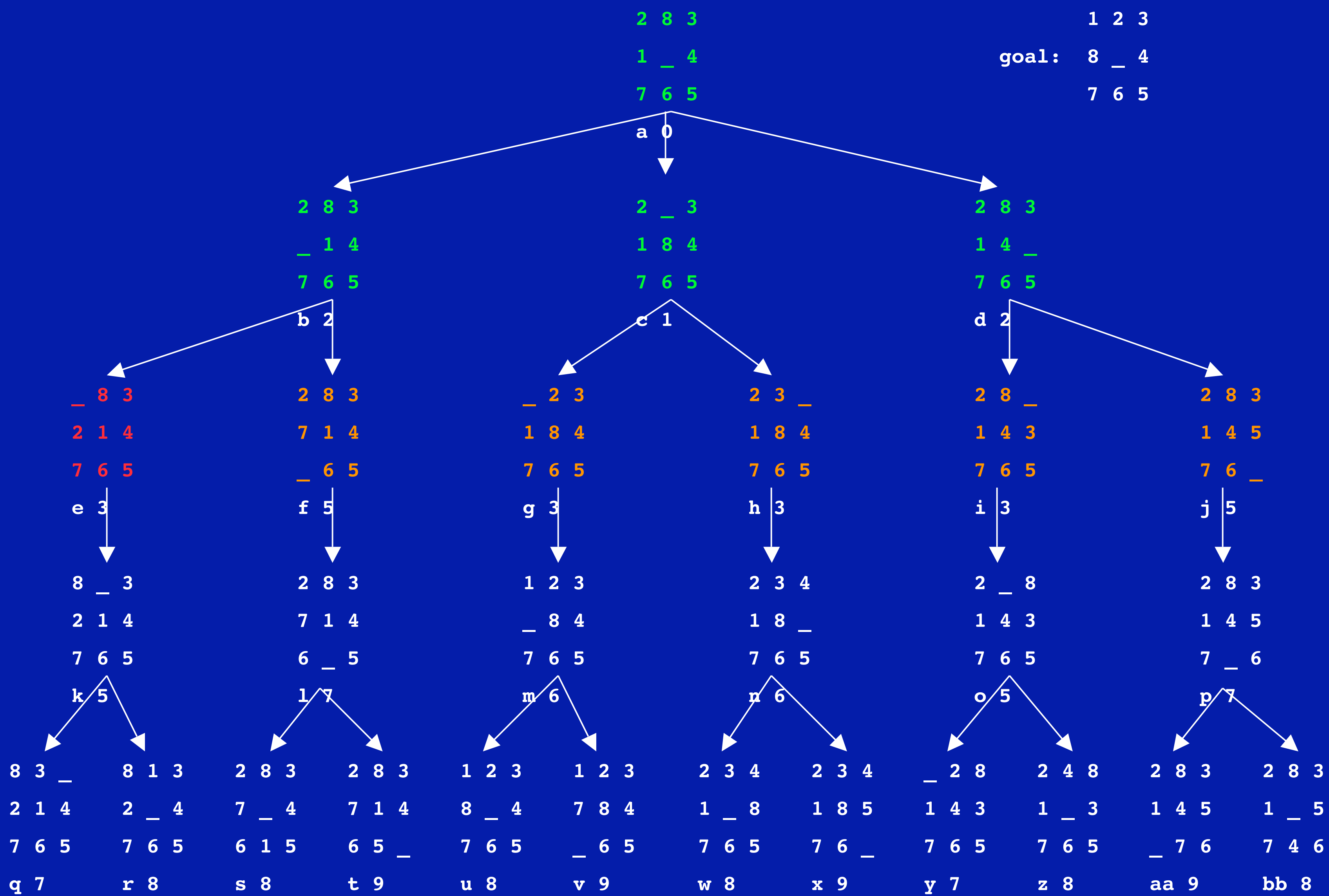
frontier: [i 3,j 5,e 3,g 3,h 3,f 5]



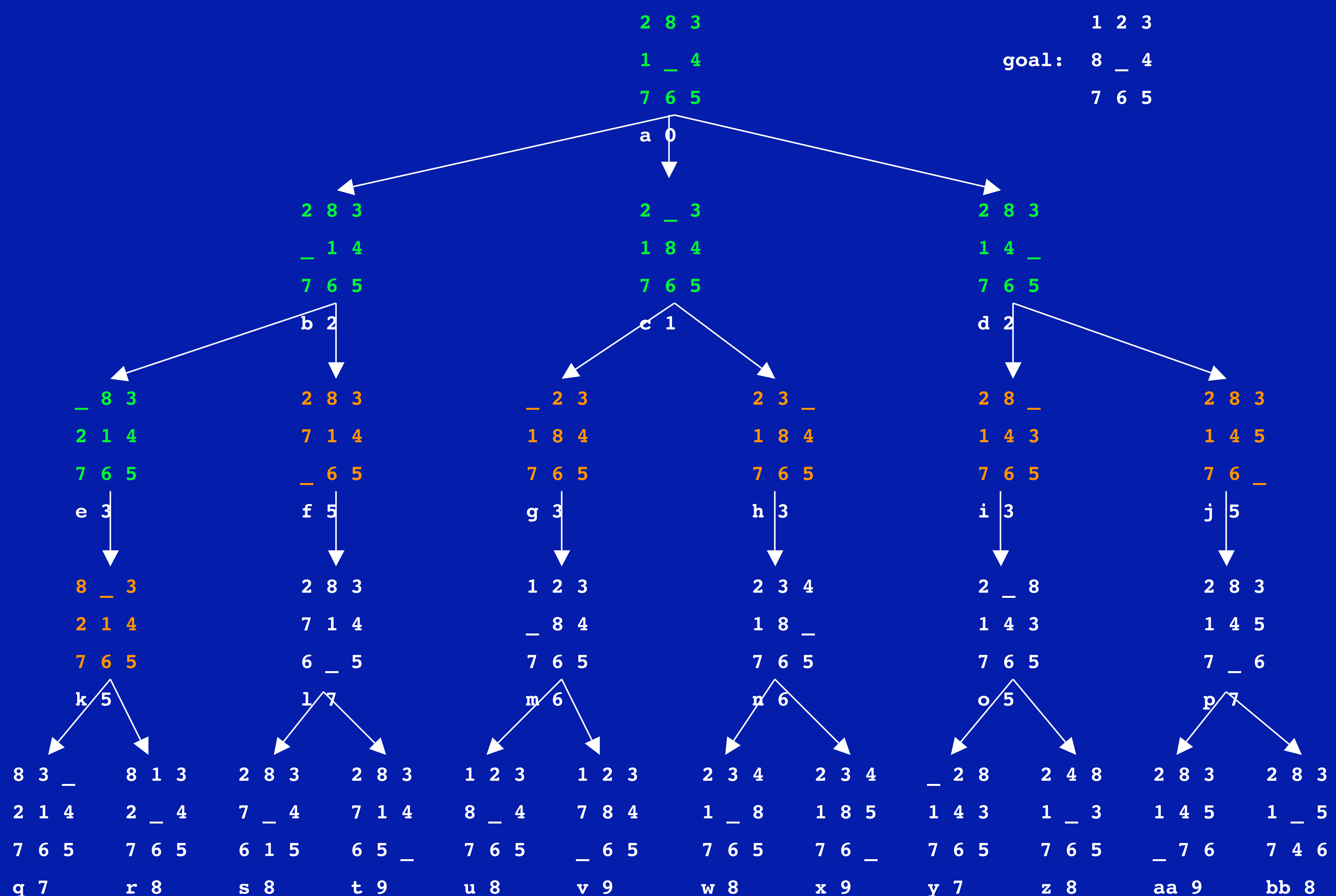
frontier: [e 3,g 3,h 3,i 3,f 5,j 5] *



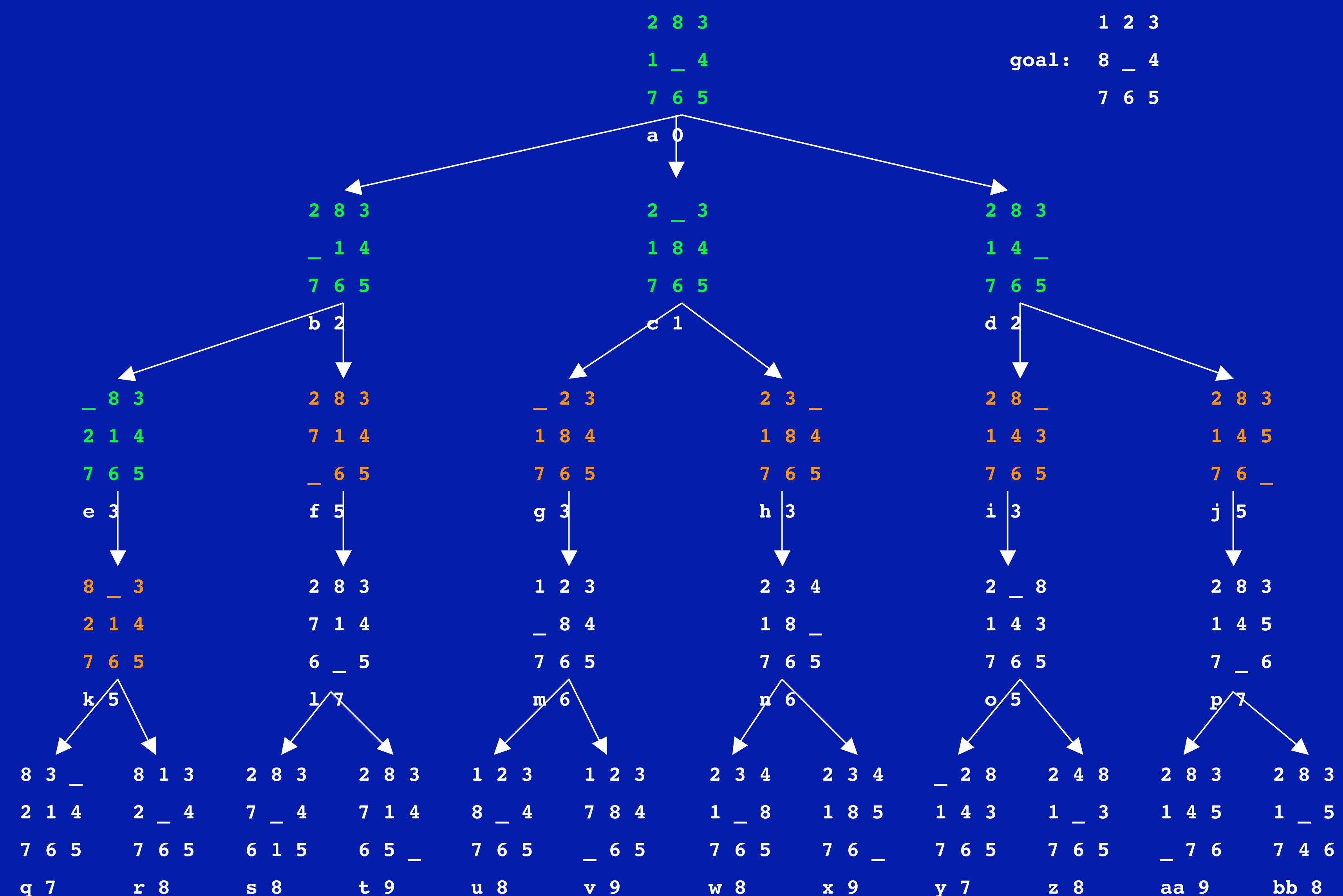
frontier: [g 3,h 3,i 3,f 5,j 5]



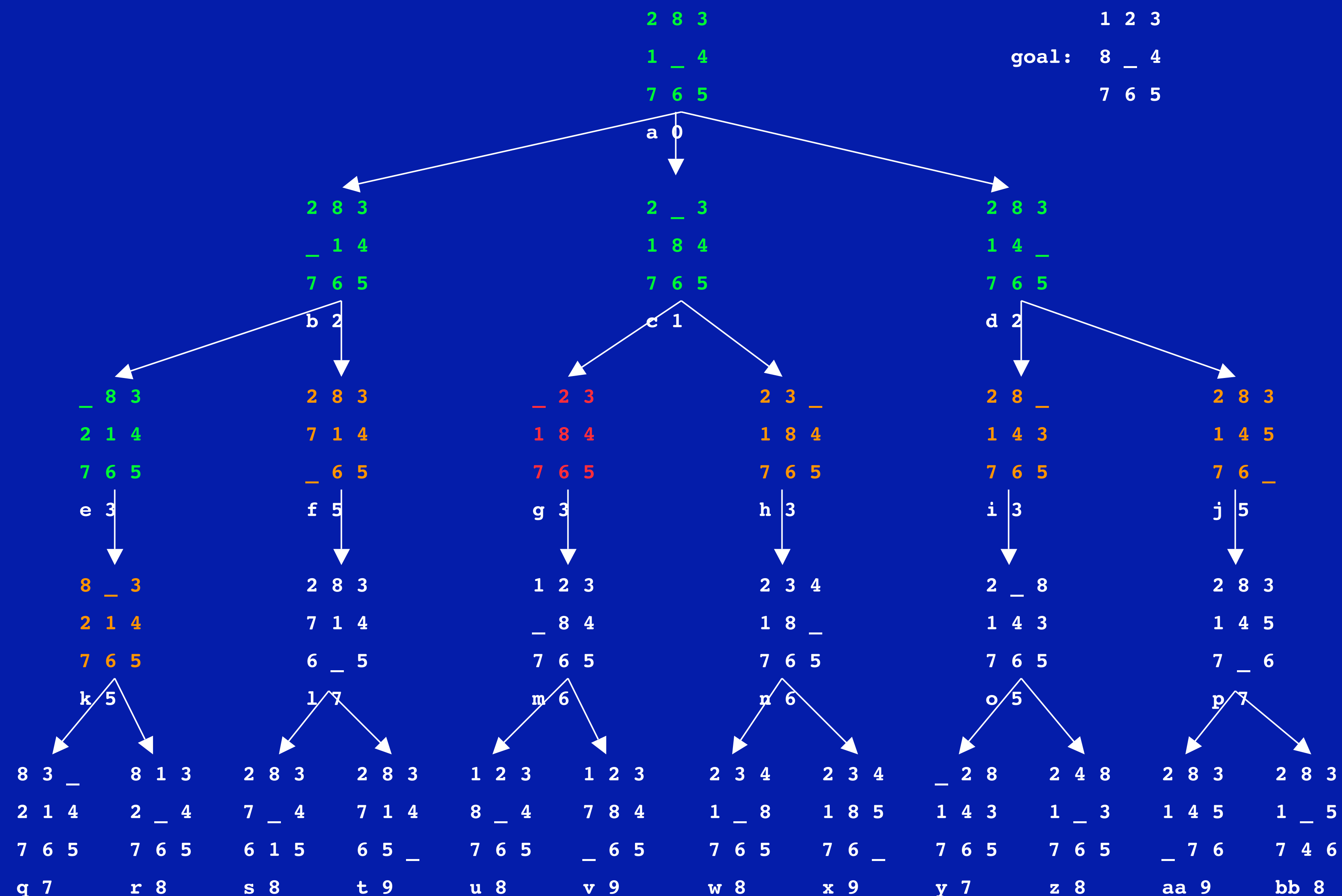
frontier: [k 5,g 3,h 3,i 3,f 5,j 5]



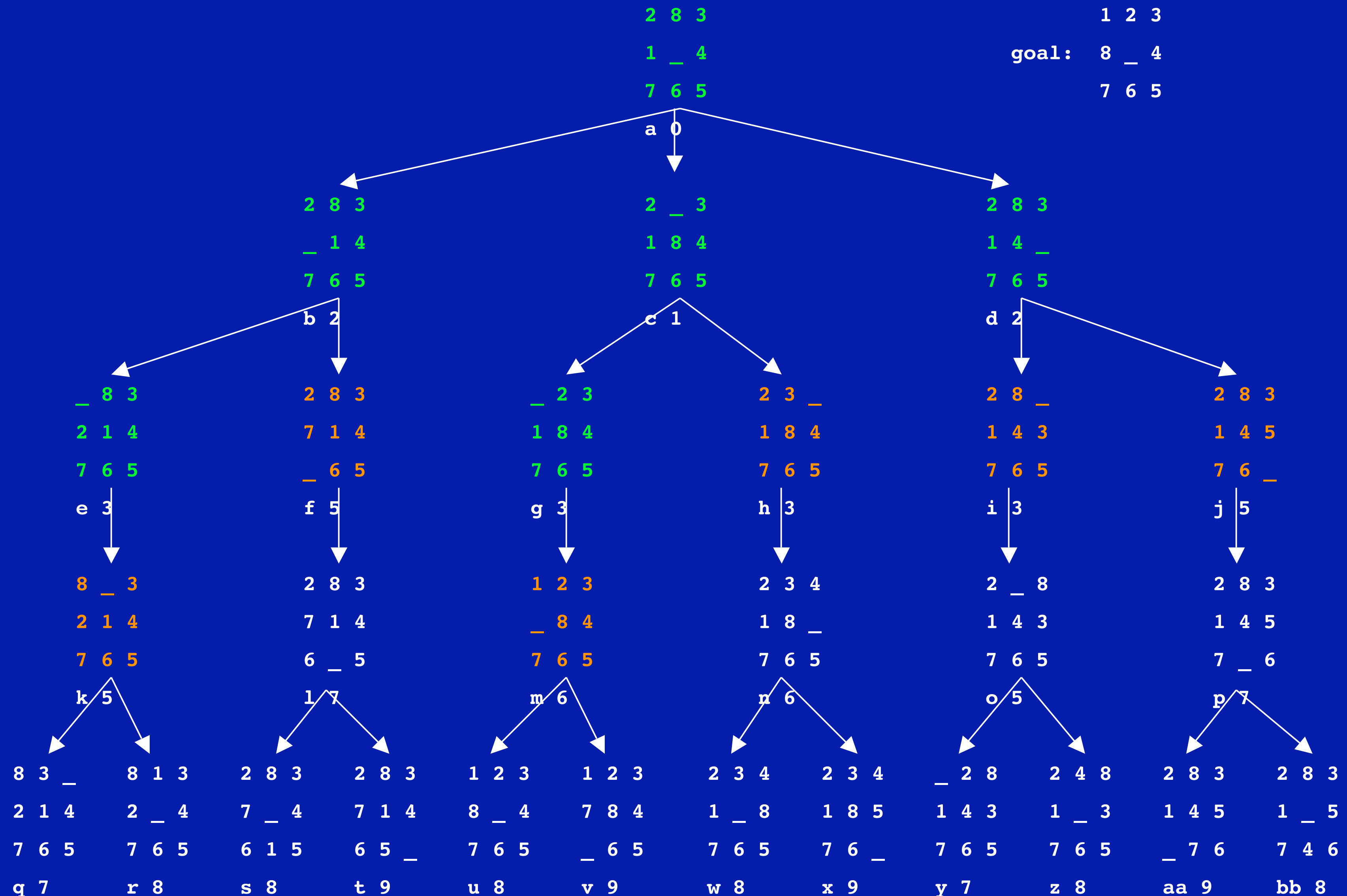
frontier: [g 3,h 3,i 3,f 5,j 5,k 5] *



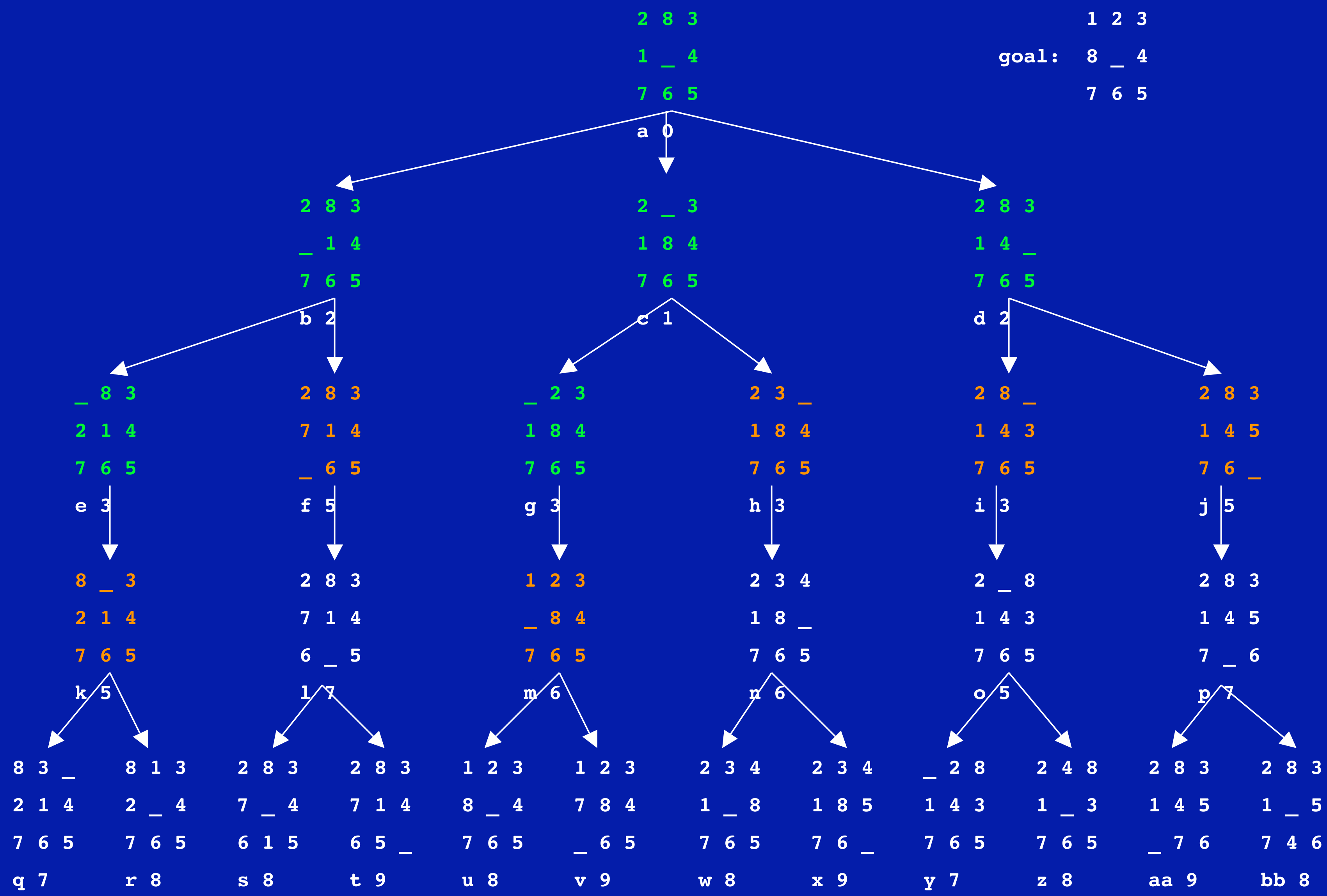
frontier: [h 3,i 3,f 5,j 5,k 5]



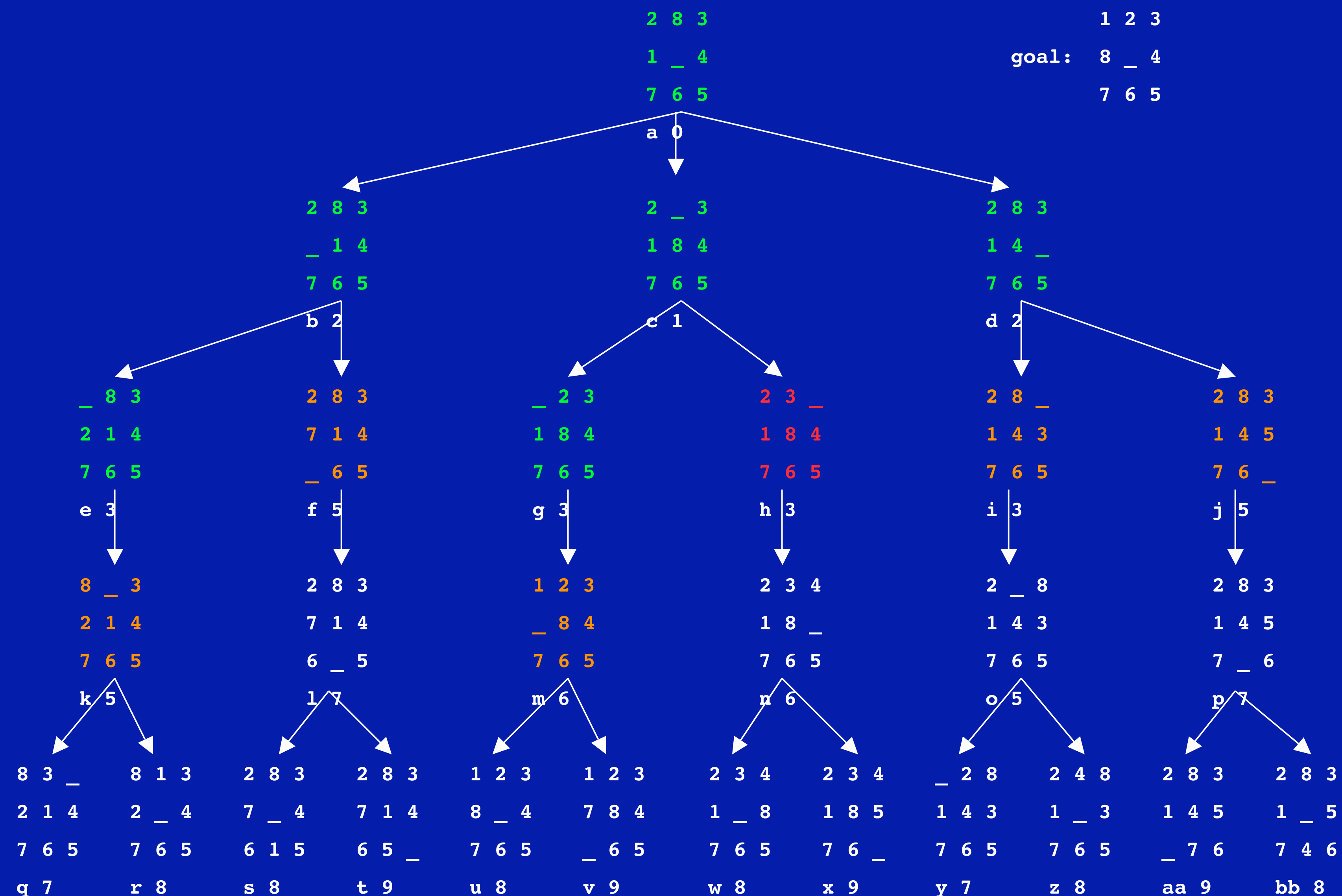
frontier: [m 6,h 3,i 3,f 5,j 5,k 5]



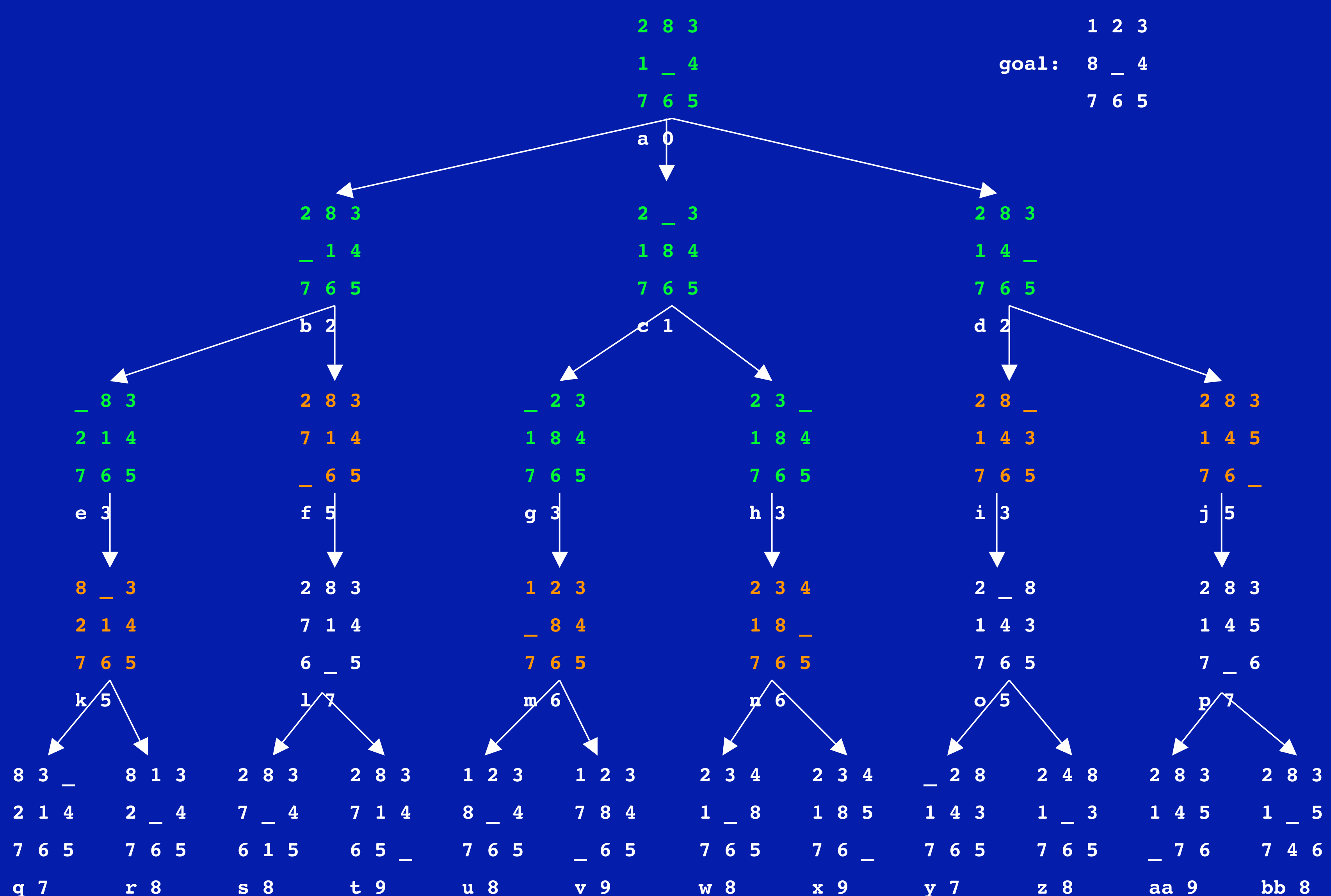
frontier: [h 3,i 3,f 5,j 5,k 5,m 6] *



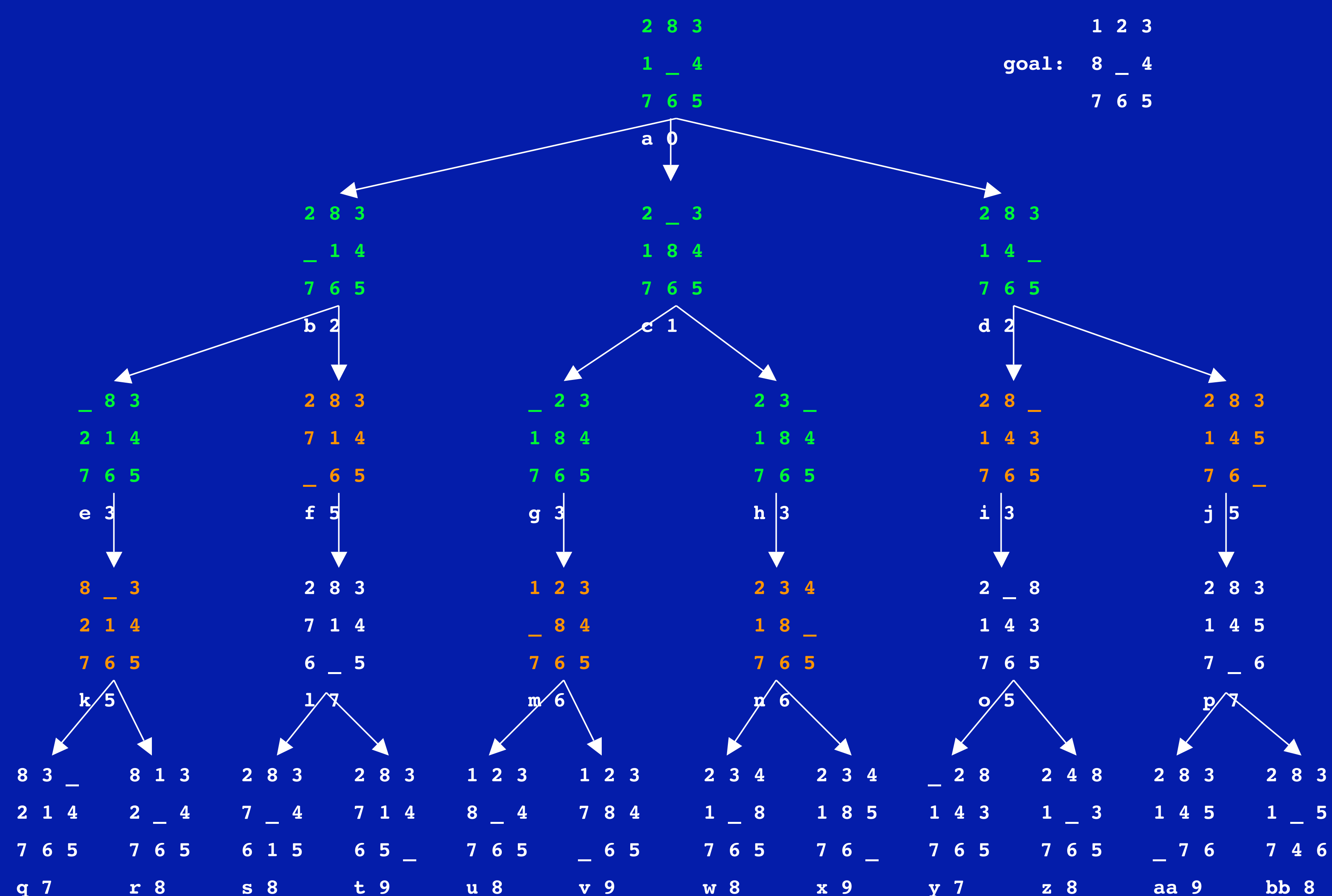
frontier: [i 3,f 5,j 5,k 5,m 6]



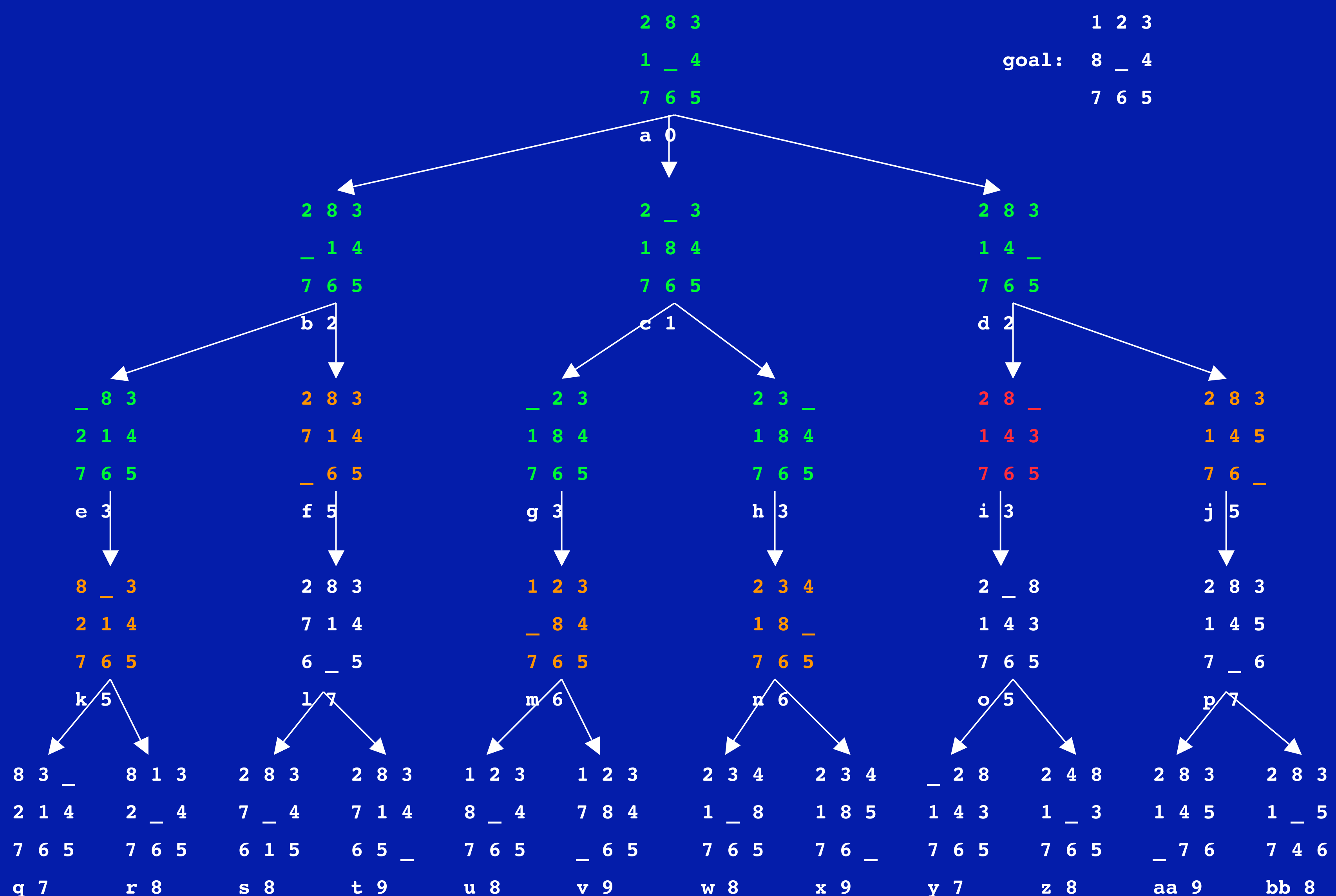
frontier: [n 6,i 3,f 5,j 5,k 5,m 6]



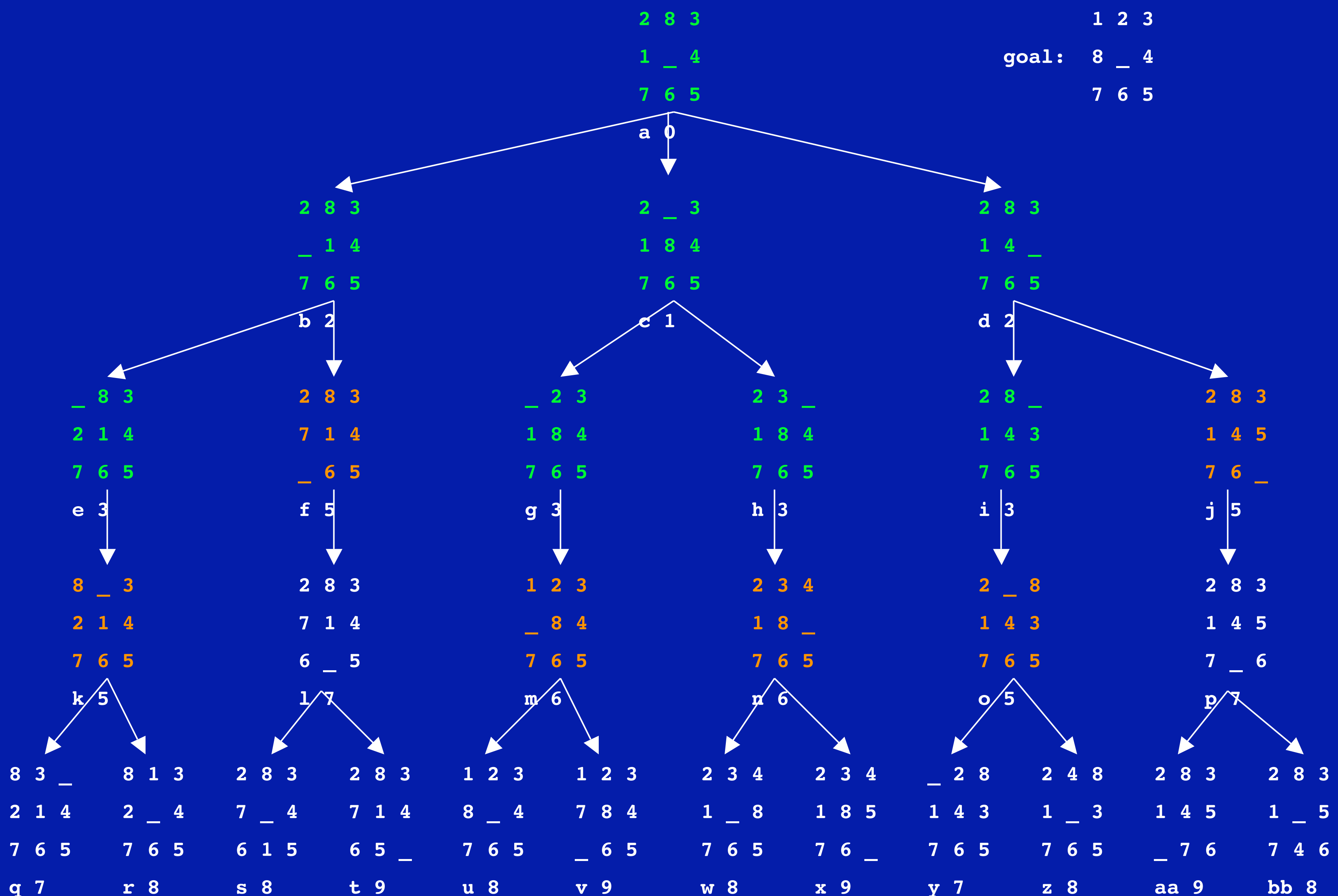
frontier: [i 3,f 5,j 5,k 5,m 6,n 6] *



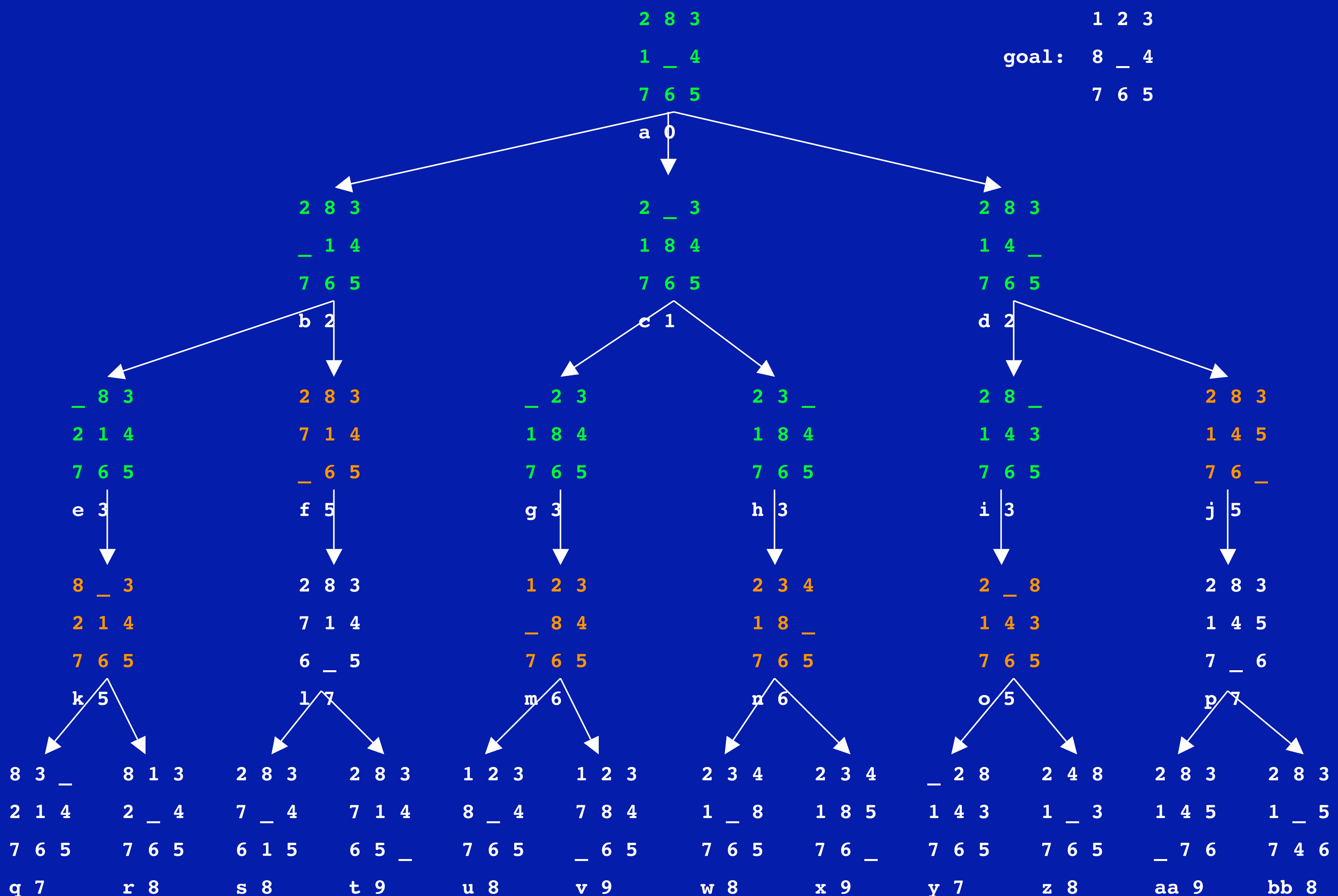
frontier: [f 5,j 5,k 5,m 6,n 6]



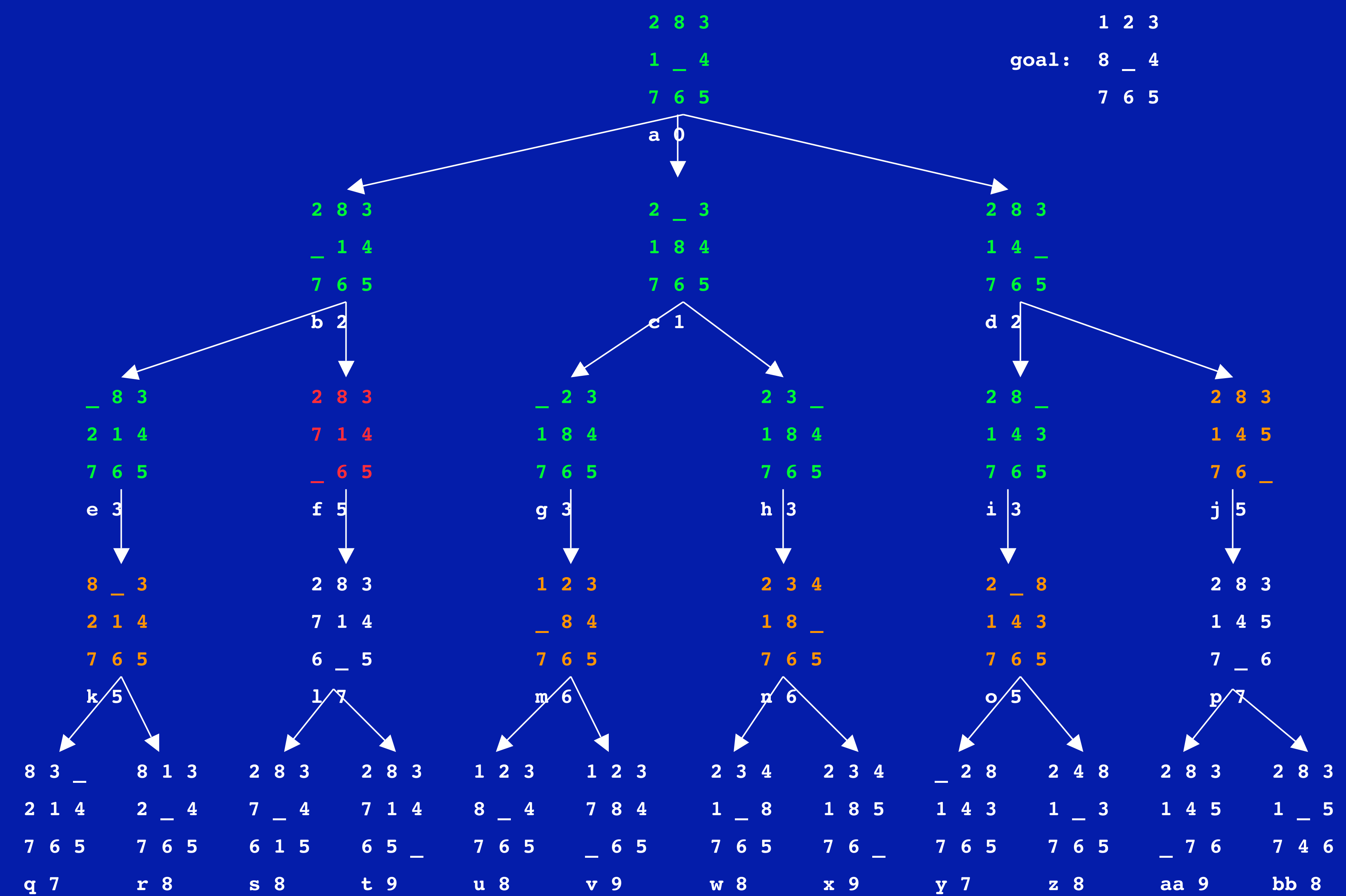
frontier: [o 5,f 5,j 5,k 5,m 6,n 6]



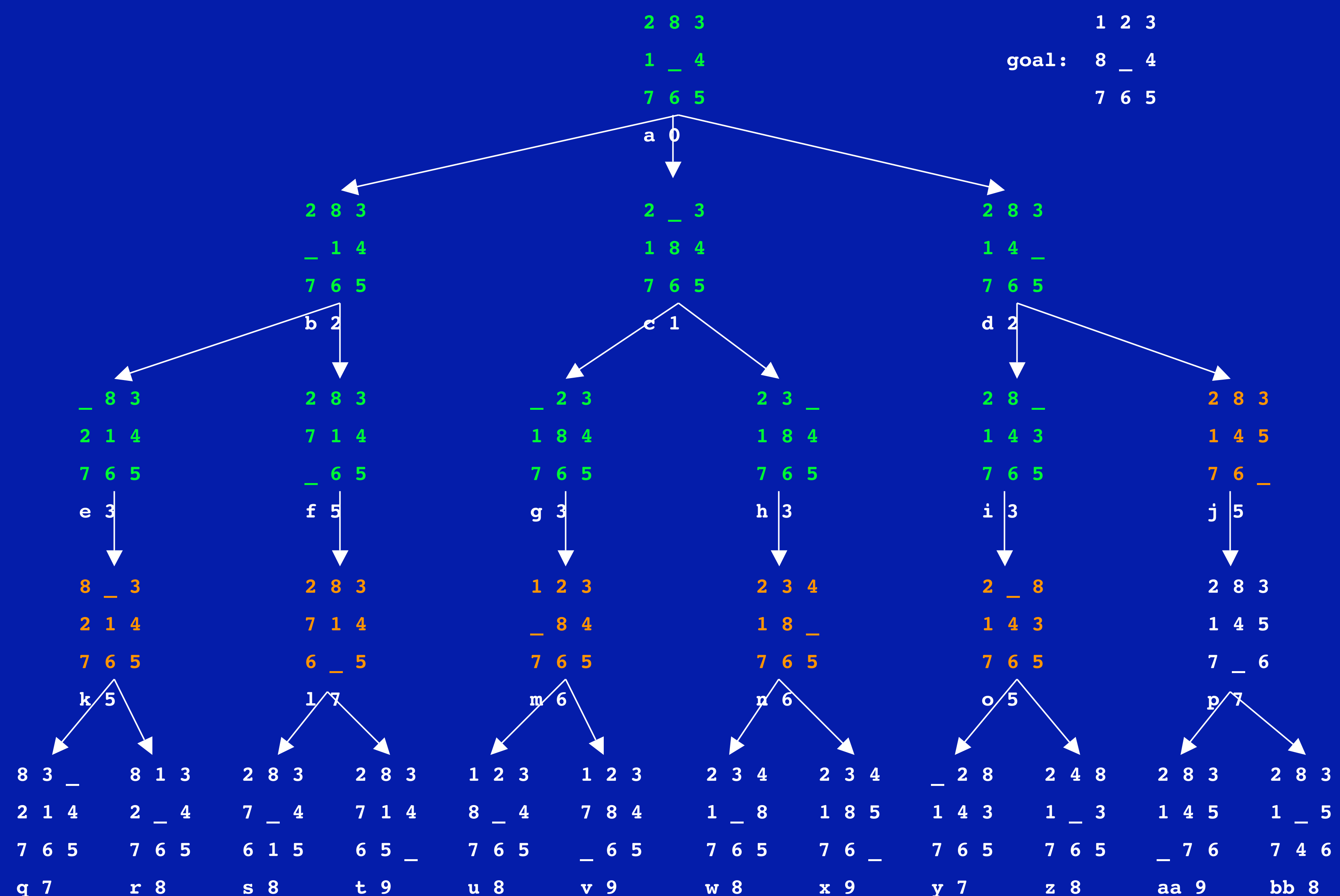
frontier: [f 5,j 5,k 5,o 5,m 6,n 6] *



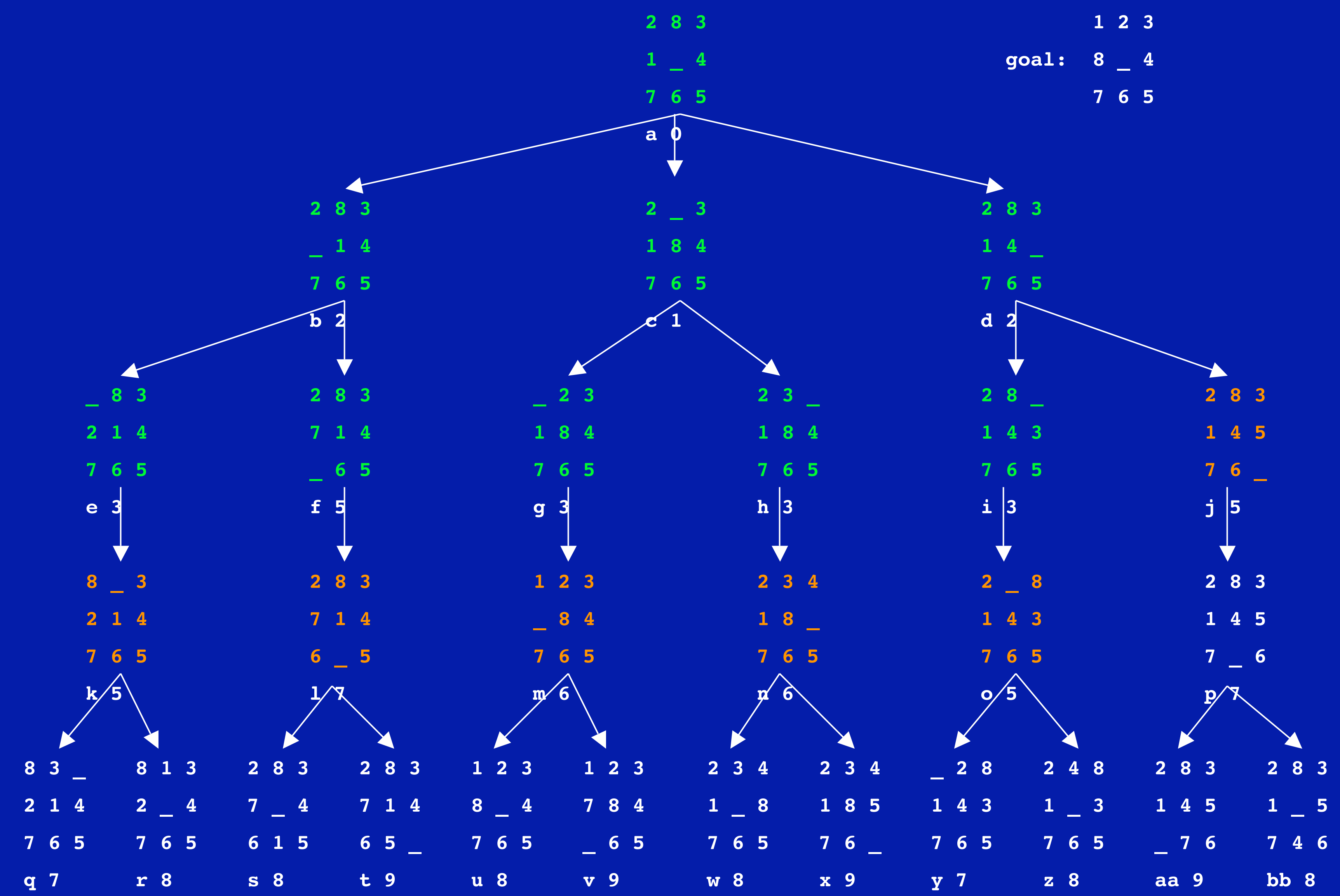
frontier: [j 5,k 5,o 5,m 6,n 6]



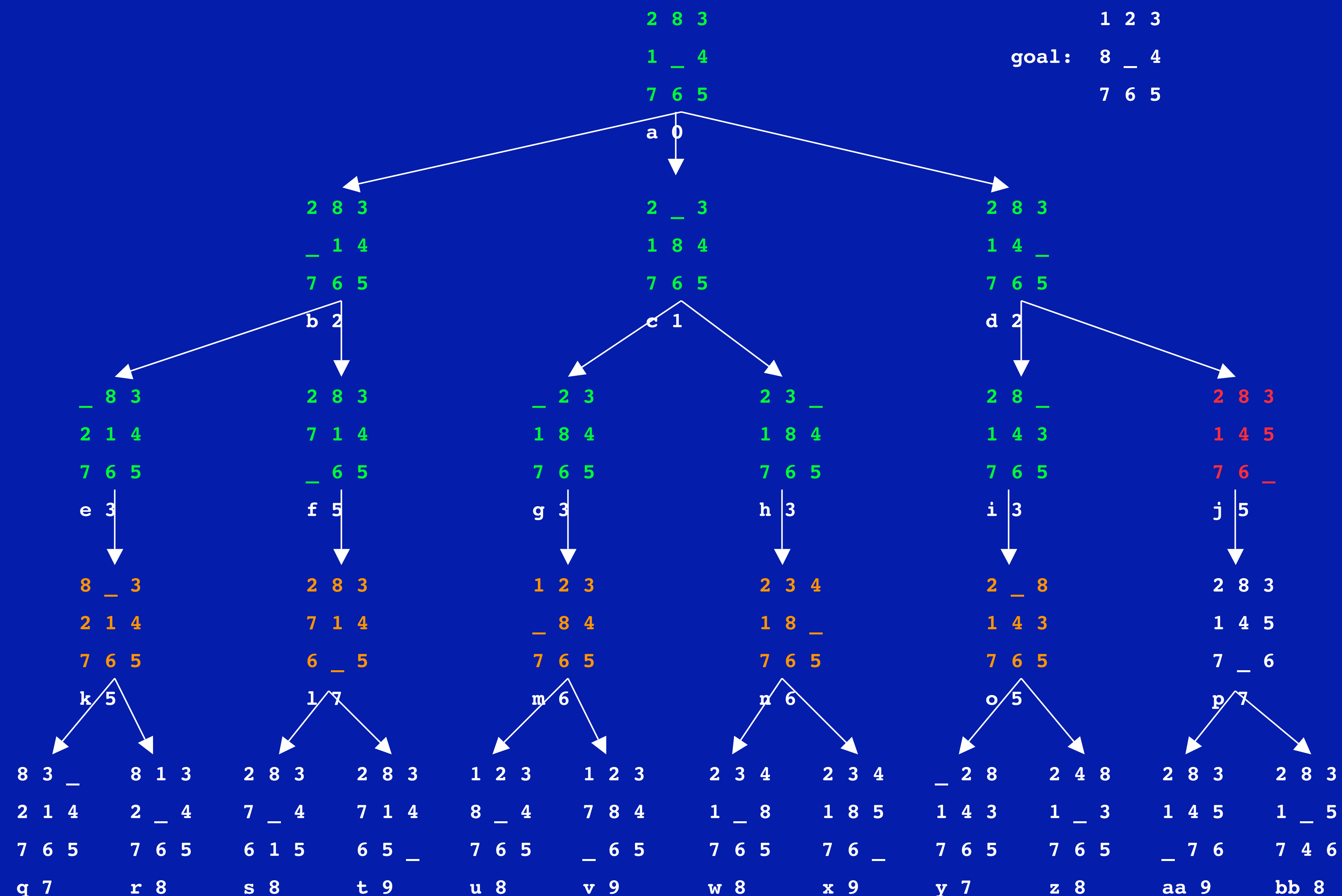
frontier: [l 7,j 5,k 5,o 5,m 6,n 6]



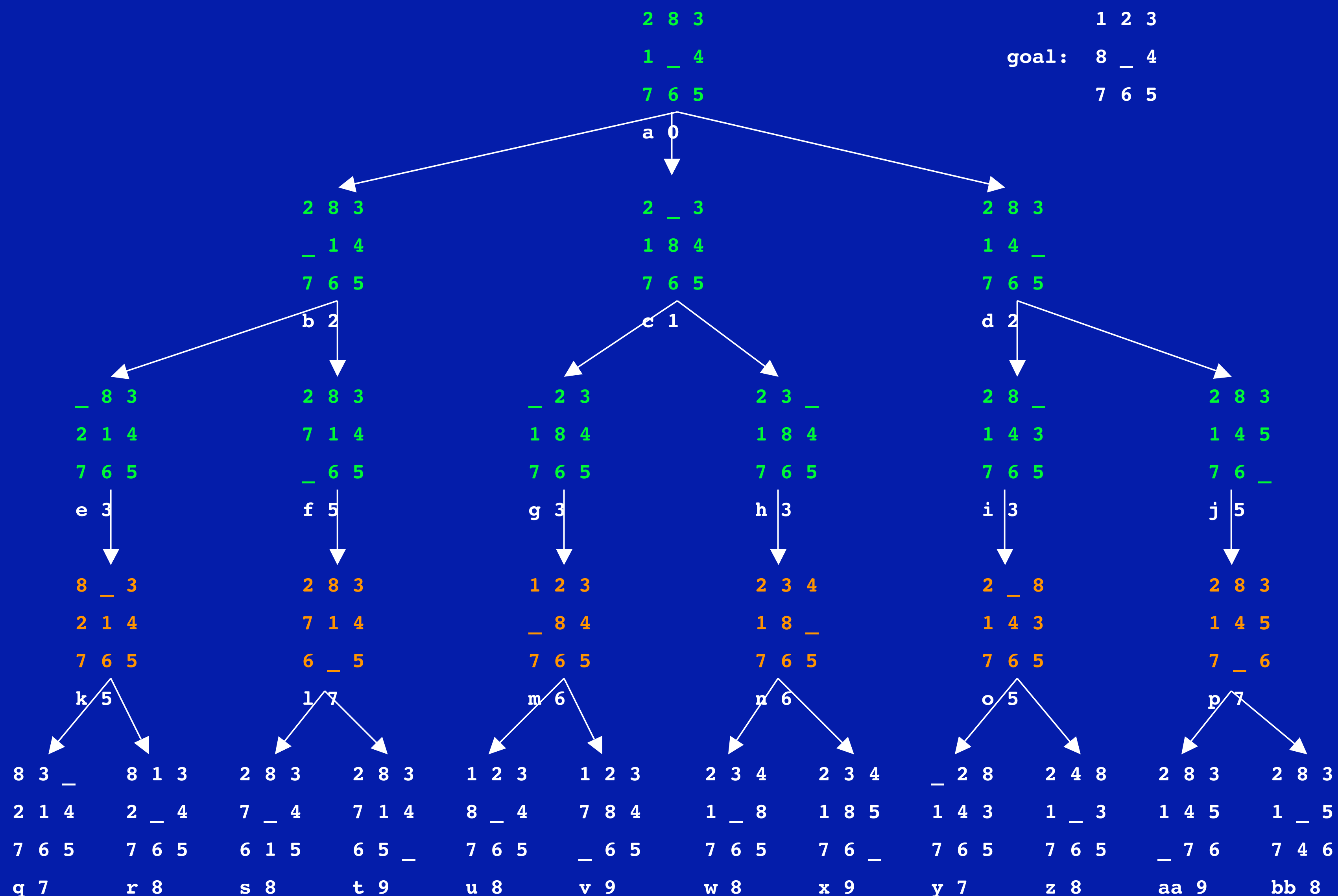
frontier: [j 5,k 5,o 5,m 6,n 6,l 7] *



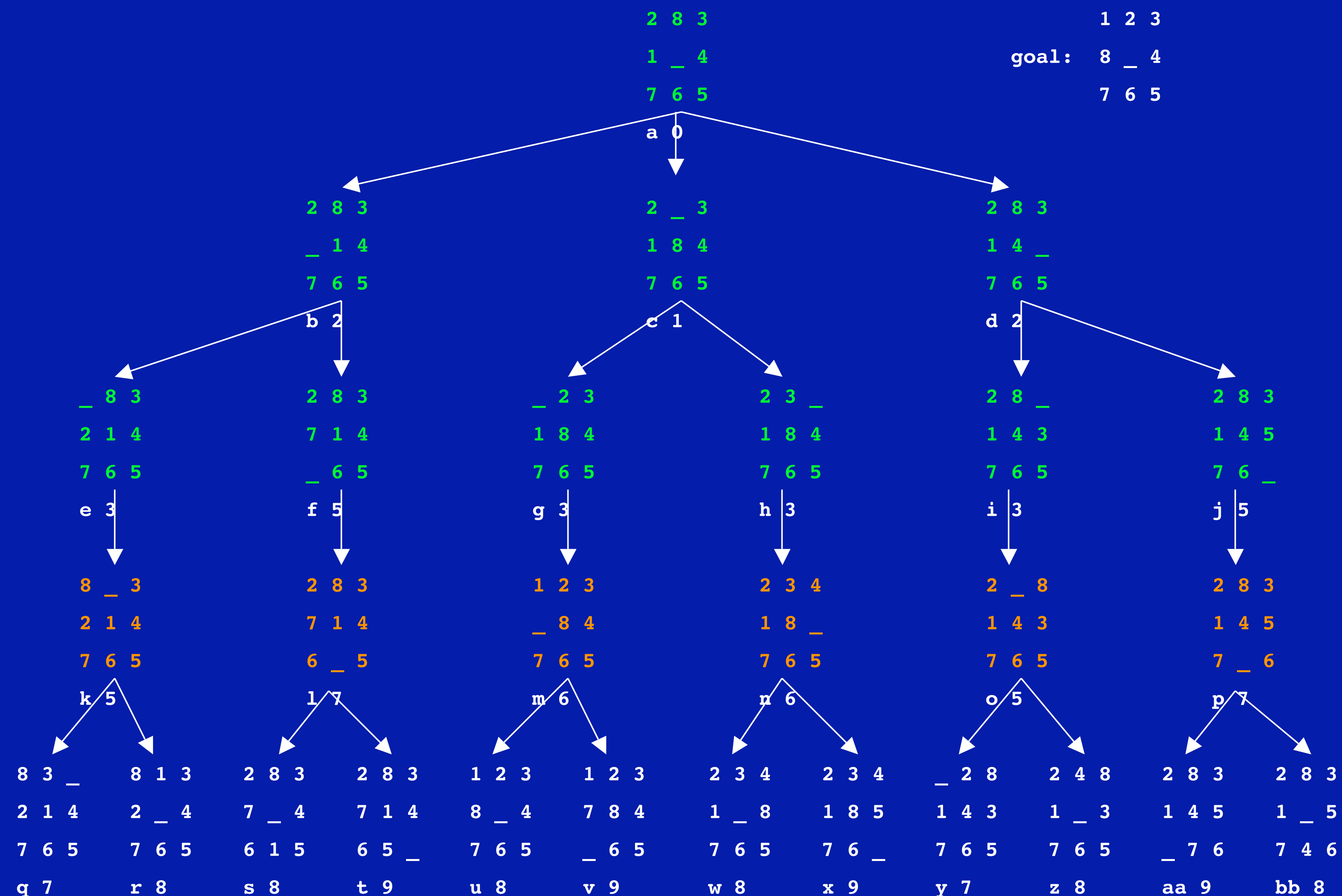
frontier: [k 5,o 5,m 6,n 6,l 7]



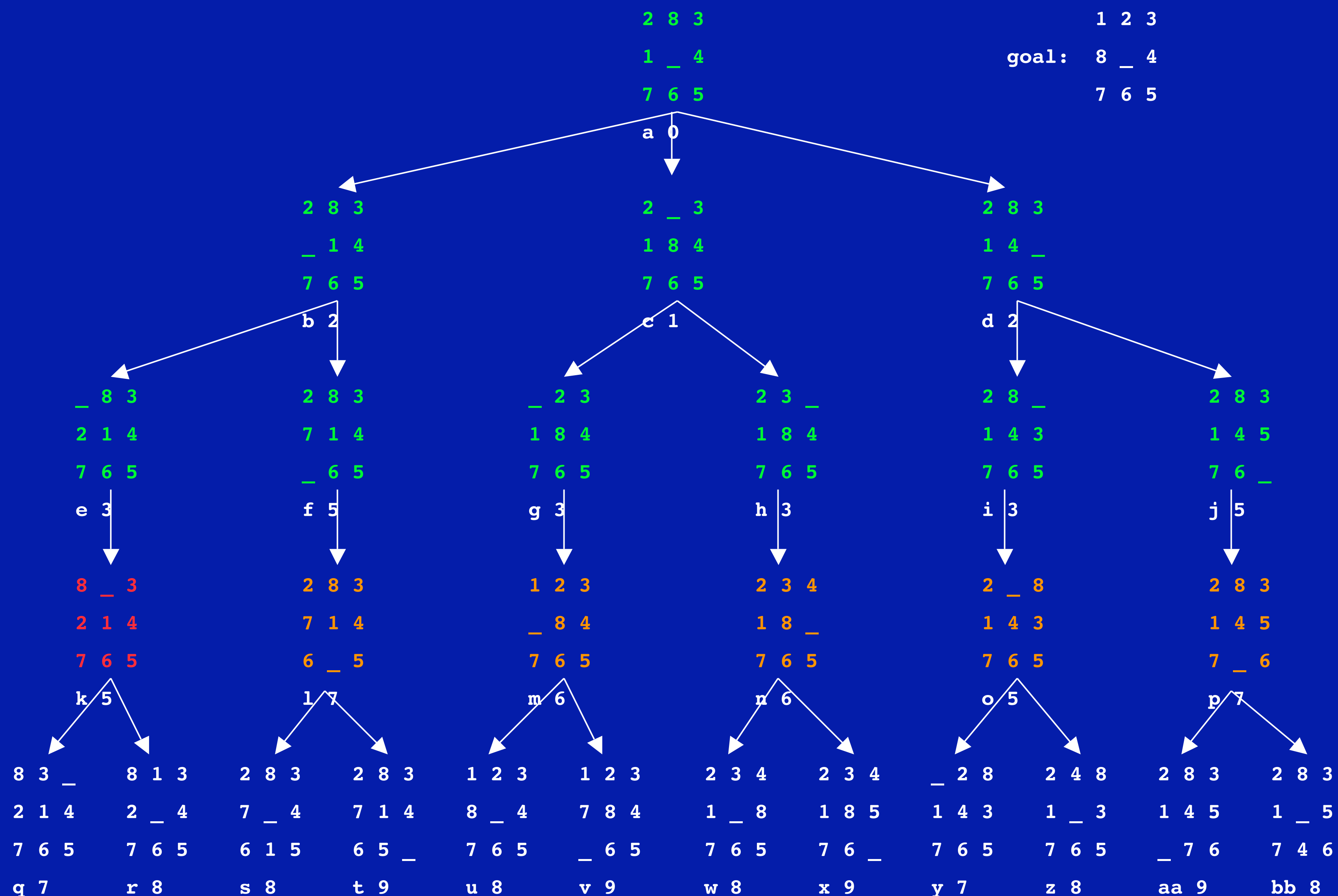
frontier: [p 7,k 5,o 5,m 6,n 6,l 7]



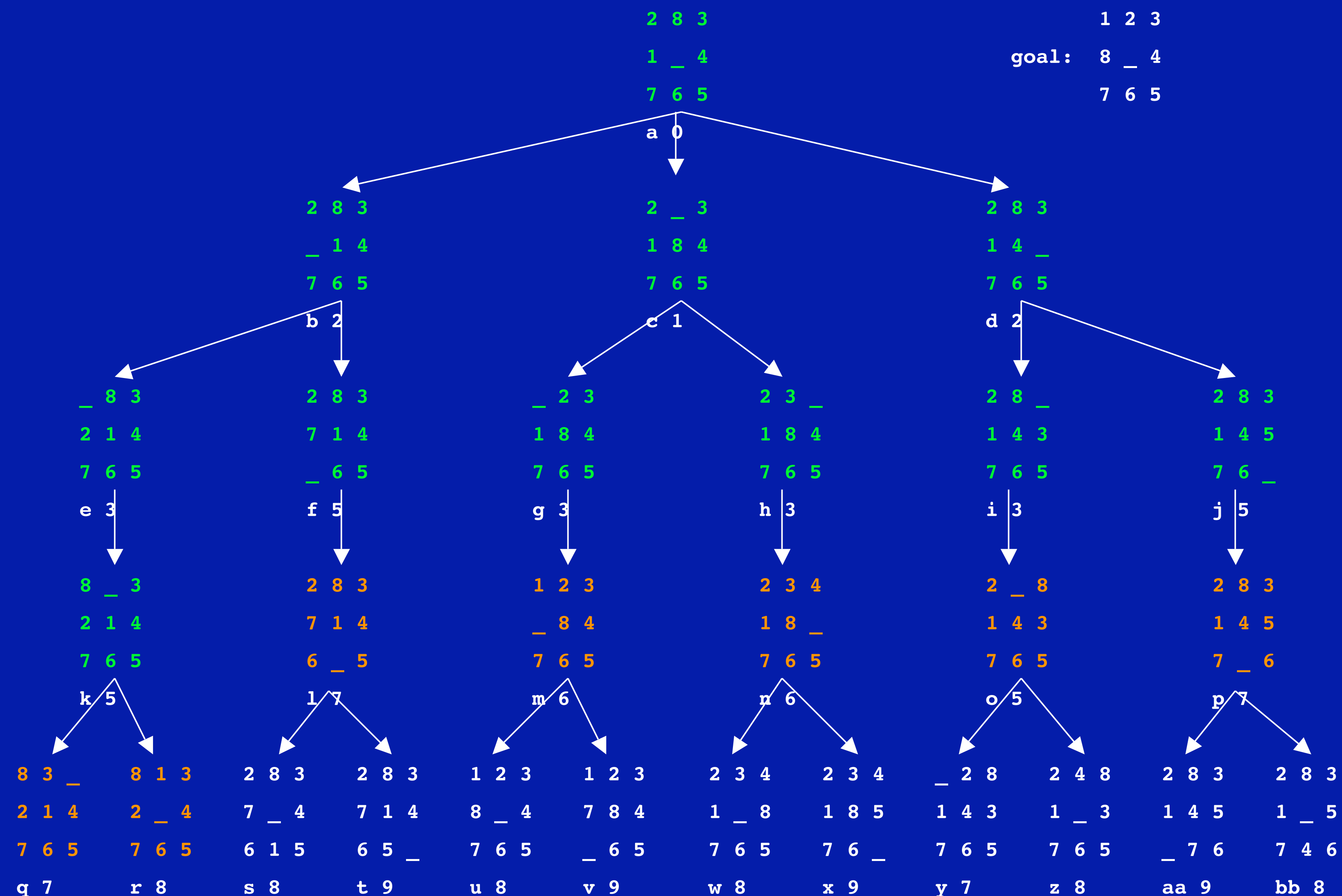
frontier: [k 5,o 5,m 6,n 6,l 7,p 7] *



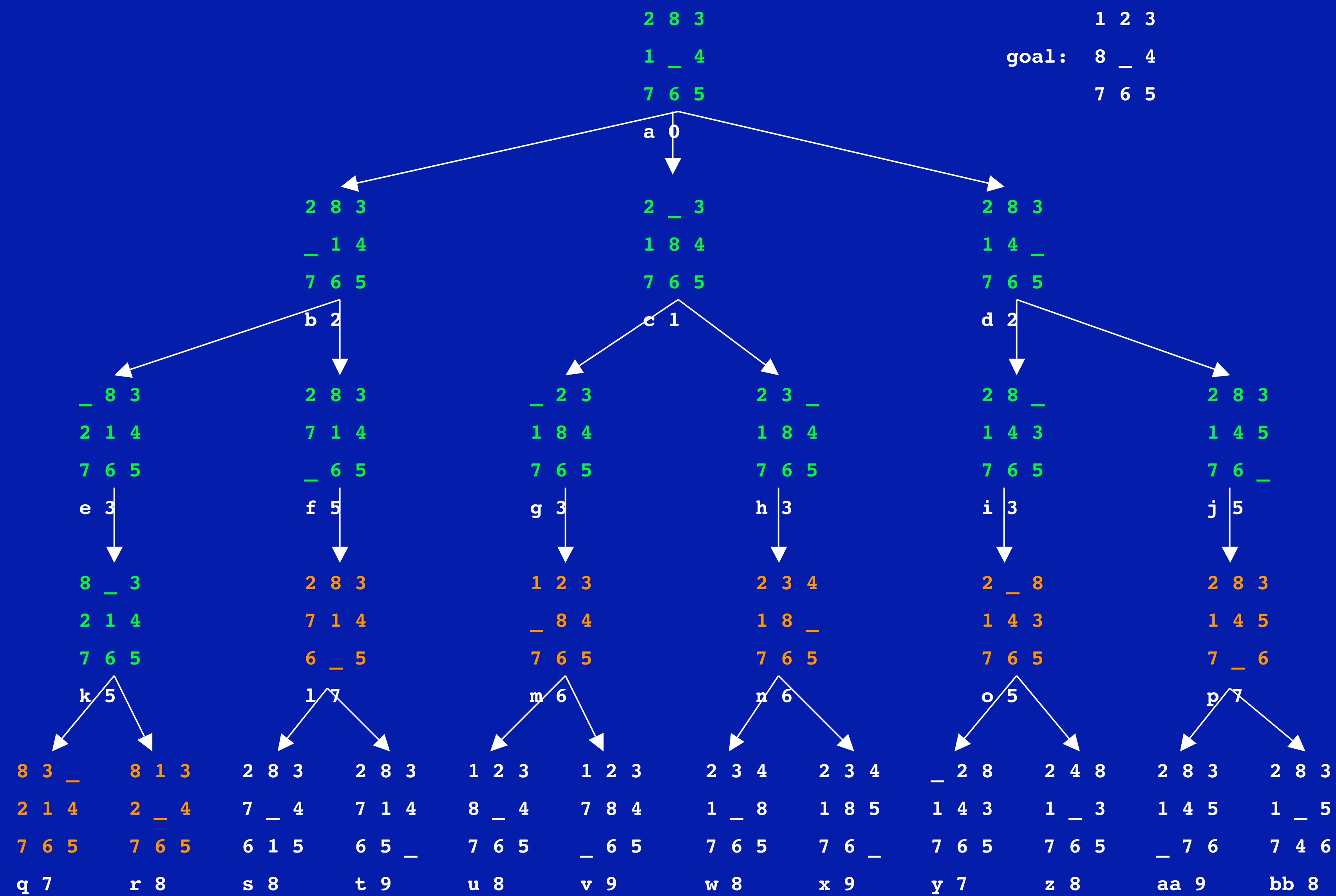
frontier: [o 5,m 6,n 6,l 7,p 7]



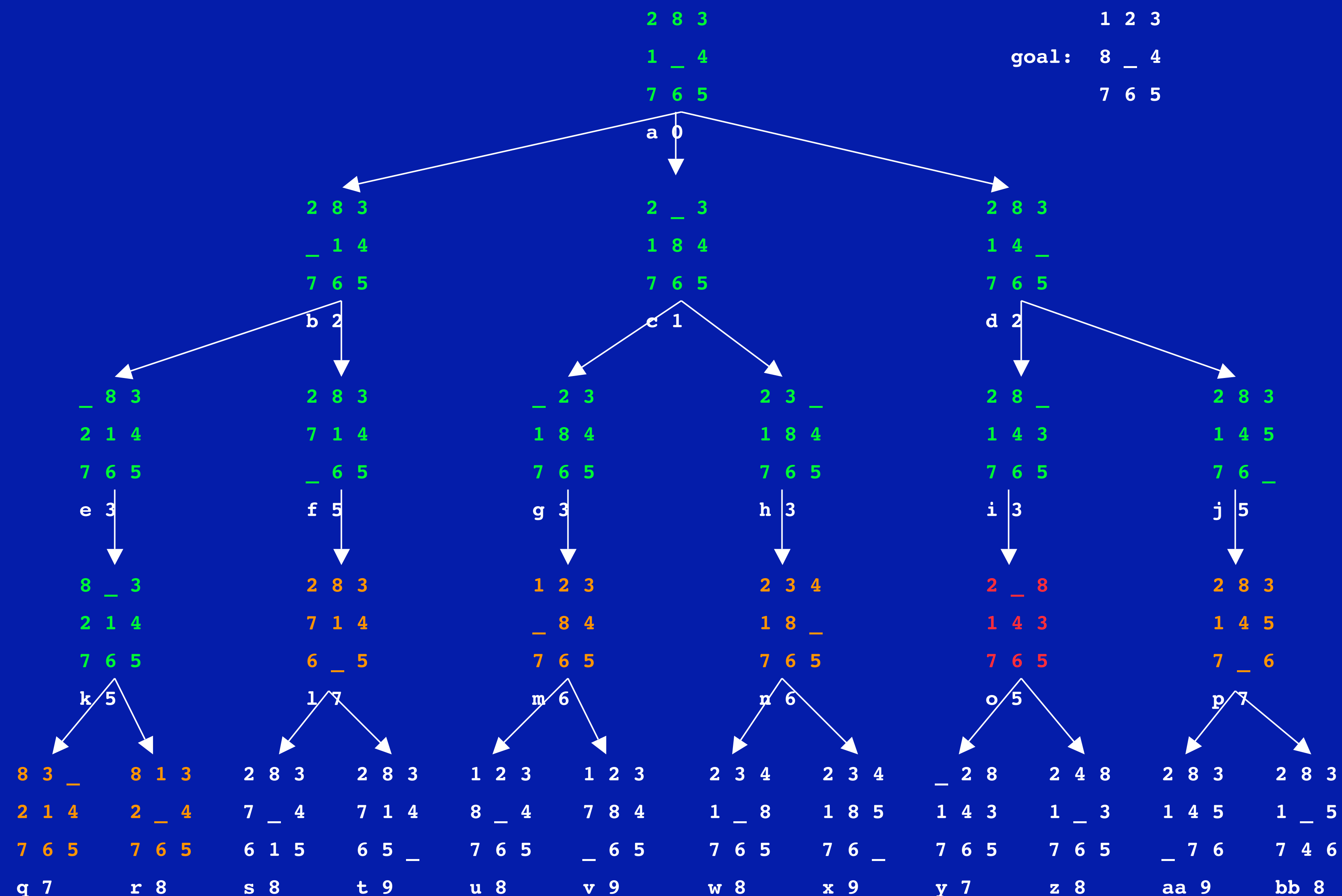
frontier: [q 7,r 8,o 5,m 6,n 6,l 7,p 7]



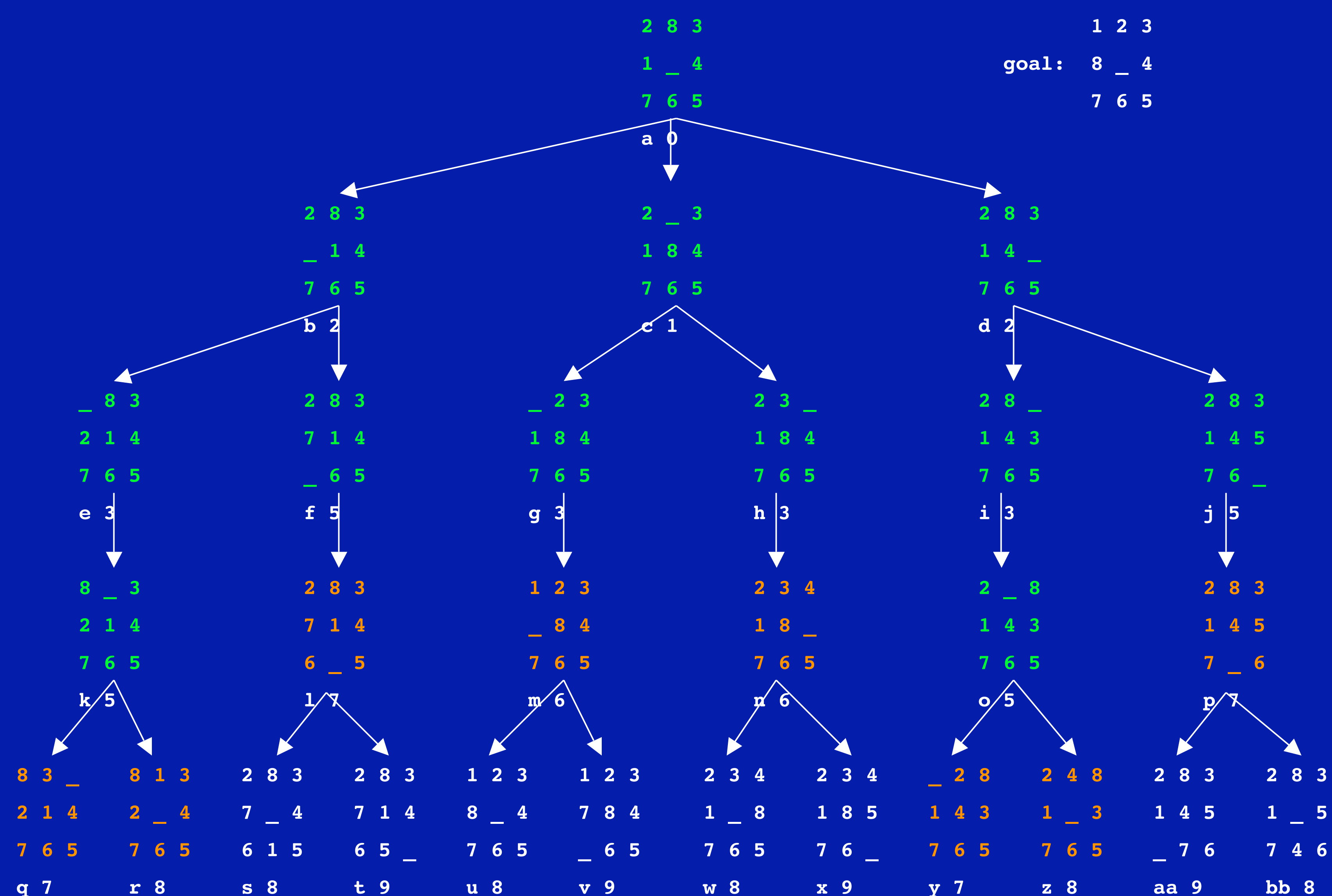
frontier: [o 5,m 6,n 6,l 7,p 7,q 7,r 8] *



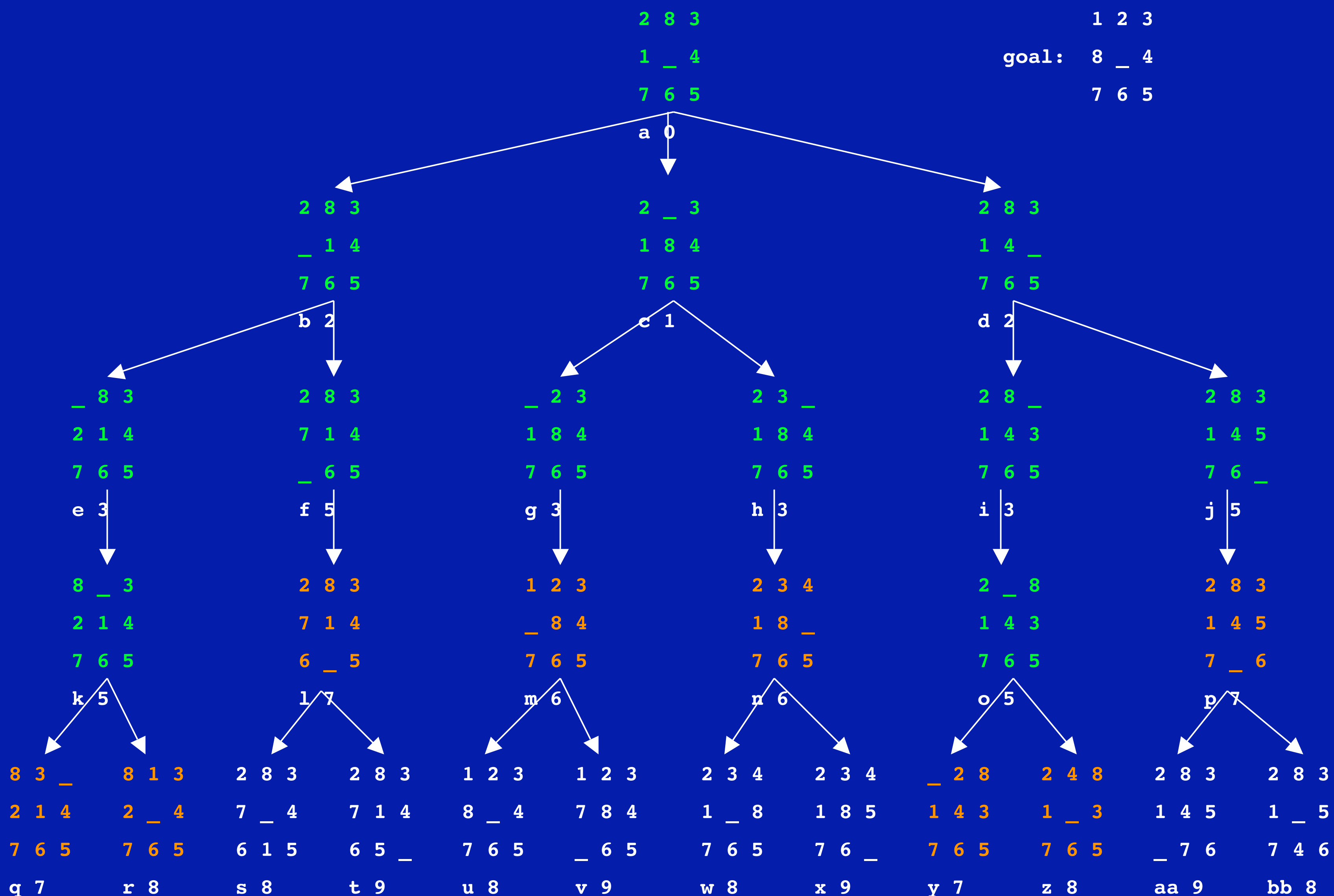
frontier: [m 6,n 6,l 7,p 7,q 7,r 8]



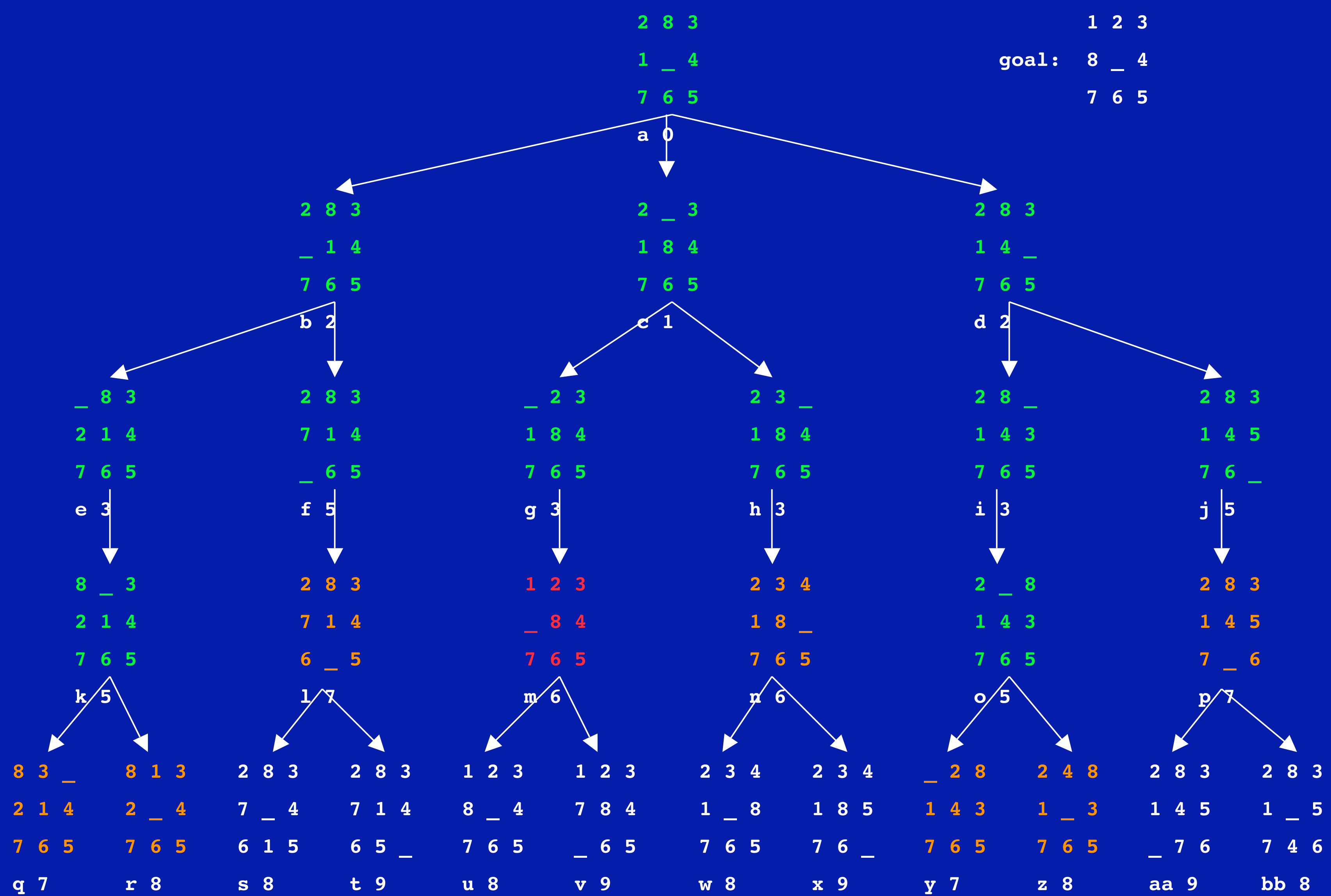
frontier: [y 7,z 8,m 6,n 6,l 7,p 7,q 7,r 8]



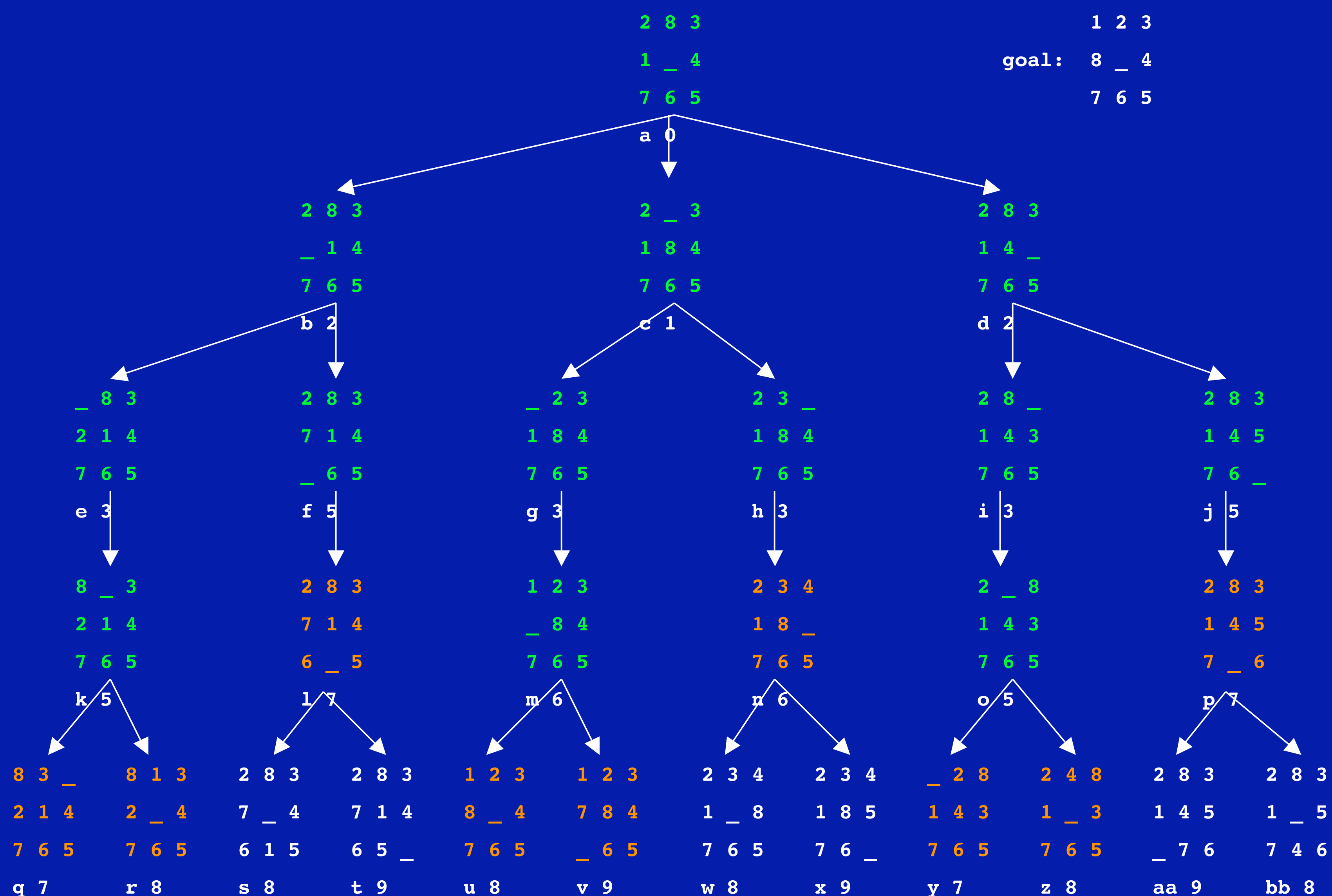
frontier: [m 6,n 6,l 7,p 7,q 7,y 7,r 8,z 8] *



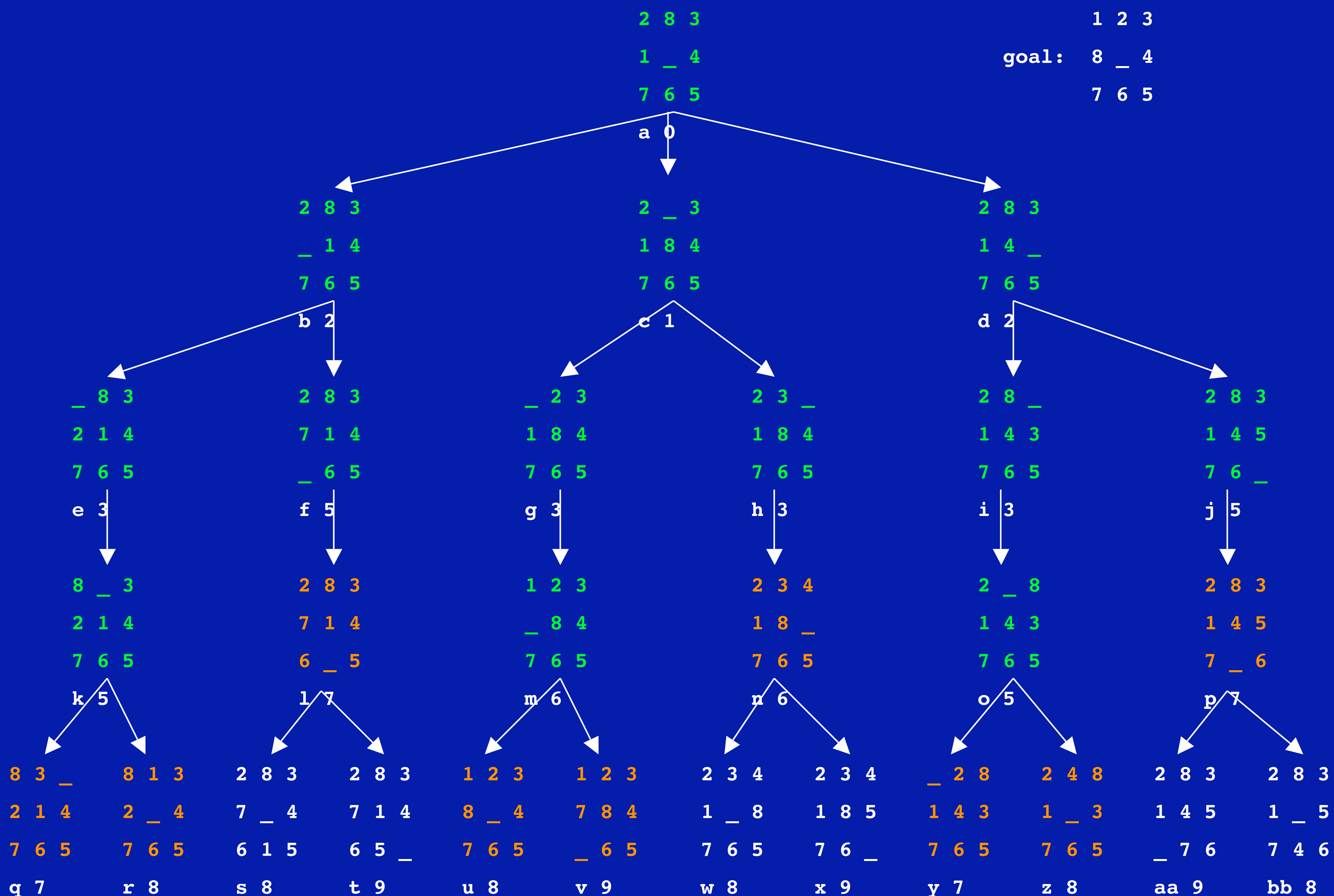
frontier: [n 6,l 7,p 7,q 7,y 7,r 8,z 8]



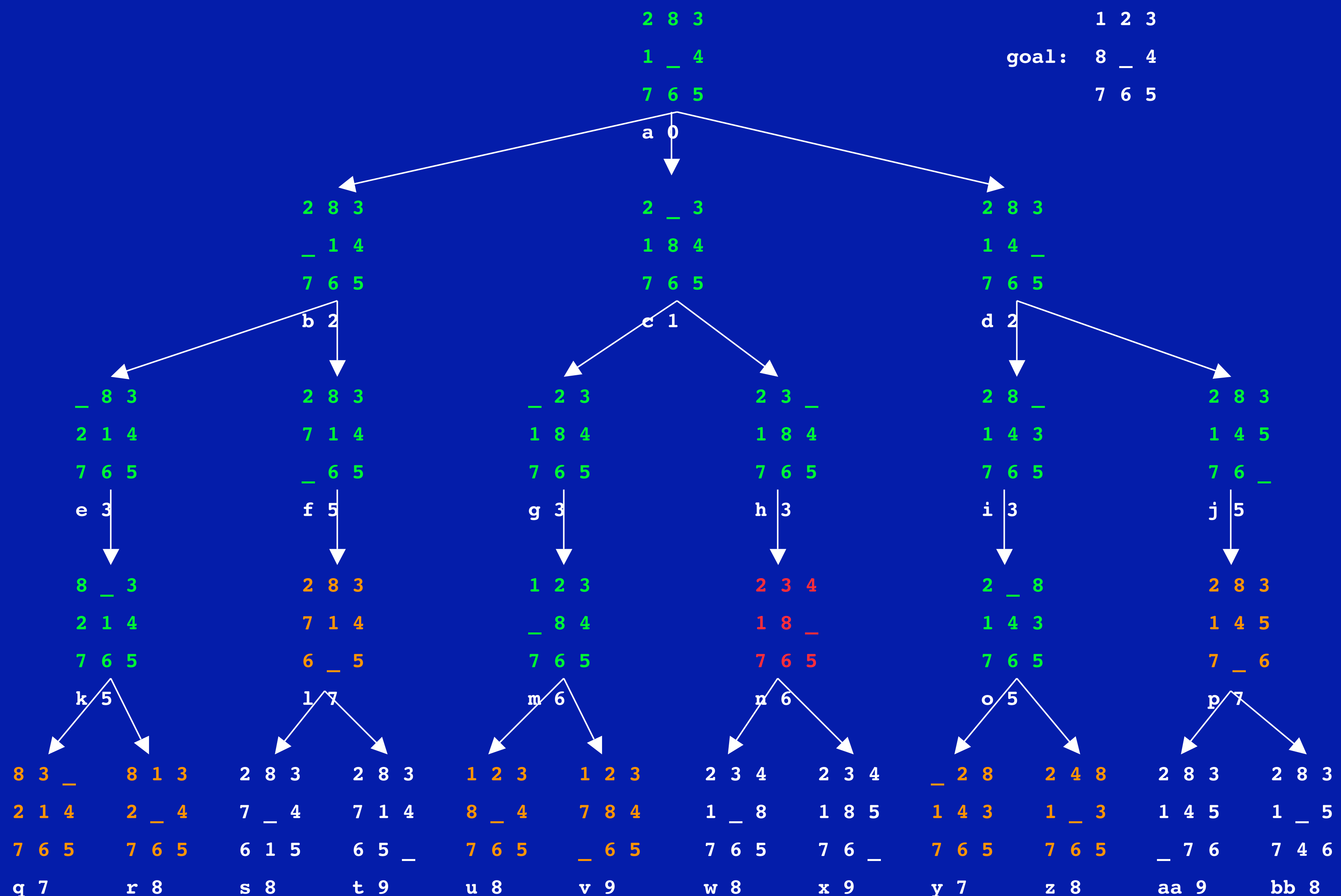
frontier: [u 8,v 9,n 6,l 7,p 7,q 7,y 7,r 8,z 8]



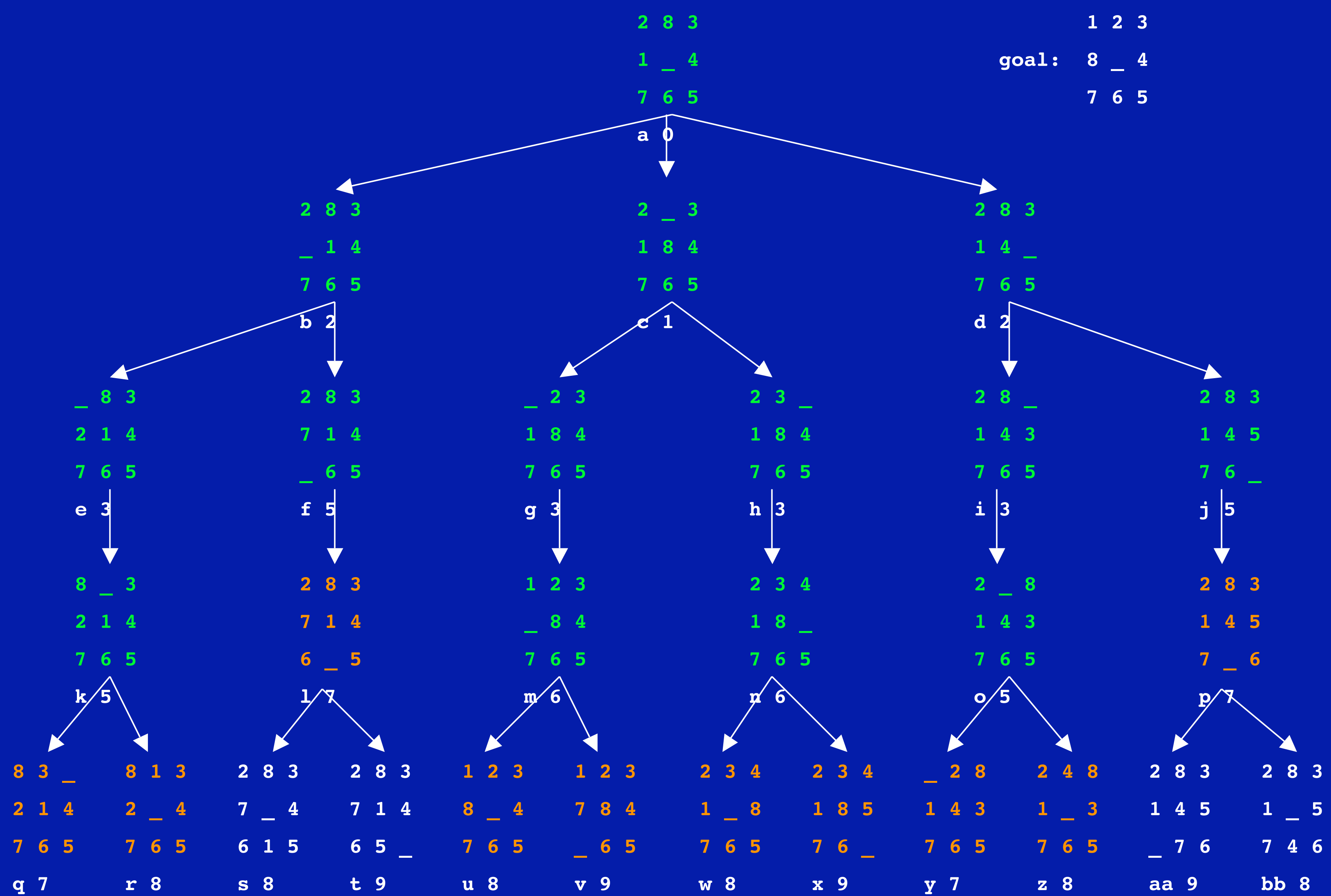
frontier: [n 6,l 7,p 7,q 7,y 7,r 8,u 8,z 8,v 9] *



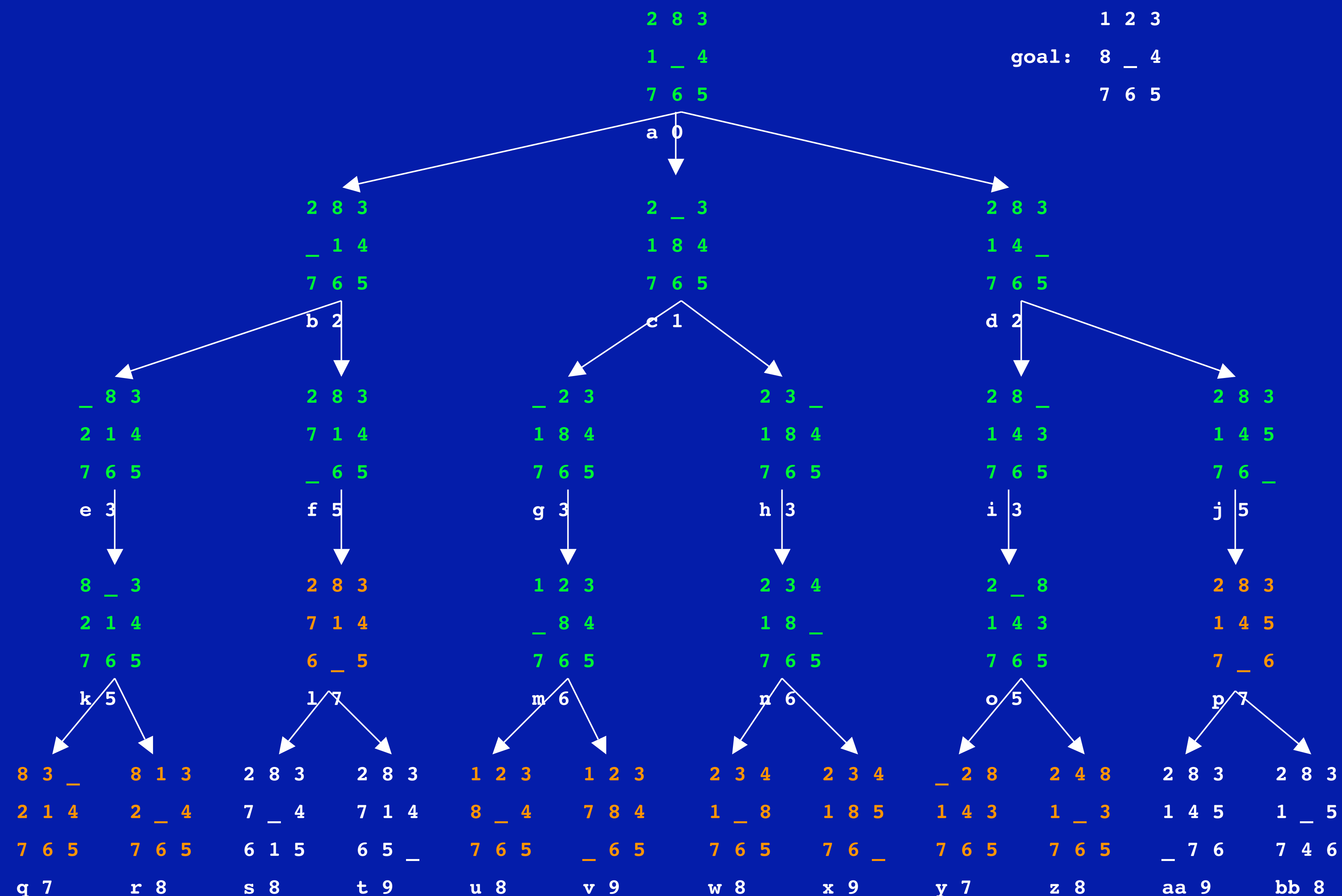
frontier: [l 7,p 7,q 7,y 7,r 8,u 8,z 8,v 9]



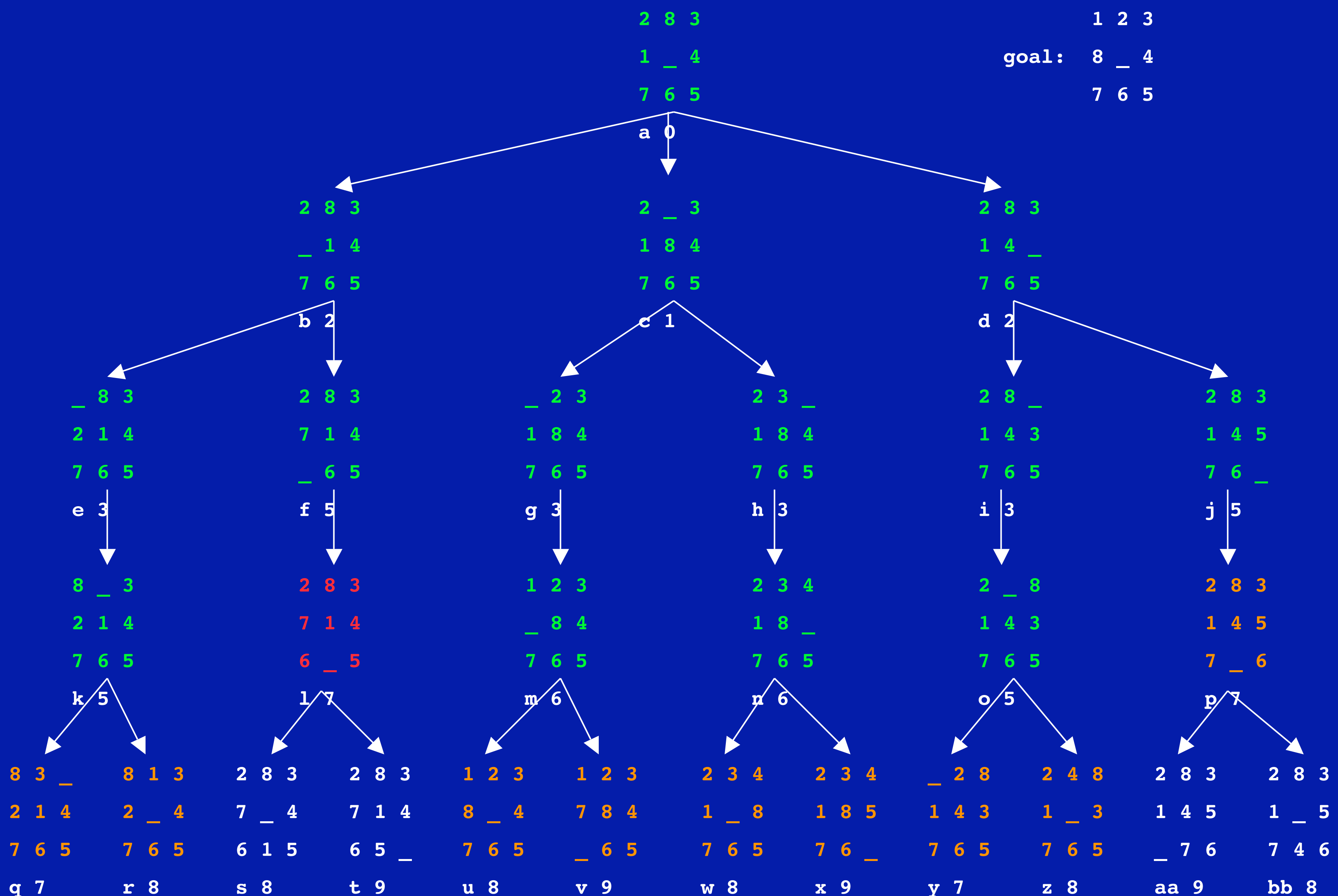
frontier: [w 8,x 9,l 7,p 7,q 7,y 7,r 8,u 8,z 8,v 9]



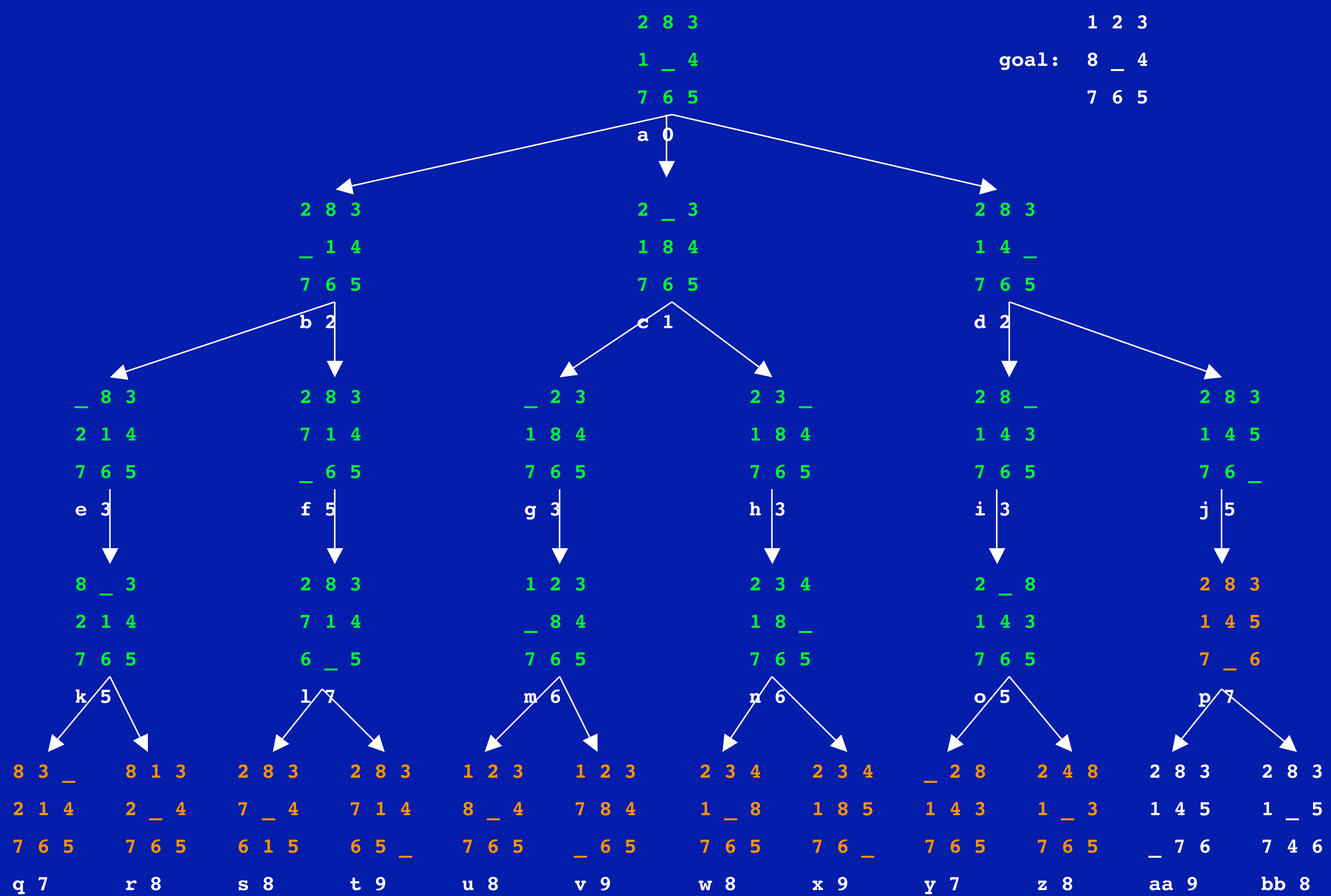
frontier: [l 7,p 7,q 7,y 7,r 8,u 8,w 8,z 8,v 9,x 9] *



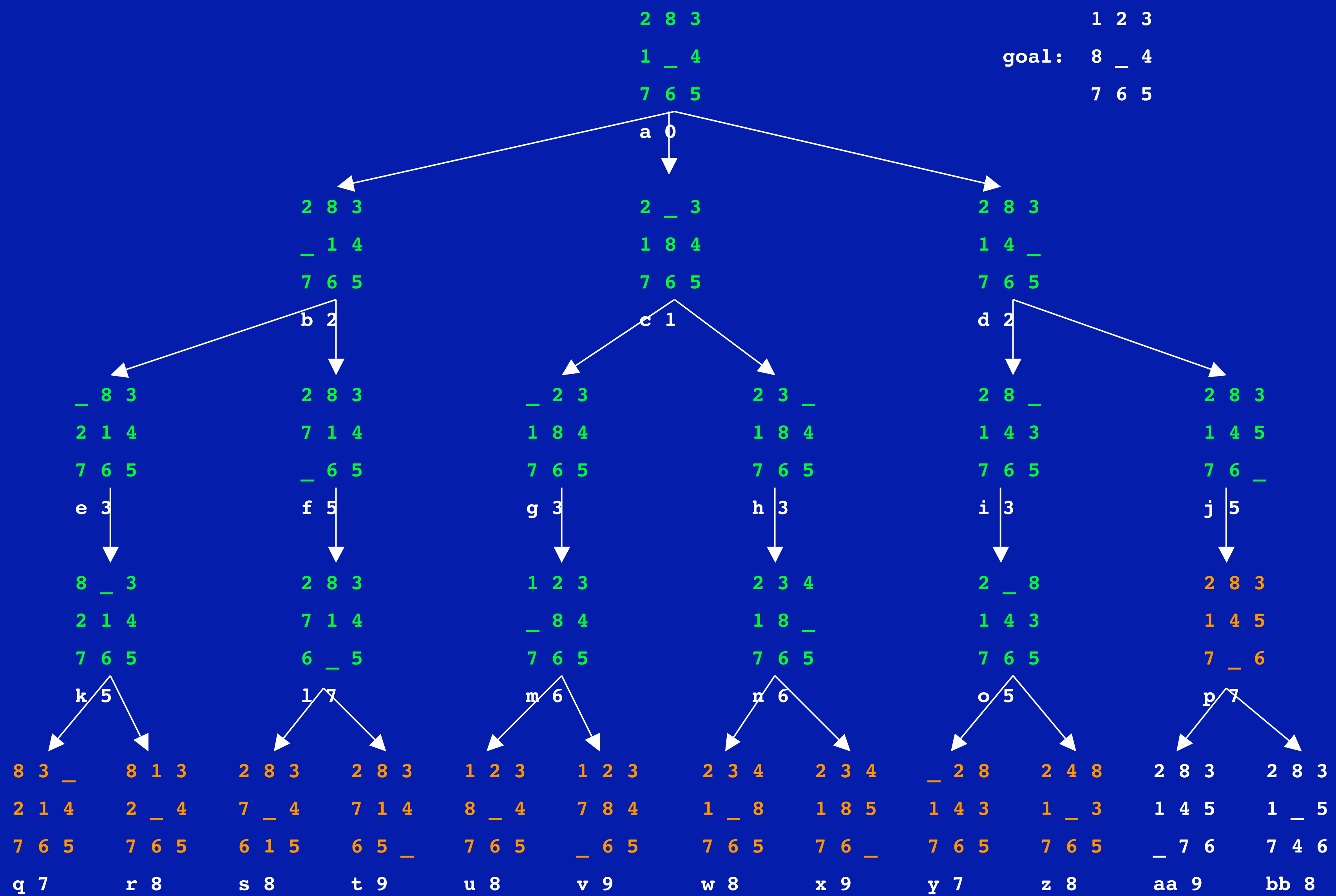
frontier: [p 7,q 7,y 7,r 8,u 8,w 8,z 8,v 9,x 9]



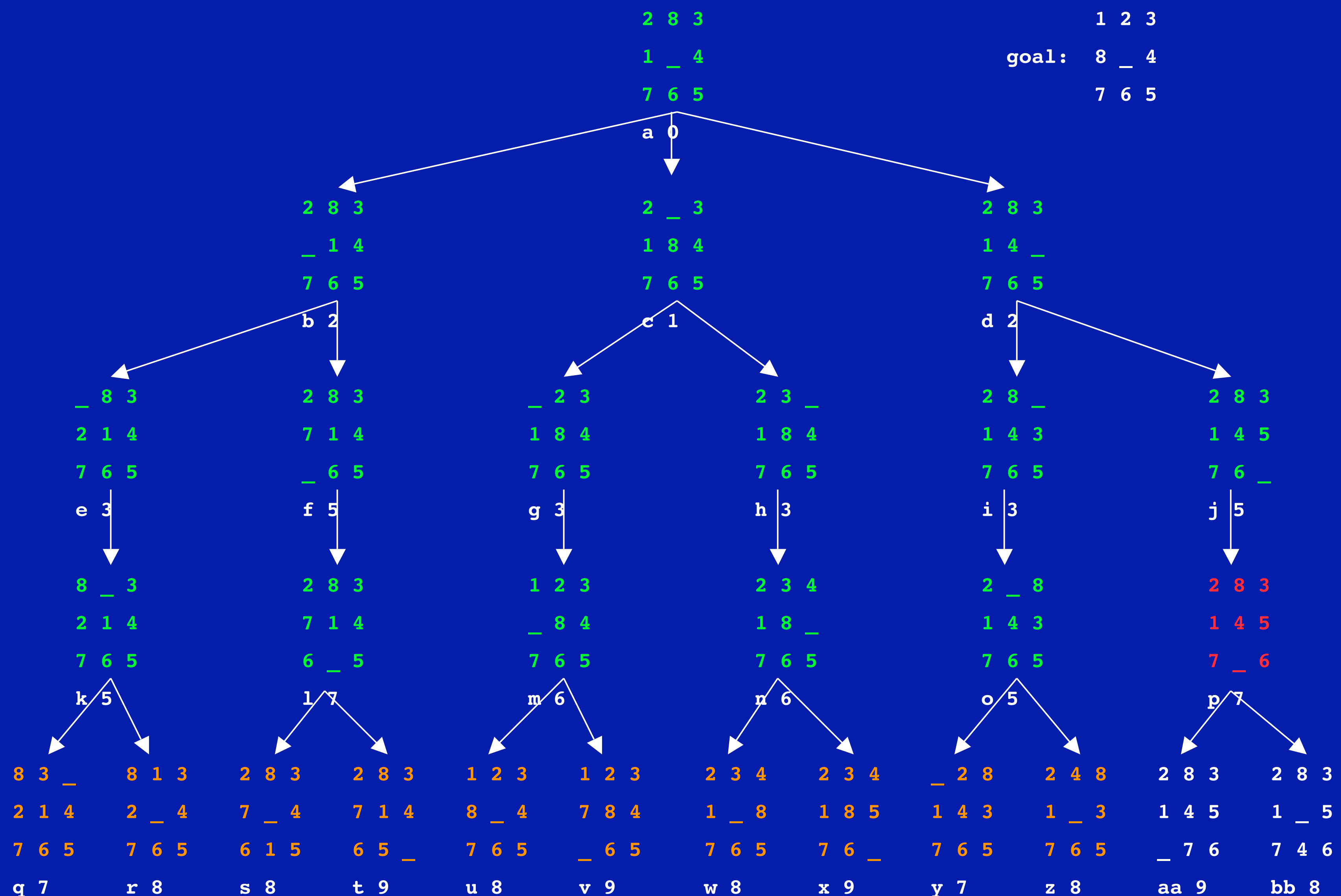
frontier: [s 8,t 9,p 7,q 7,y 7,r 8,u 8,w 8,z 8,v 9,x 9]



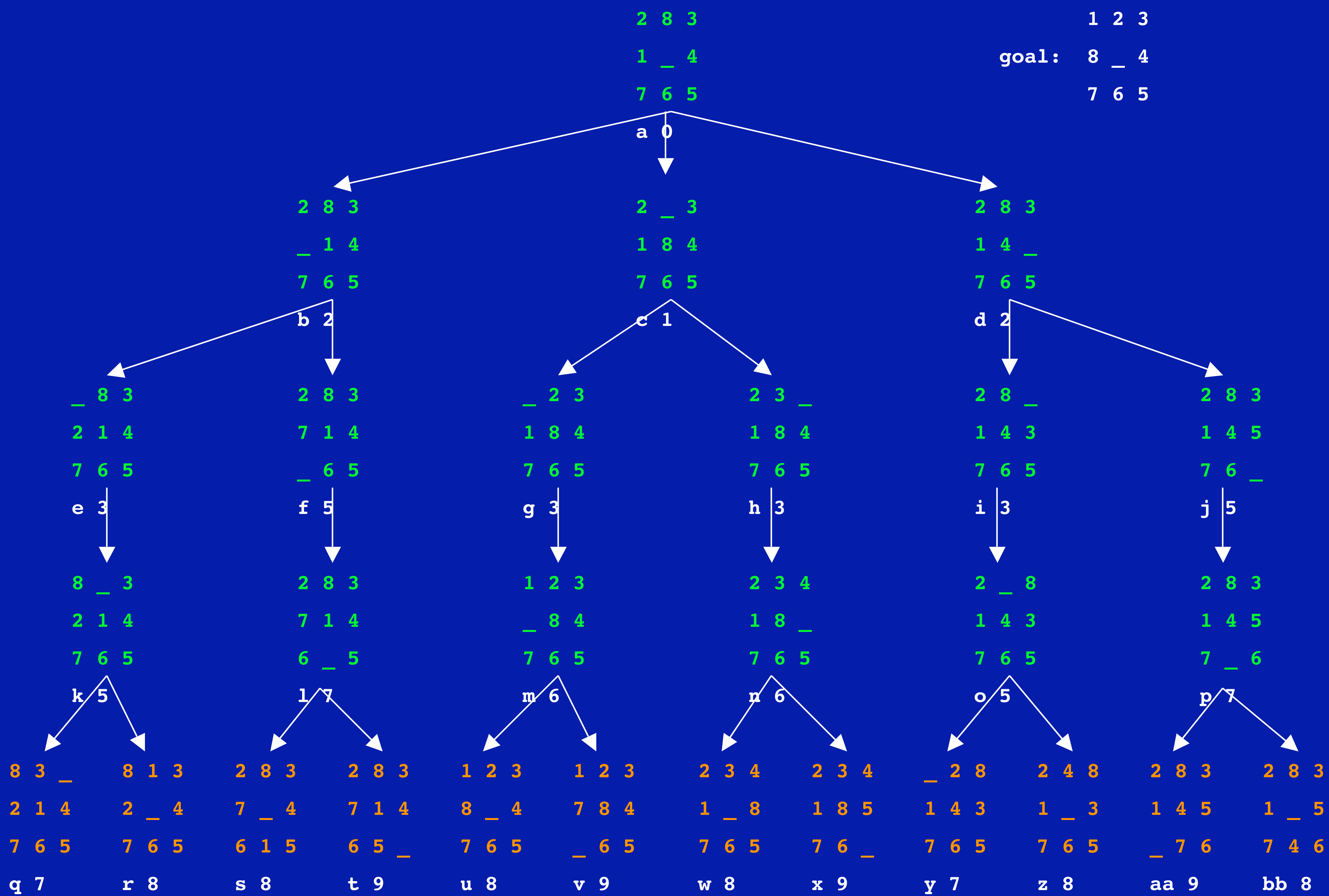
frontier: [p 7,q 7,y 7,r 8,s 8,u 8,w 8,z 8,t 9,v 9,x 9] *



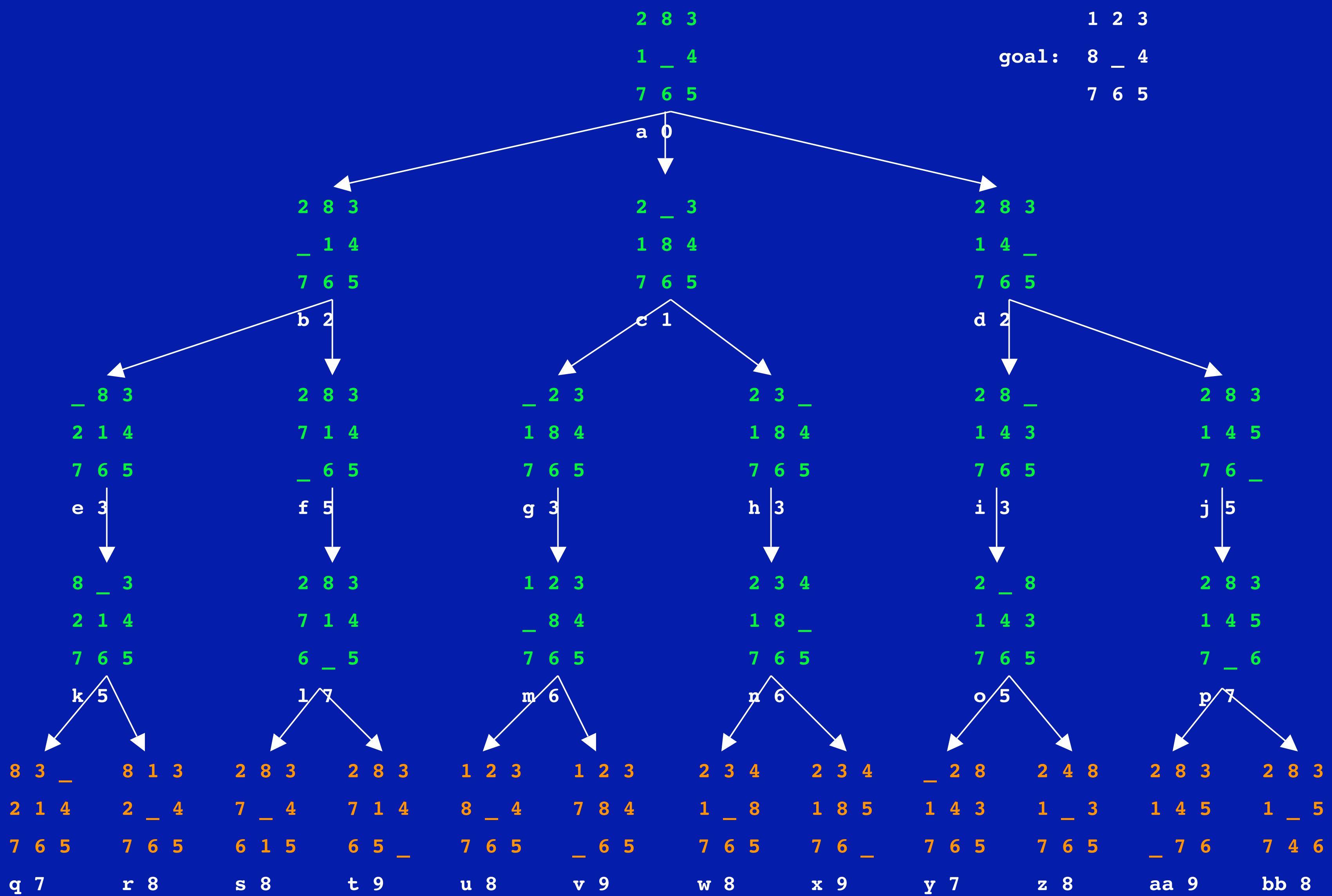
frontier: [q 7,y 7,r 8,s 8,u 8,w 8,z 8,t 9,v 9,x 9]



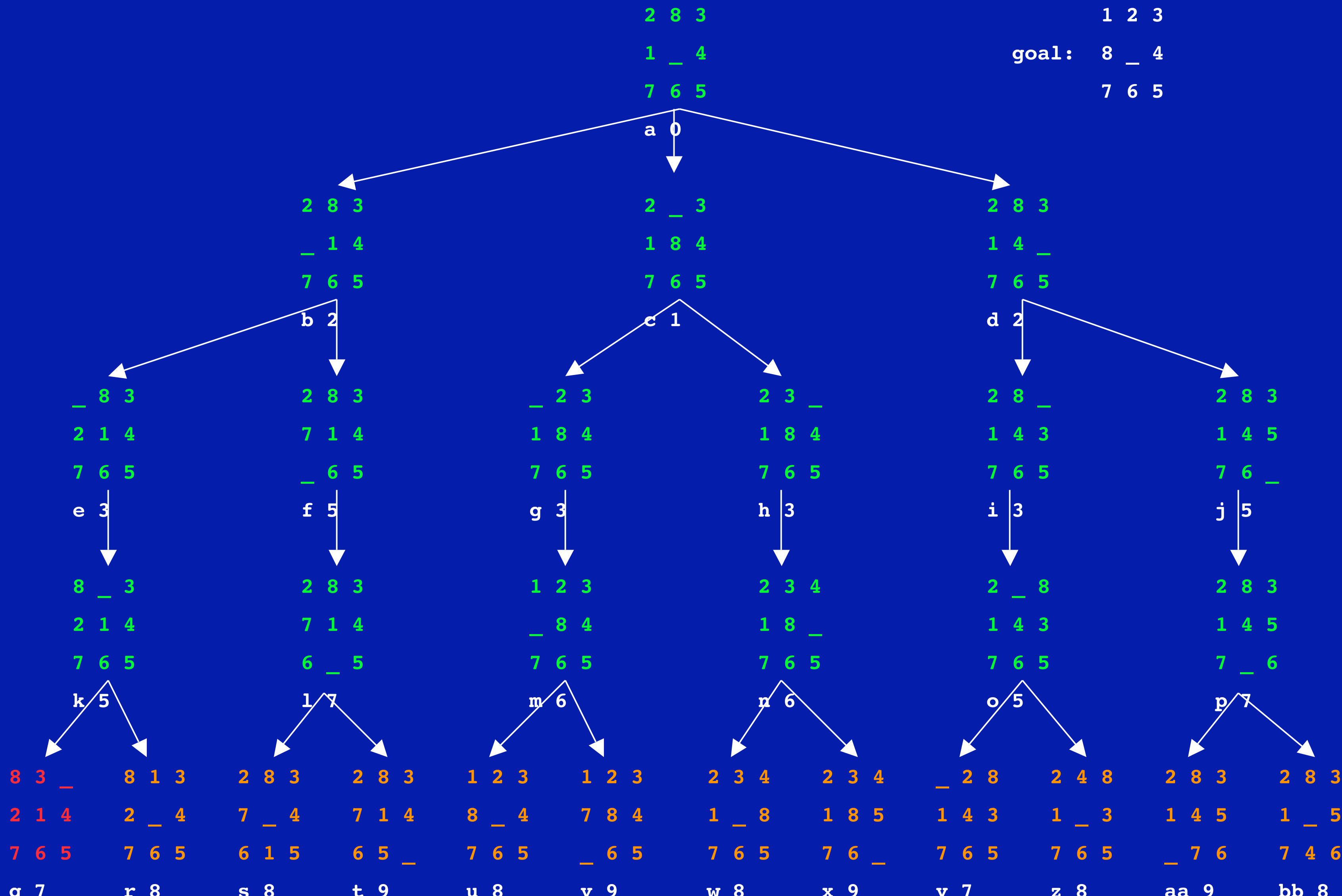
frontier: [aa 9,bb 8,q 7,y 7,r 8,s 8,u 8,w 8,z 8,t 9,v 9,x 9]



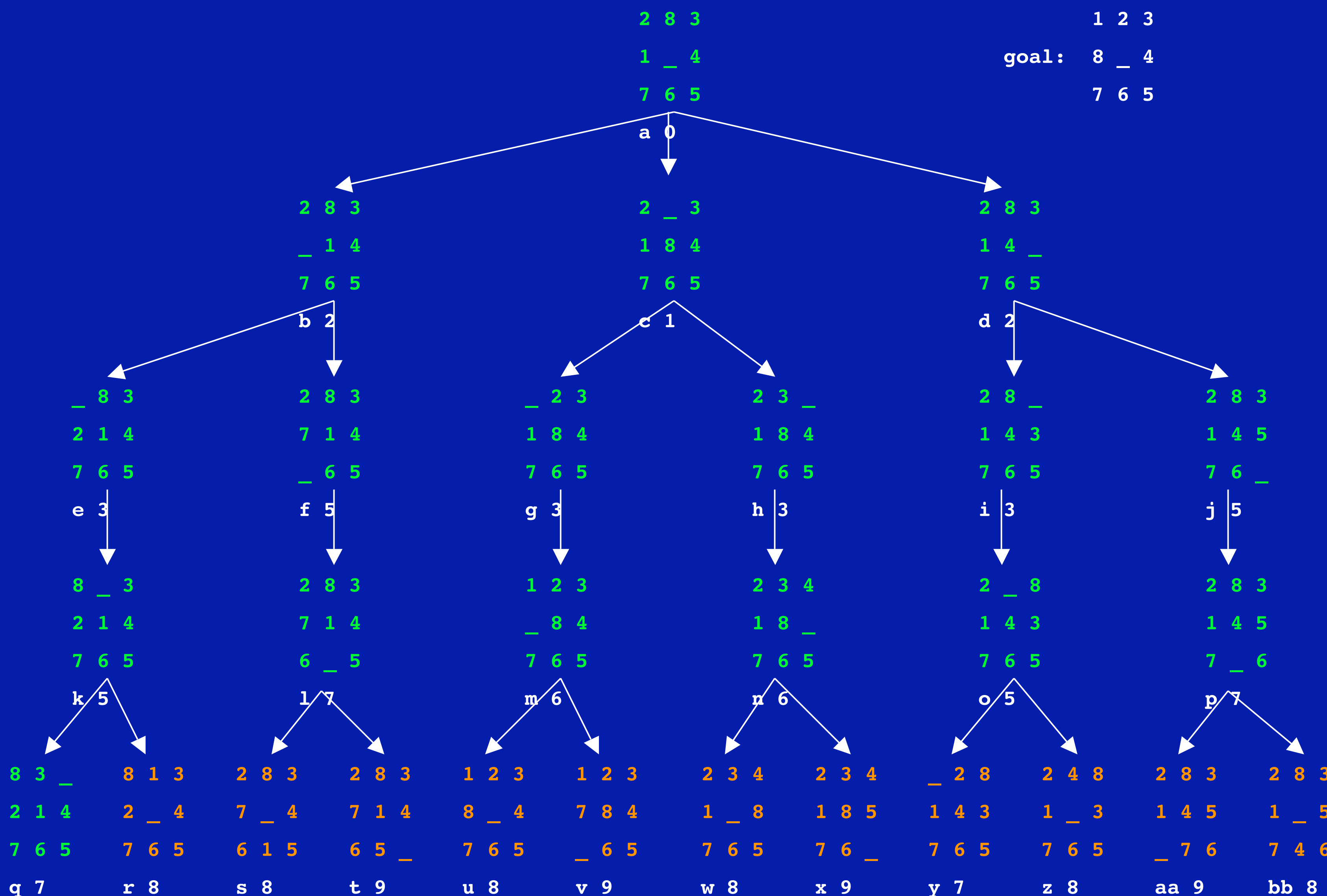
frontier: [q 7,y 7,r 8,s 8,u 8,w 8,z 8,bb 8,t 9,v 9,x 9,aa 9] *



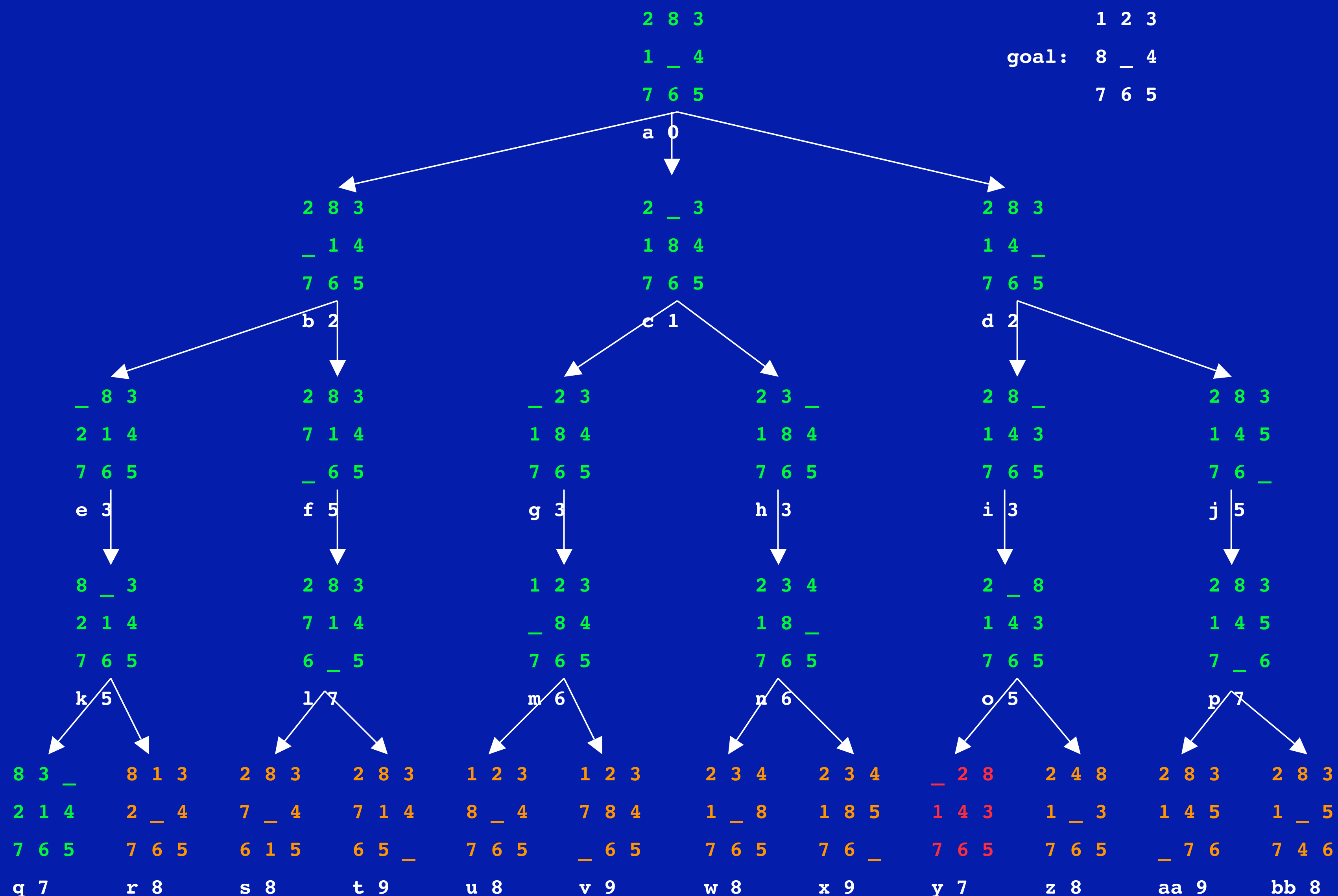
frontier: [y 7,r 8,s 8,u 8,w 8,z 8,bb 8,t 9,v 9,x 9,aa 9]



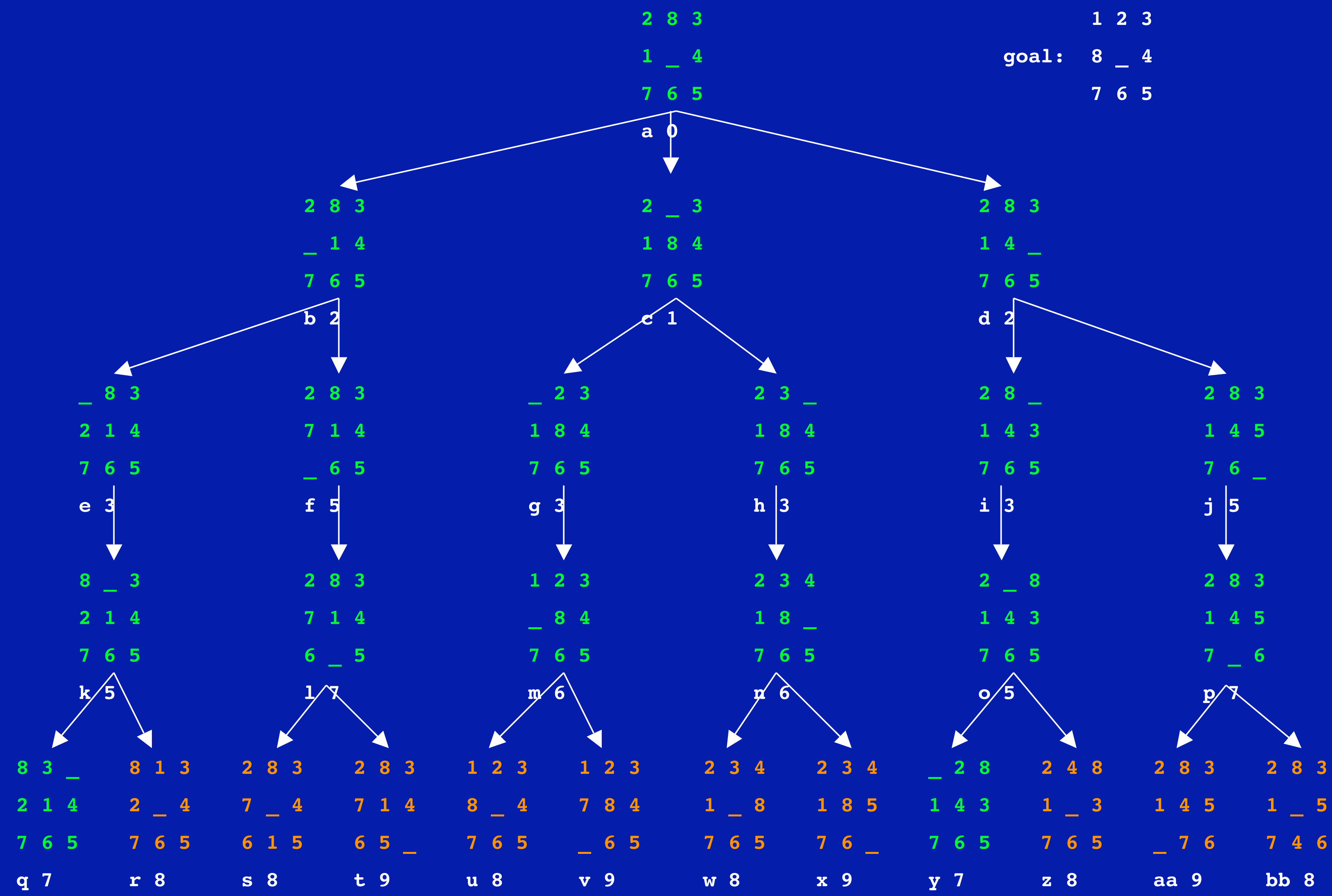
frontier: [y 7,r 8,s 8,u 8,w 8,z 8,bb 8,t 9,v 9,x 9,aa 9]



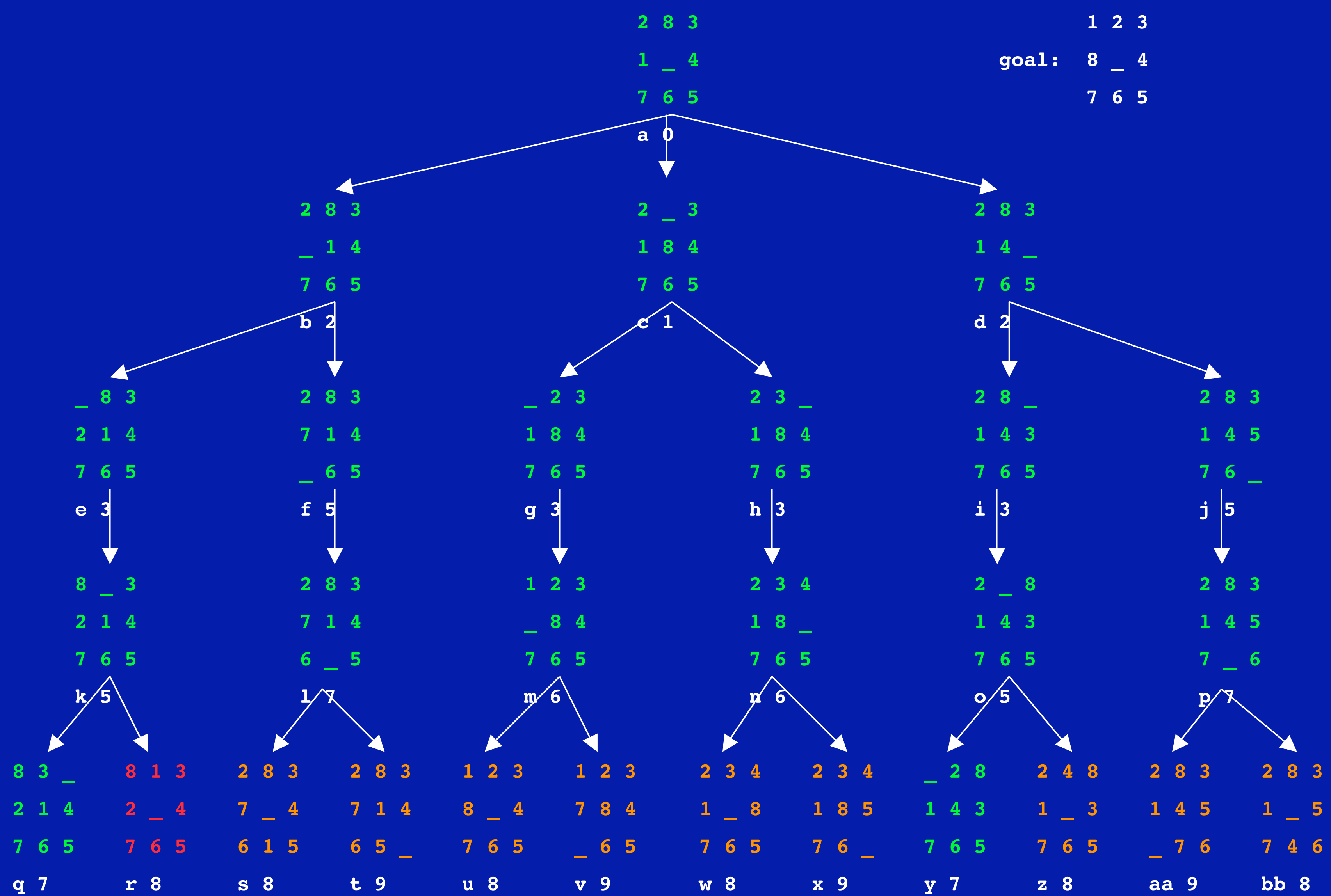
frontier: [r 8,s 8,u 8,w 8,z 8,bb 8,t 9,v 9,x 9,aa 9]



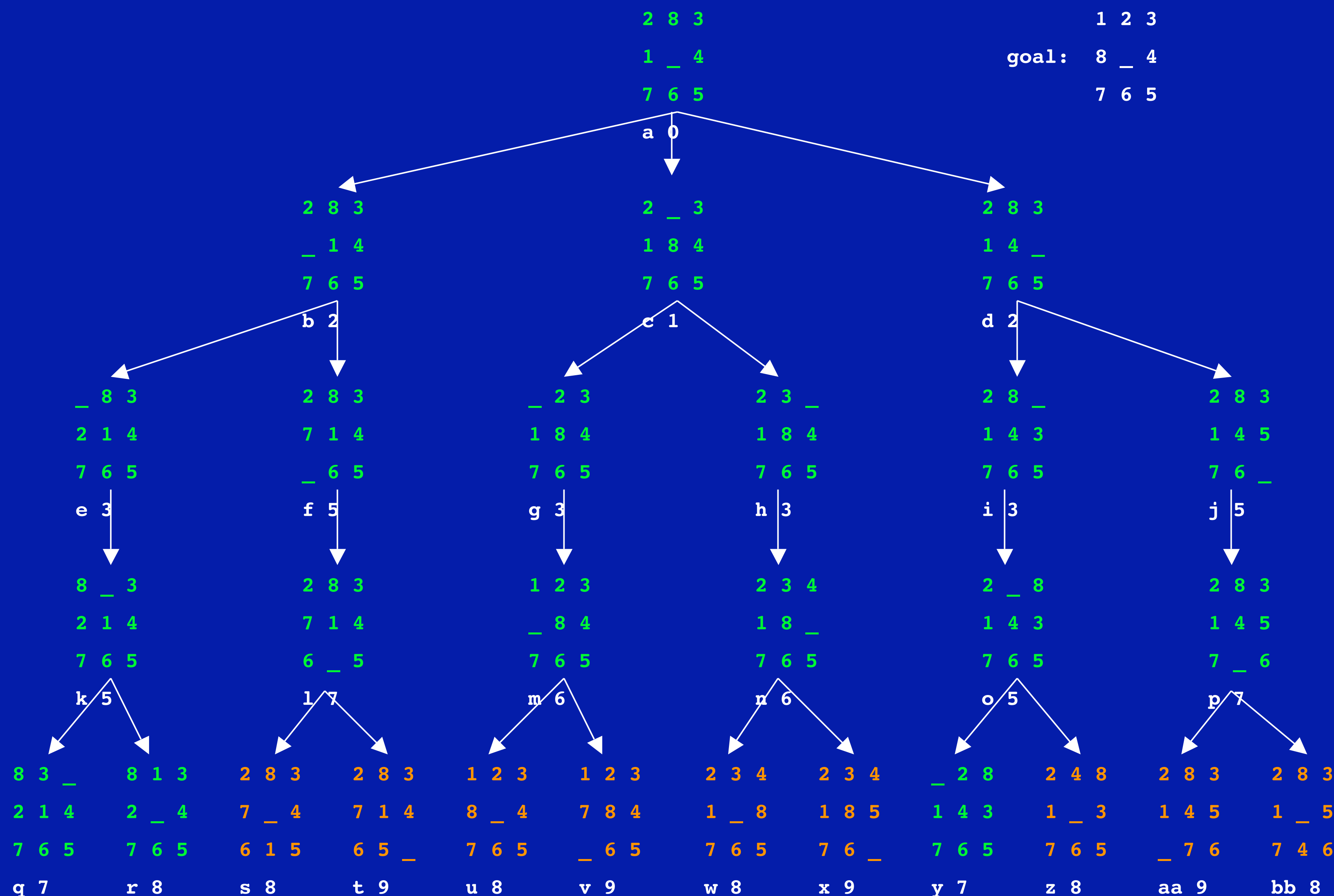
frontier: [r 8,s 8,u 8,w 8,z 8,bb 8,t 9,v 9,x 9,aa 9]



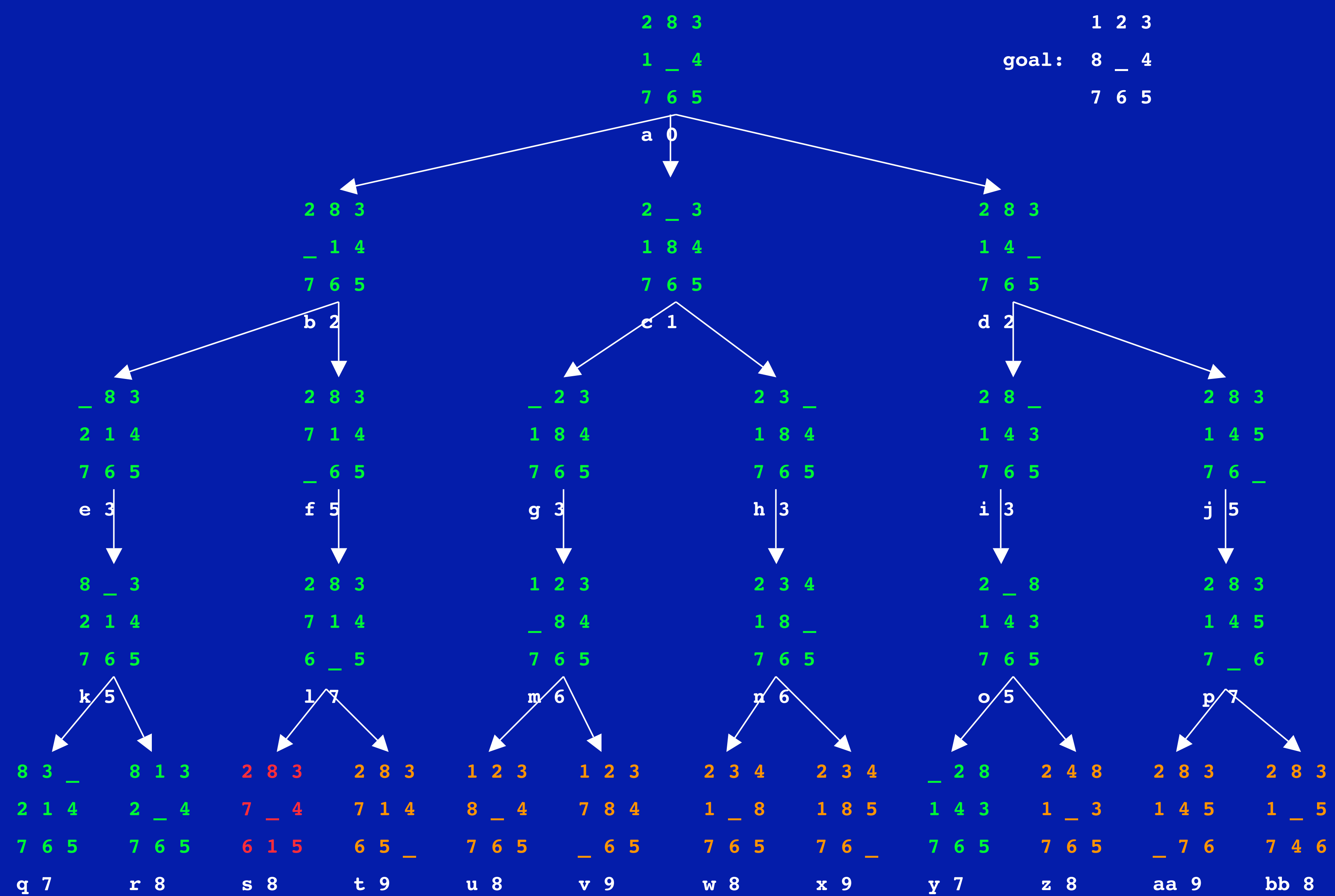
frontier: [s 8,u 8,w 8,z 8,bb 8,t 9,v 9,x 9,aa 9]



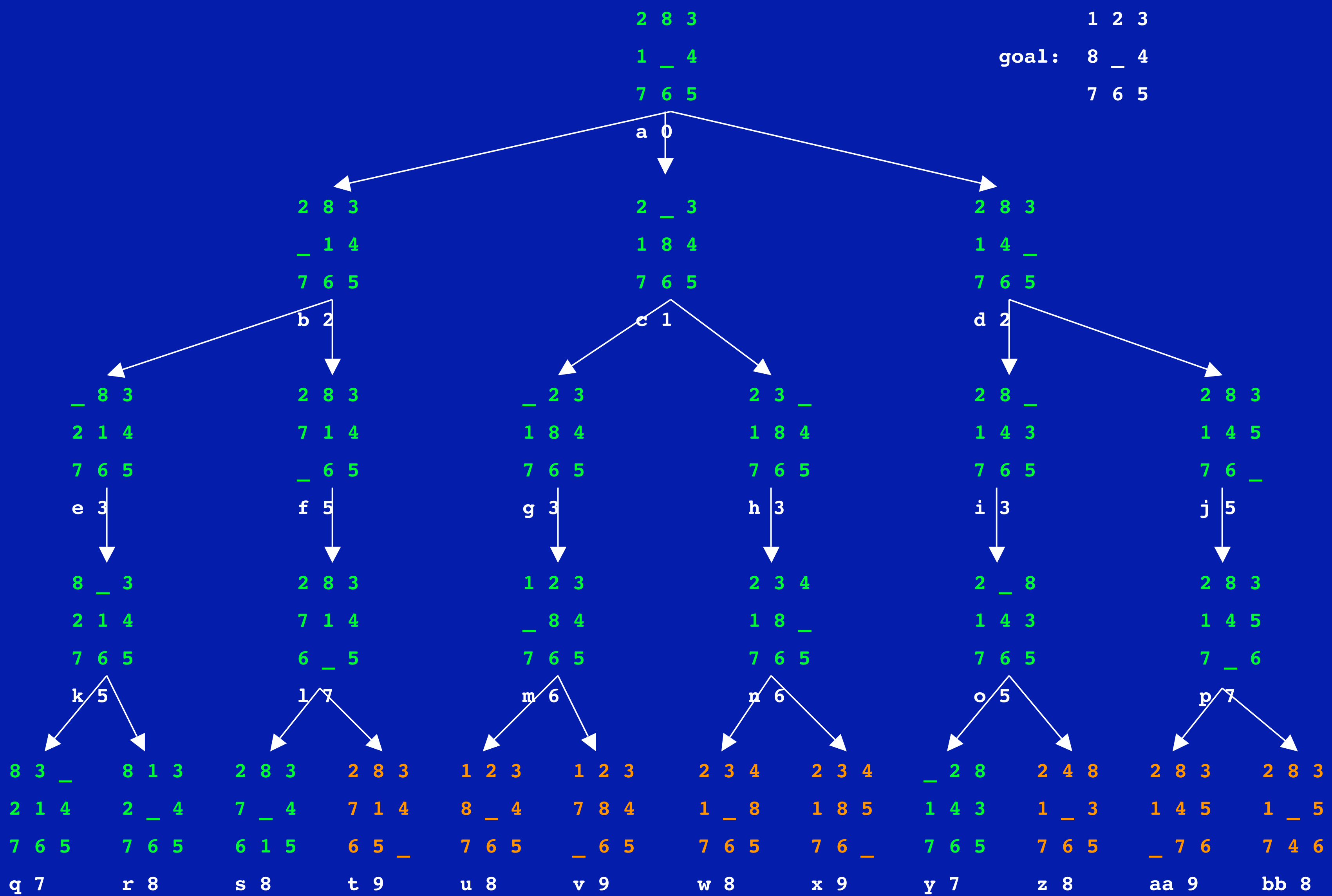
frontier: [s 8,u 8,w 8,z 8,bb 8,t 9,v 9,x 9,aa 9]



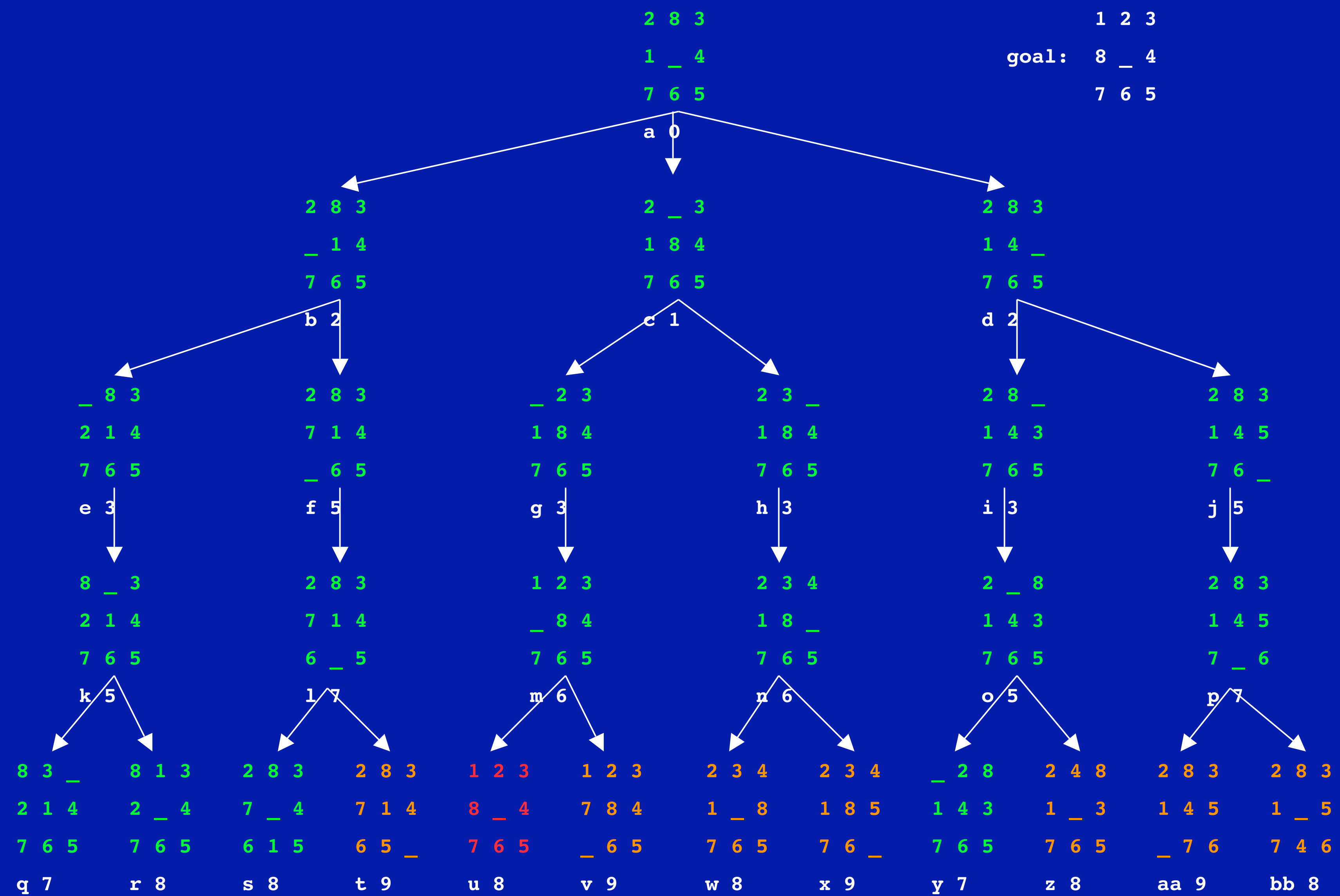
frontier: [u 8,w 8,z 8,bb 8,t 9,v 9,x 9,aa 9]



frontier: [u 8,w 8,z 8,bb 8,t 9,v 9,x 9,aa 9]



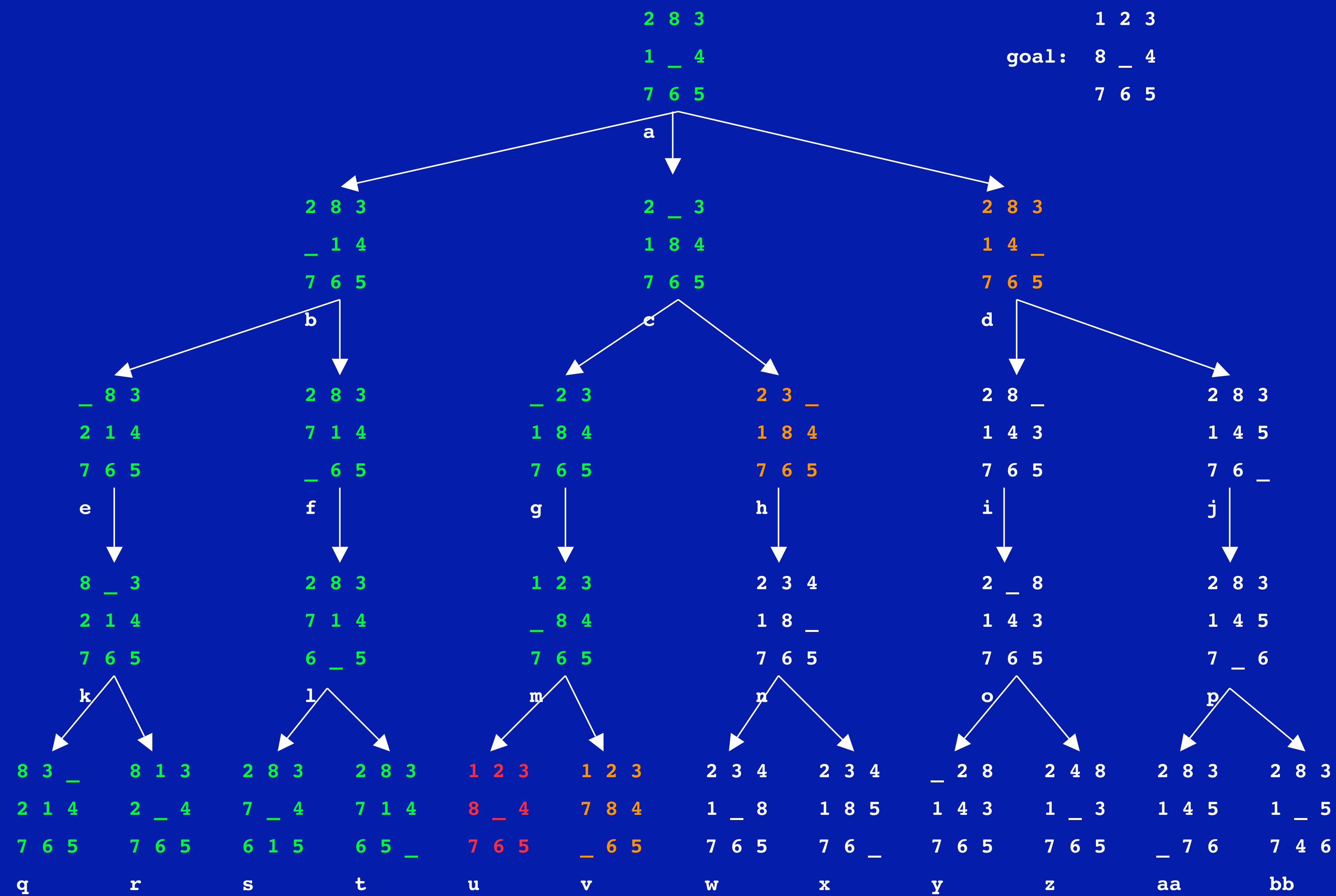
frontier: [w 8,z 8,bb 8,t 9,v 9,x 9,aa 9]



Review how each one finishes

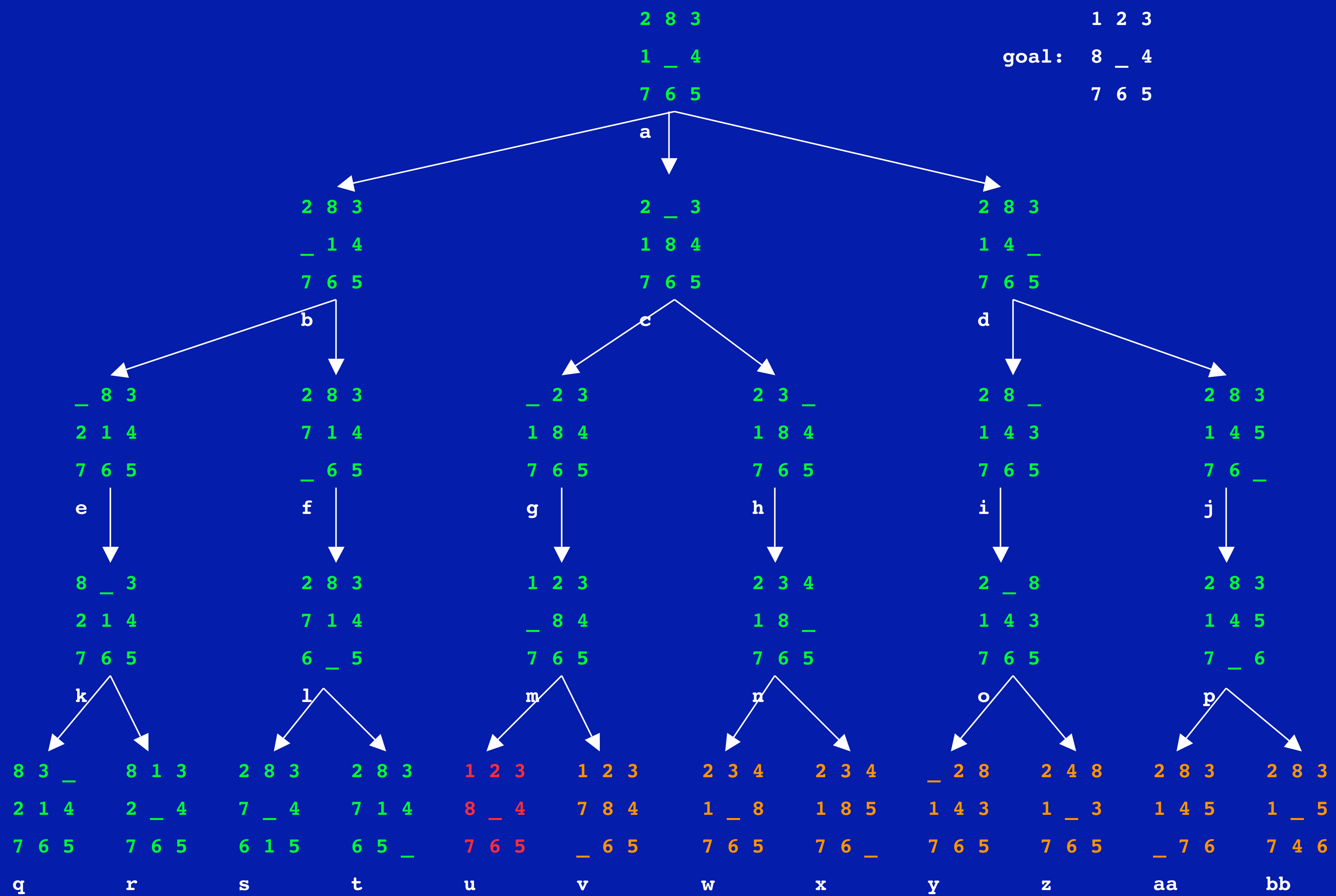
depth-first

frontier: [v,h,d]



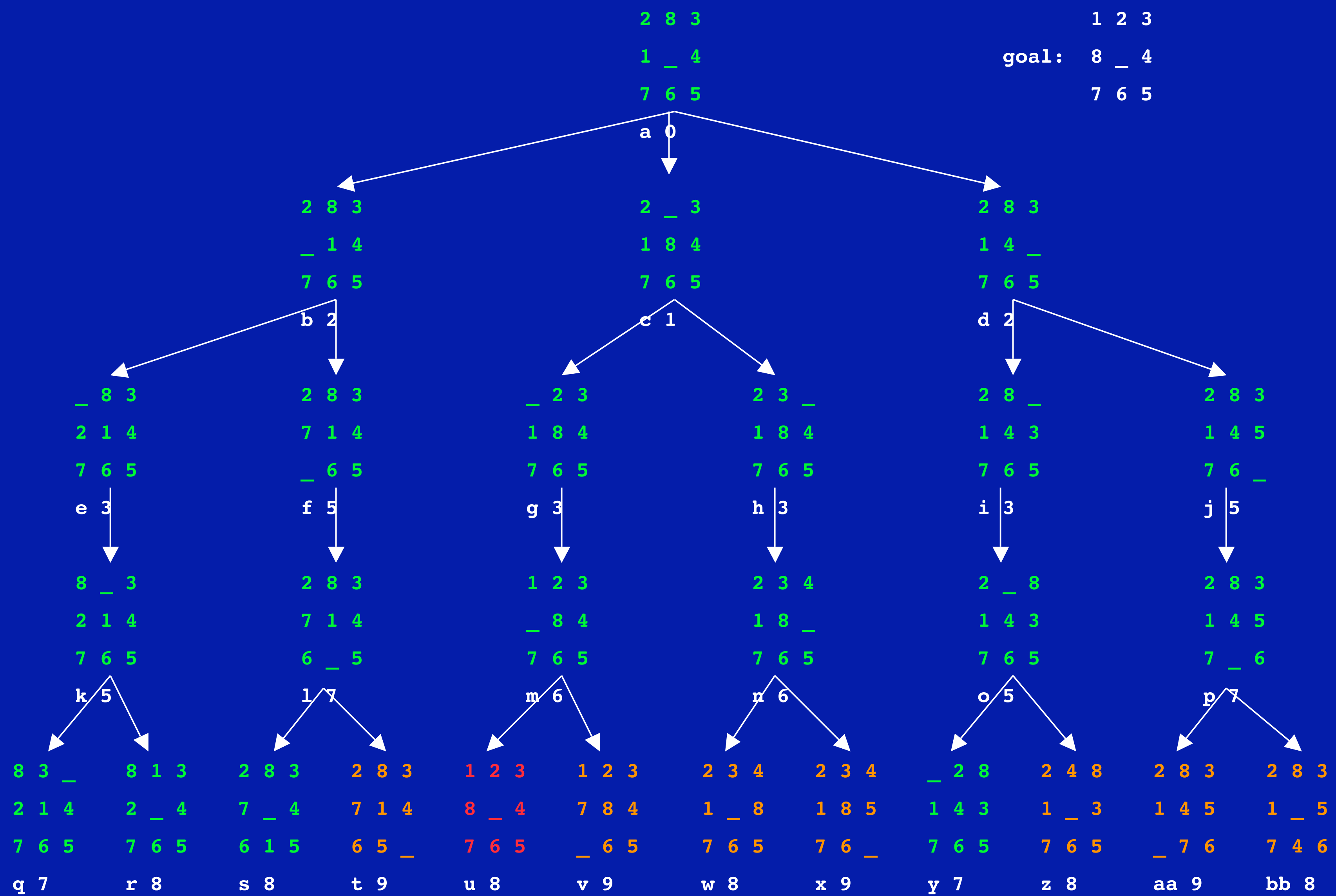
breadth-first

frontier: [v,w,x,y,z,aa,bb]



lowest-cost-first

frontier: [w 8,z 8,bb 8,t 9,v 9,x 9,aa 9]



Comparing blind search strategies

All will find a solution in a finite space if a solution exists

Depth-first can get trapped in infinite recursion in an infinite space (except in CILOG)

Breadth-first and lowest-cost will find a solution even in an infinite space, even if one exists

Breadth-first will find path to goal with fewest arcs

Lowest-cost-first will find lowest cost path (of course) when arcs have different costs

Breadth-first is just depth-first but adding to frontier differently

Lowest-cost-first is just breadth-first with more info and sorting

Comparing blind search strategies

Breadth-first and lowest-cost-first seem wonderful, but they're gigantic space hogs in terms of how big frontier can be

There's a whole lot more of this good vs. bad stuff in your textbook (chapter 4.4)...read it

Ultimately, all three are expensive approaches to search