CPSC 322
Introduction to Artificial Intelligence

October 8, 2004

Things to look for

Your final exam will be from
noon to 3:00pm on Friday, December 10

location to be determined

Search

A physical symbol system exercises its intelligence in problem solving by search -- that is, by generating and progressively modifying symbol structures until it produces a solution structure.

Allen Newell and Herbert A. Simon, “Computer Science as Empirical Inquiry: Symbols and Search”

Graph search - informal

Most problem-solving tasks can be mapped onto the problem of finding a path in a graph

- nodes represent partial solutions
- arcs represent transformations from one partial solution to another
- try to find path along arcs from a start node to a goal node

Graph search - formal

A directed graph consists of a set of nodes and a set of ordered pairs called arcs (links?).

A node n_2 is a neighbor of n_1 if there is an arc from n_1 to n_2.

A path is an ordered sequence of nodes such that there is an arc between consecutive nodes in the path

A cycle is a nonempty path with start node and end node the same.

A directed graph without cycles is called a directed acyclic graph.

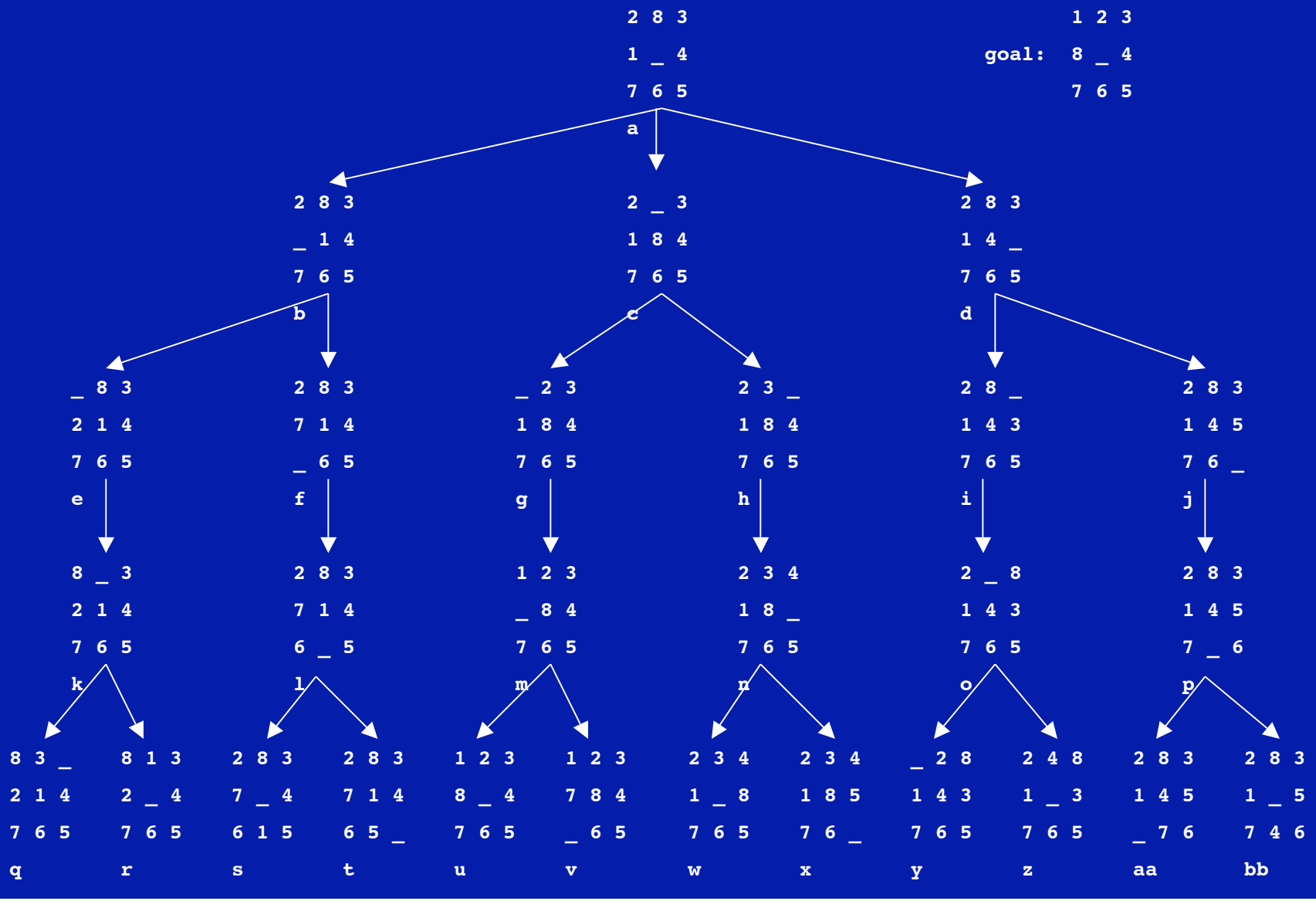
How do I do this graph search stuff?

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

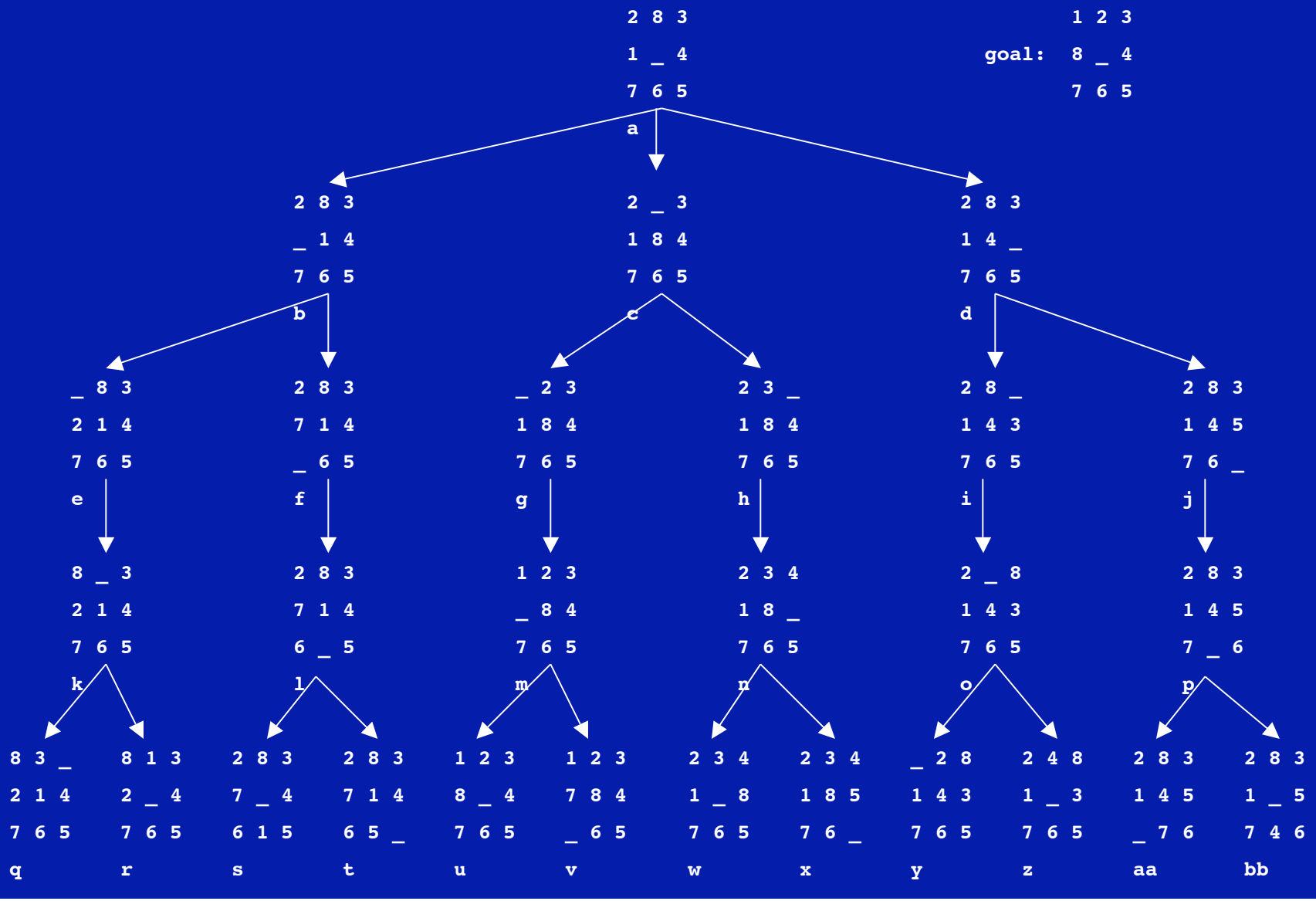
make a “list” of the start nodes - let’s call it the “frontier”

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

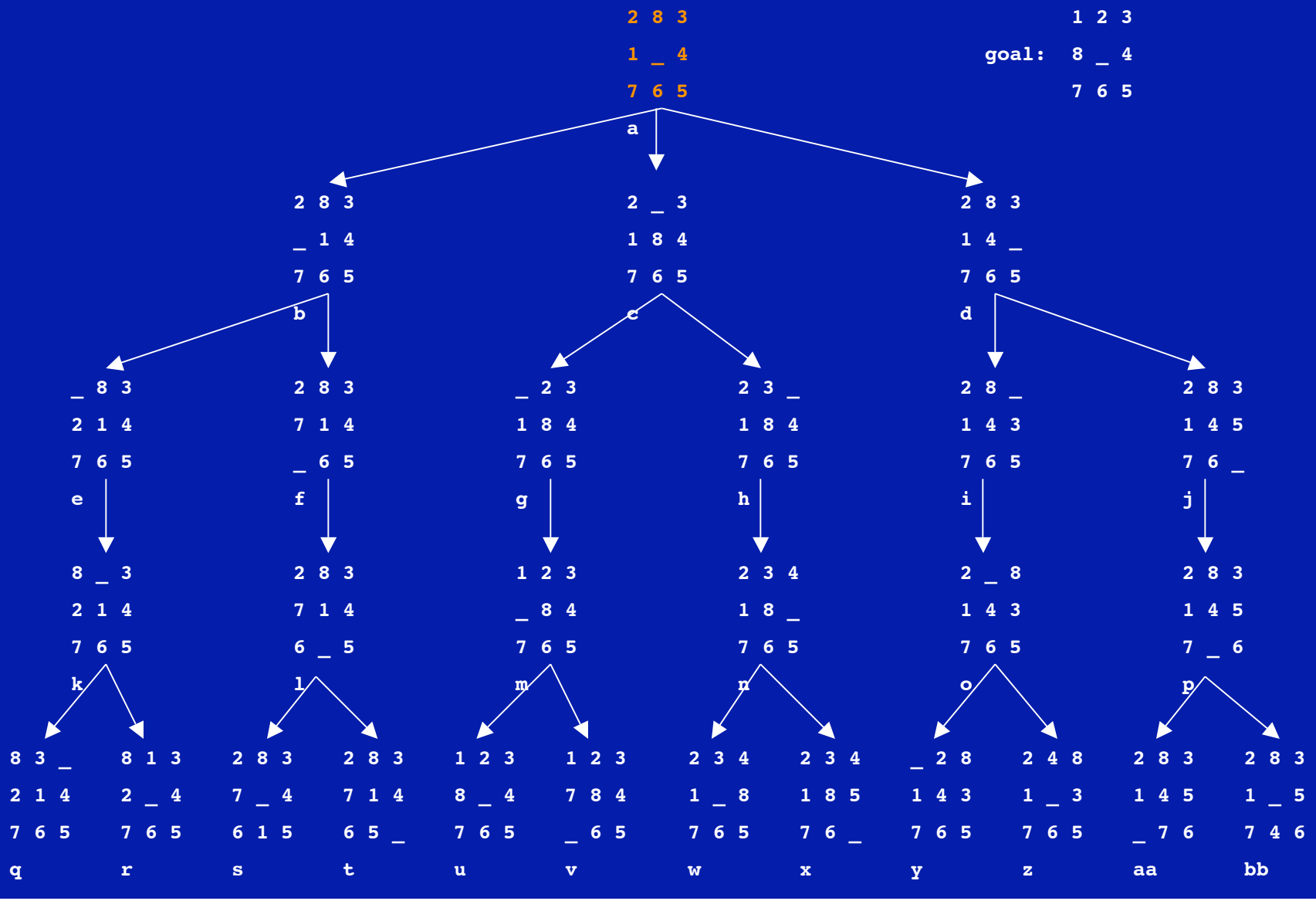
8-Tile Puzzle



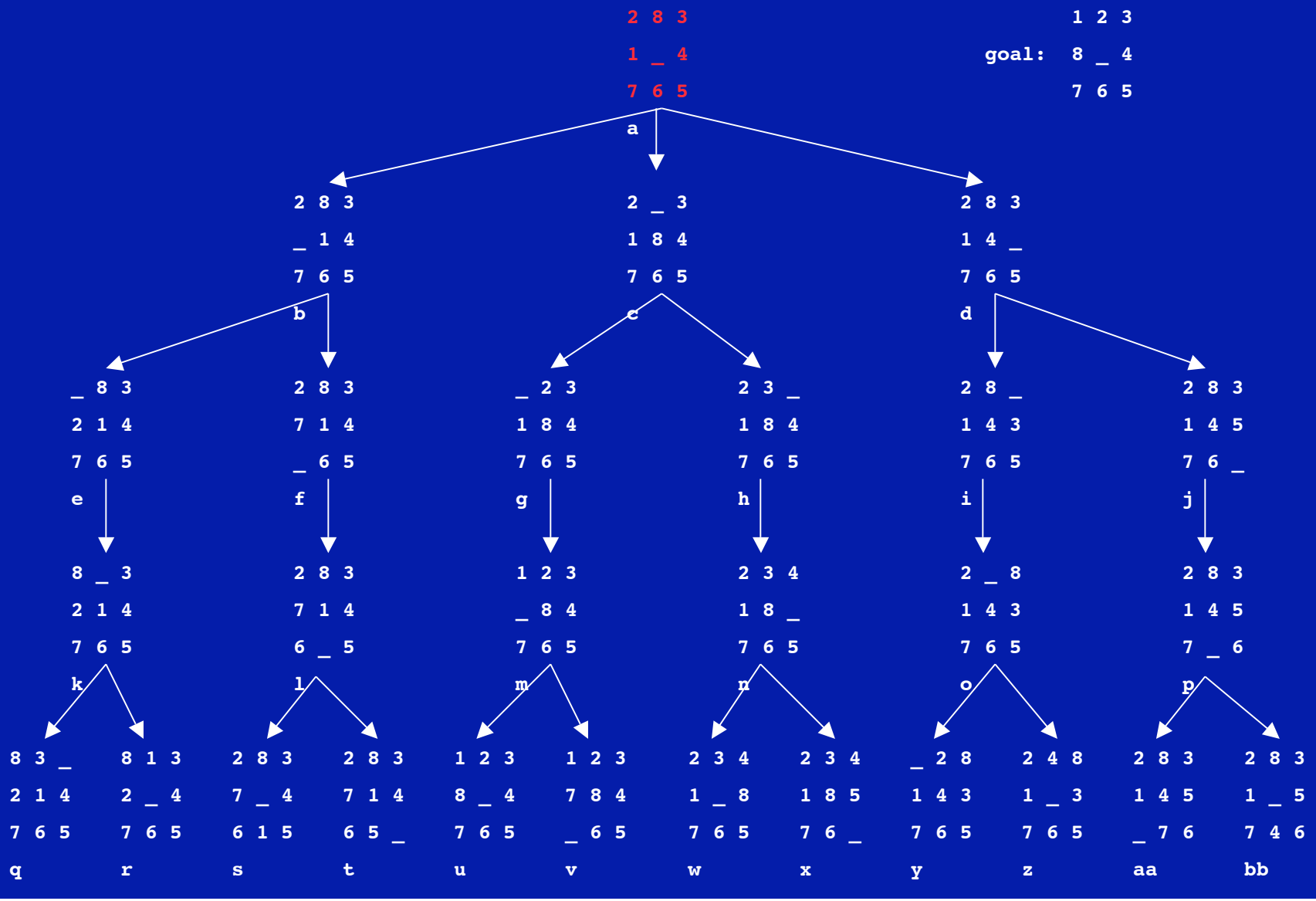
frontier: [ ]


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

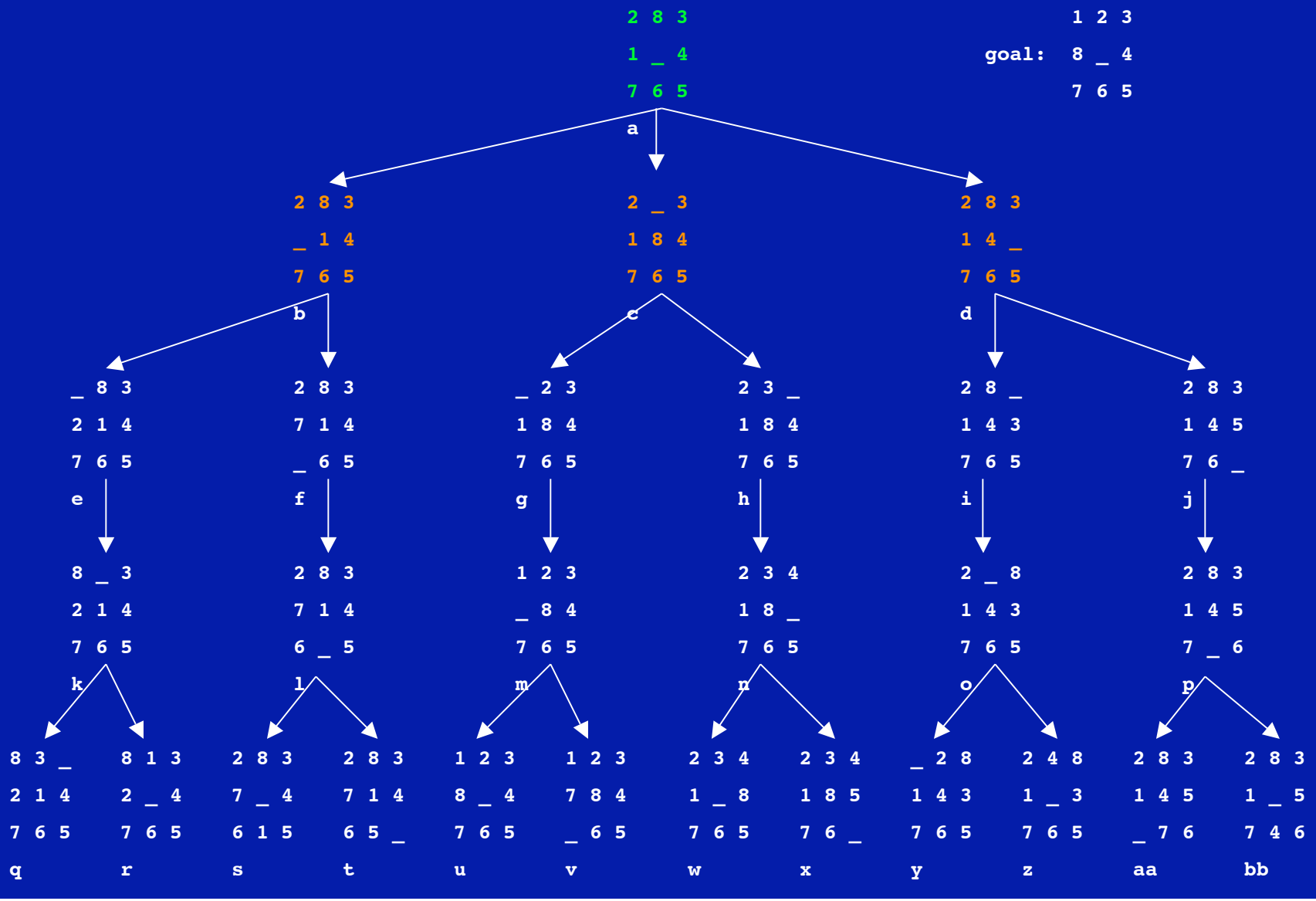
frontier: [a]


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

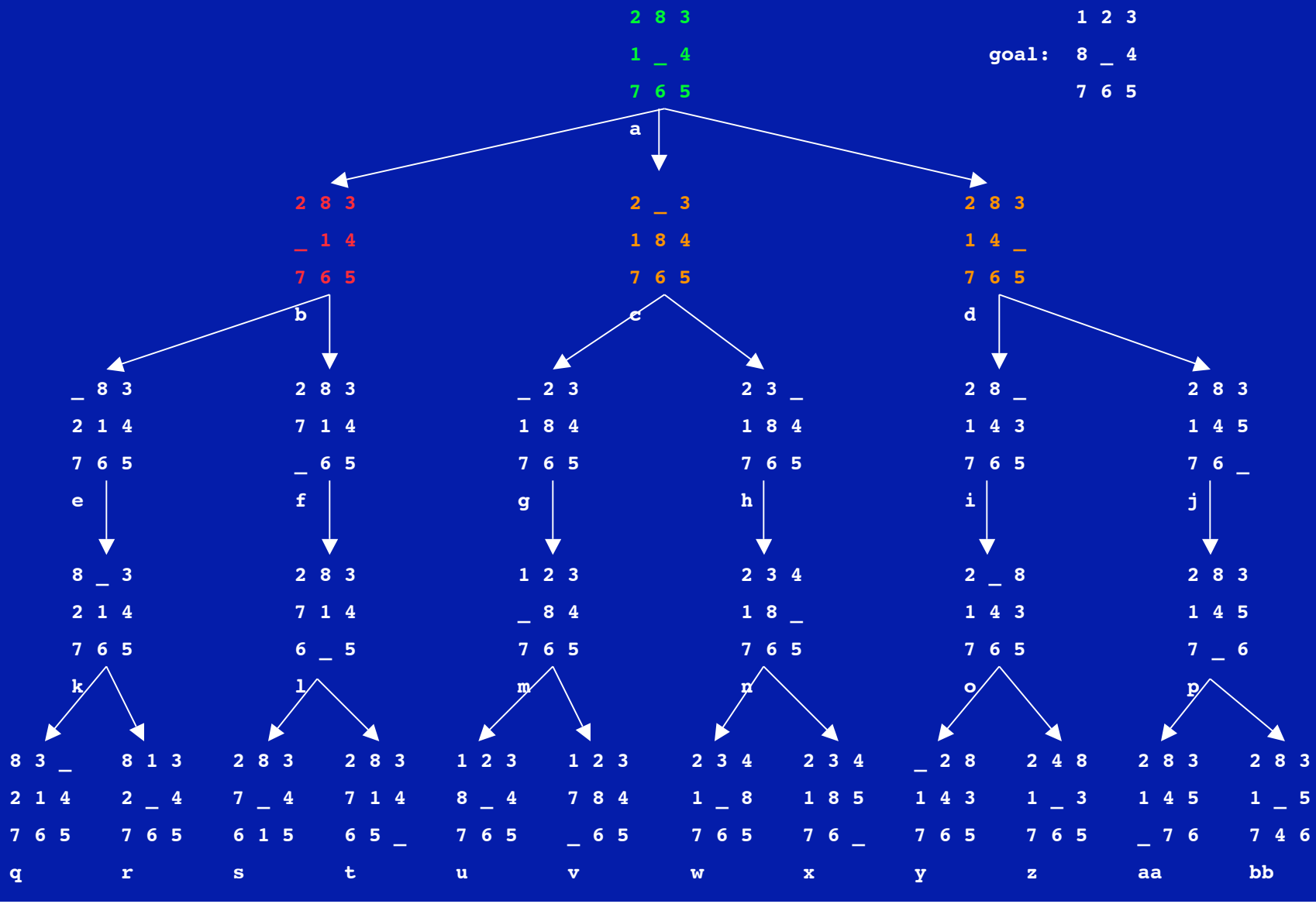
frontier: [ ]


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

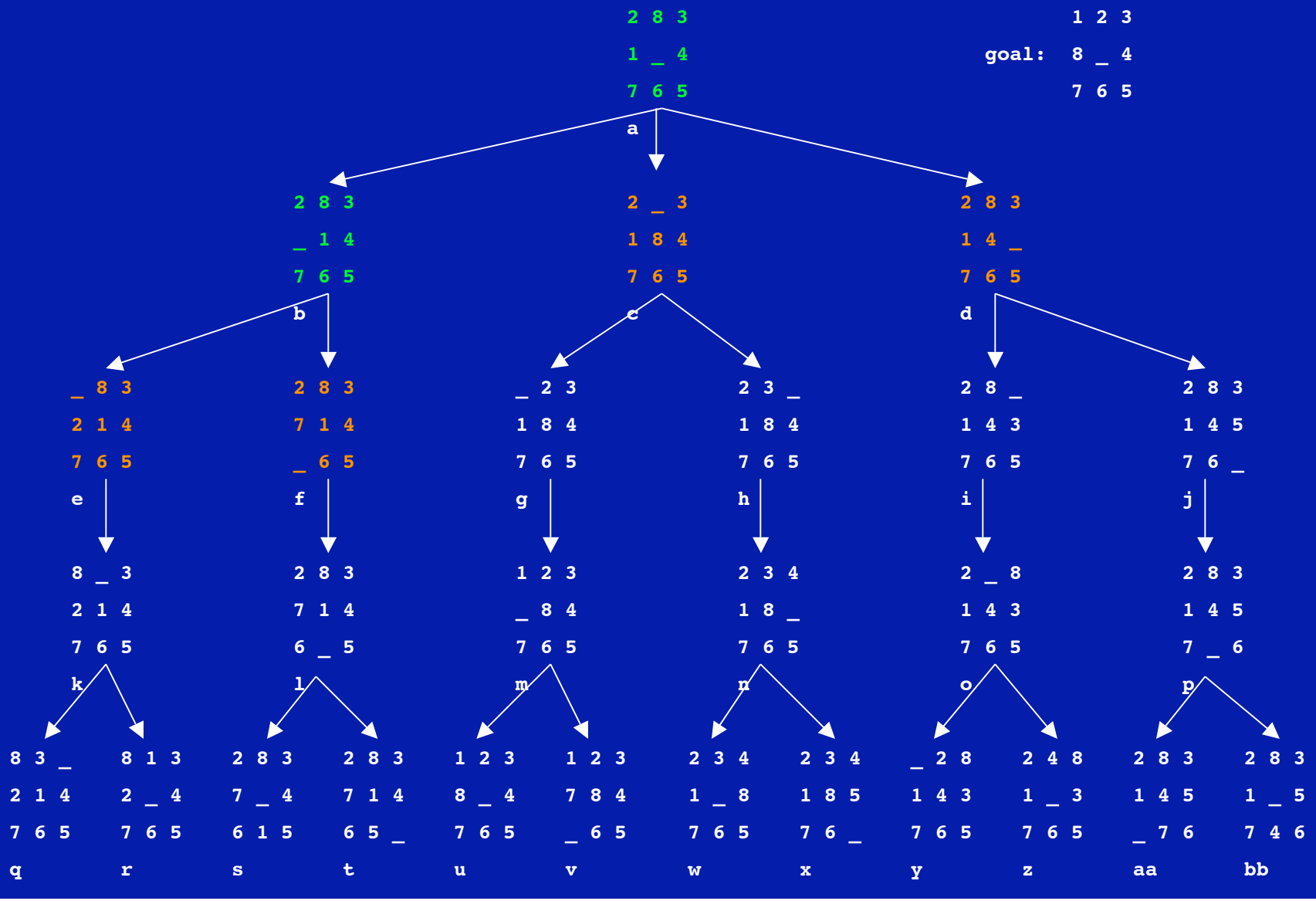
frontier: [b,c,d]



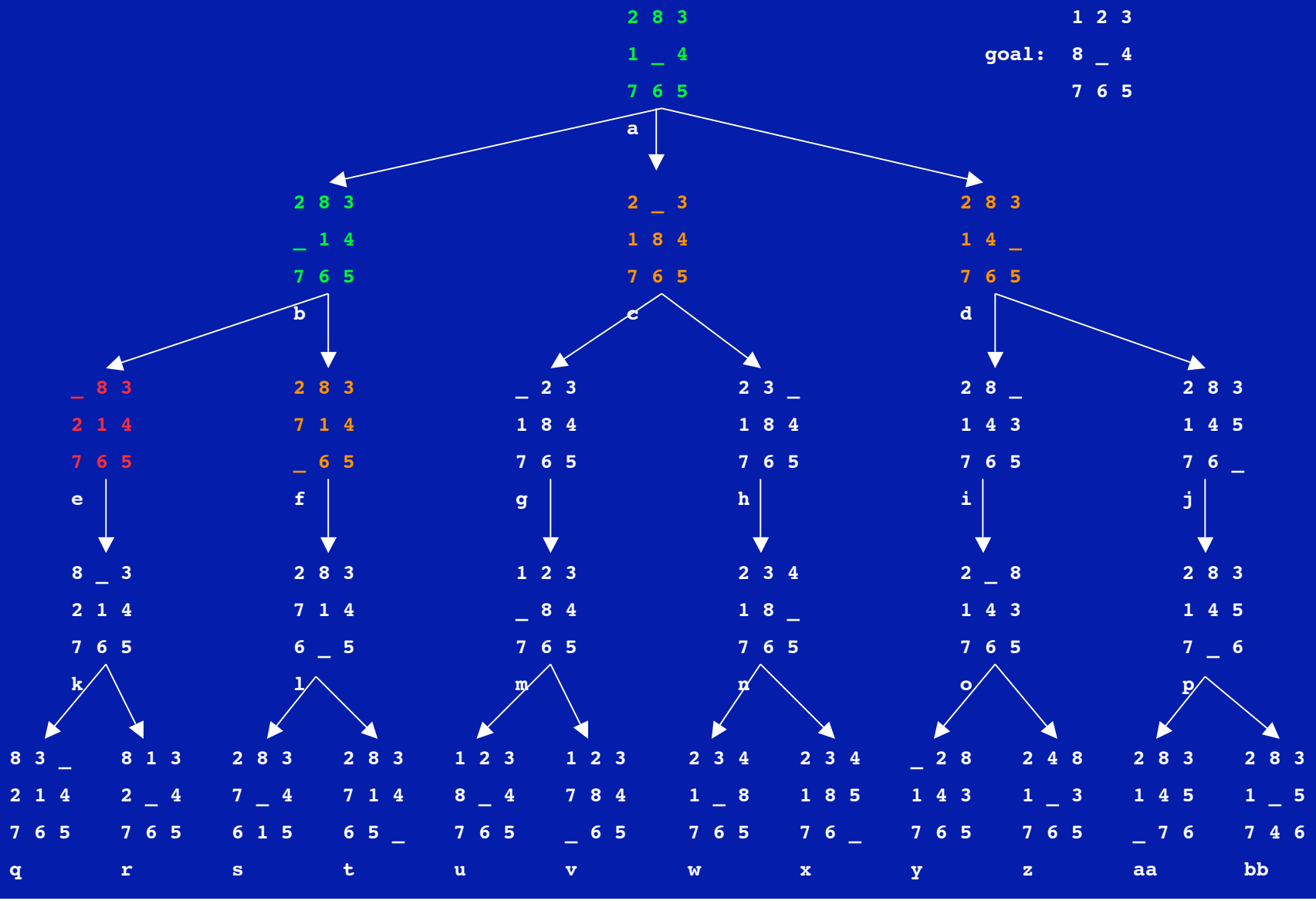
frontier: [c,d]


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

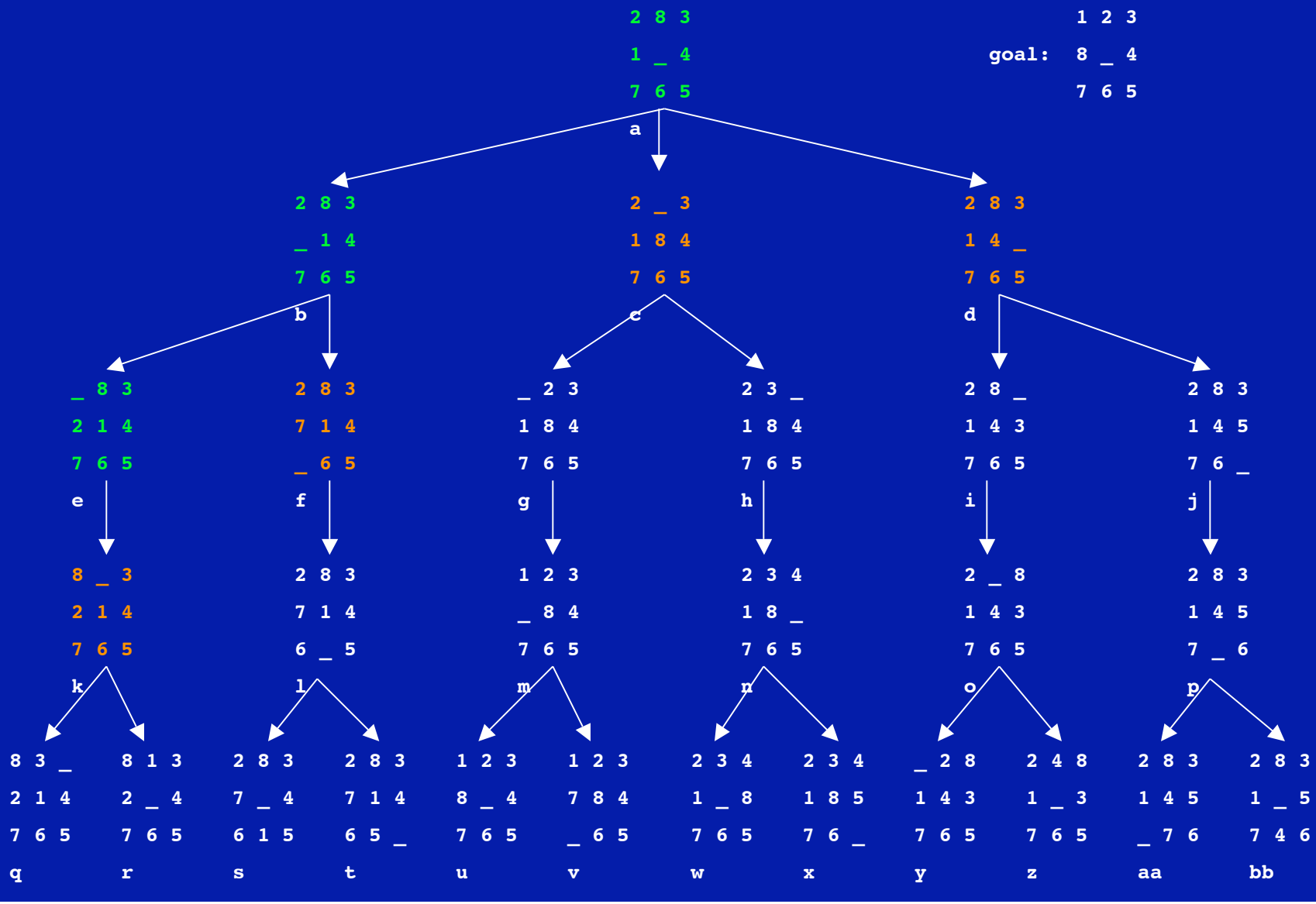
frontier: [e,f,c,d]



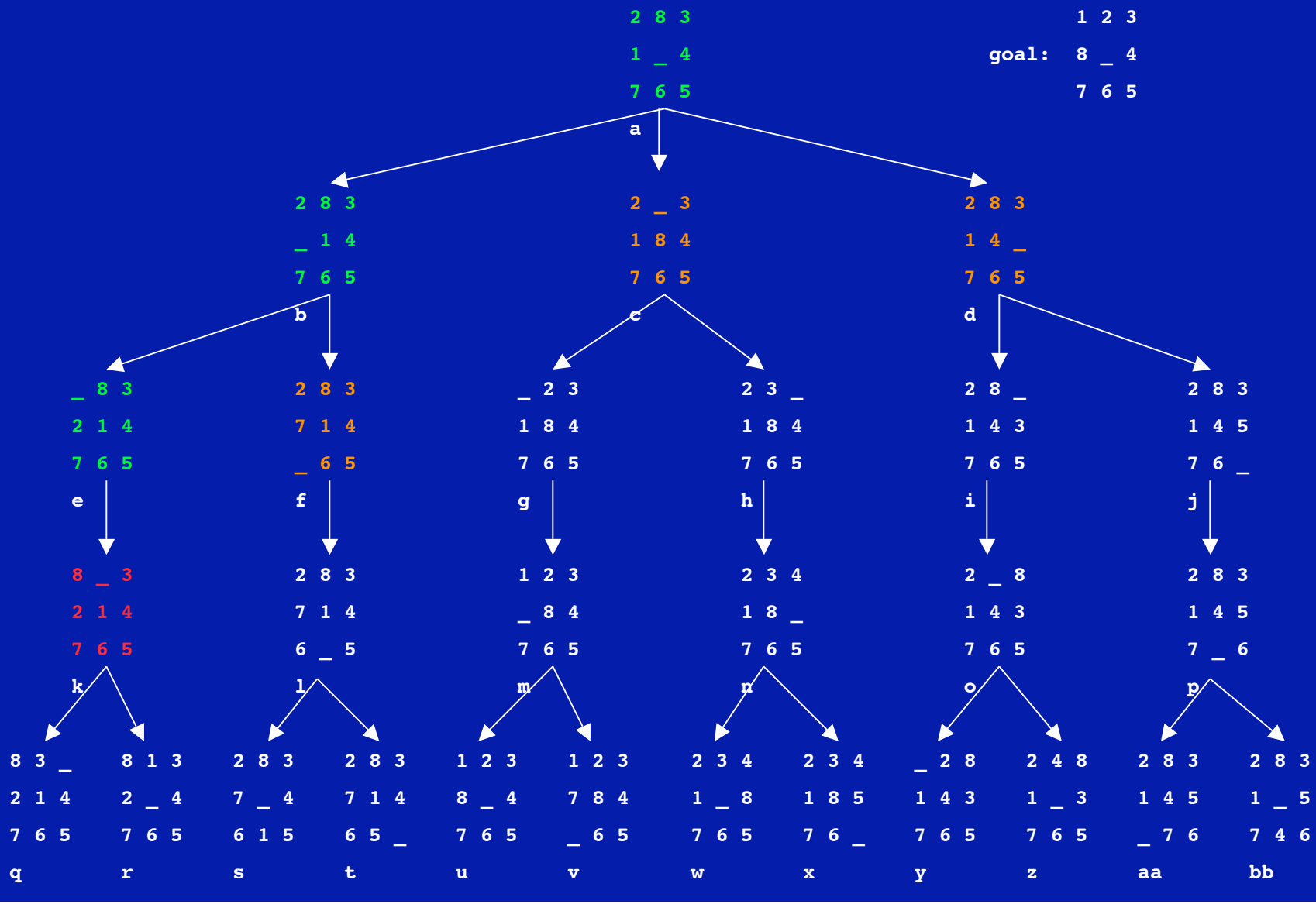
frontier: [f,c,d]



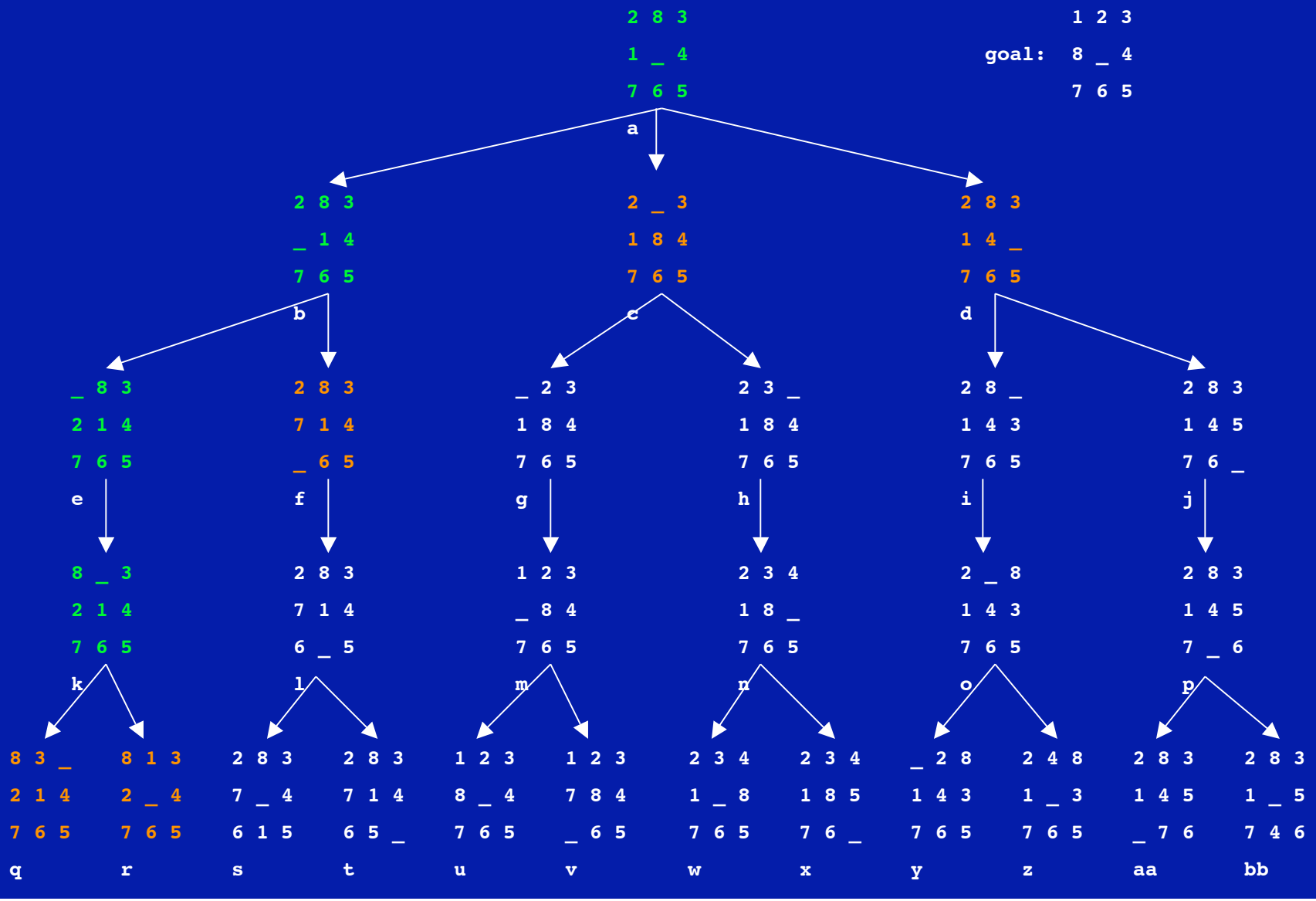
frontier: [k,f,c,d]



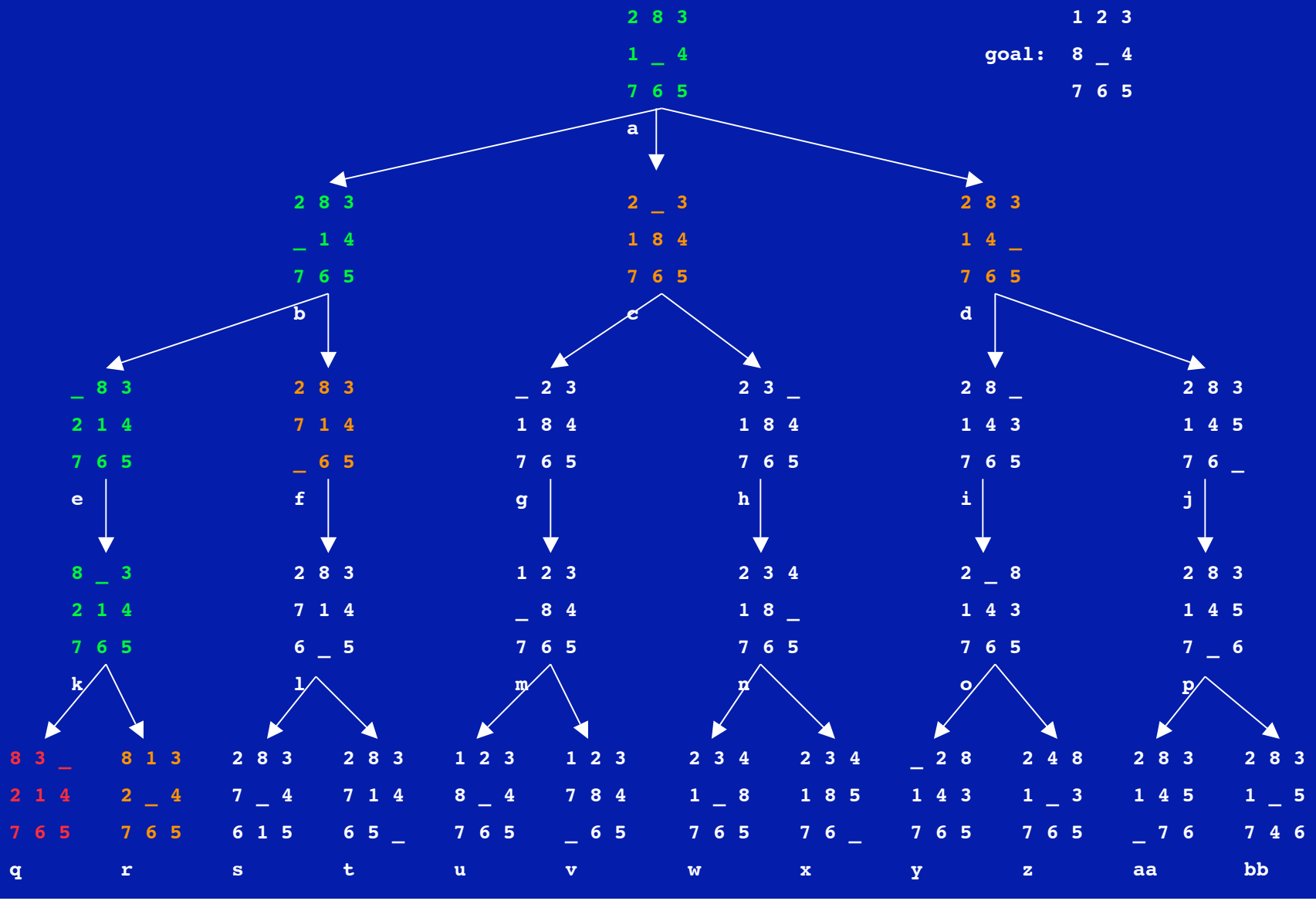
frontier: [f,c,d]



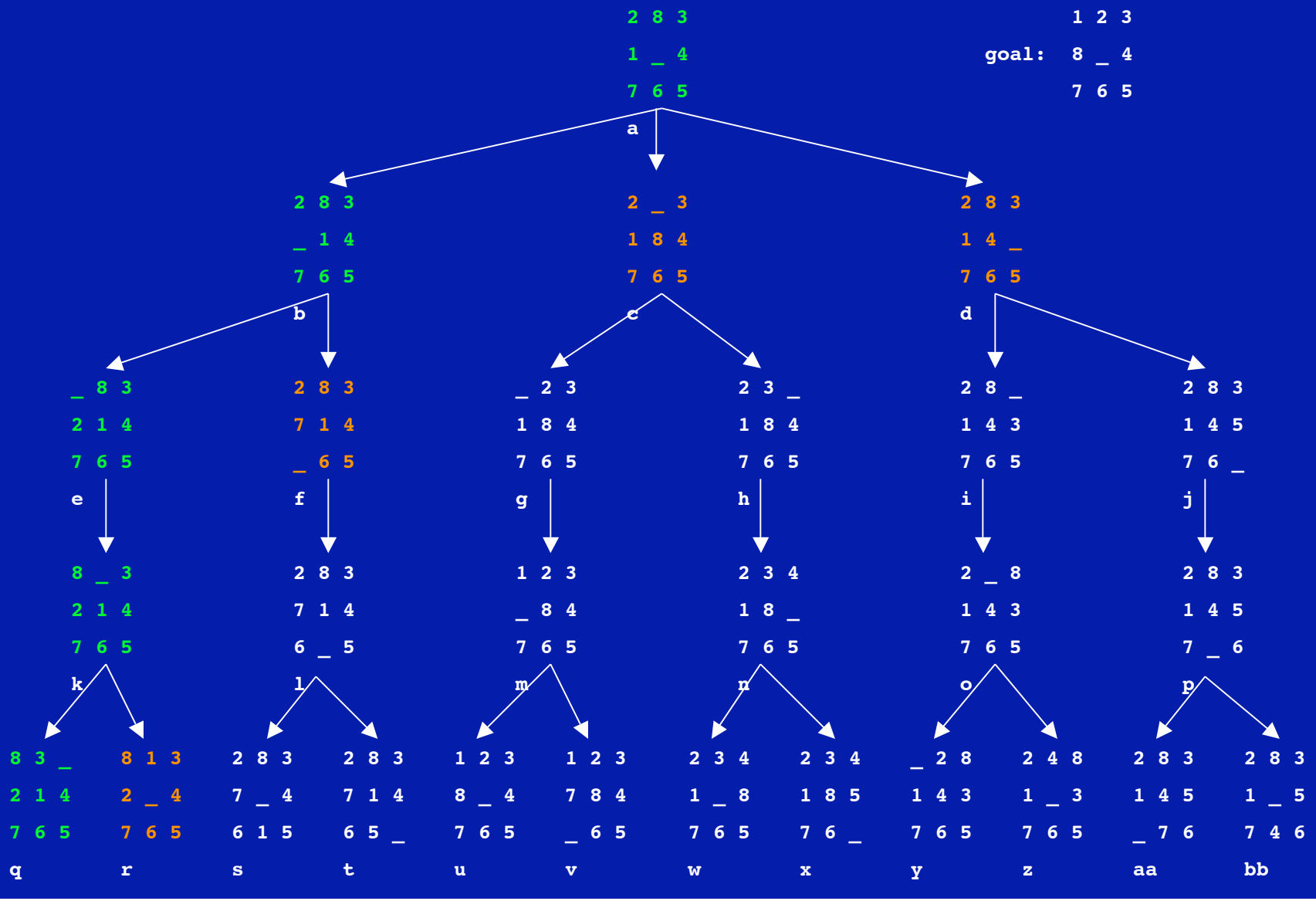
frontier: [q,r,f,c,d]



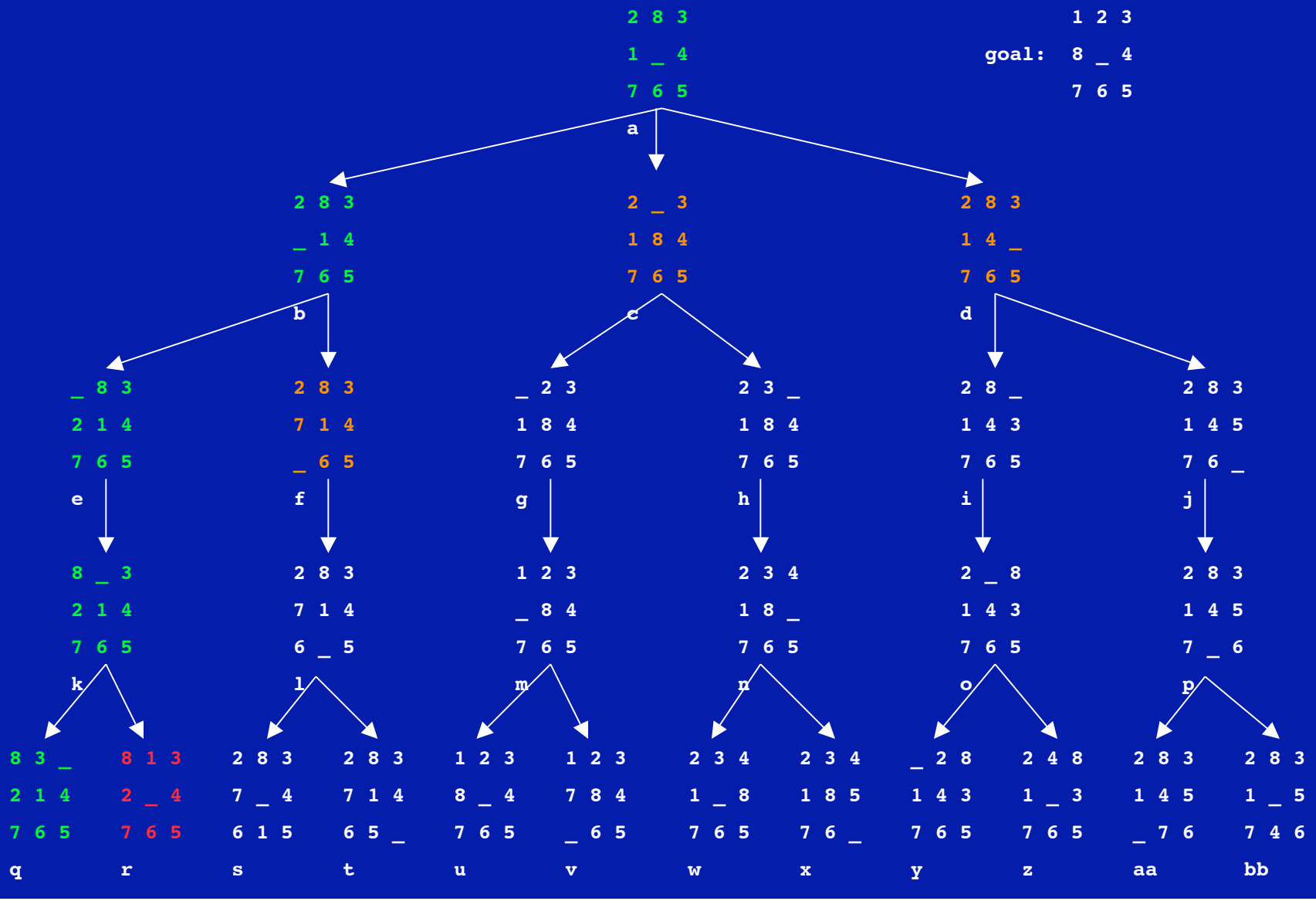
frontier: [r,f,c,d]



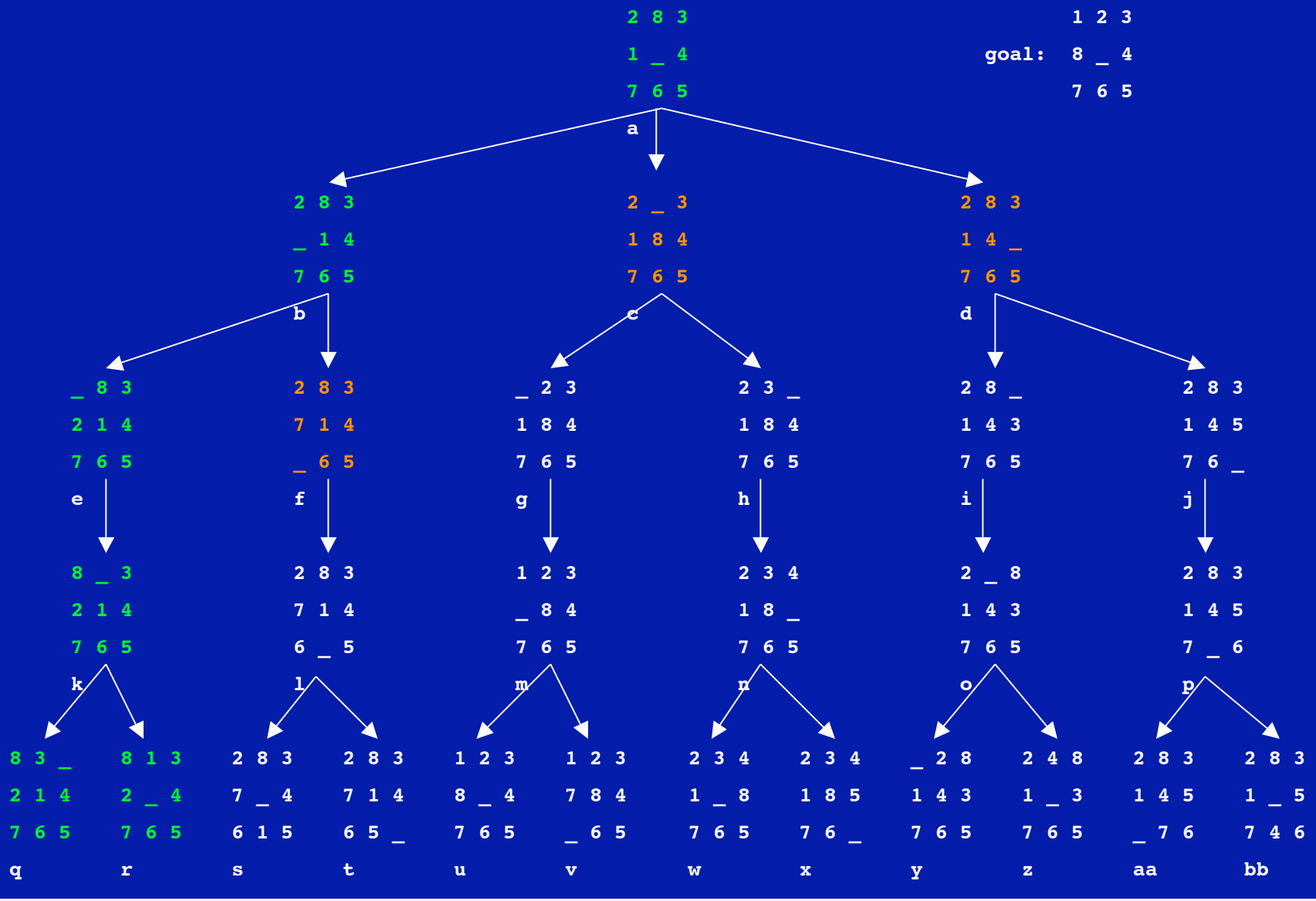
frontier: [r,f,c,d]



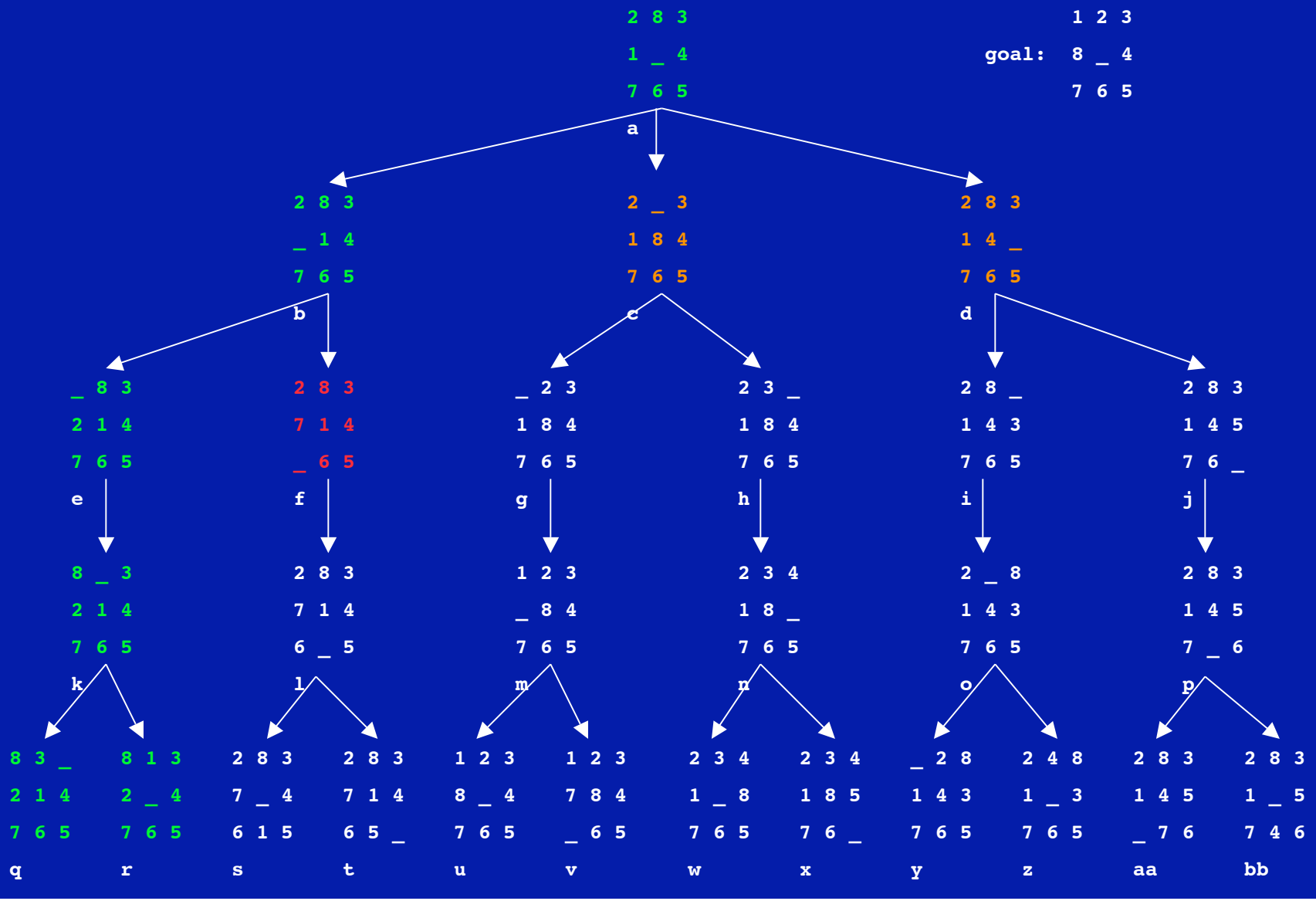
frontier: [f,c,d]



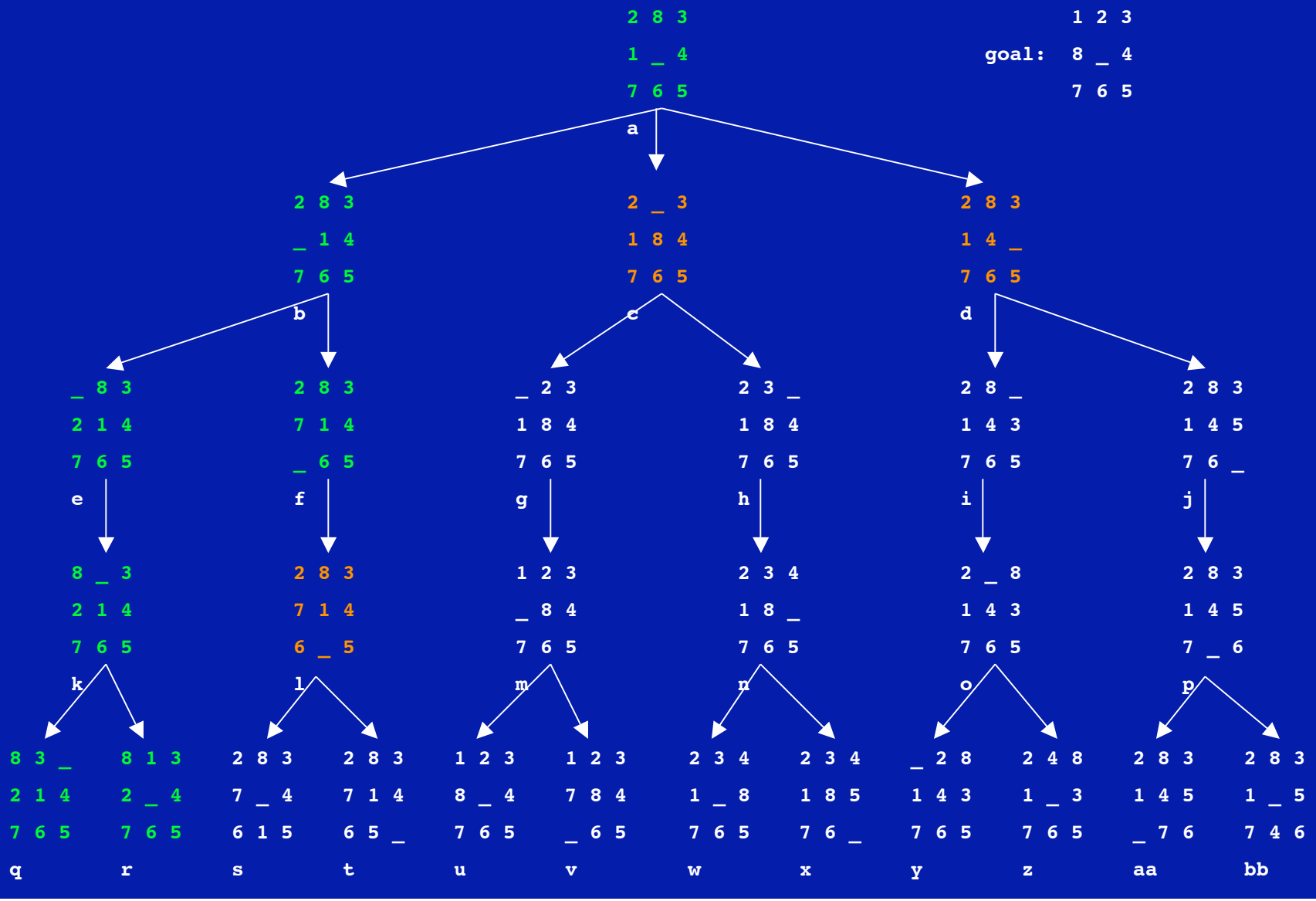
frontier: [f,c,d]



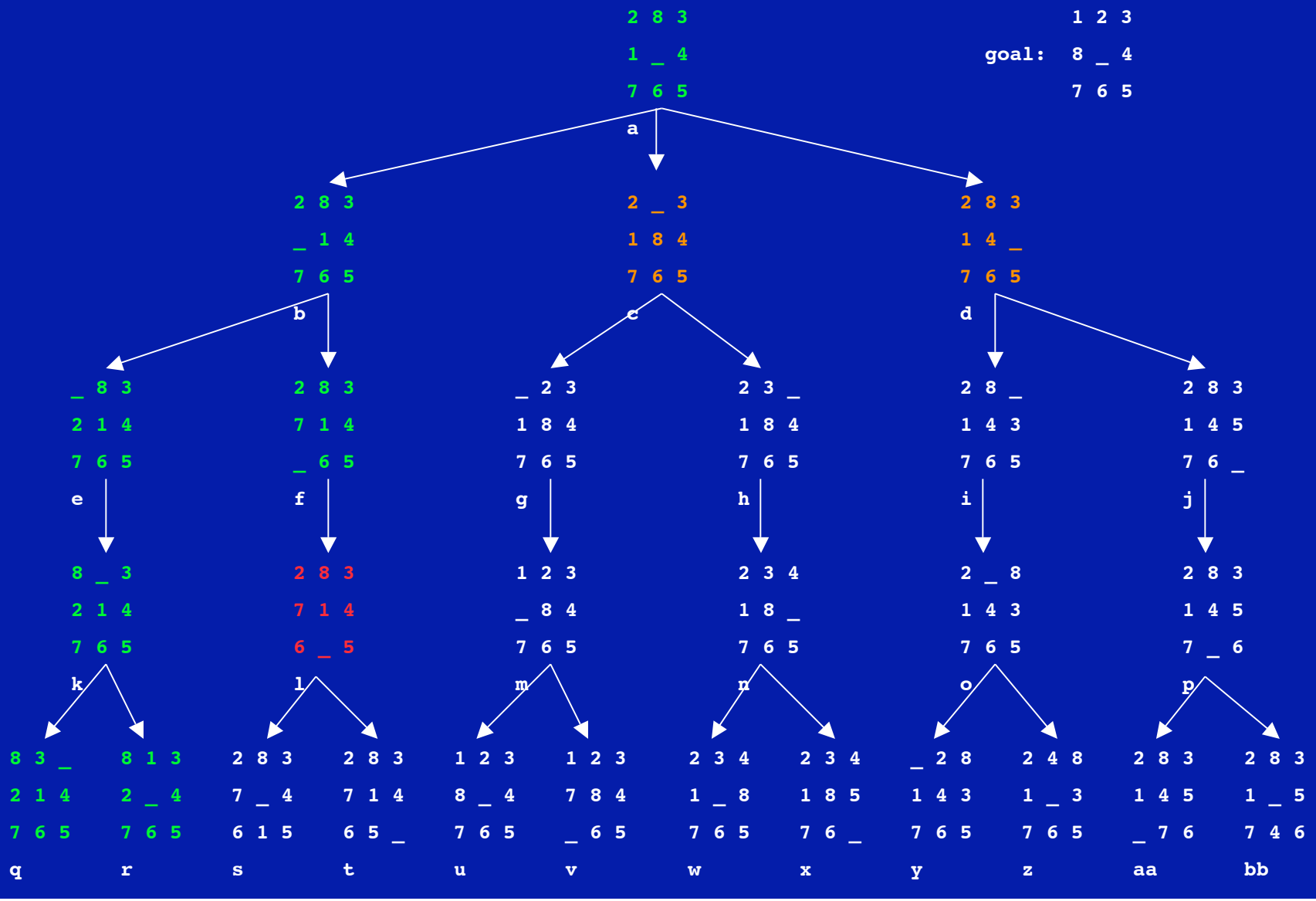
frontier: [c,d]


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

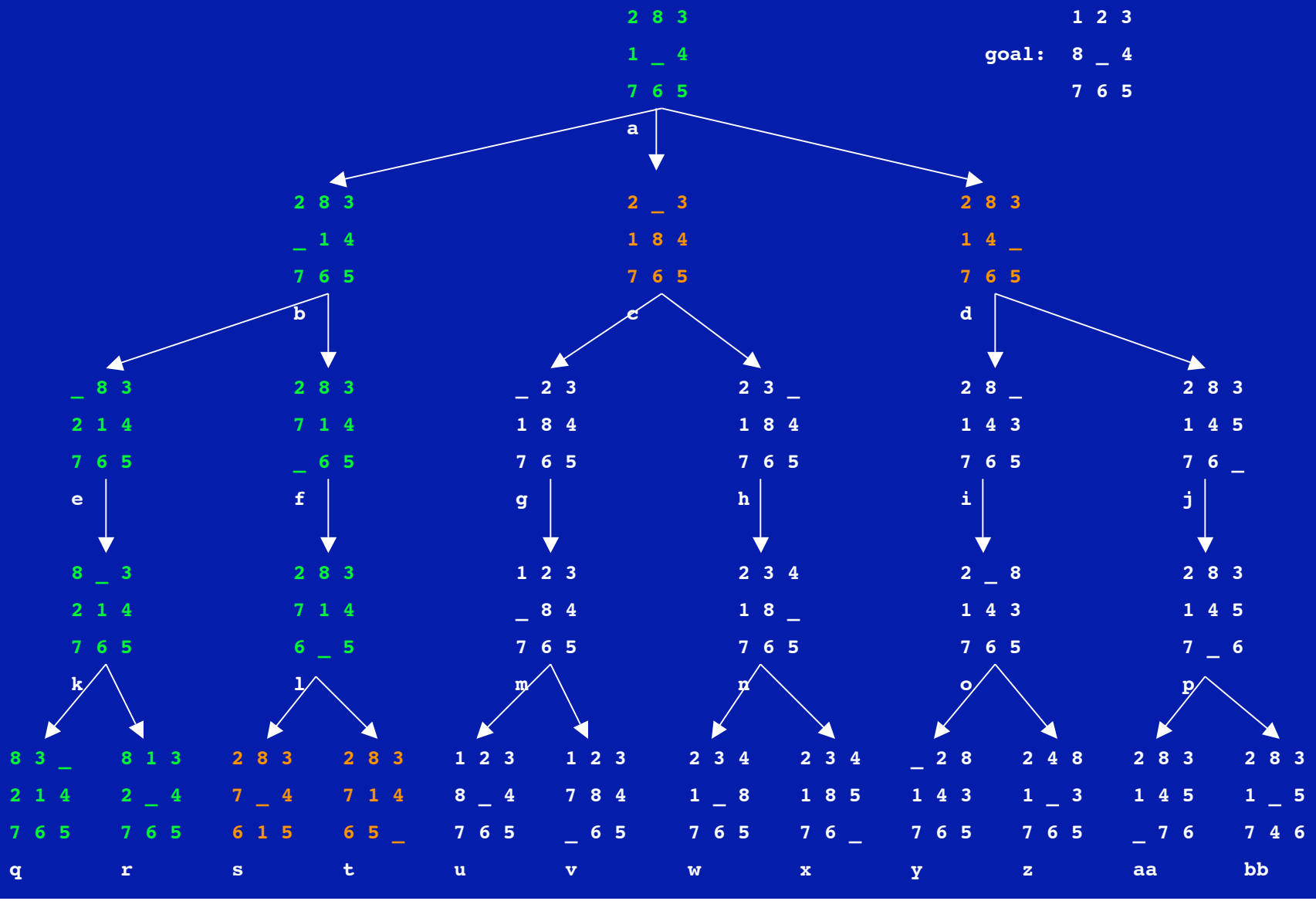
frontier: [l,c,d]



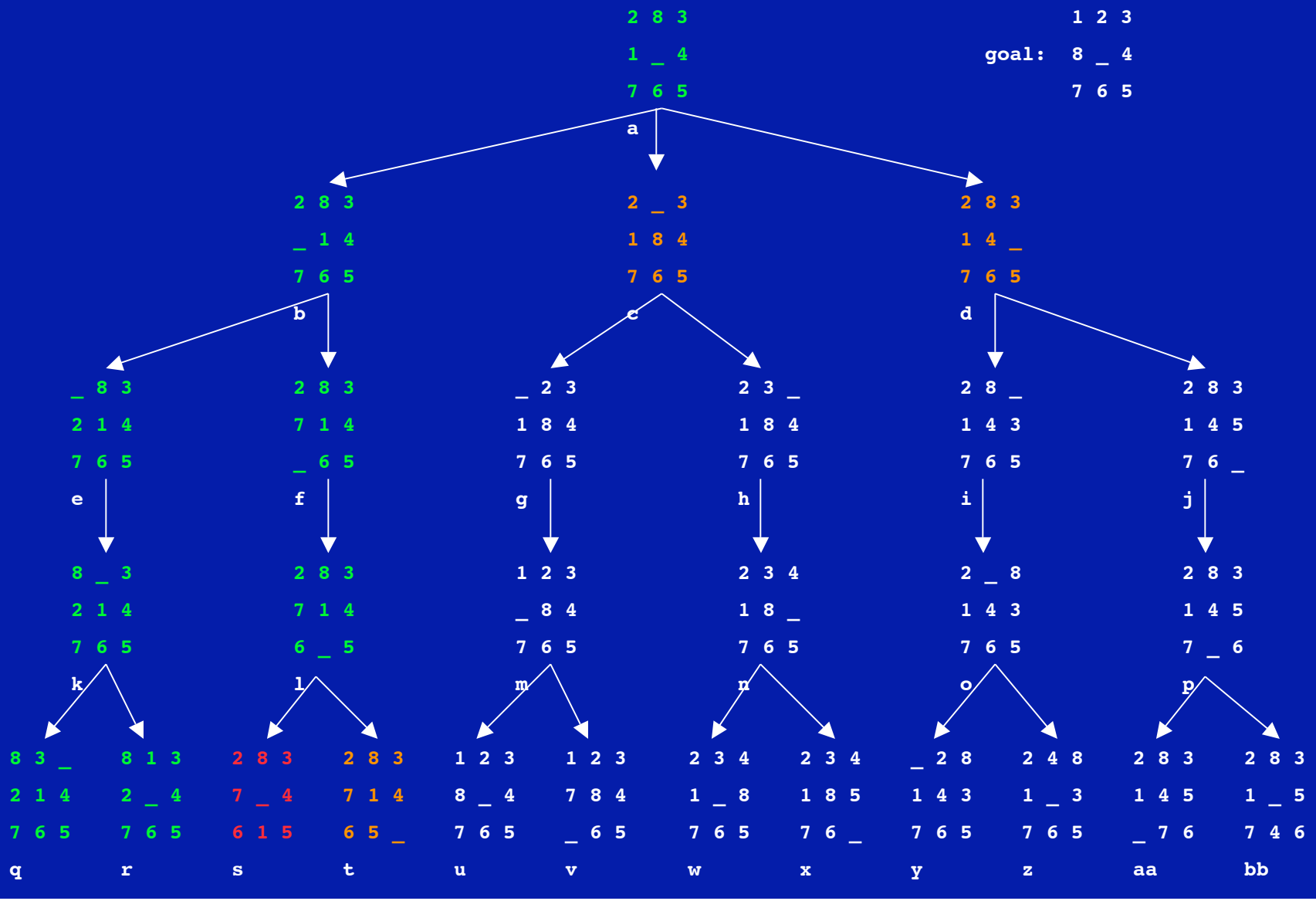
frontier: [c,d]


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

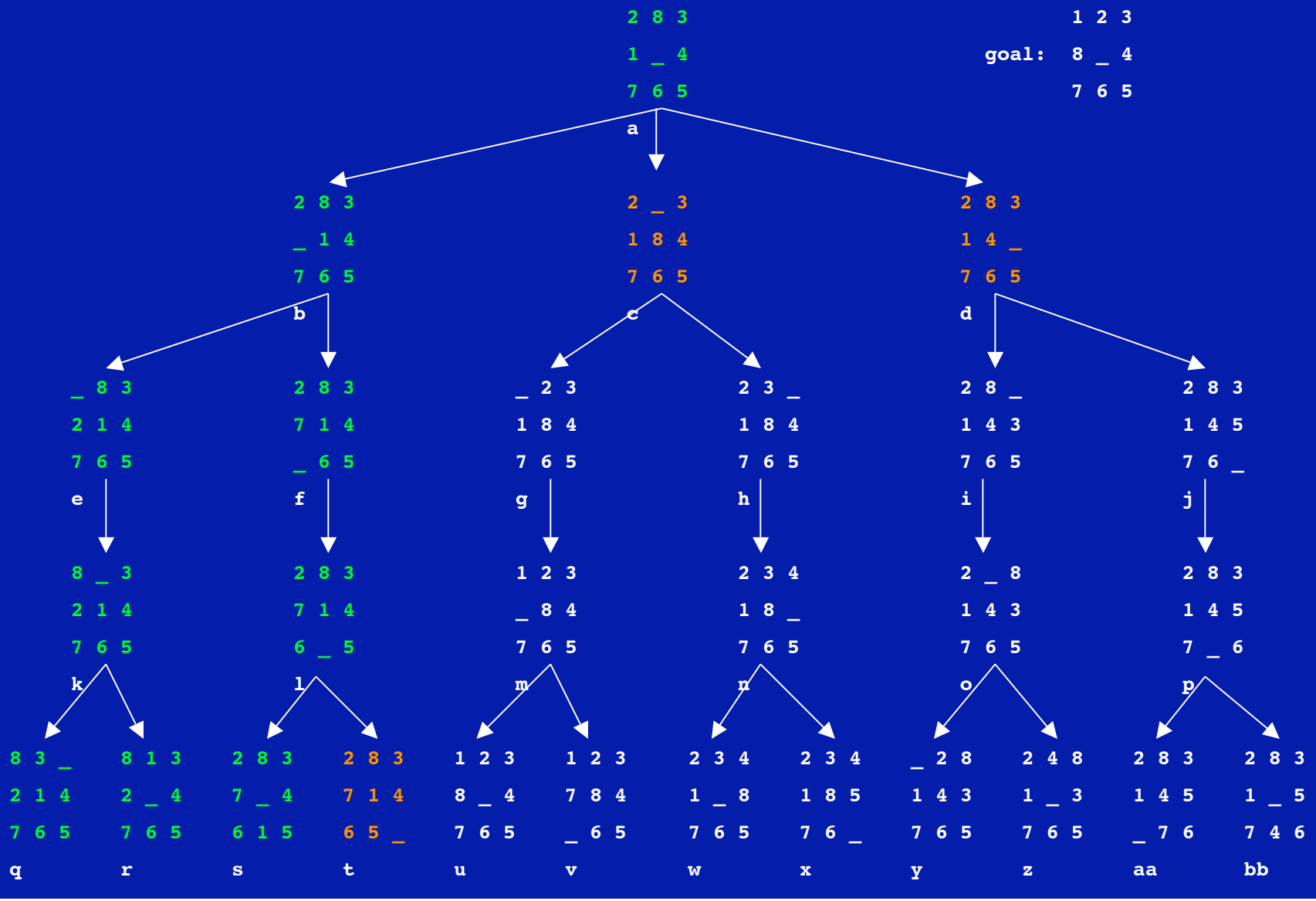
frontier: [s,t,c,d]



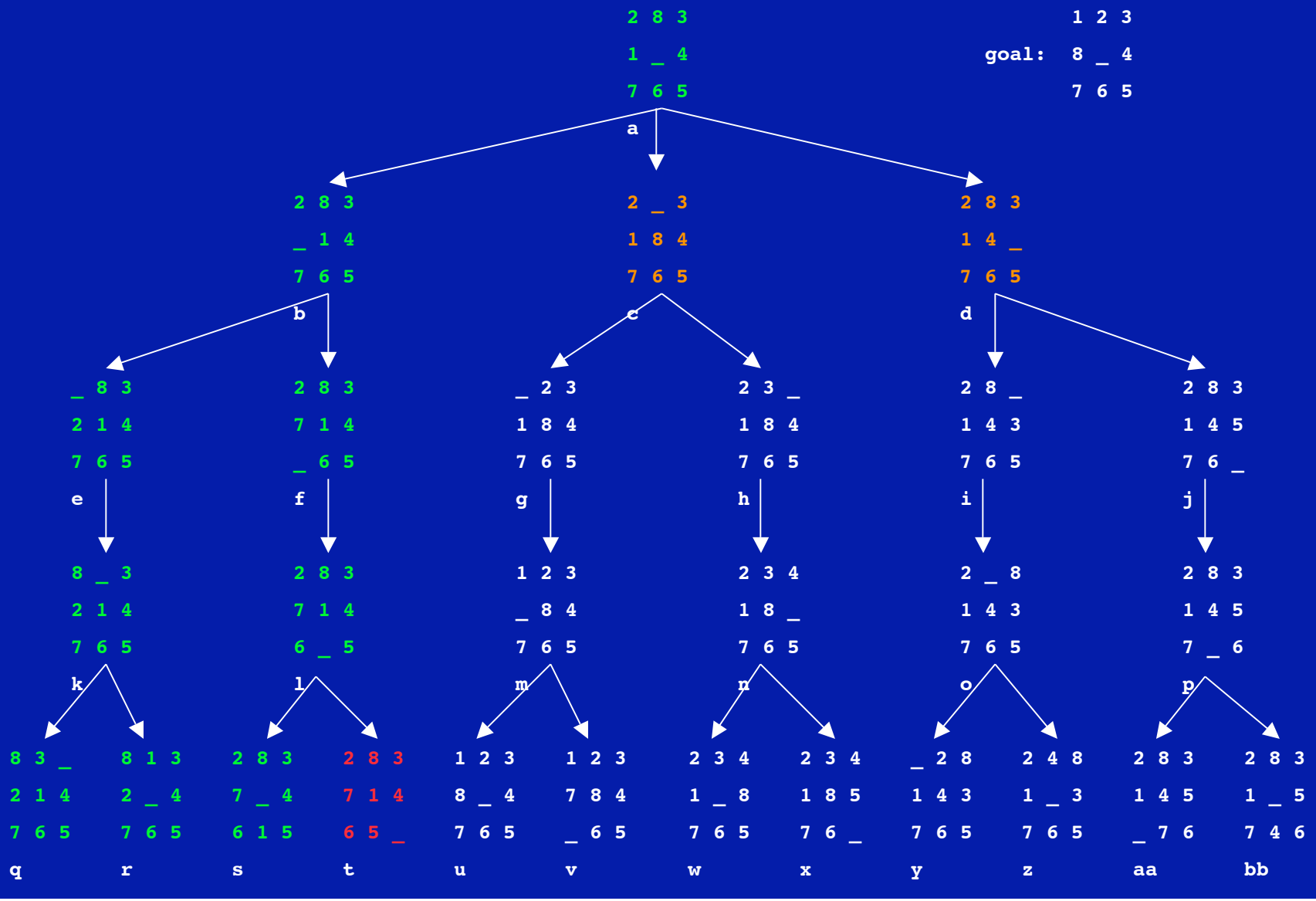
frontier: [t,c,d]



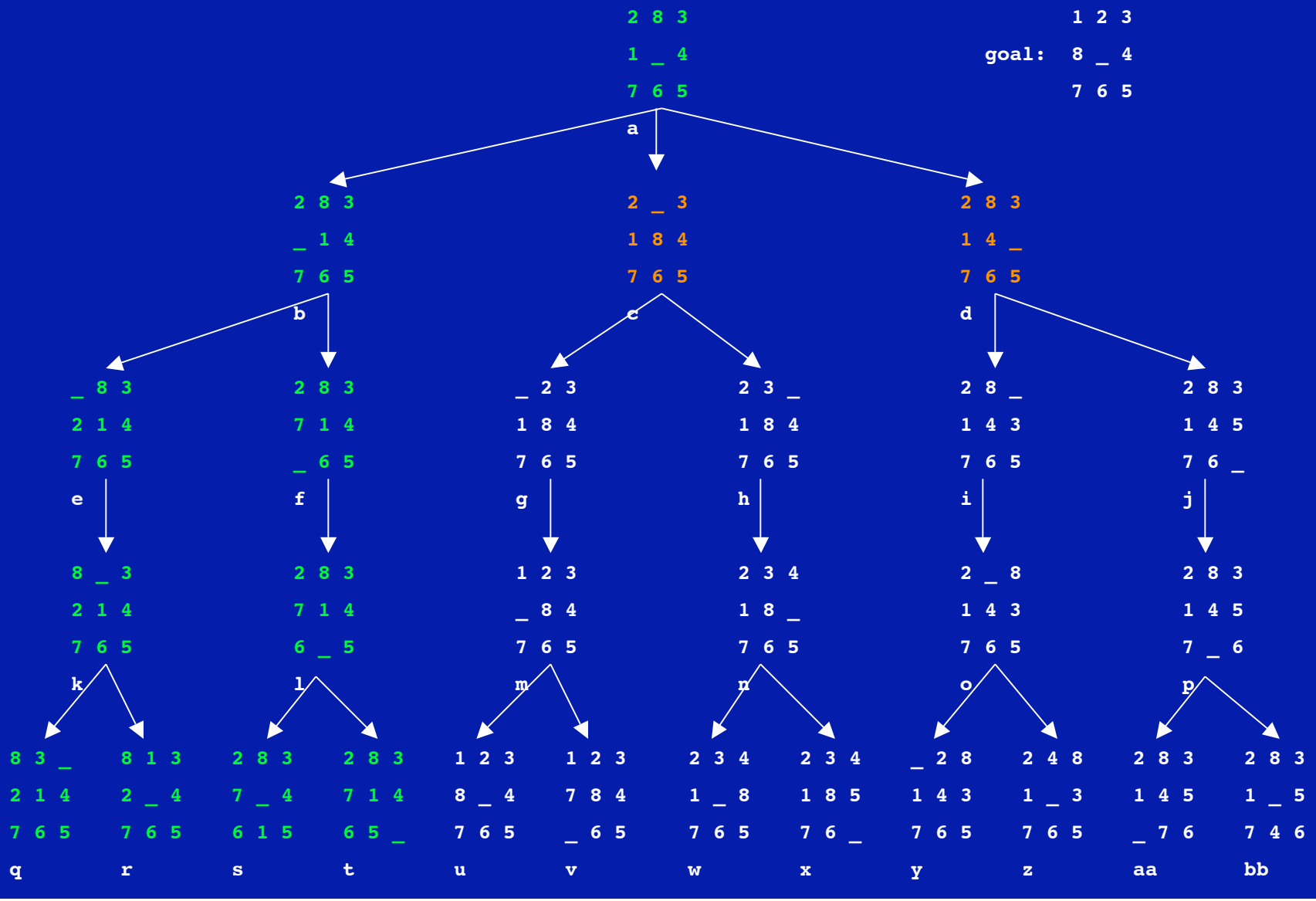
frontier: [t,c,d]



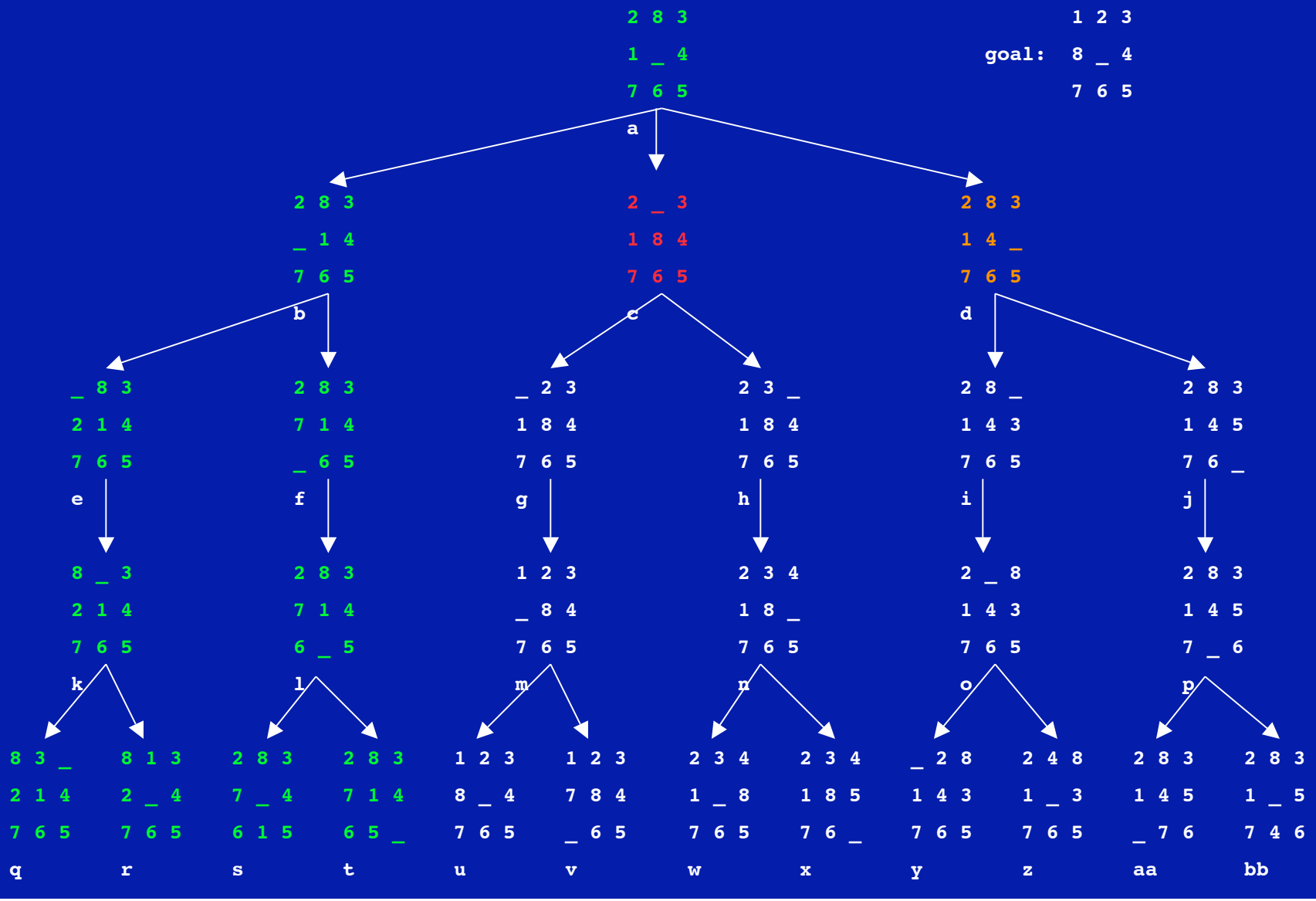
frontier: [c,d]


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

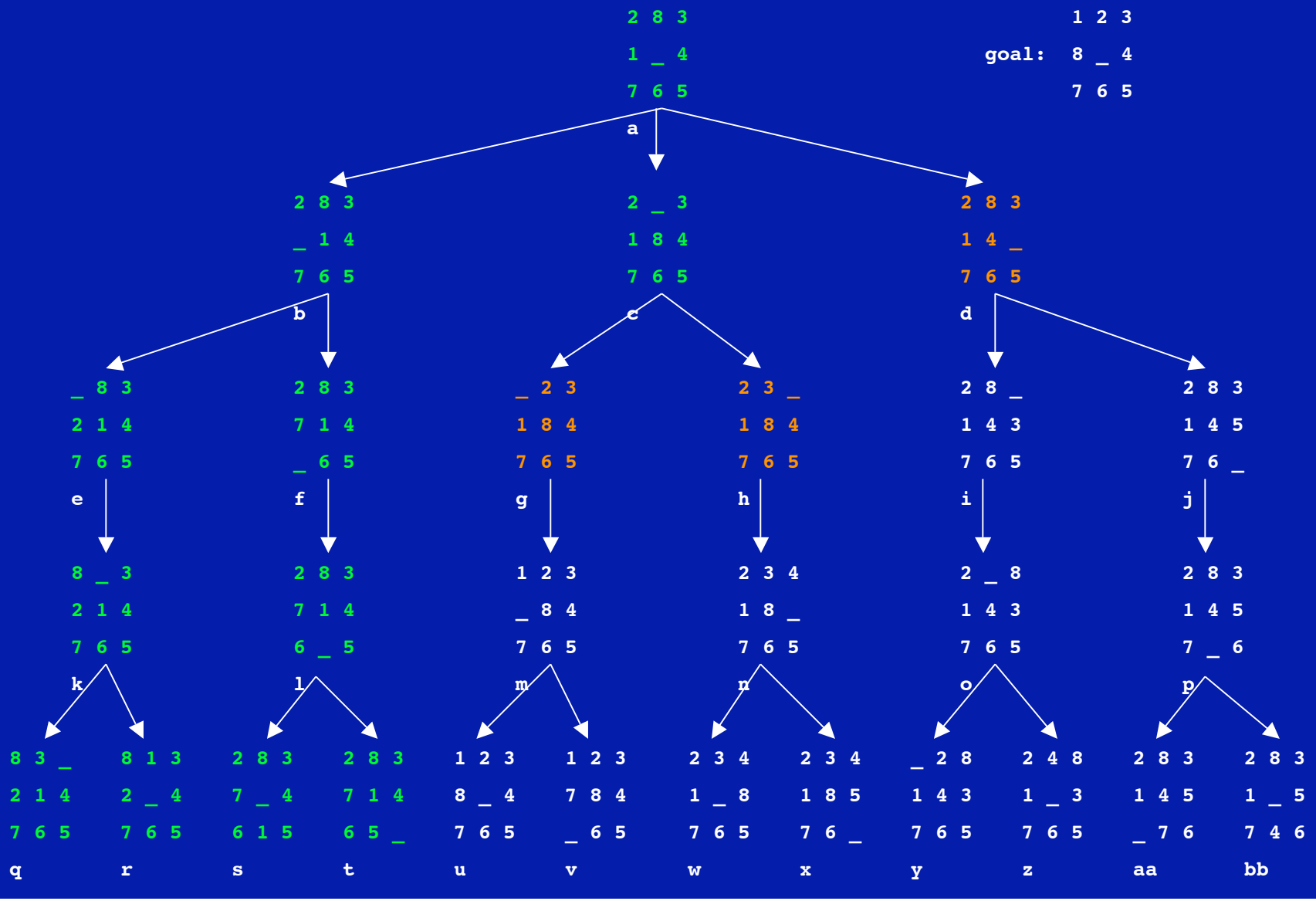
frontier: [c,d]


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

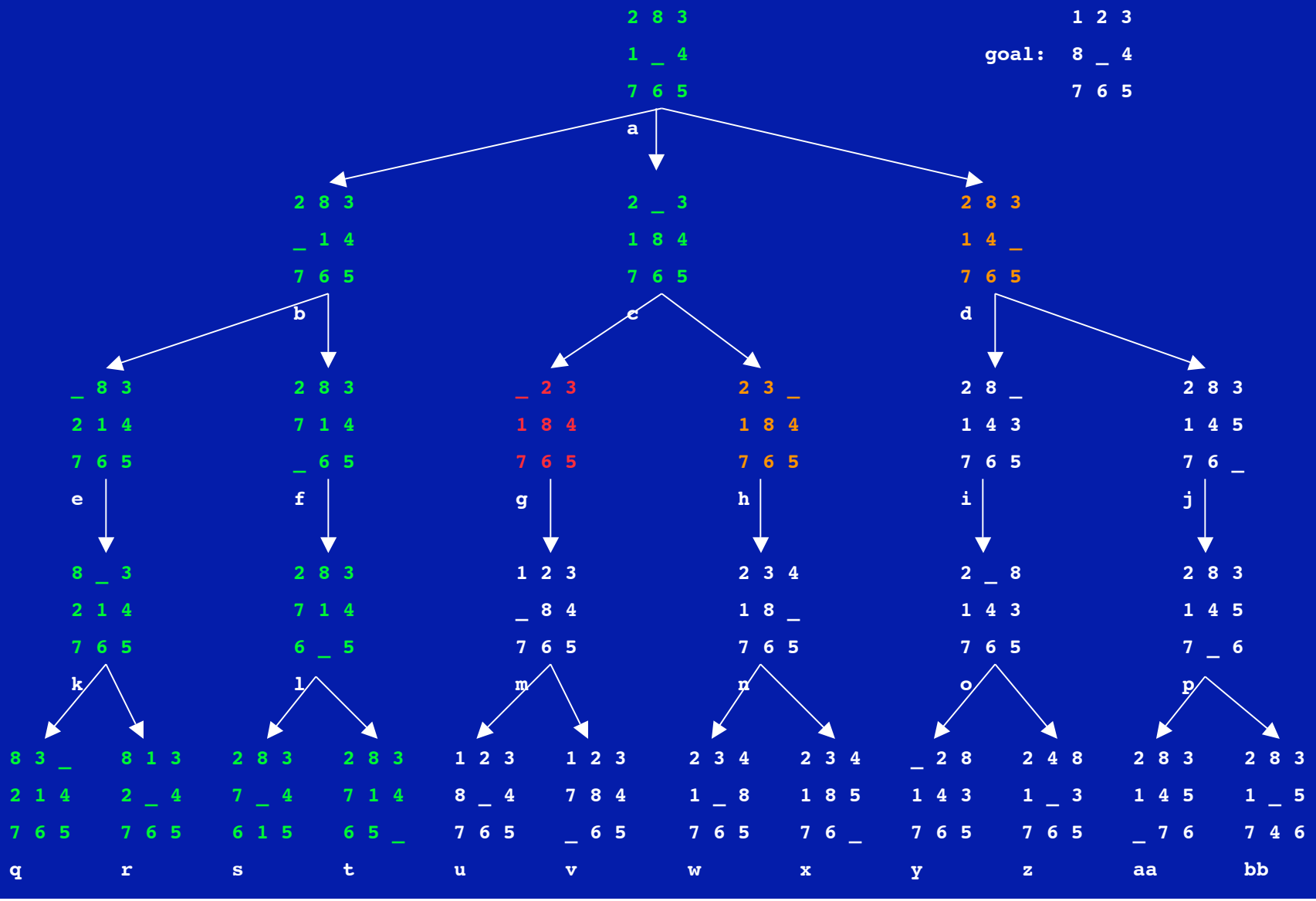
frontier: [d]


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

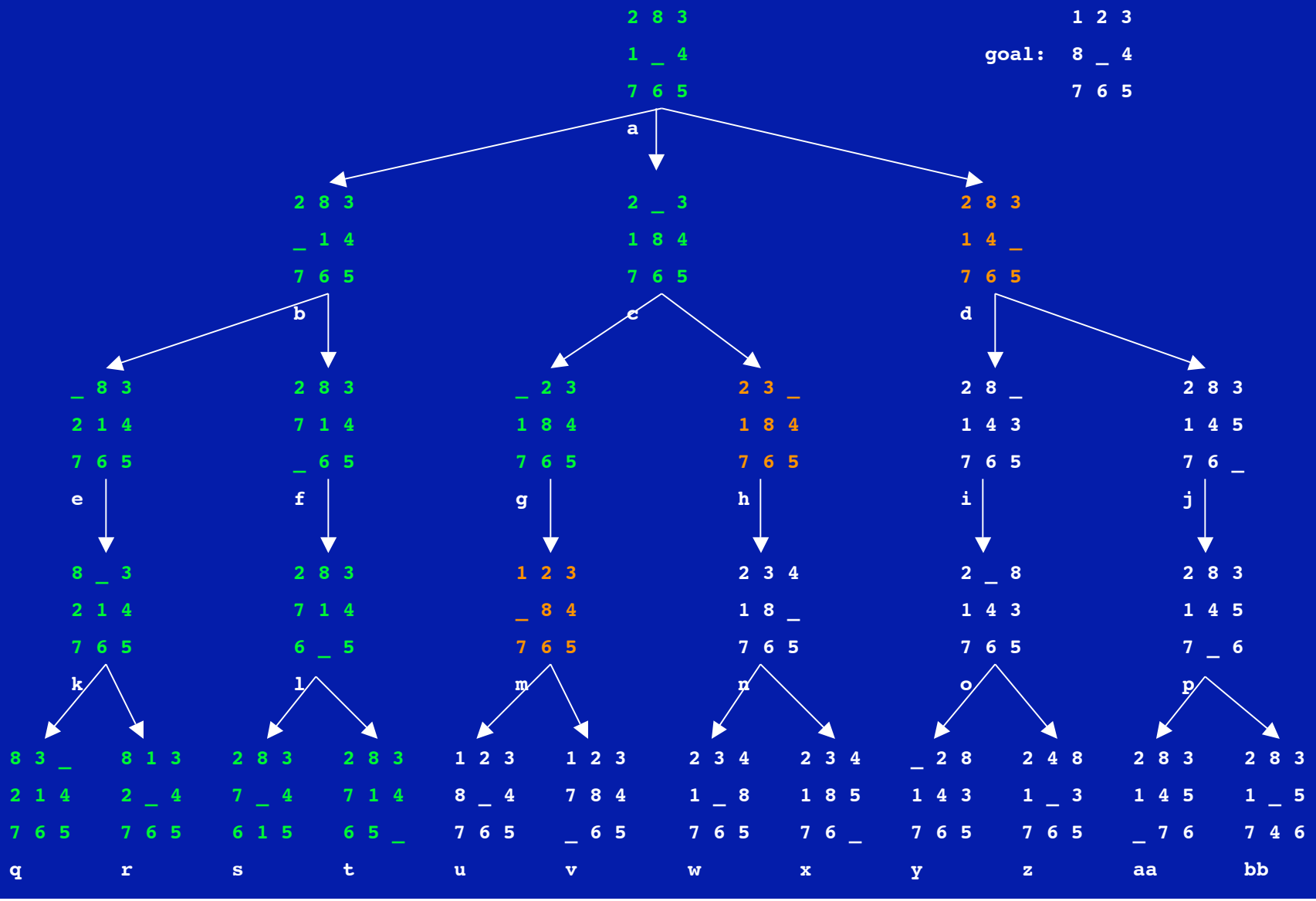
frontier: [g,h,d]



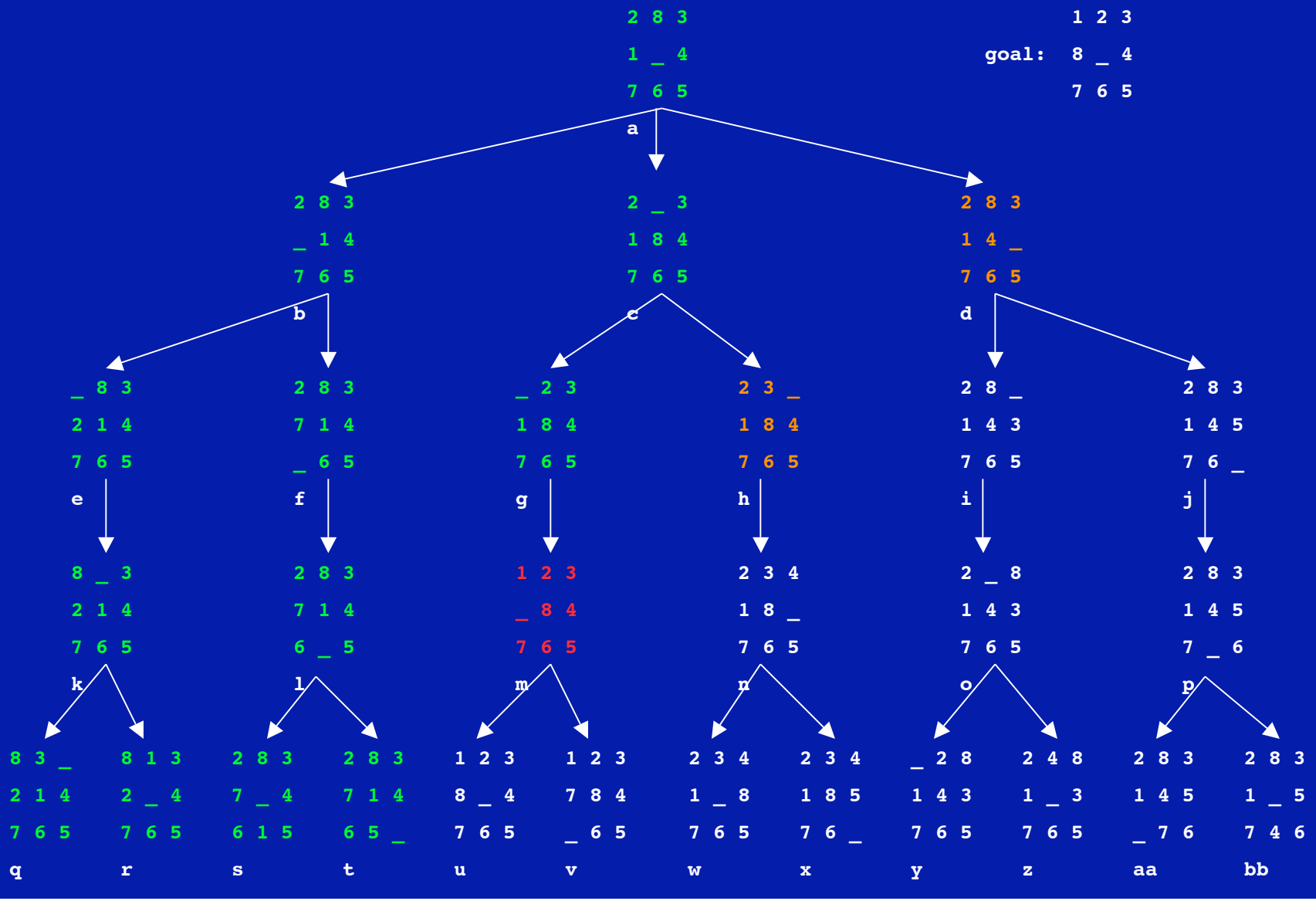
frontier: [h,d]


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

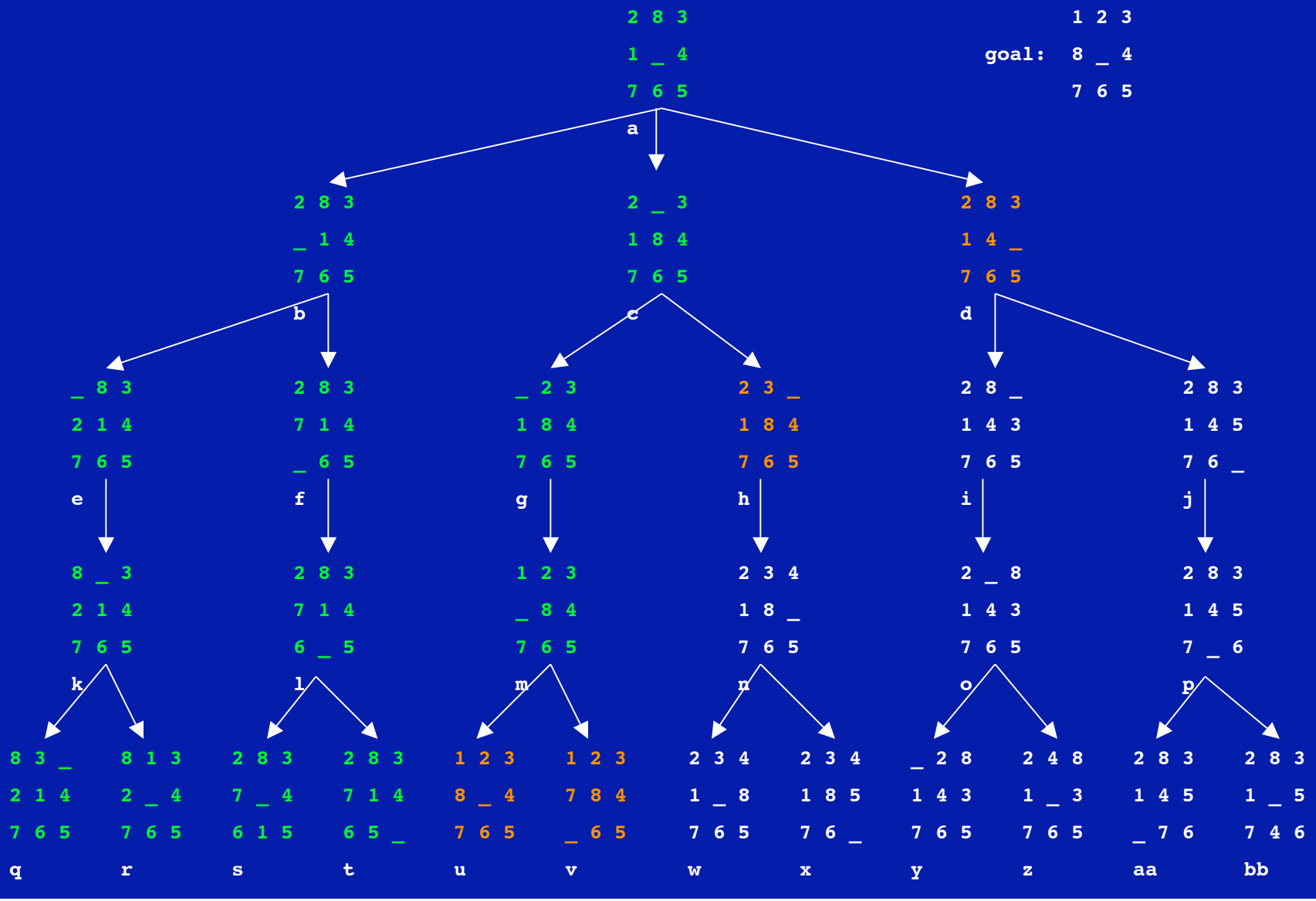
frontier: [m,h,d]



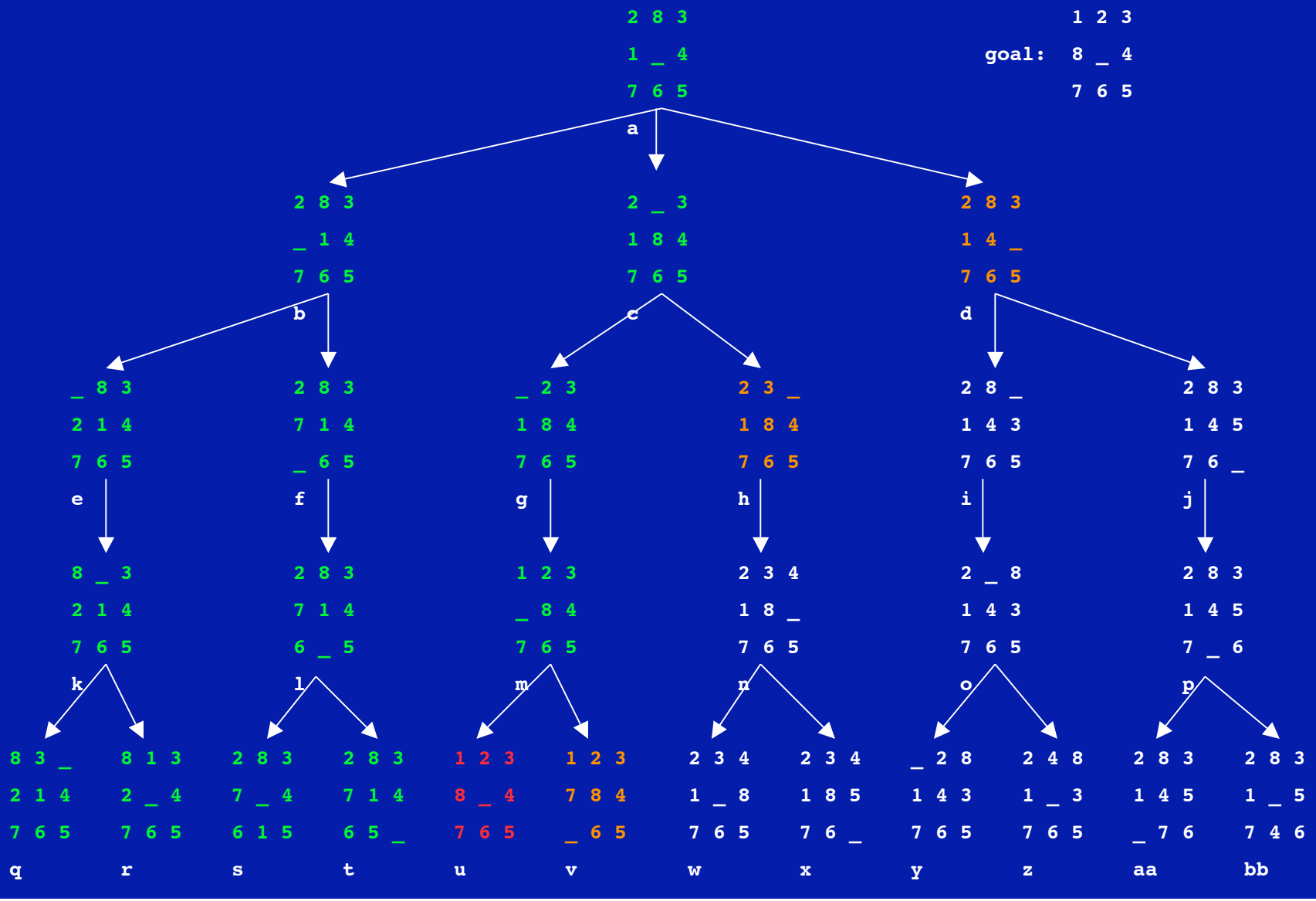
frontier: [h,d]


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

frontier: [u,v,h,d]



frontier: [v,h,d]



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

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 front of frontier and remove it
  if the chosen node matches the goal node
    then terminate with success
    else put next nodes (neighbors) on front of frontier
end repeat

Depth-first 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 front of frontier and remove it
  if the chosen node matches the goal node
    then terminate with success
    else put next nodes (neighbors) on front of frontier
end repeat

Datalog graph search algorithm

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

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

Datalog graph search algorithm

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

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

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).
```

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).
```

interpreting the predicates:

search(F0) is true if there is a path from one element in the list F0 - the frontier - to a goal node

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).
```

interpreting the predicates:

is_goal(N) is true if N is a goal node

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).
```

interpreting the predicates:

neighbors(N,NN) is true if NN is the list of neighbors of node N

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).
```

interpreting the predicates:

choose(N,F0,F1) means N is some element selected from F0 - the frontier - and F1 is the set of nodes remaining when N is removed. This fails if F0 is empty.

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).
```

interpreting the predicates:

add_to_frontier(NN,F1,F2) means that F2 is the new frontier made by adding the list of nodes NN to the old frontier F1

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).
```

So how do you implement choose and add_to_frontier for depth-first search?

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).
```

Time for a CILOG break!

look at ‘puzzle.ci’ in Word

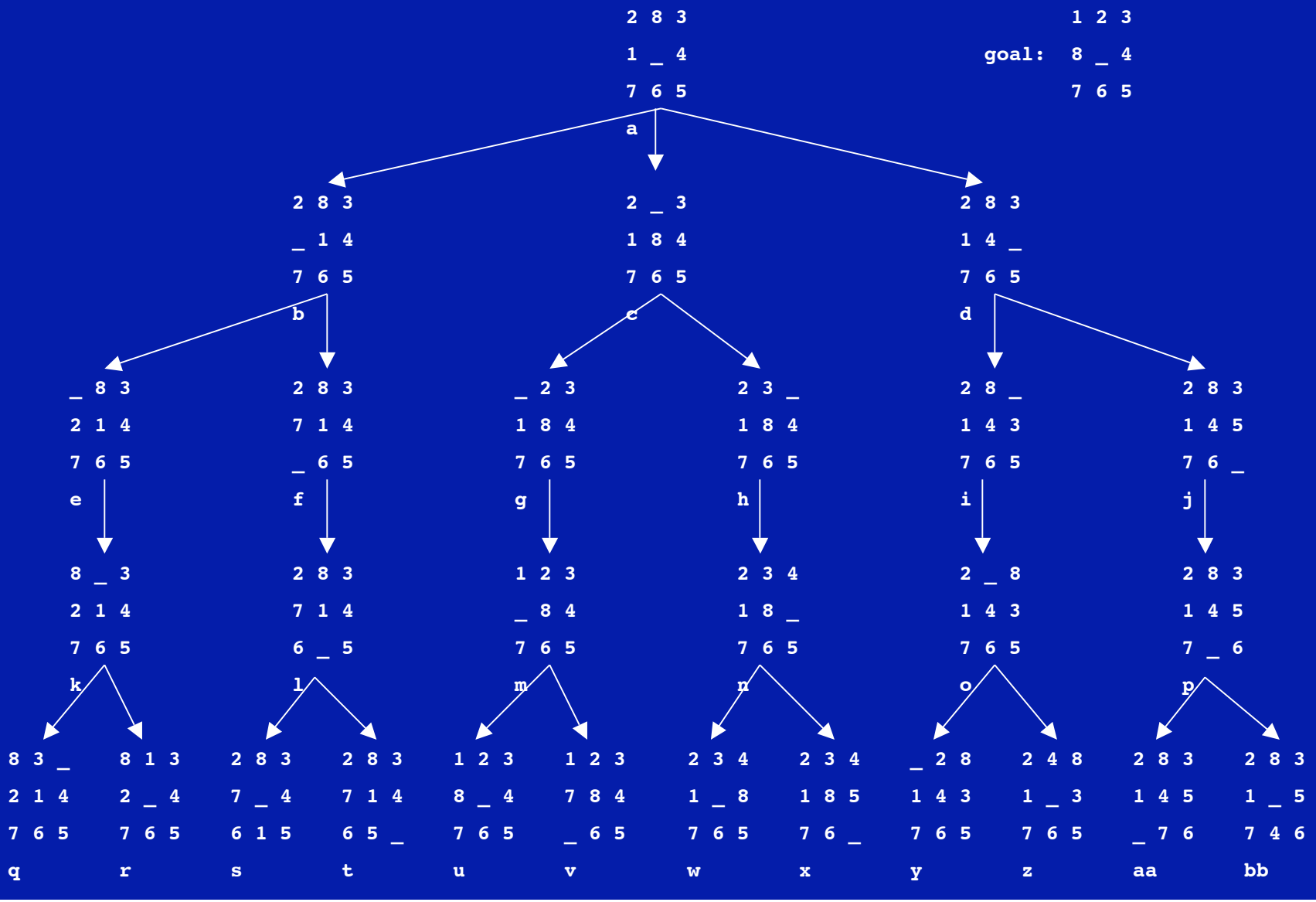
load ‘puzzle.ci’ in CILOG

cilog: ask search([[2,8,3,1,0,4,7,6,5]]).

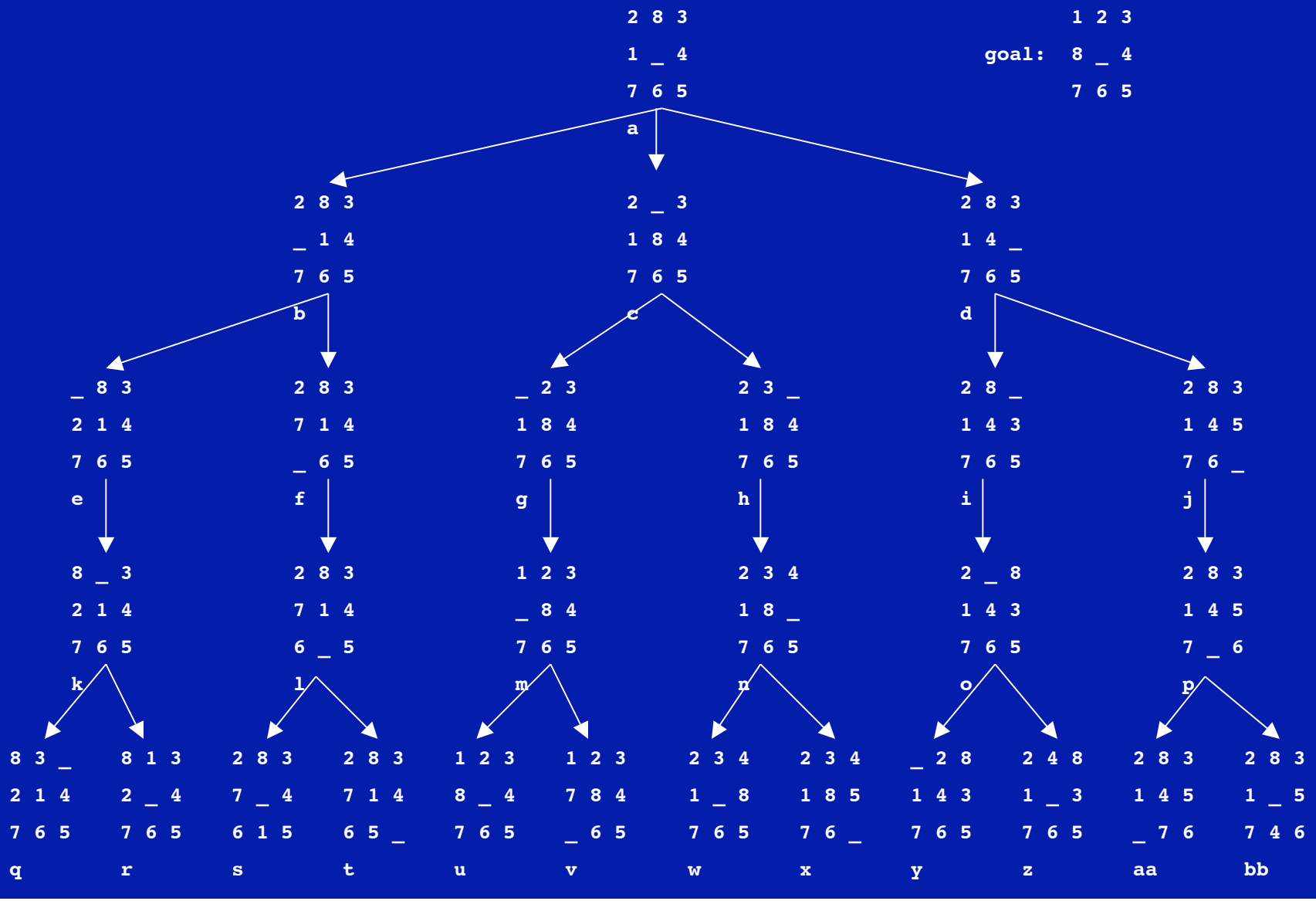
How would you change it all...

...to get breadth-first search instead of depth-first search?

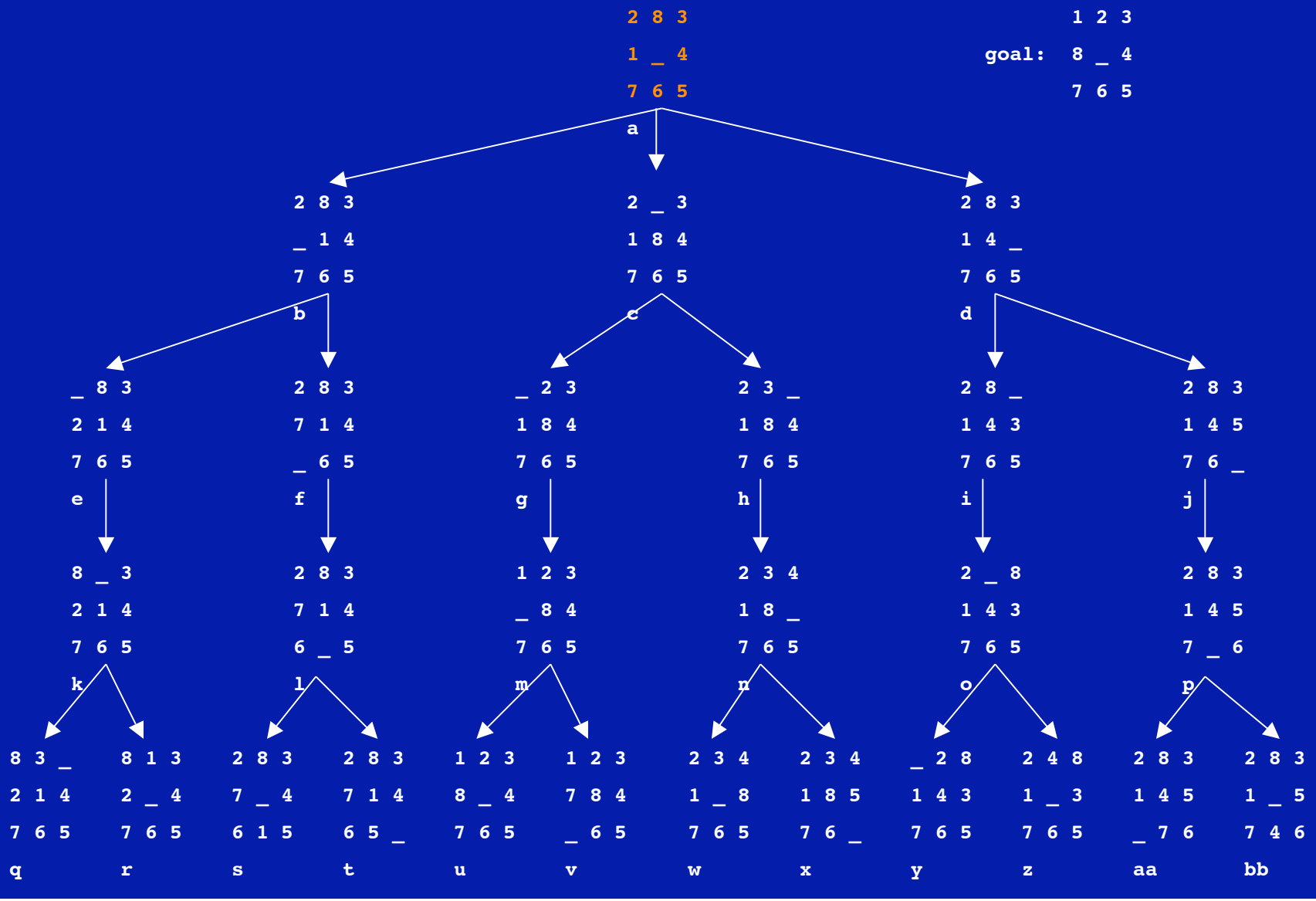
Breadth-first search



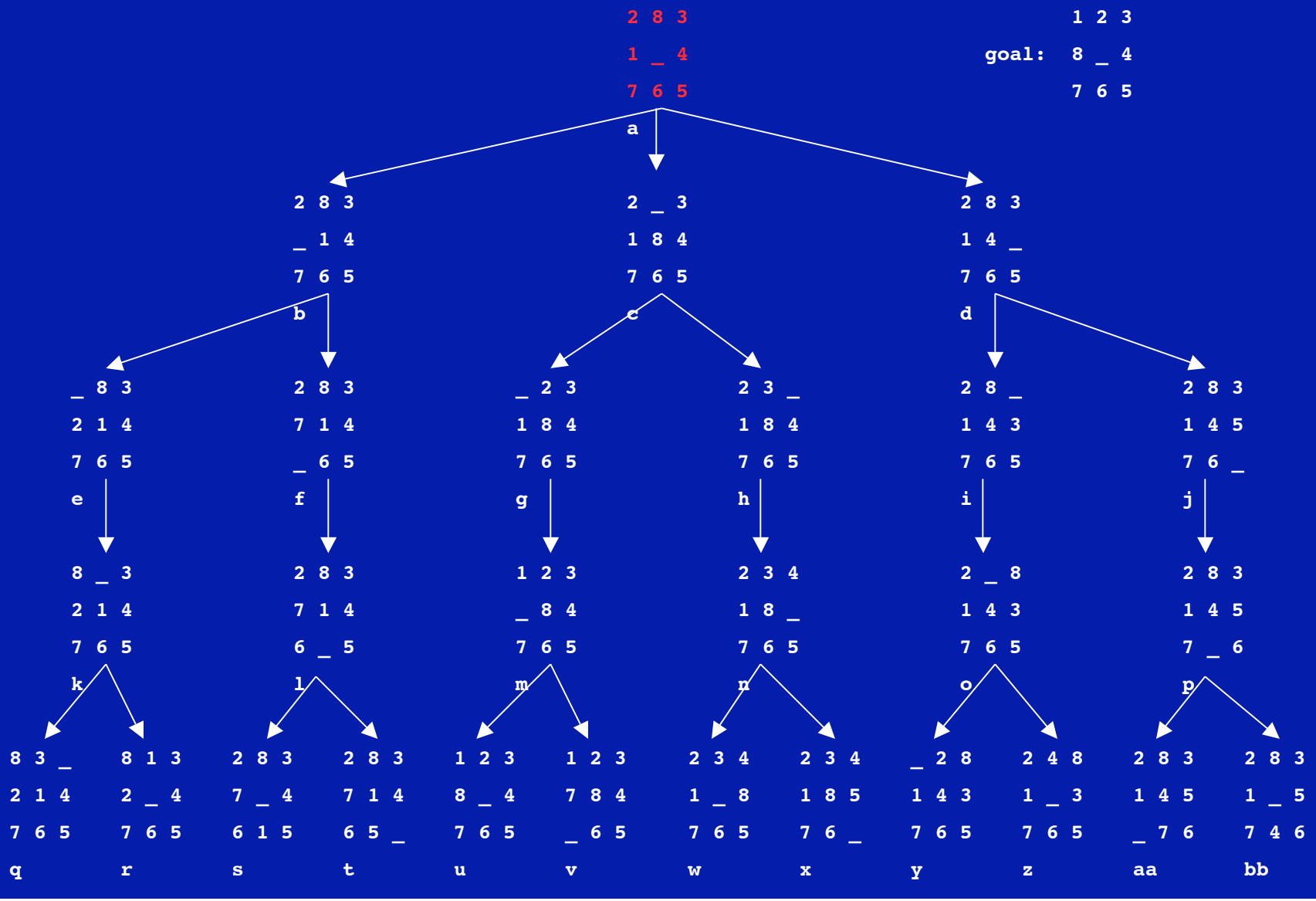
frontier: [ ]


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

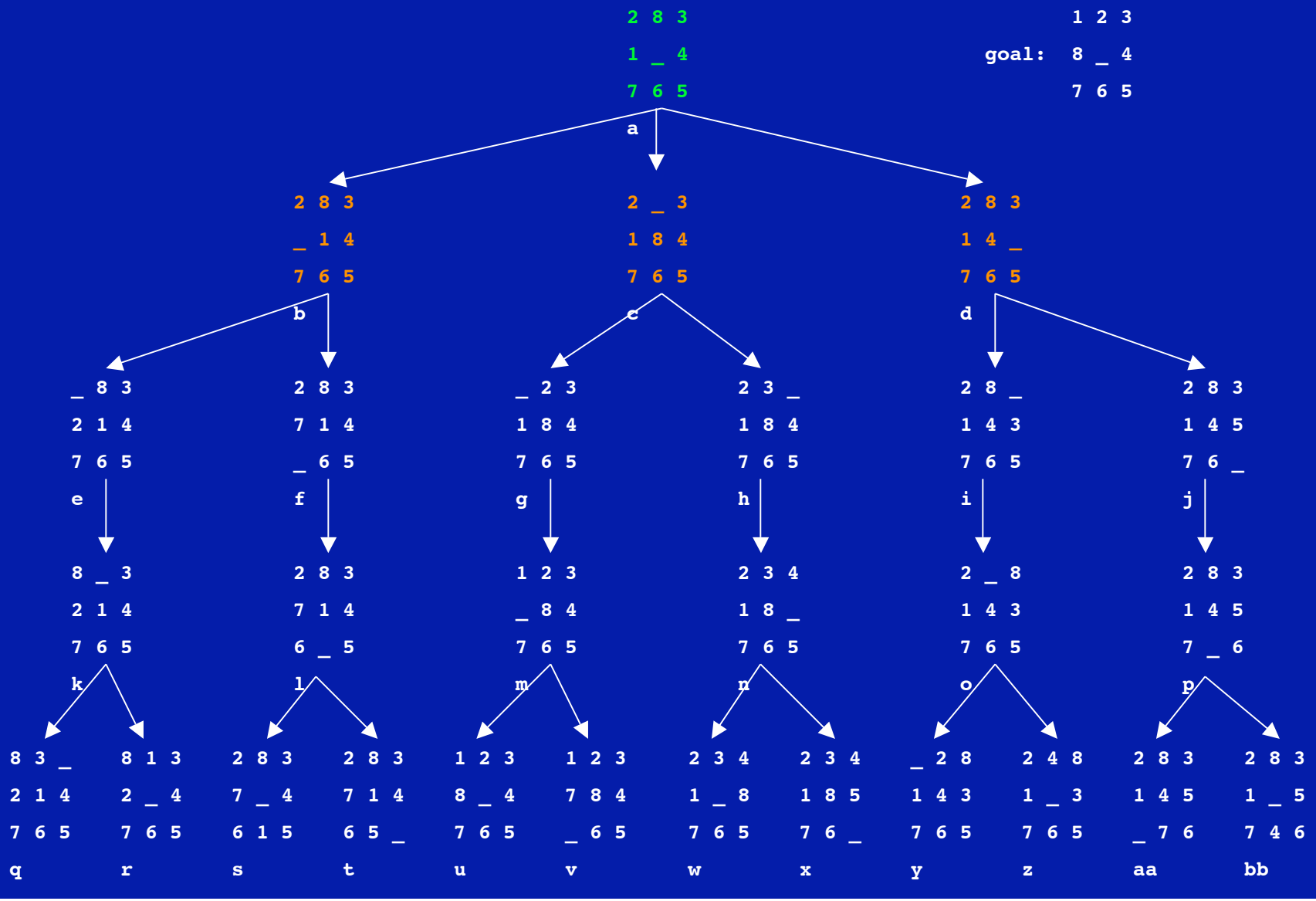
frontier: [a]


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

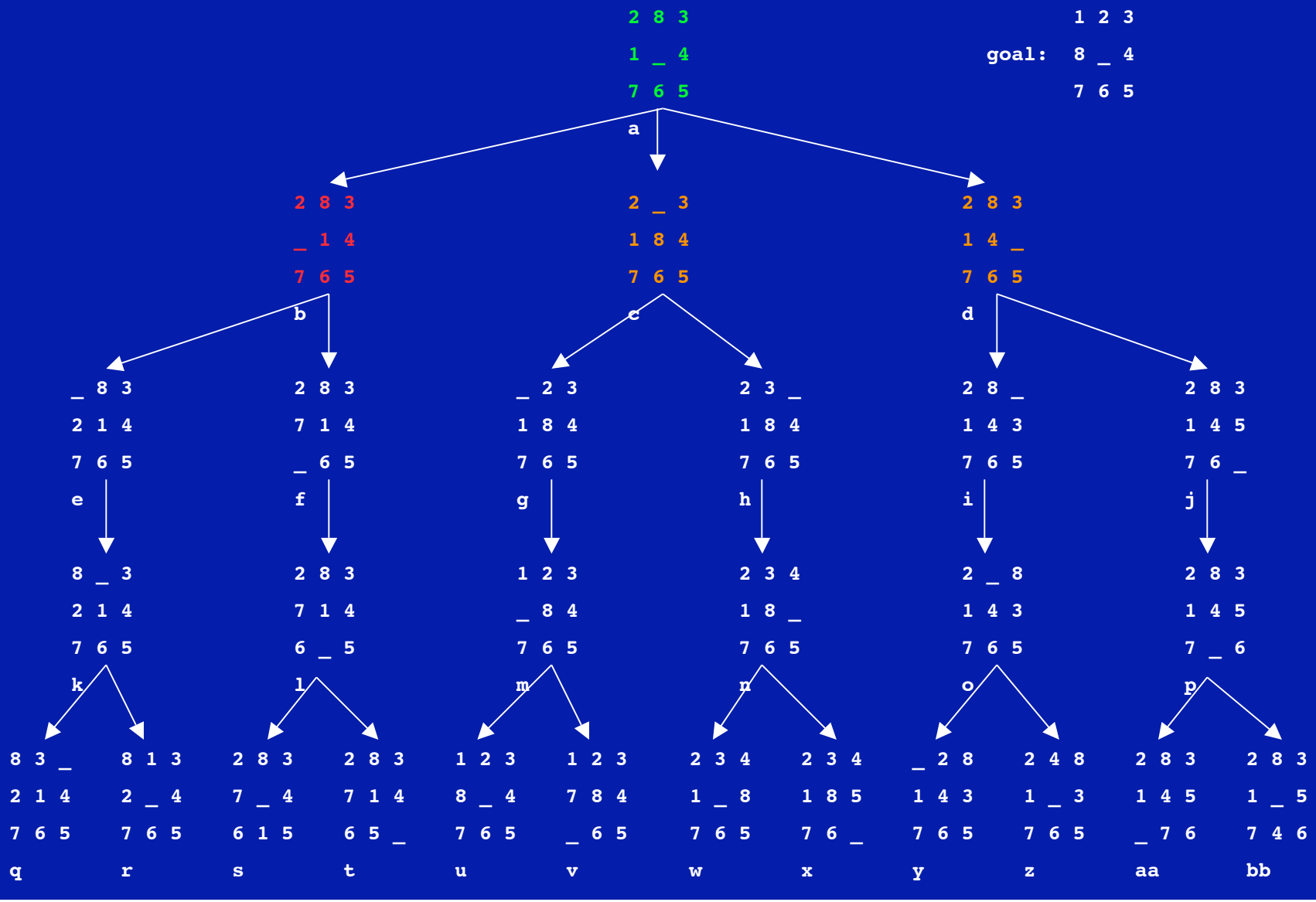
frontier: [ ]


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

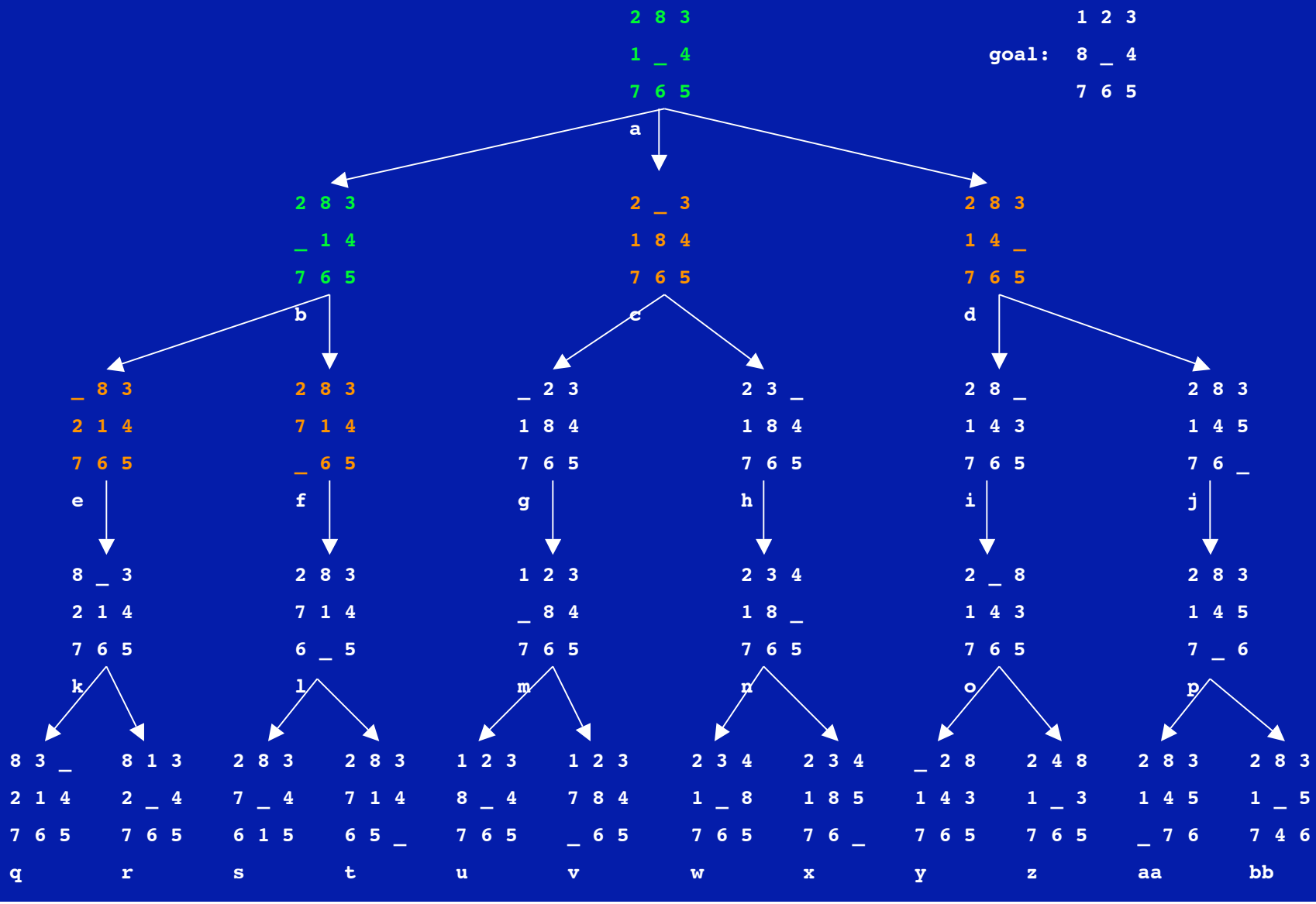
frontier: [b,c,d]



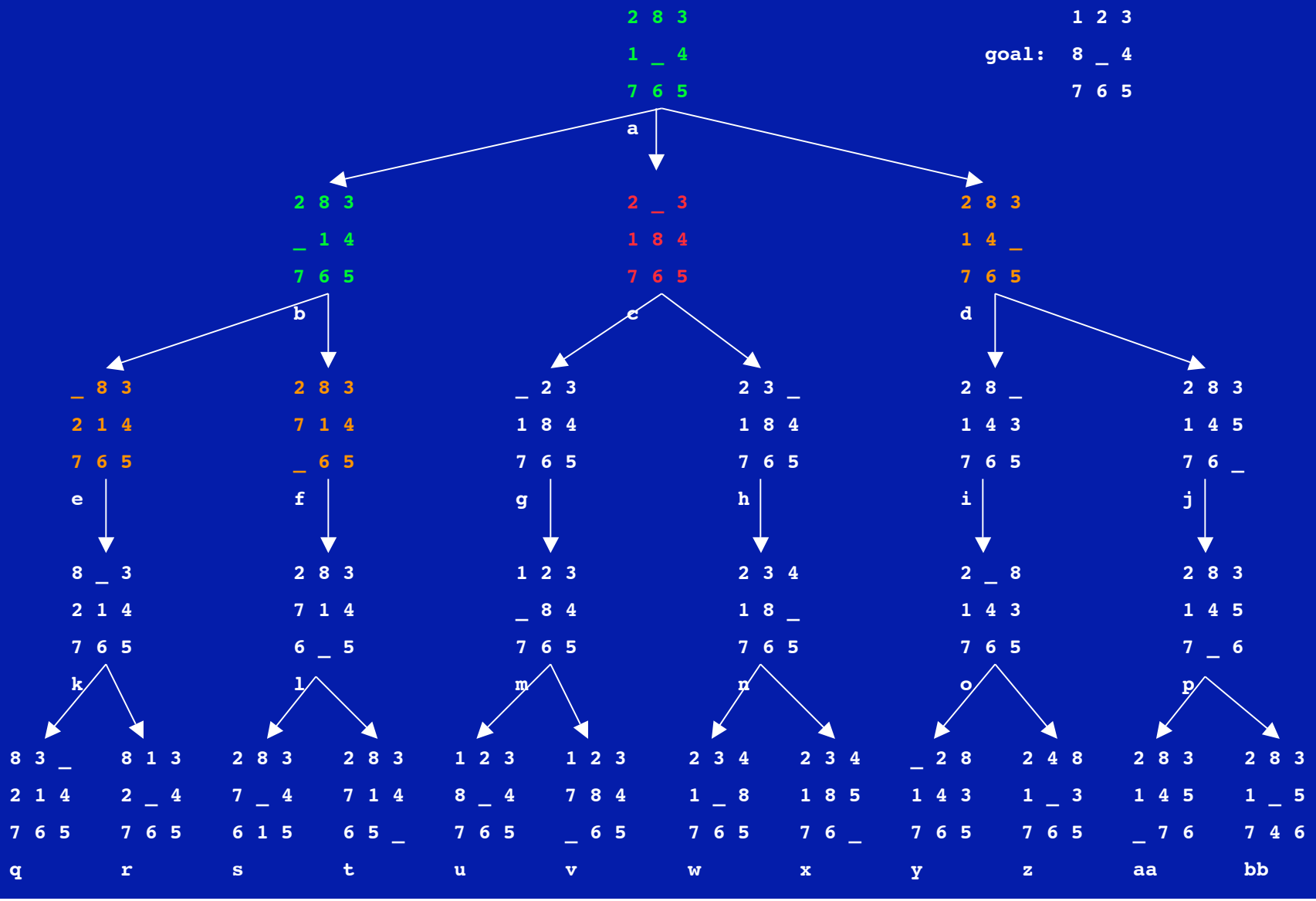
frontier: [c,d]


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

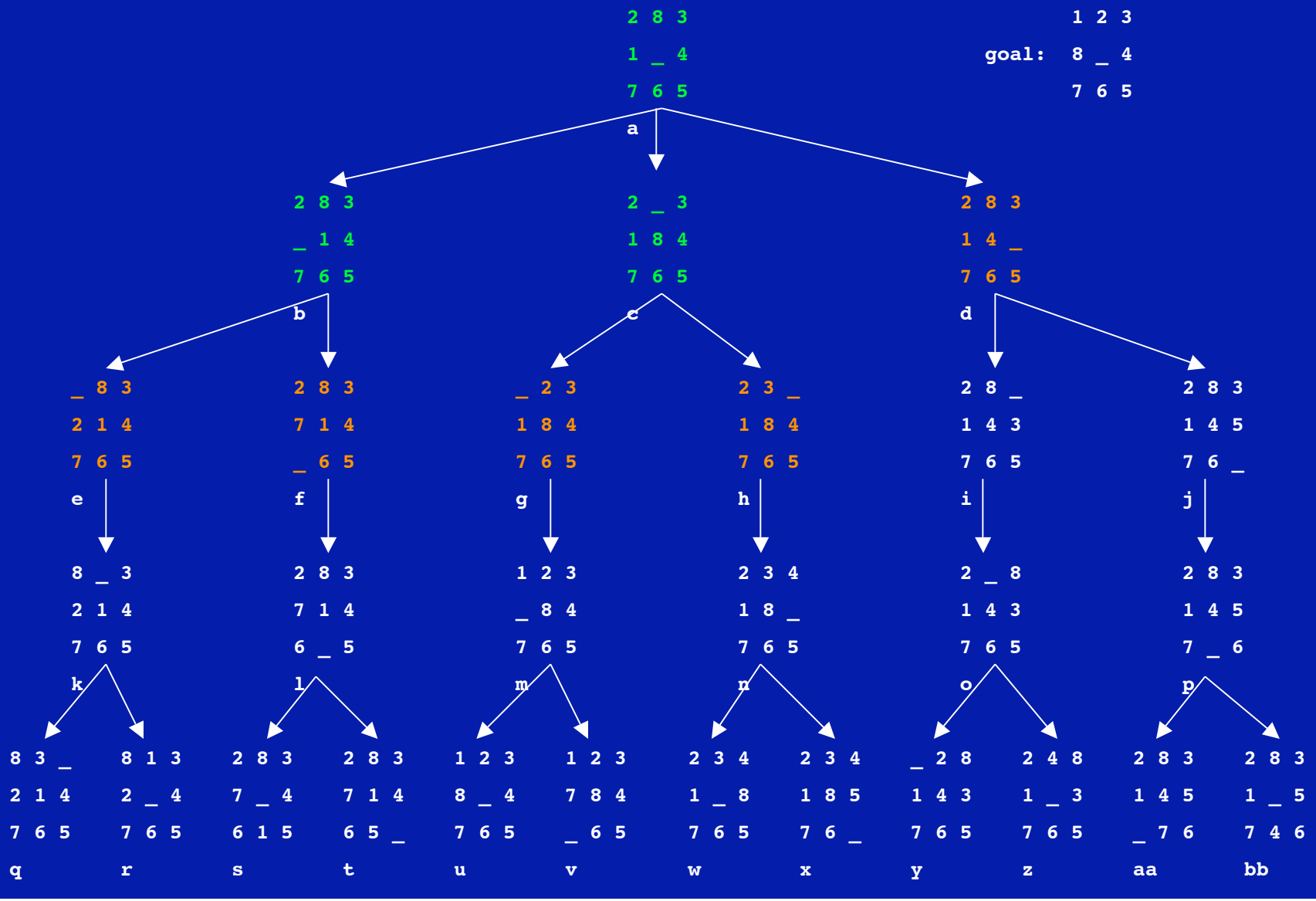
frontier: [c,d,e,f]



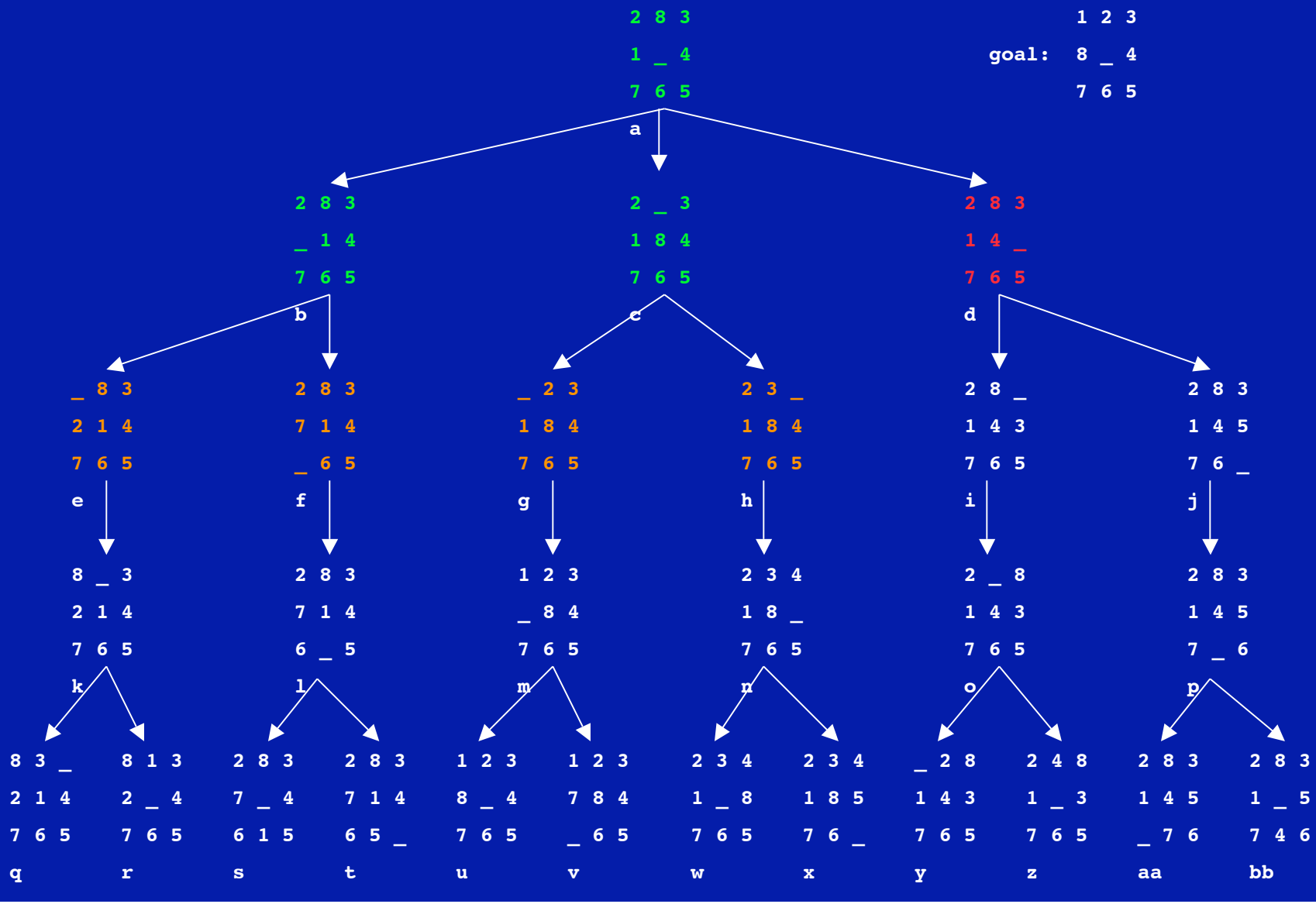
frontier: [d,e,f]



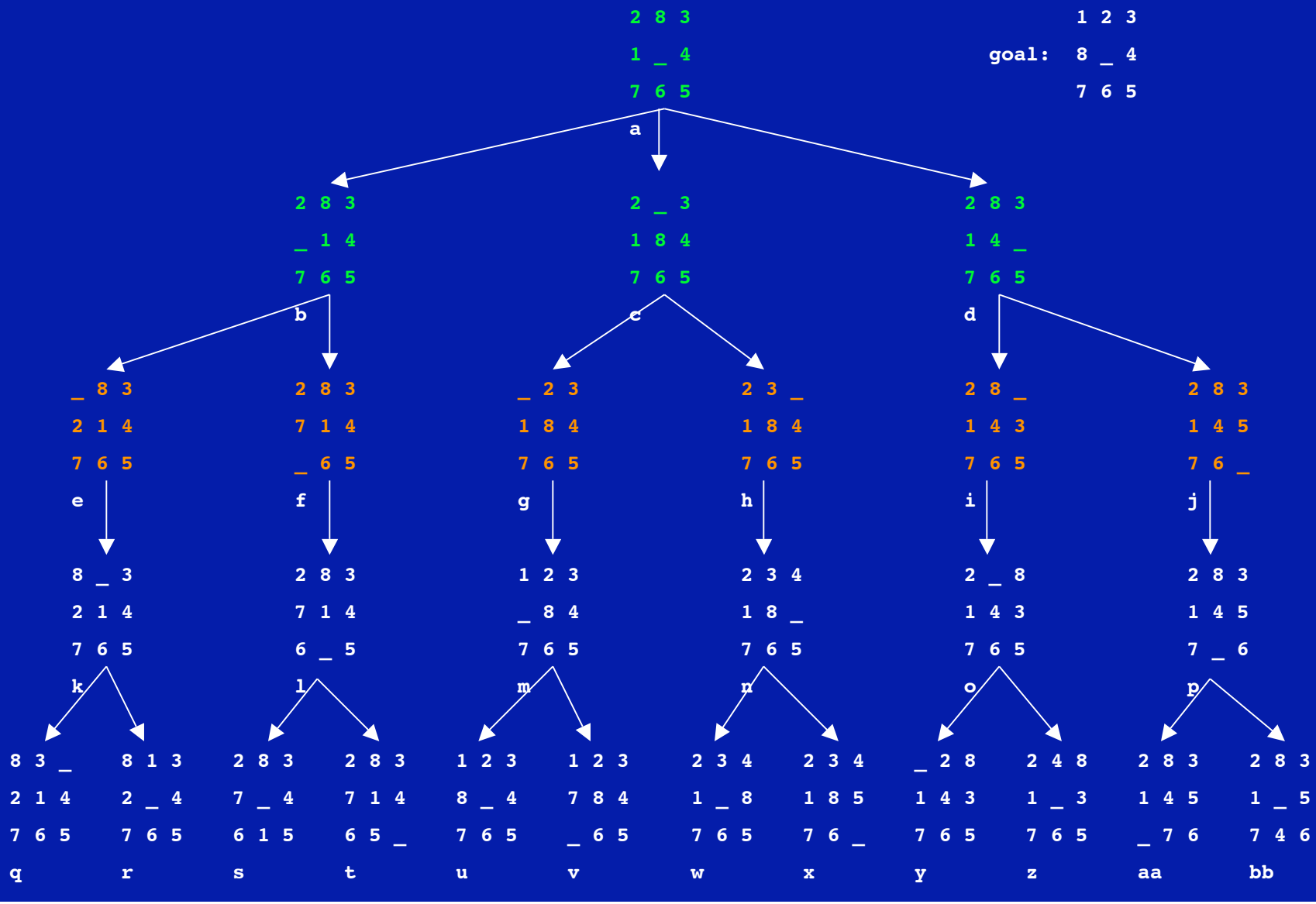
frontier: [d,e,f,g,h]



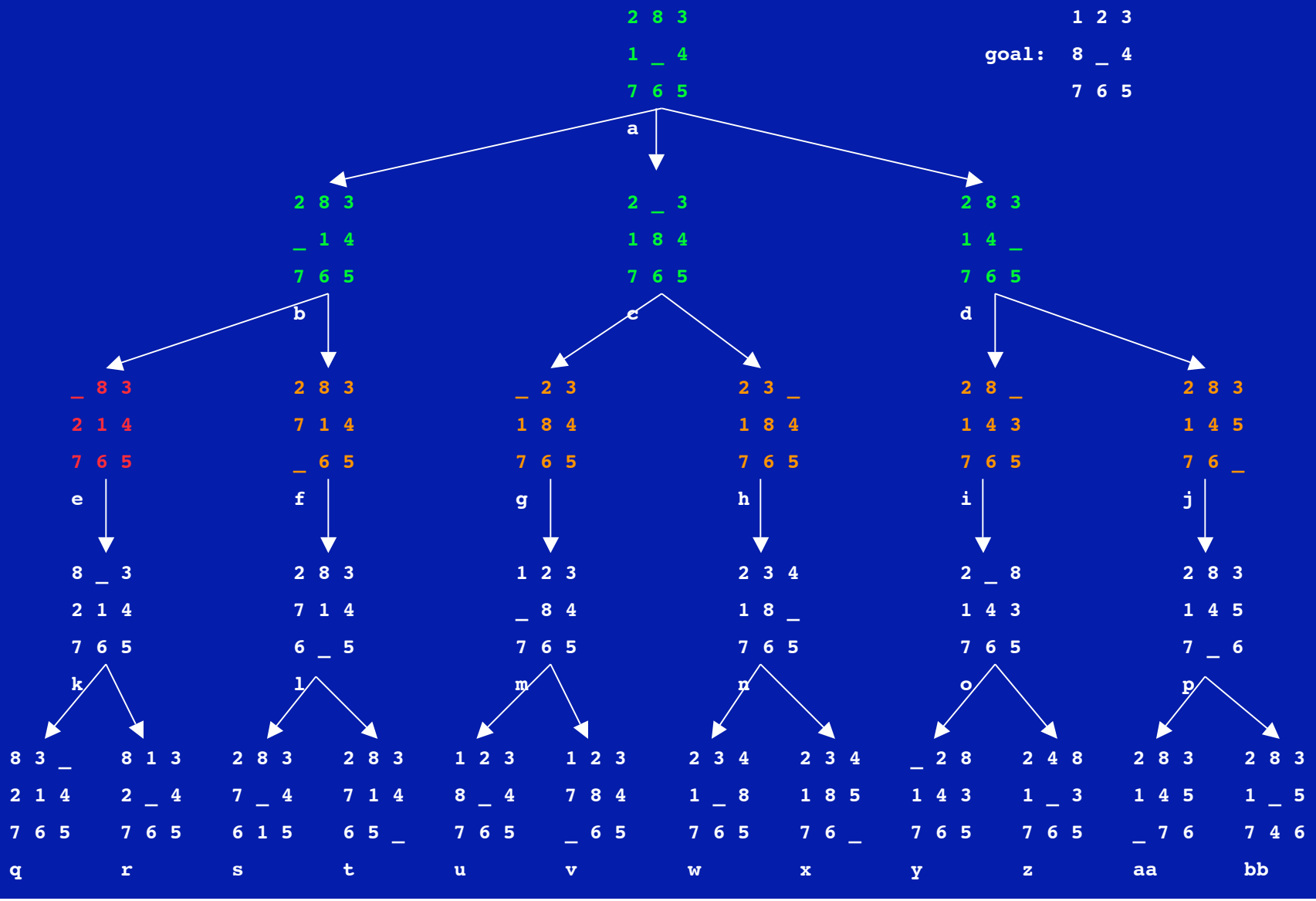
frontier: [e,f,g,h]



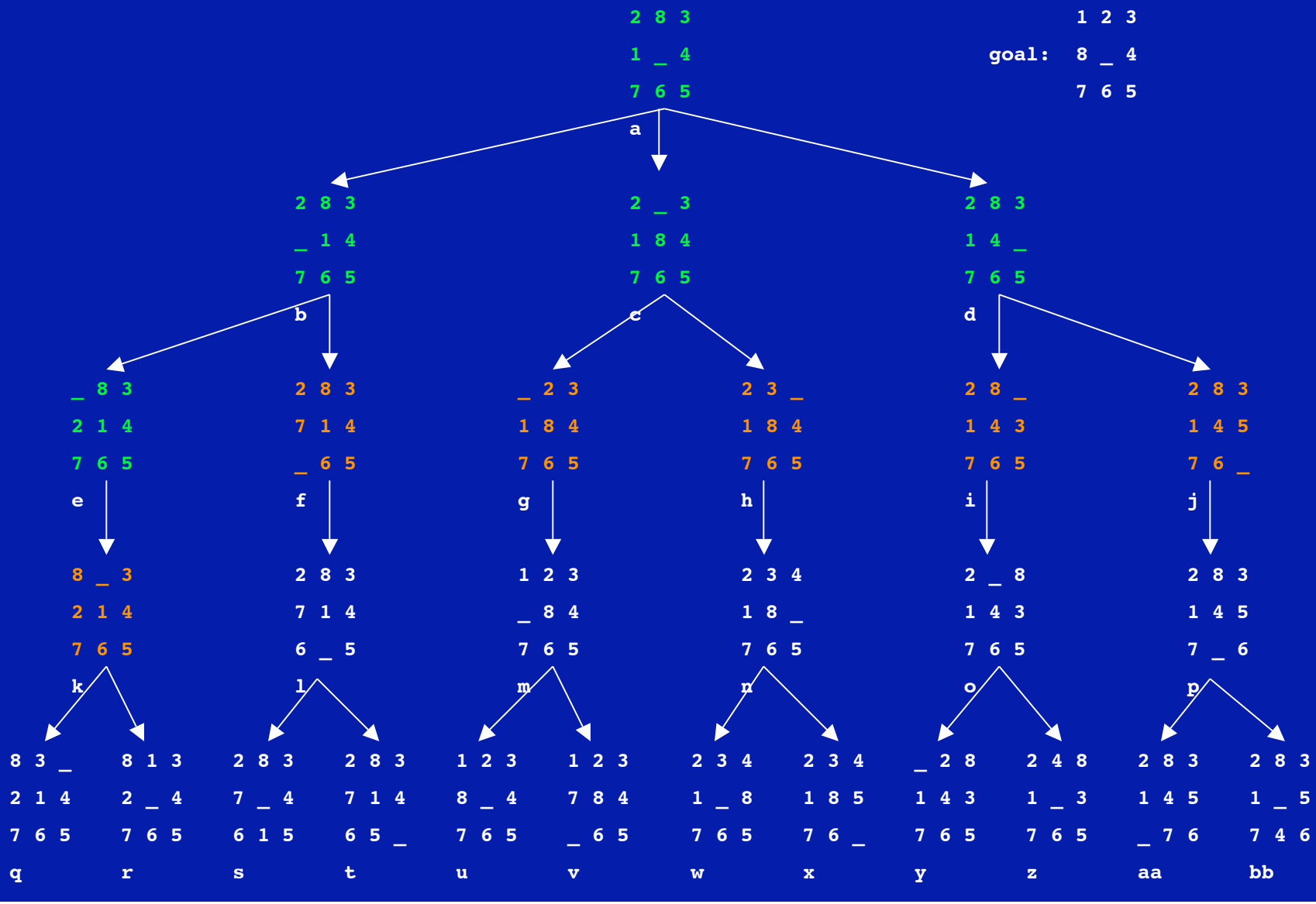
frontier: [e,f,g,h,i,j]



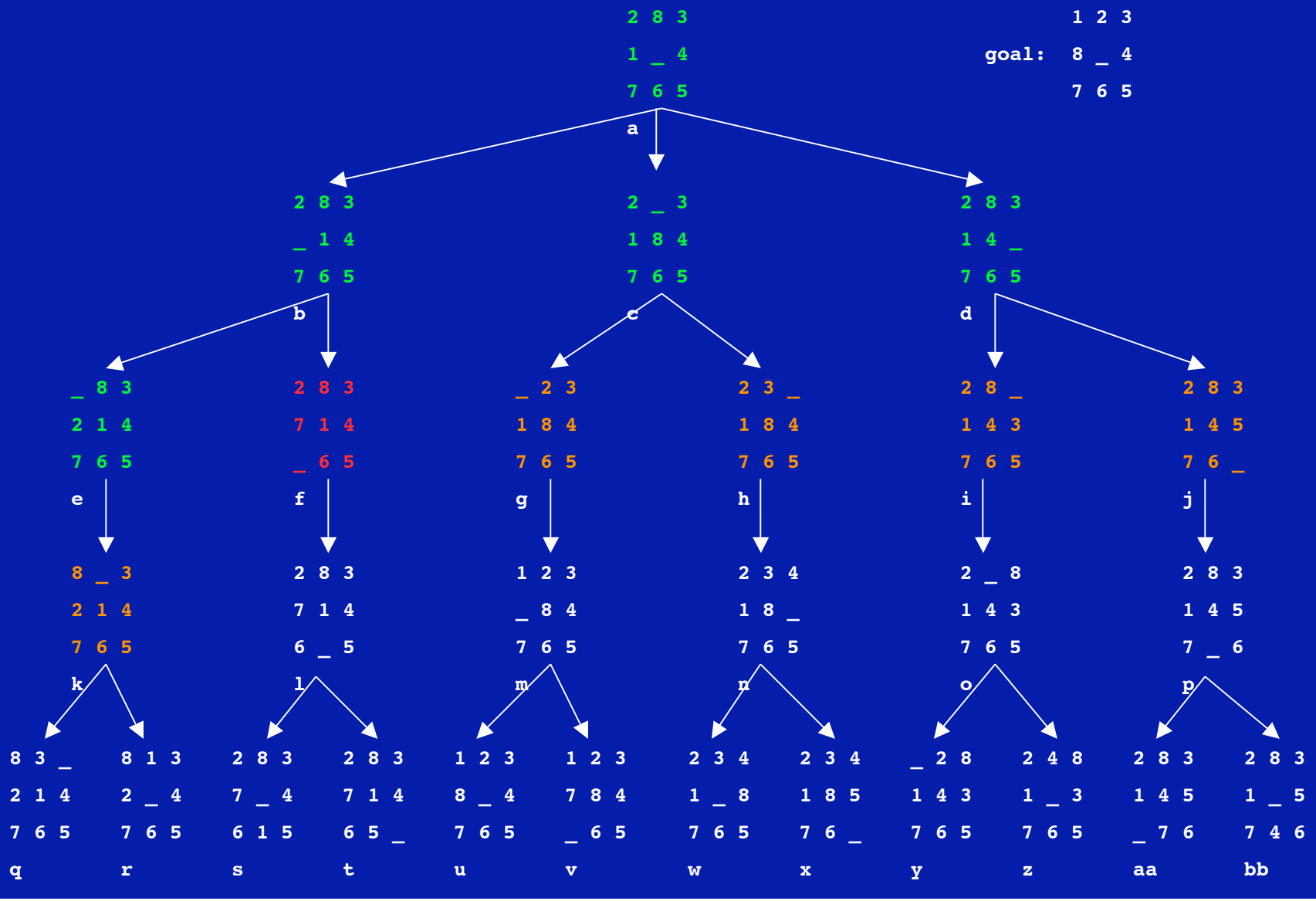
frontier: [f,g,h,i,j]



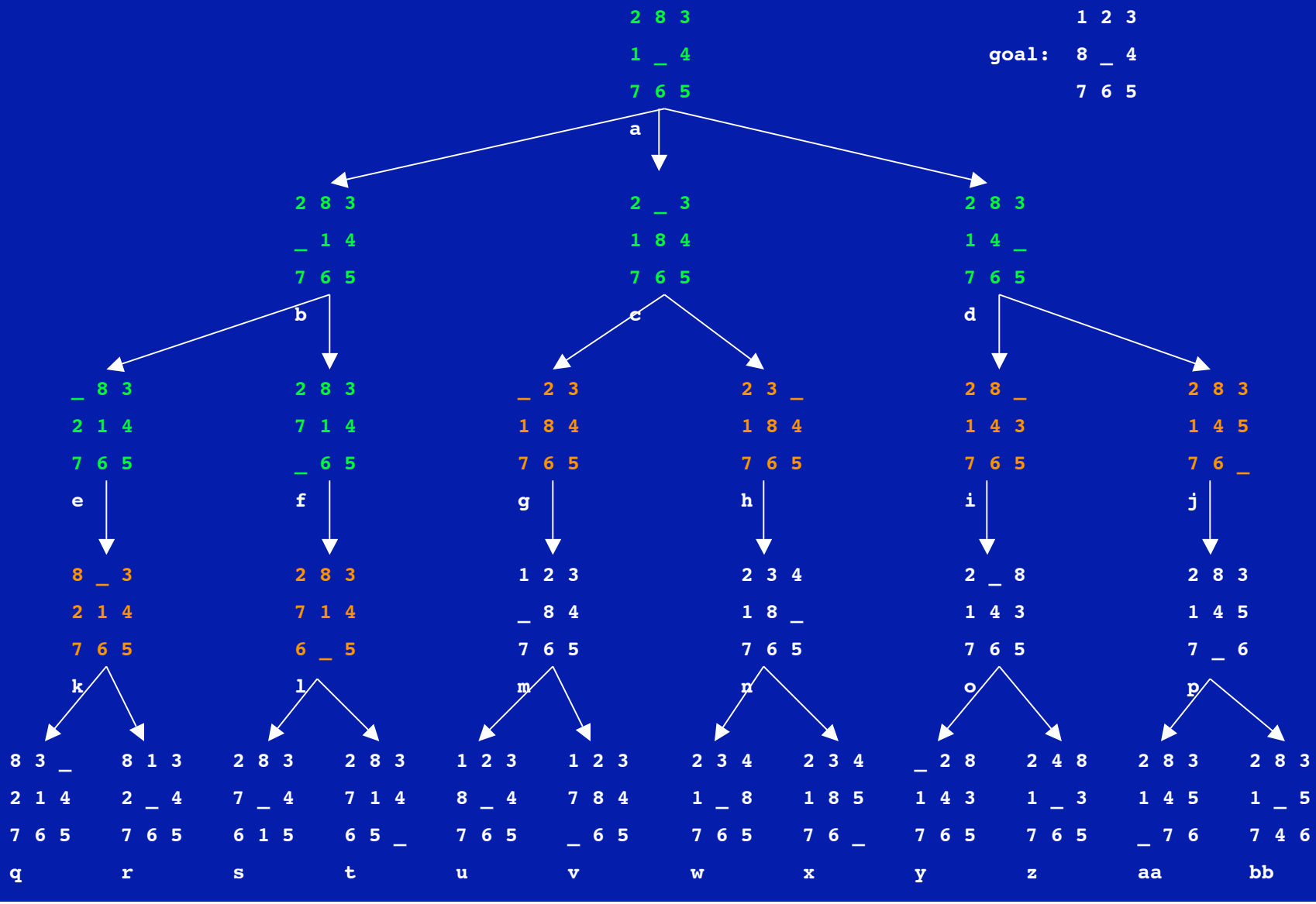
frontier: [f,g,h,i,j,k]



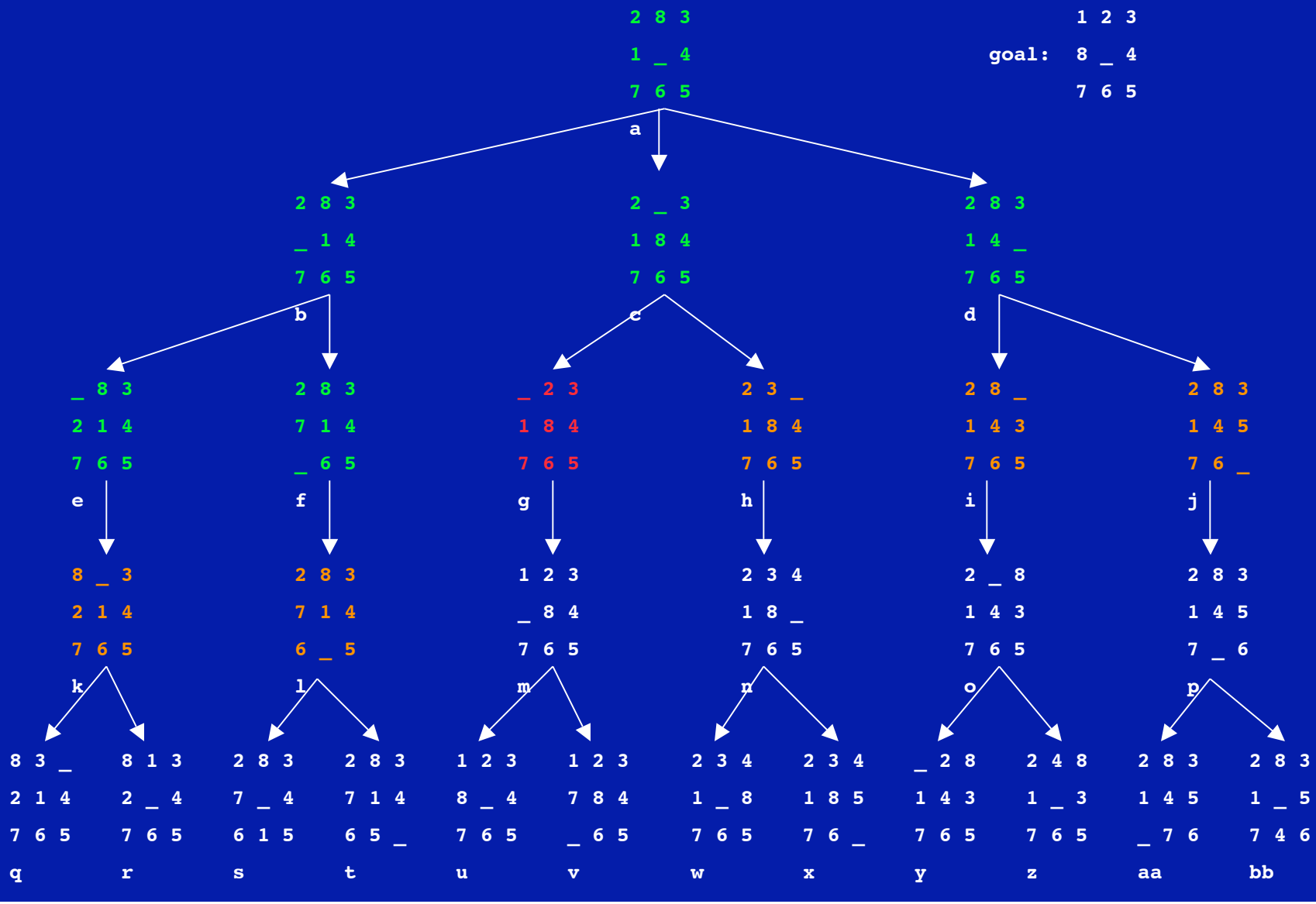
frontier: [g,h,i,j,k]



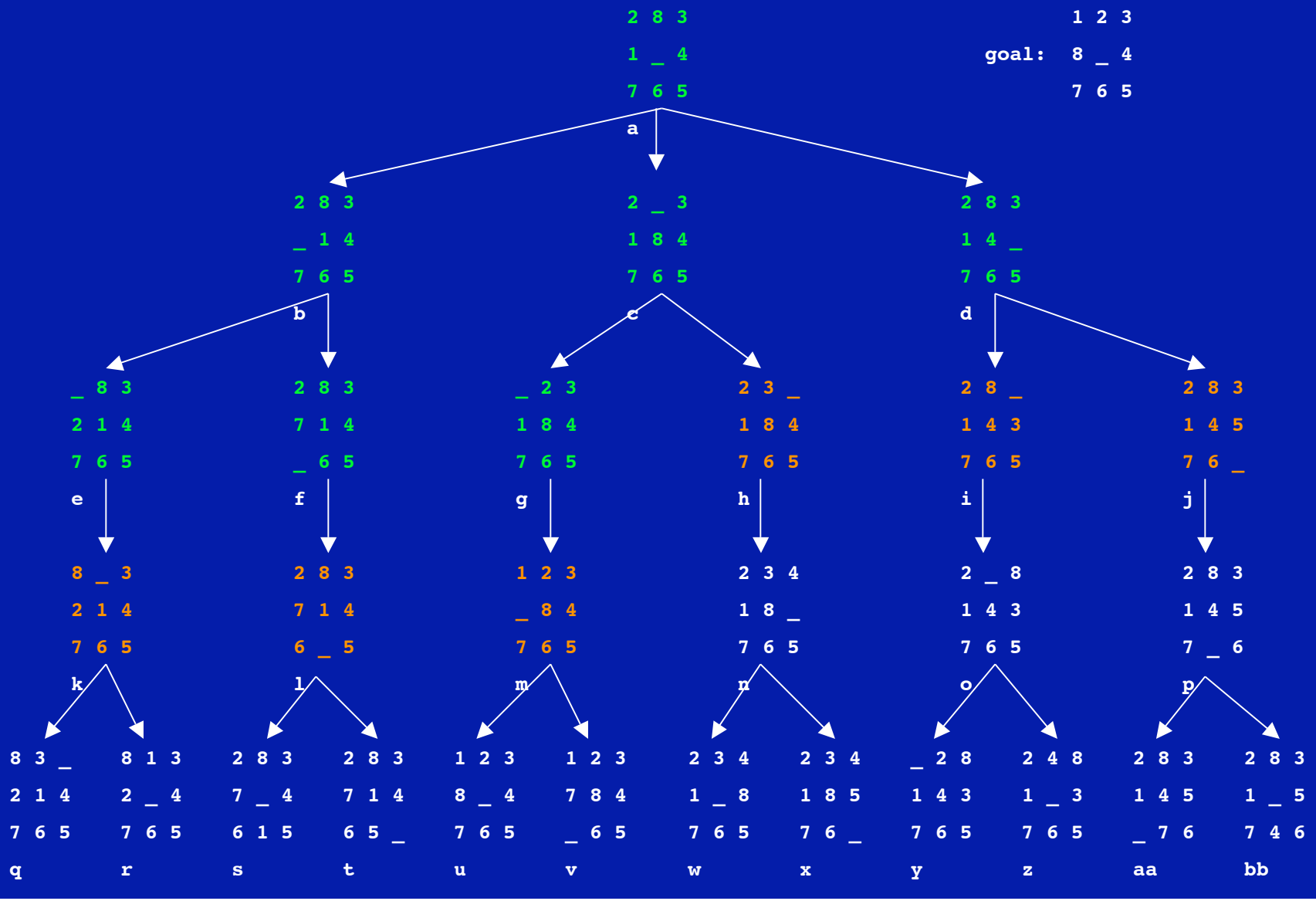
frontier: [g,h,i,j,k,l]



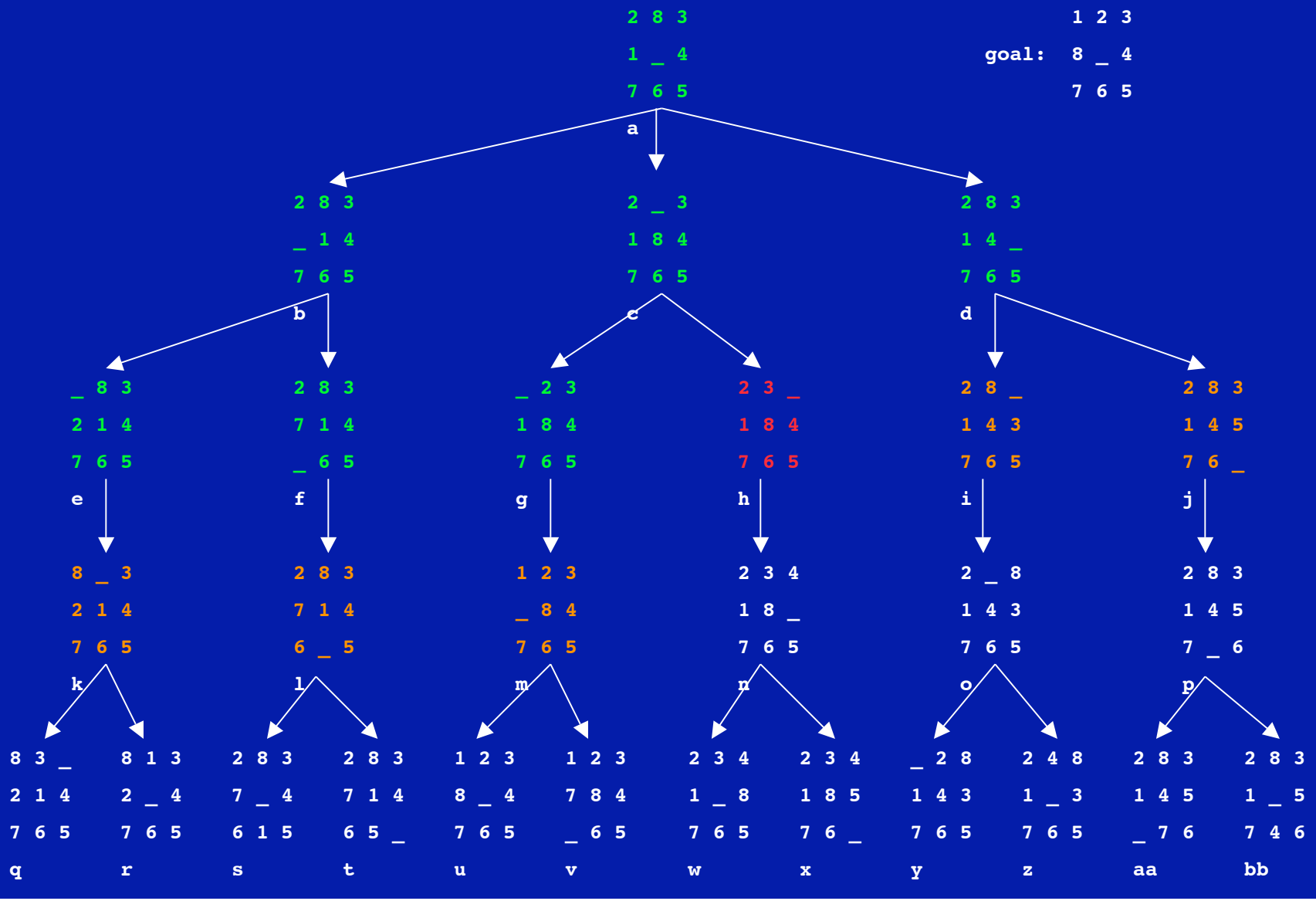
frontier: [h,i,j,k,l]



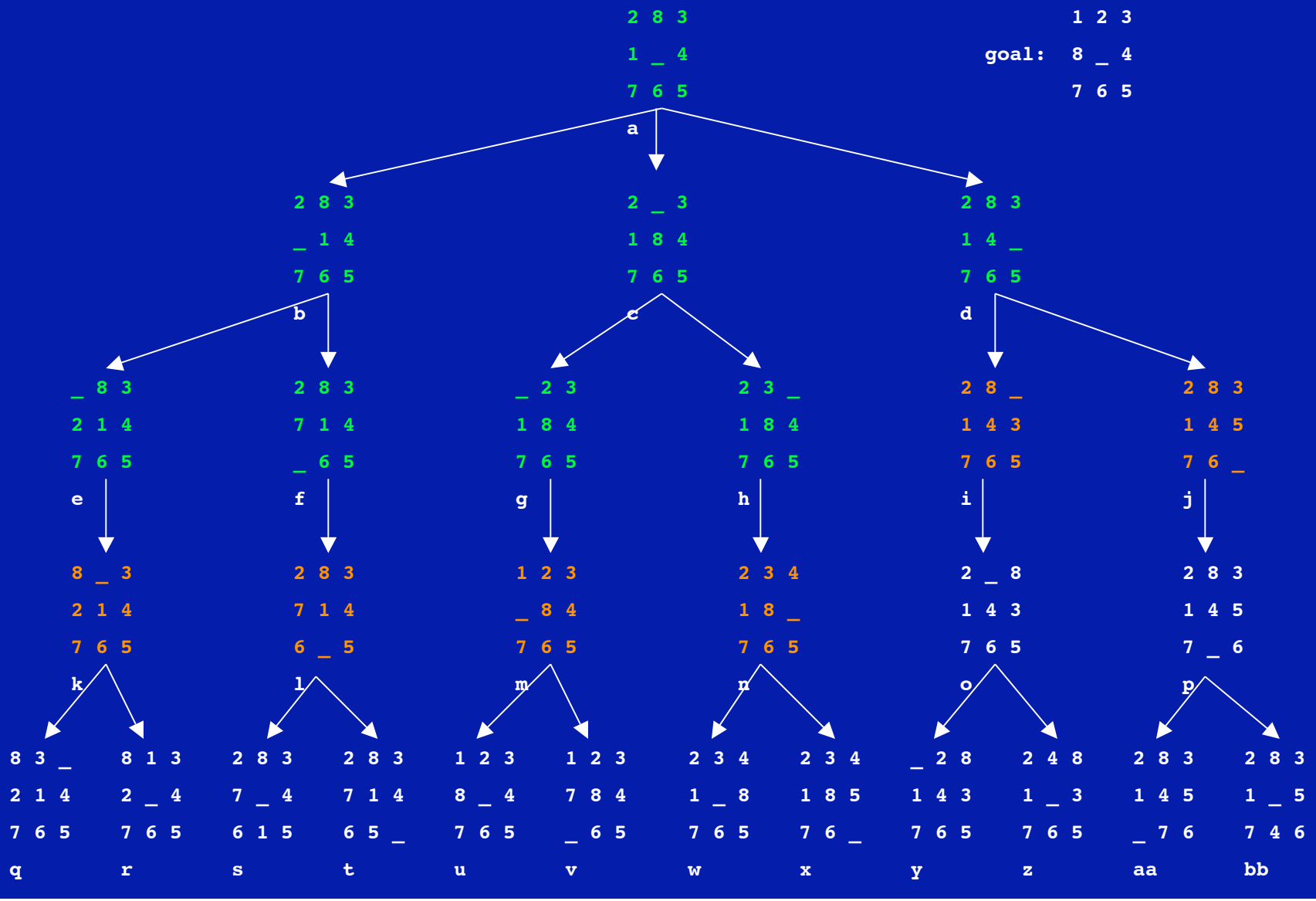
frontier: [h,i,j,k,l,m]



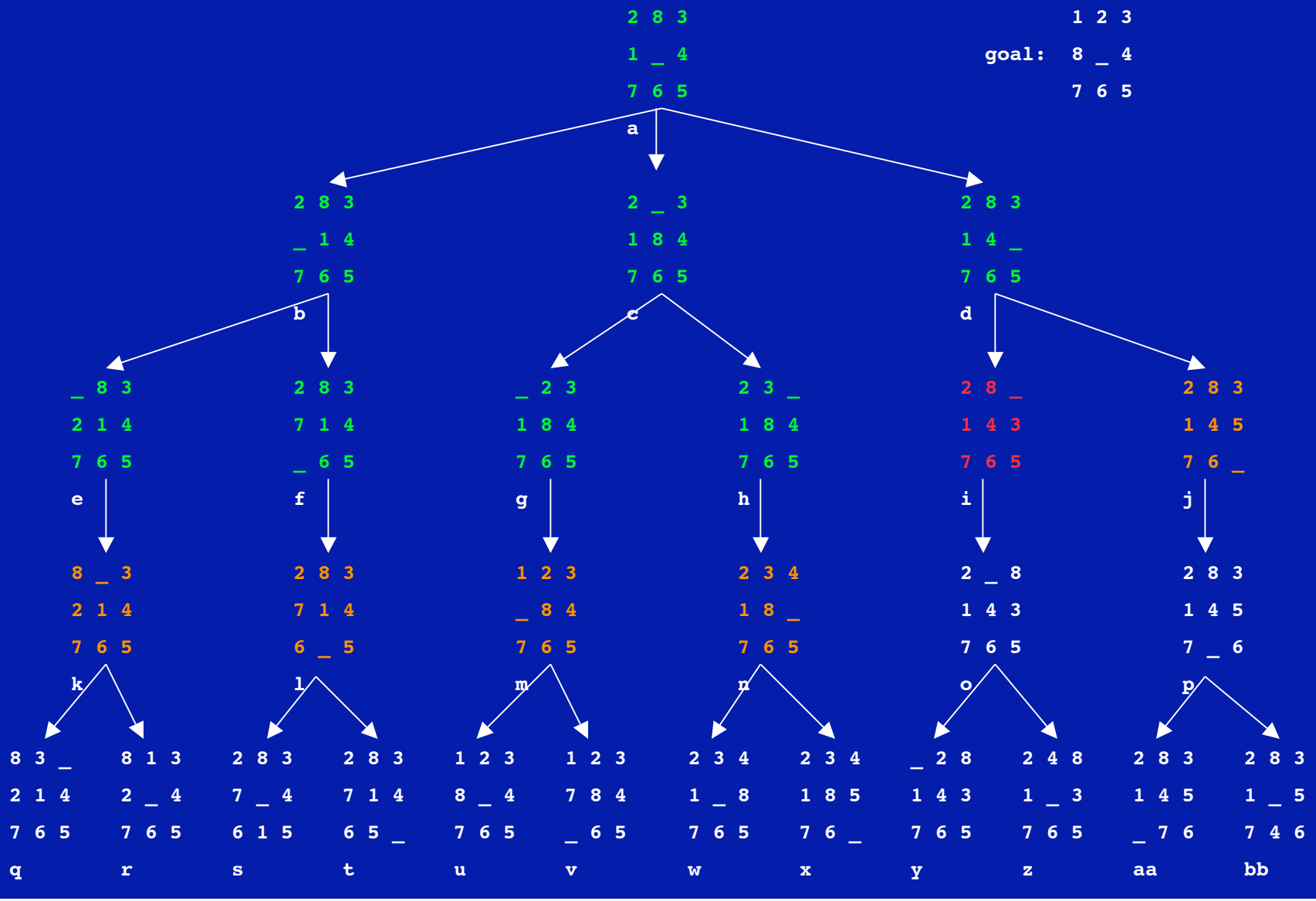
frontier: [i,j,k,l,m]



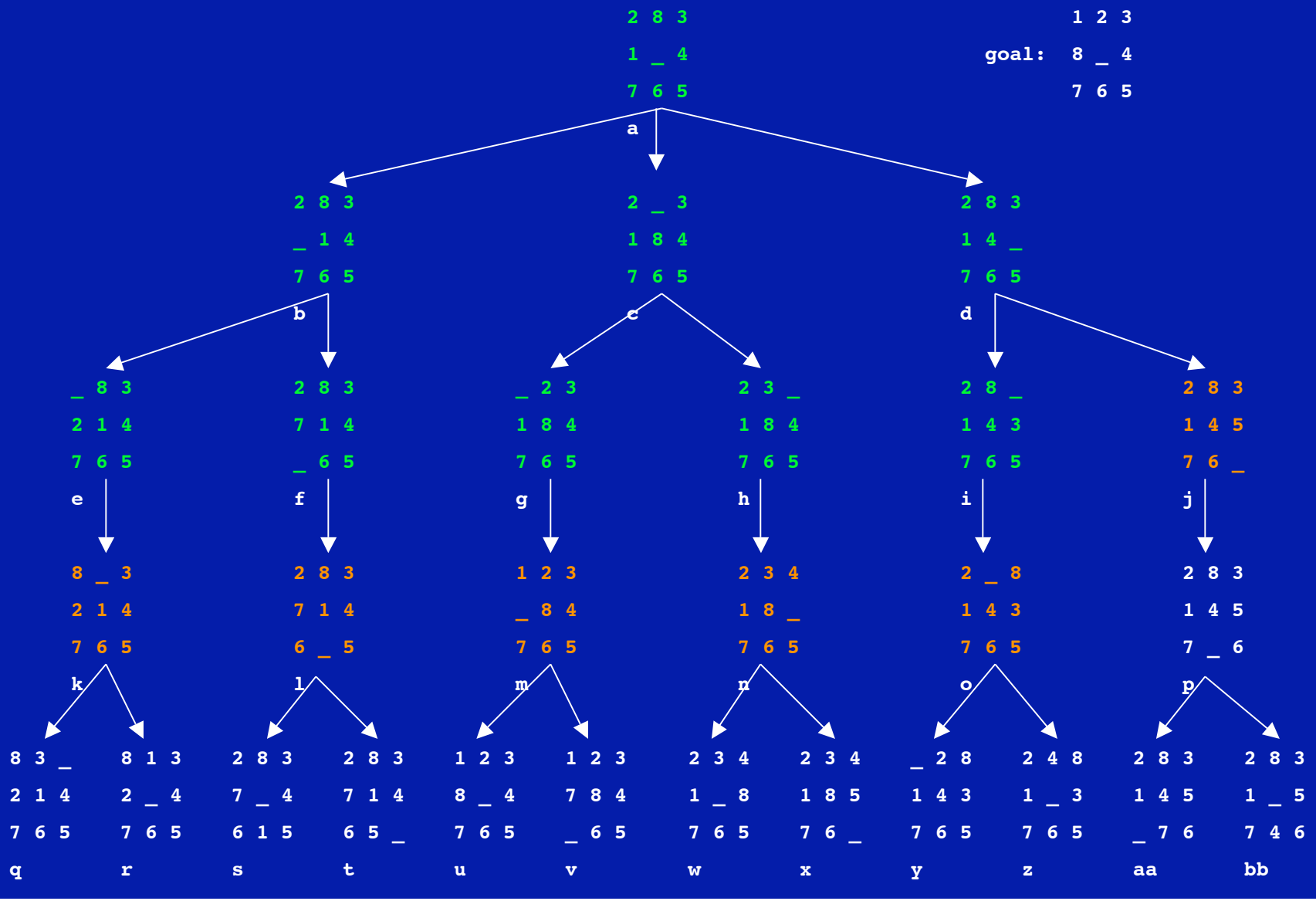
frontier: [i,j,k,l,m,n]



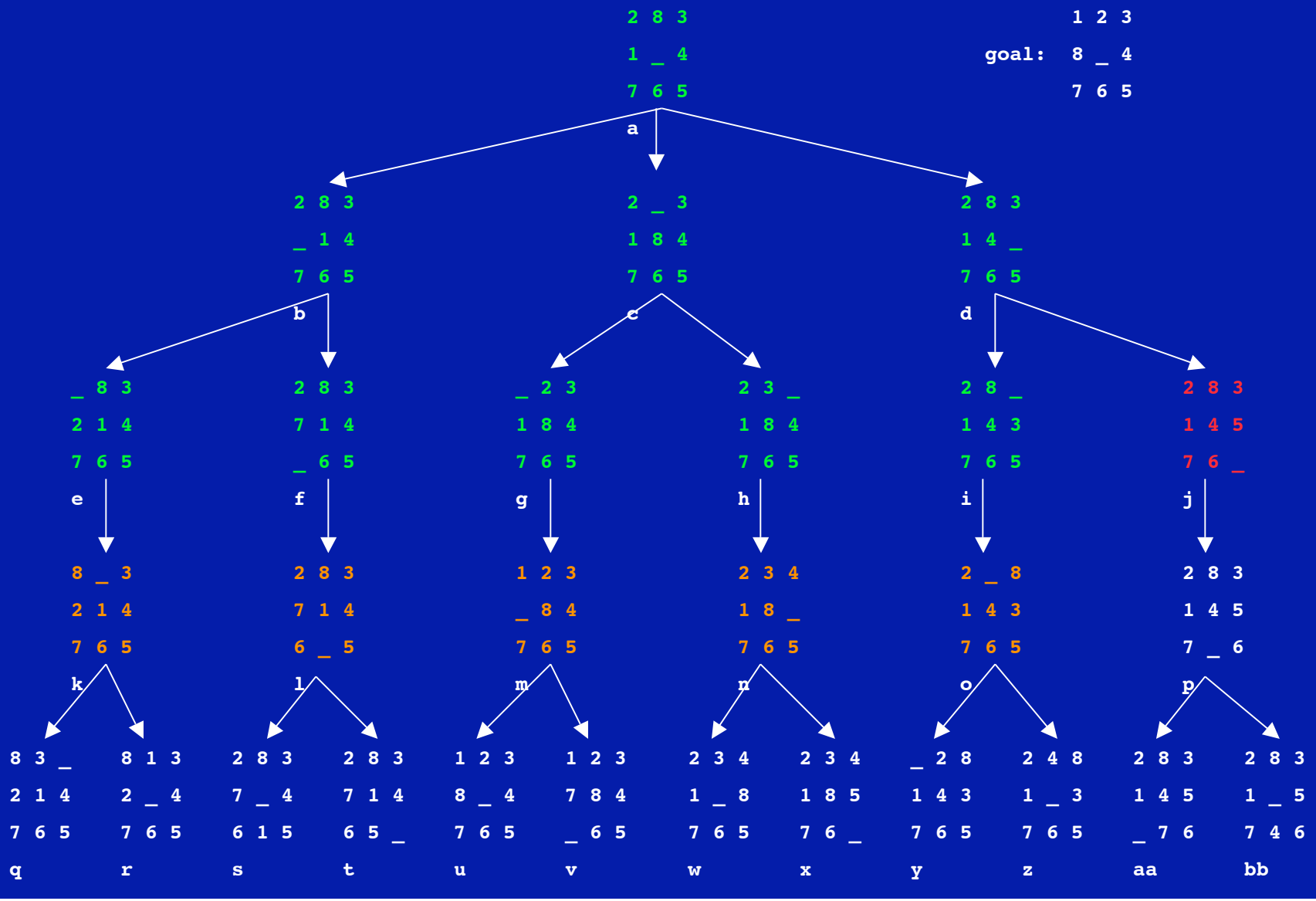
frontier: [j,k,l,m,n]



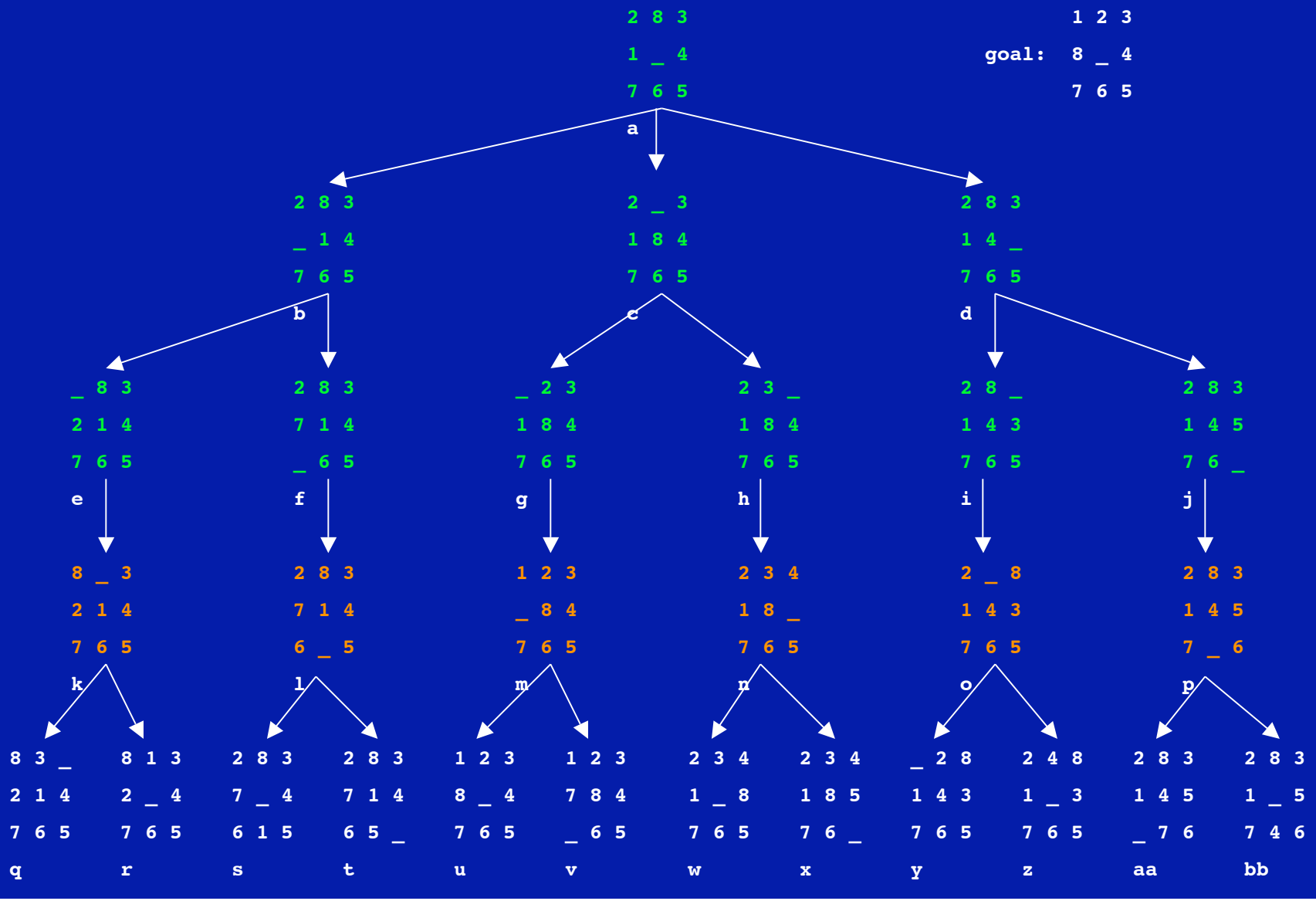
frontier: [j,k,l,m,n,o]



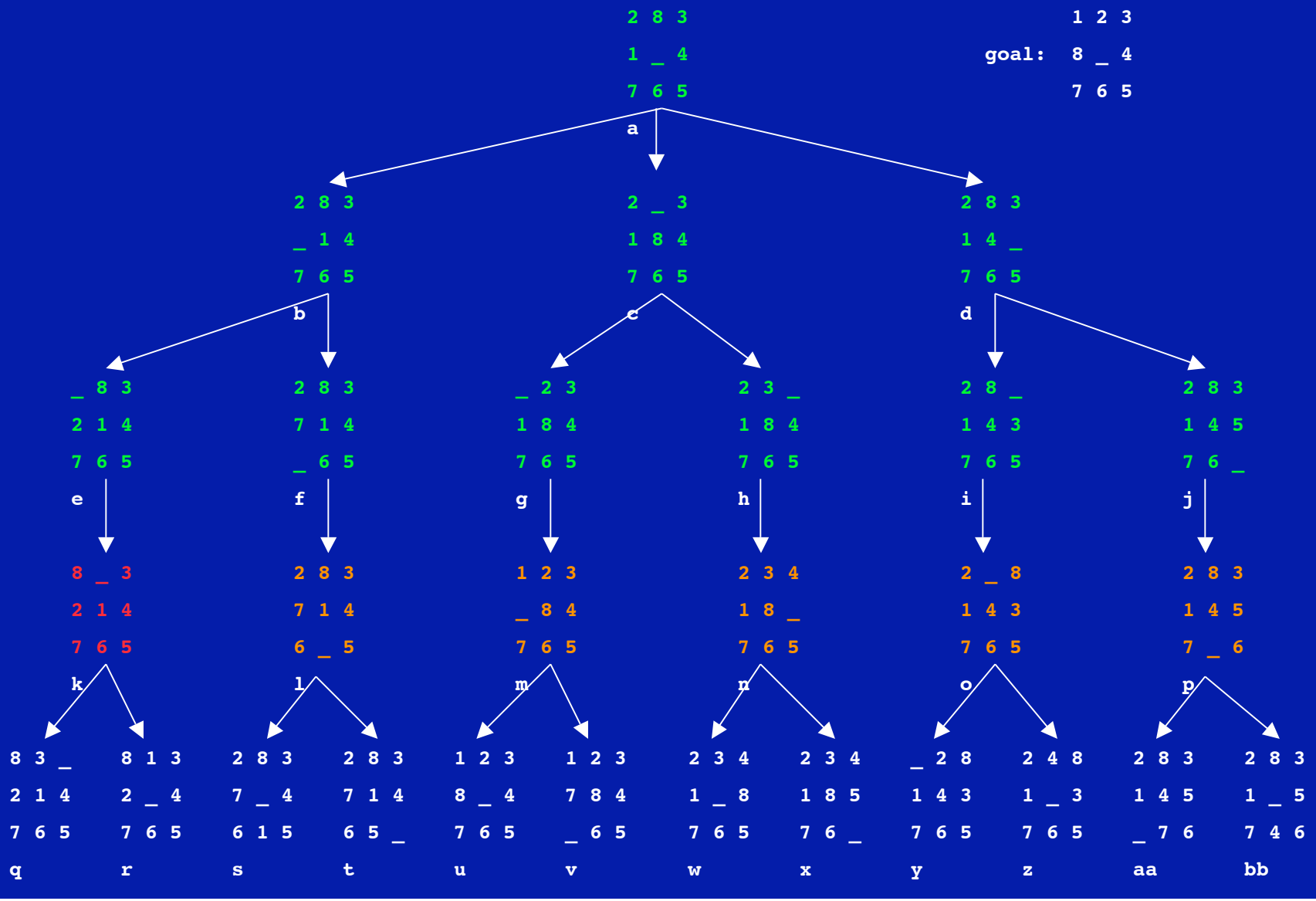
frontier: [k,l,m,n,o]



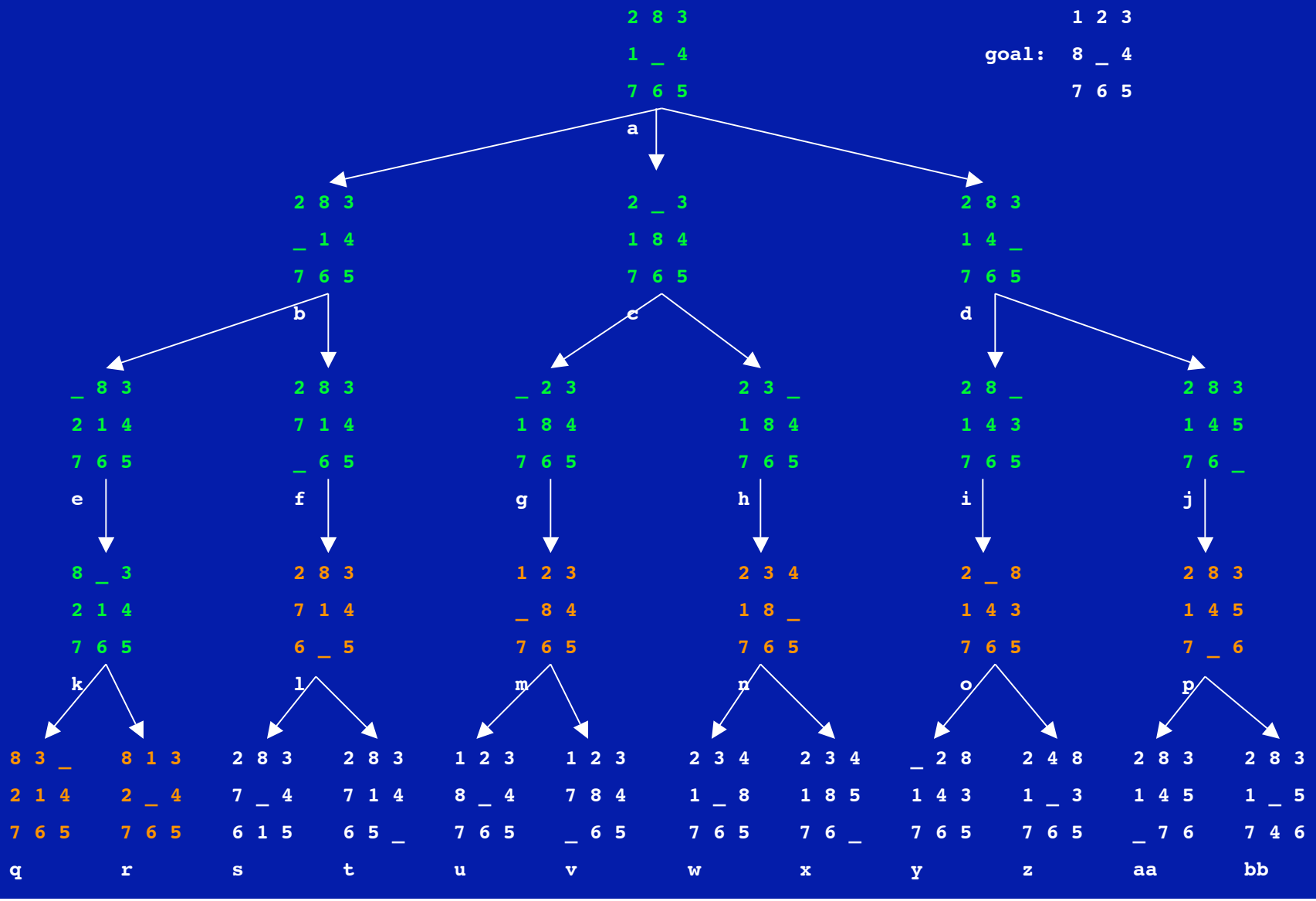
frontier: [k,l,m,n,o,p]



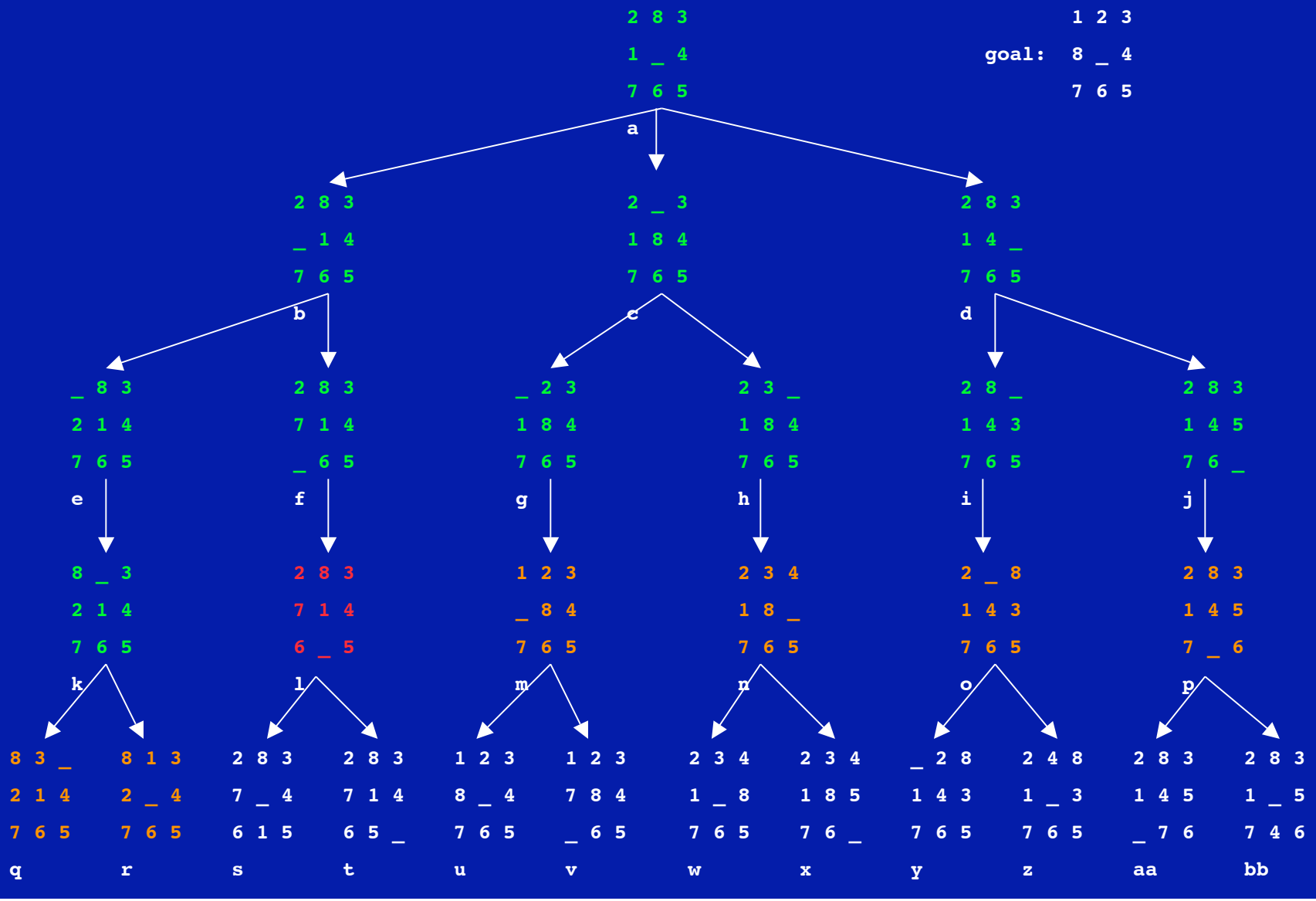
frontier: [l,m,n,o,p]



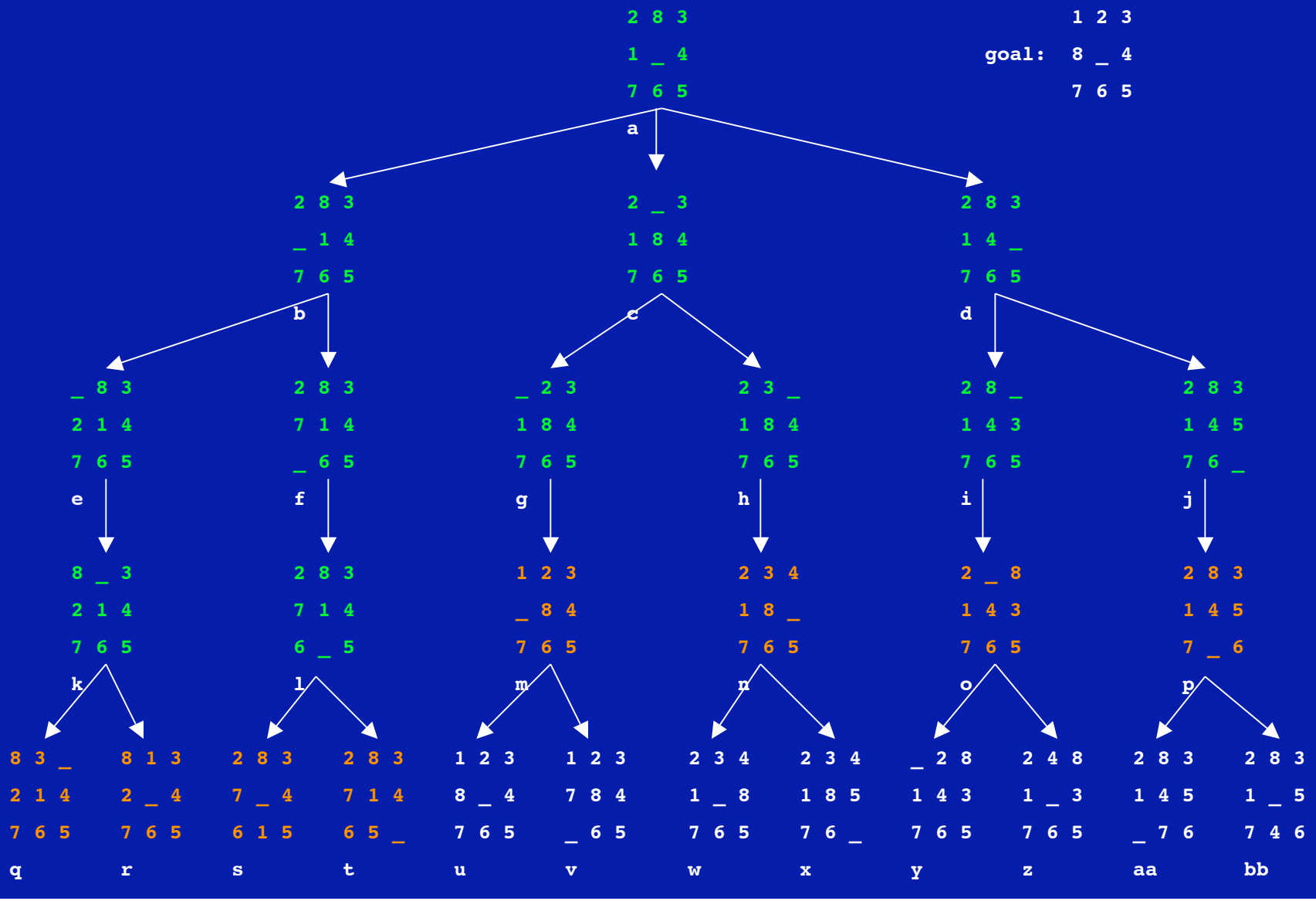
frontier: [l,m,n,o,p,q,r]



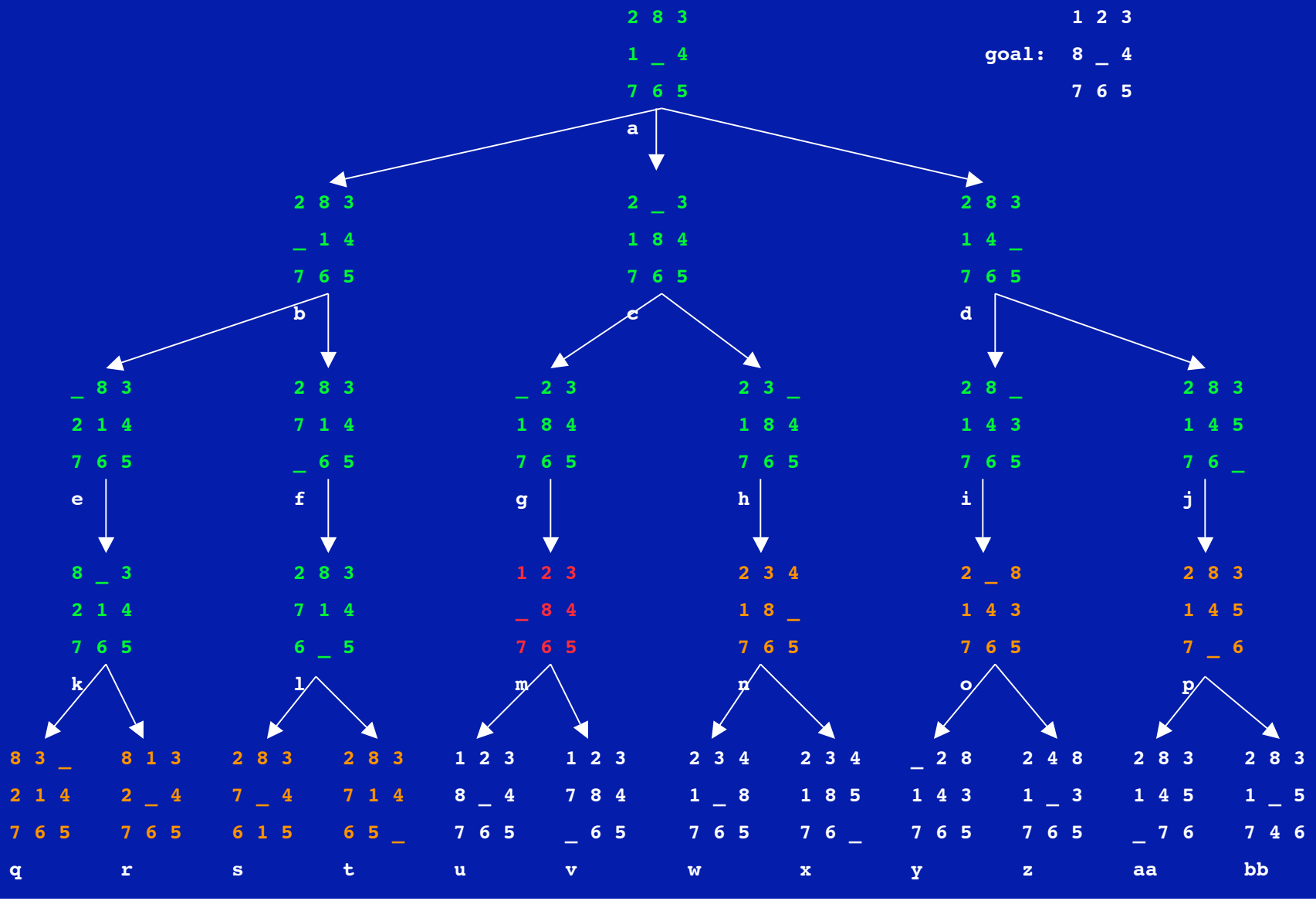
frontier: [m,n,o,p,q,r]



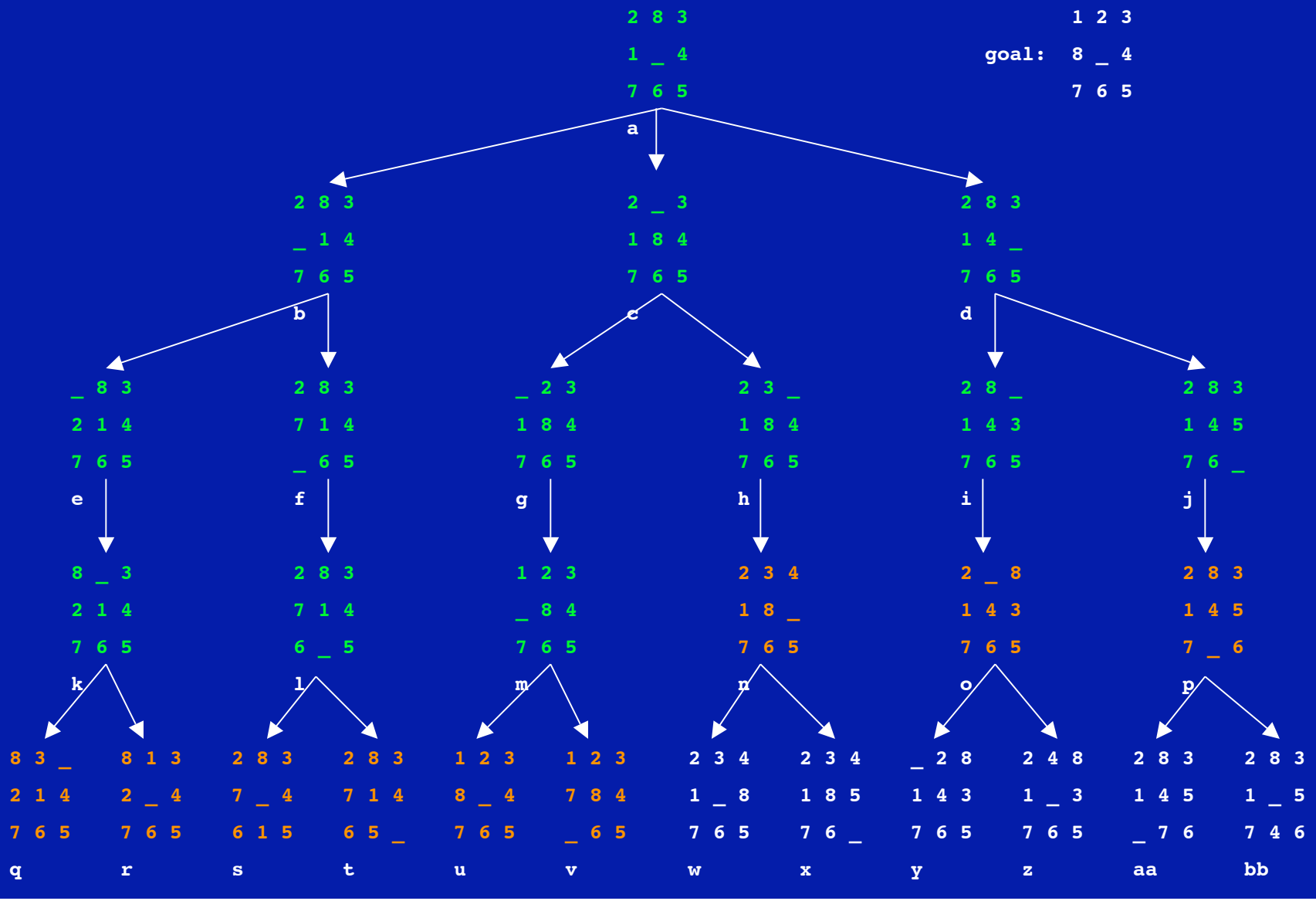
frontier: [m,n,o,p,q,r,s,t]



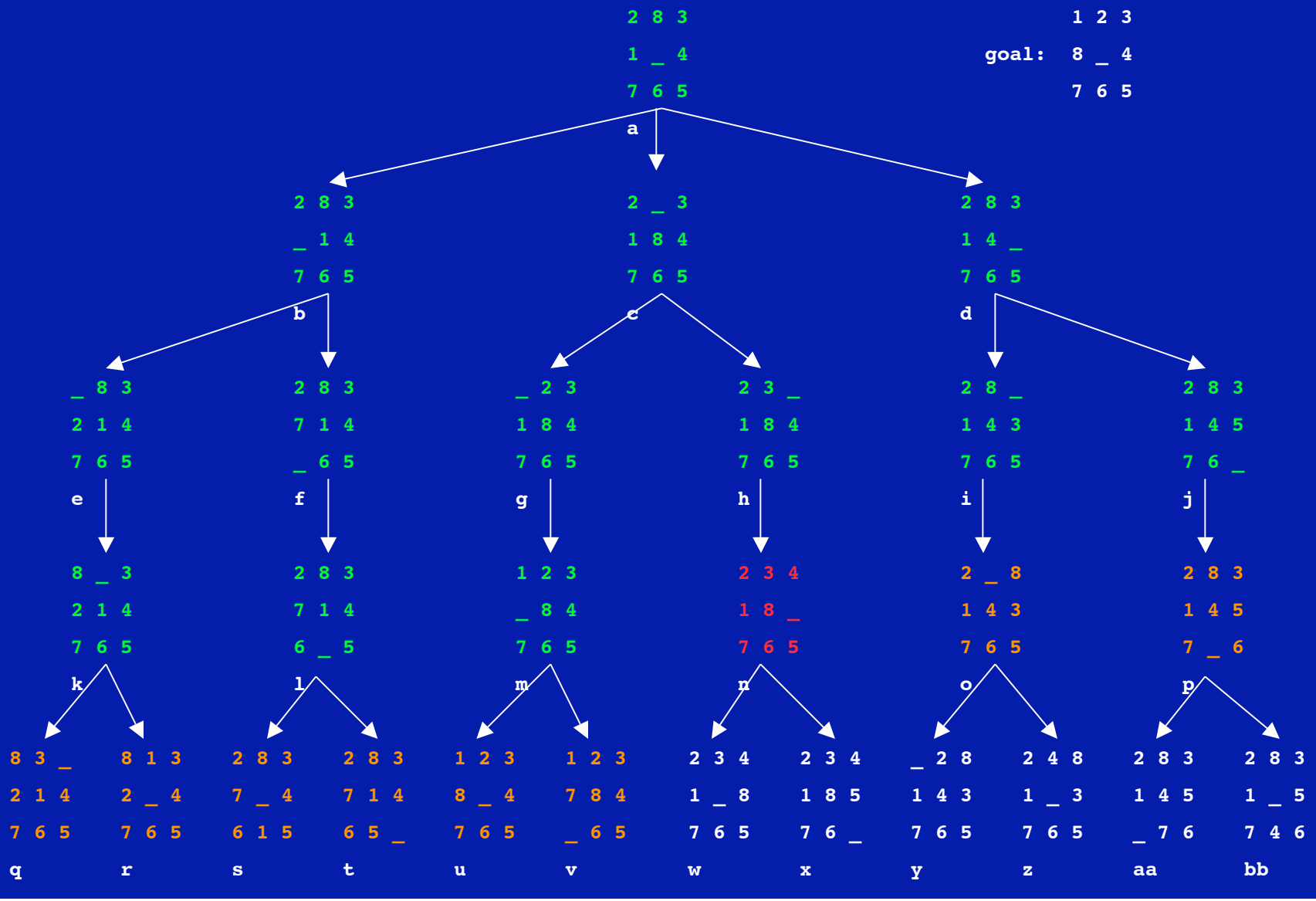
frontier: [n,o,p,q,r,s,t]



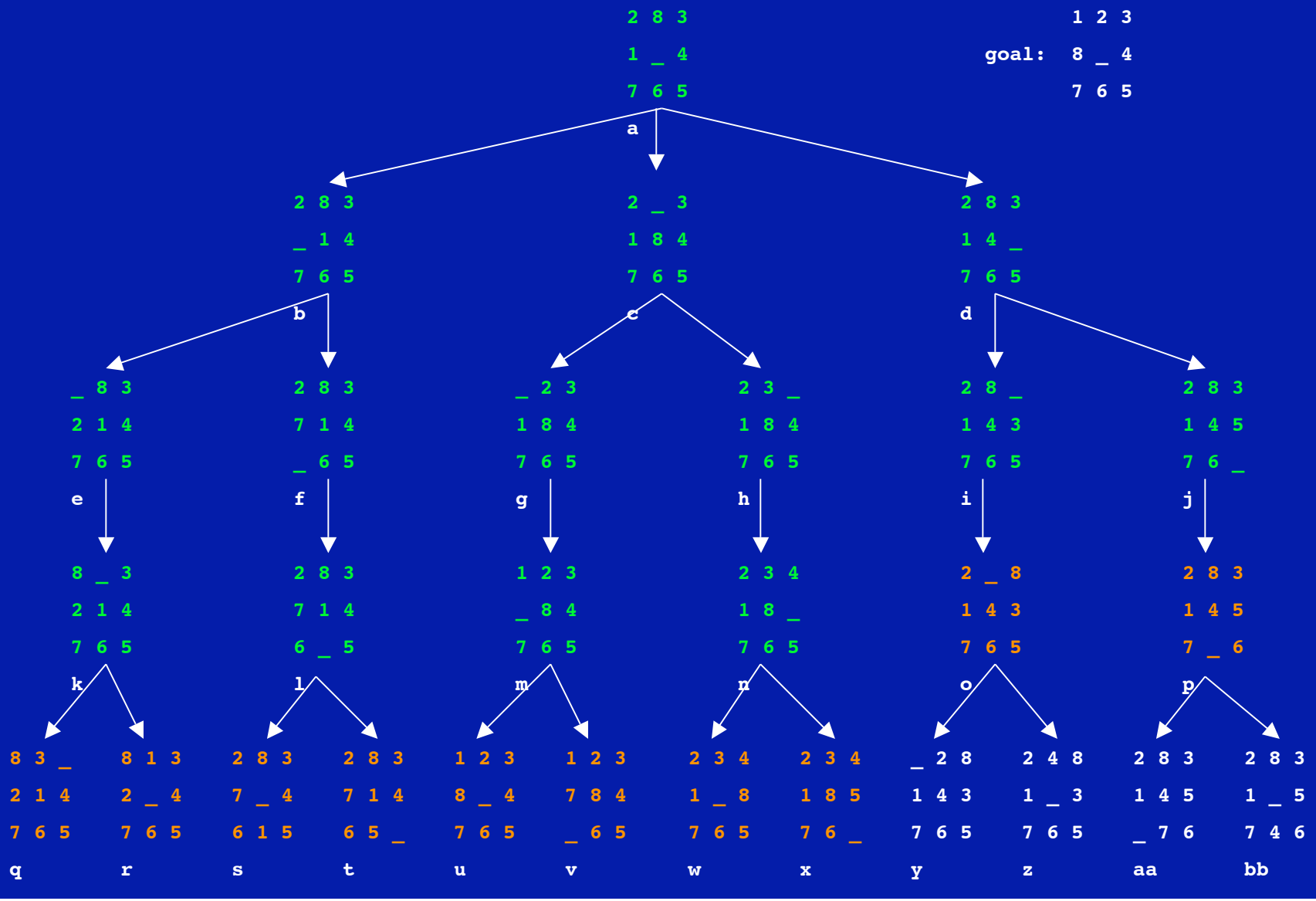
frontier: [n,o,p,q,r,s,t,u,v]



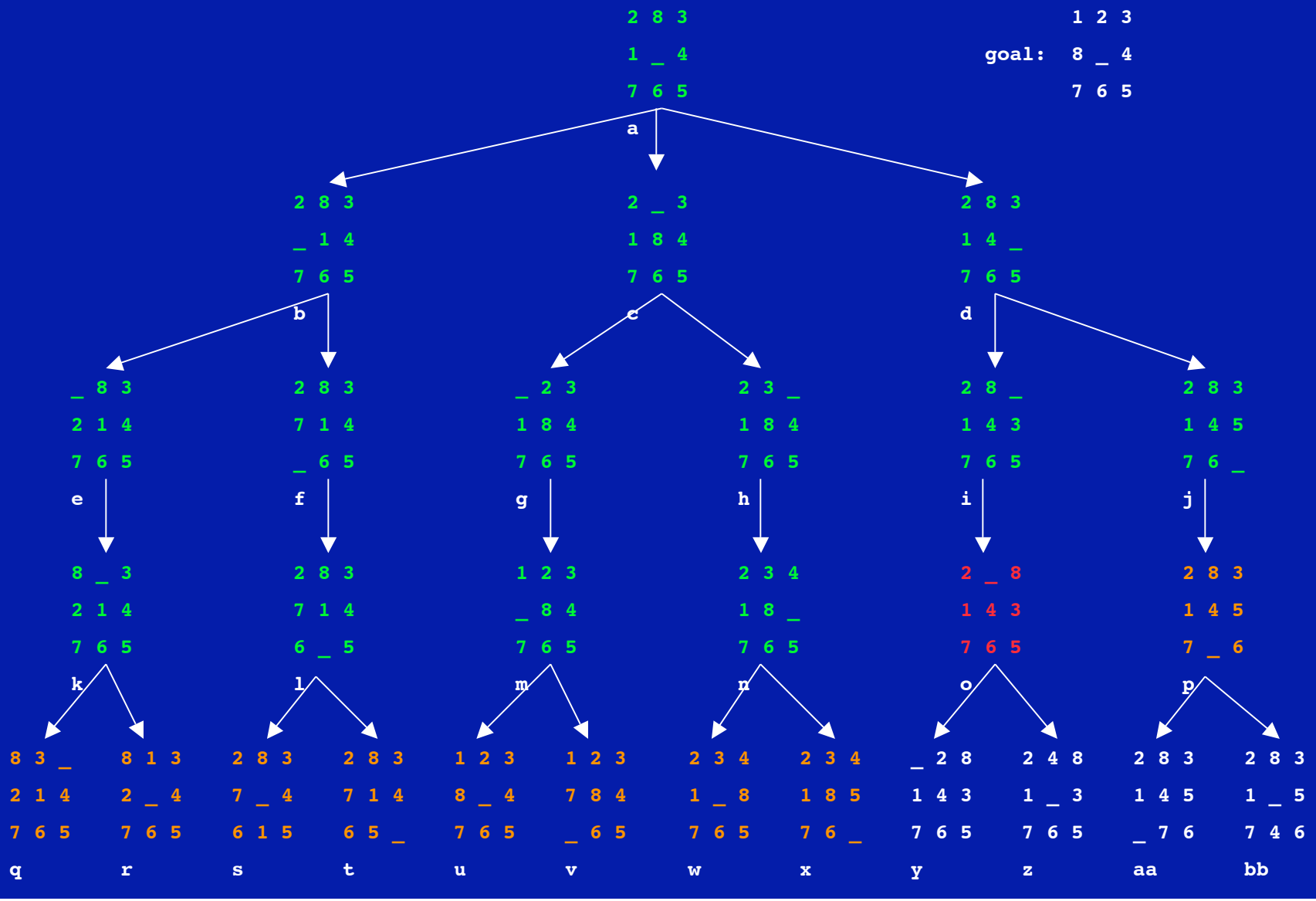
frontier: [o,p,q,r,s,t,u,v]



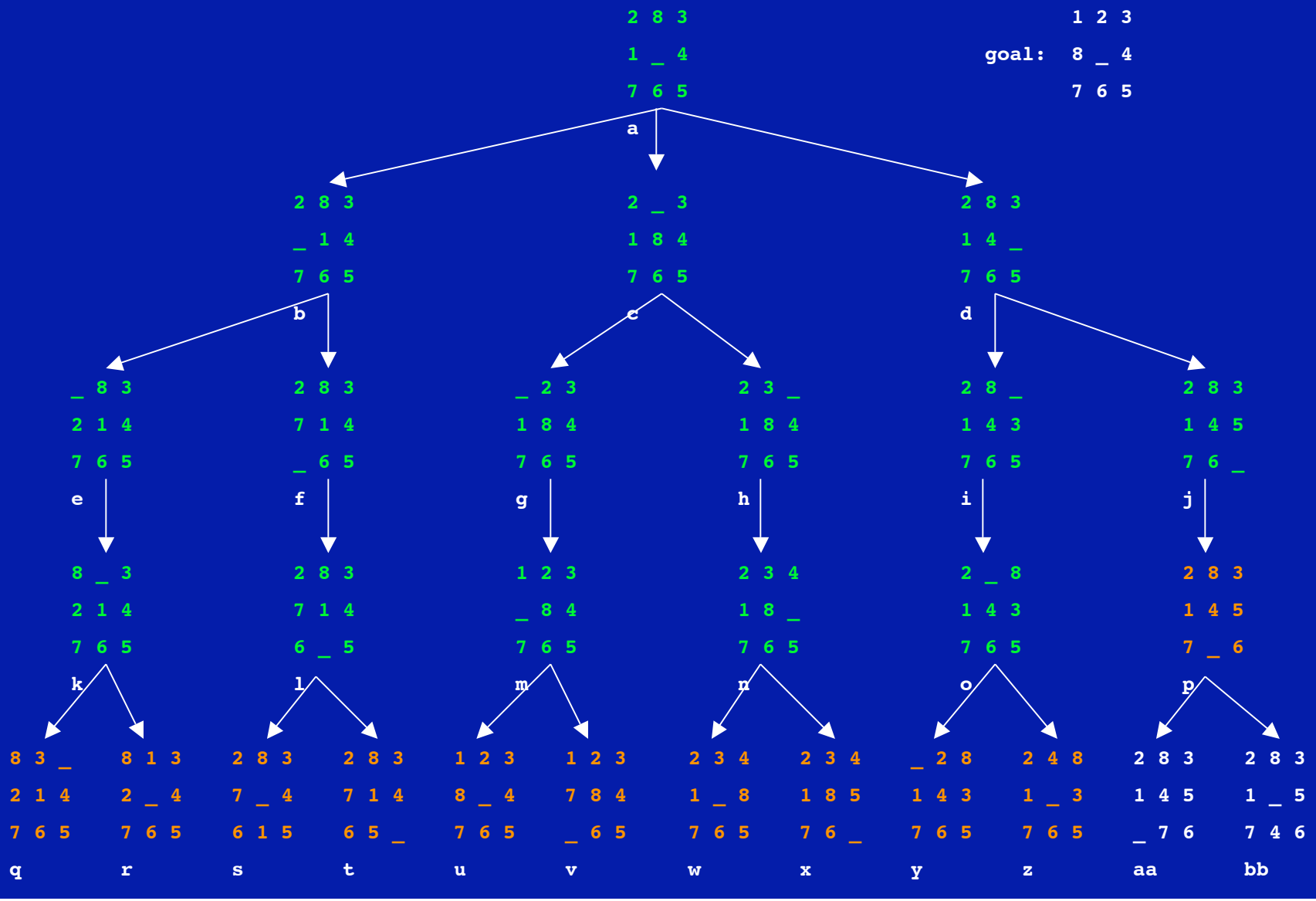
frontier: [o,p,q,r,s,t,u,v,w,x]



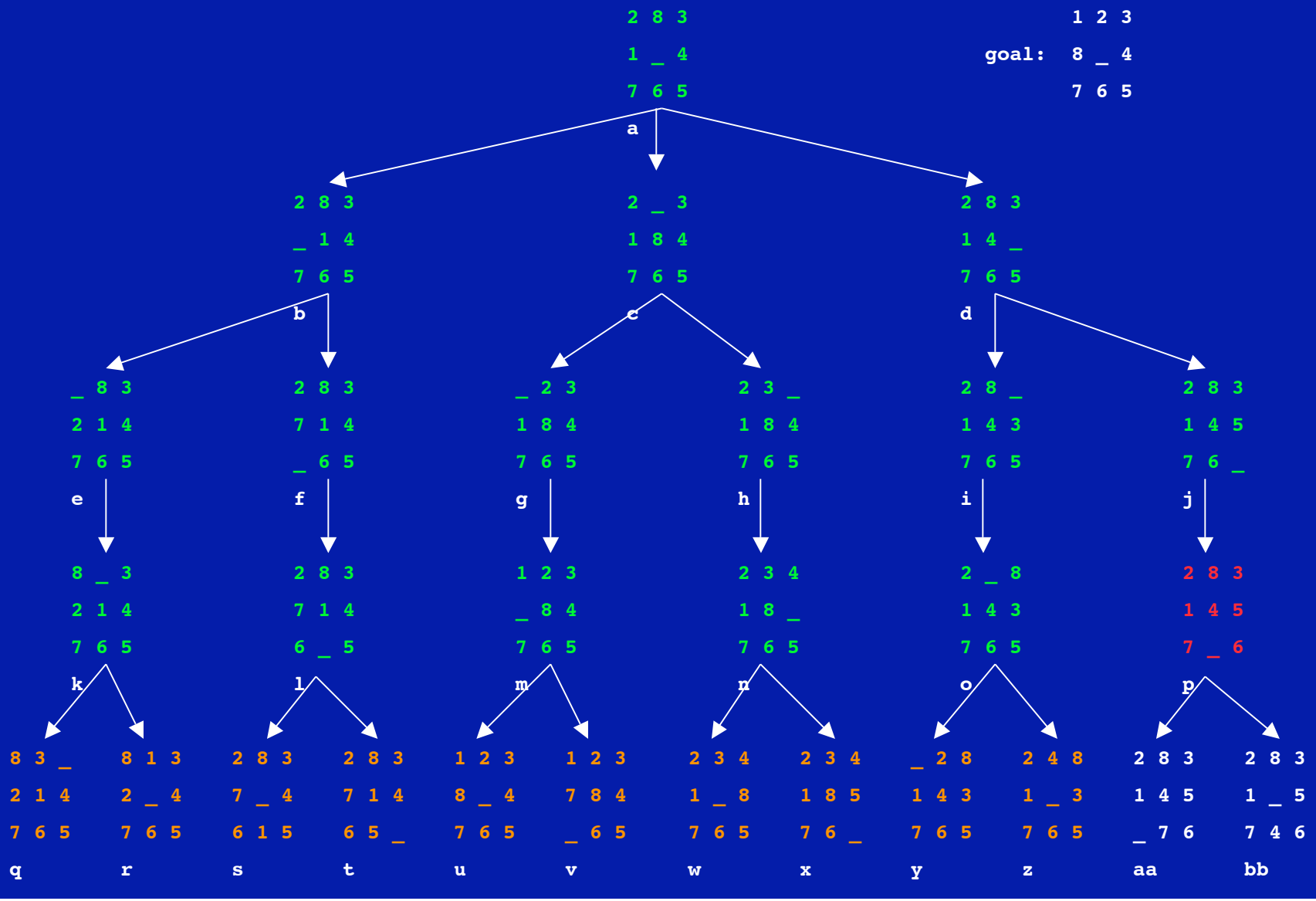
frontier: [p,q,r,s,t,u,v,w,x]



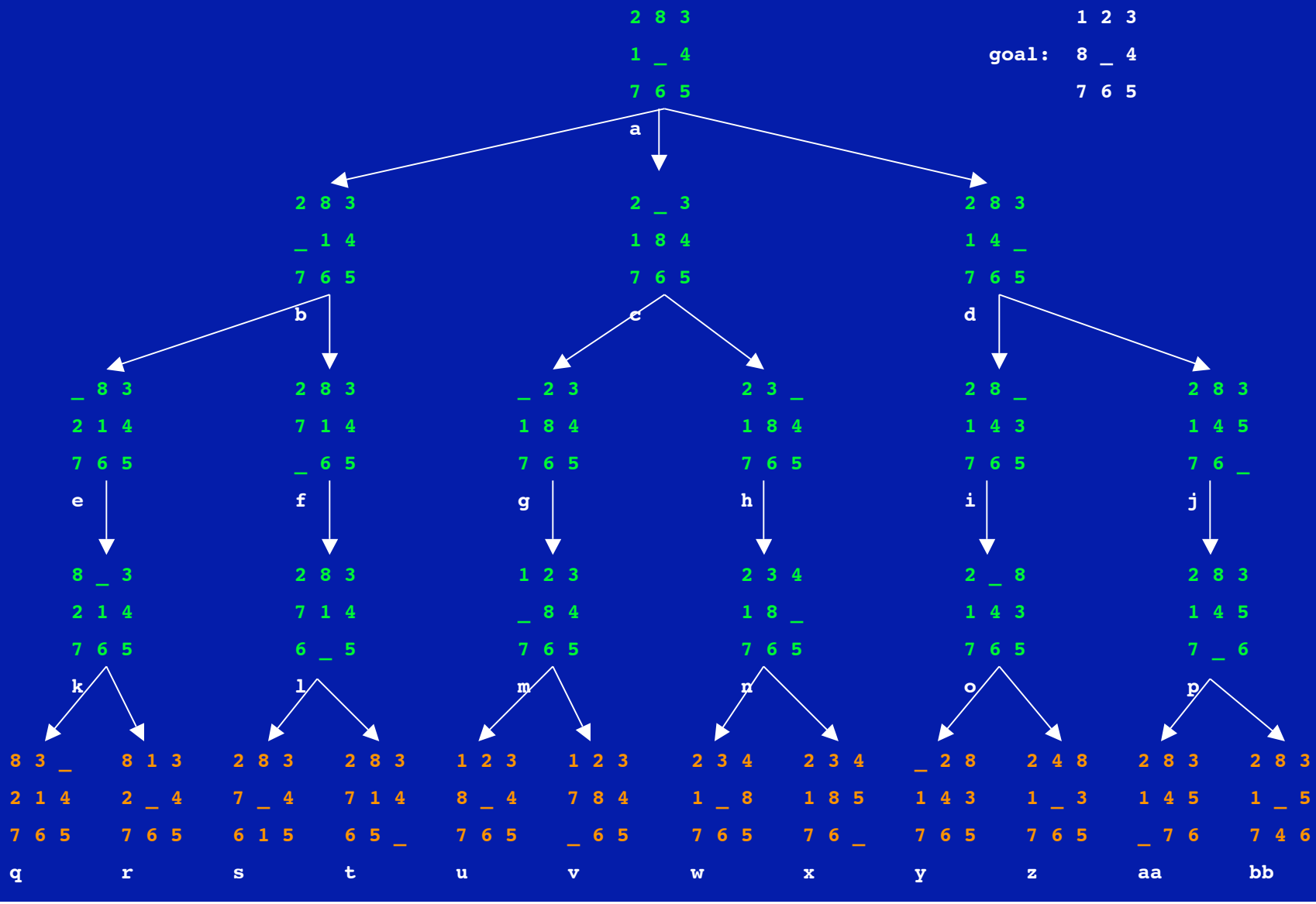
frontier: [p,q,r,s,t,u,v,w,x,y,z]



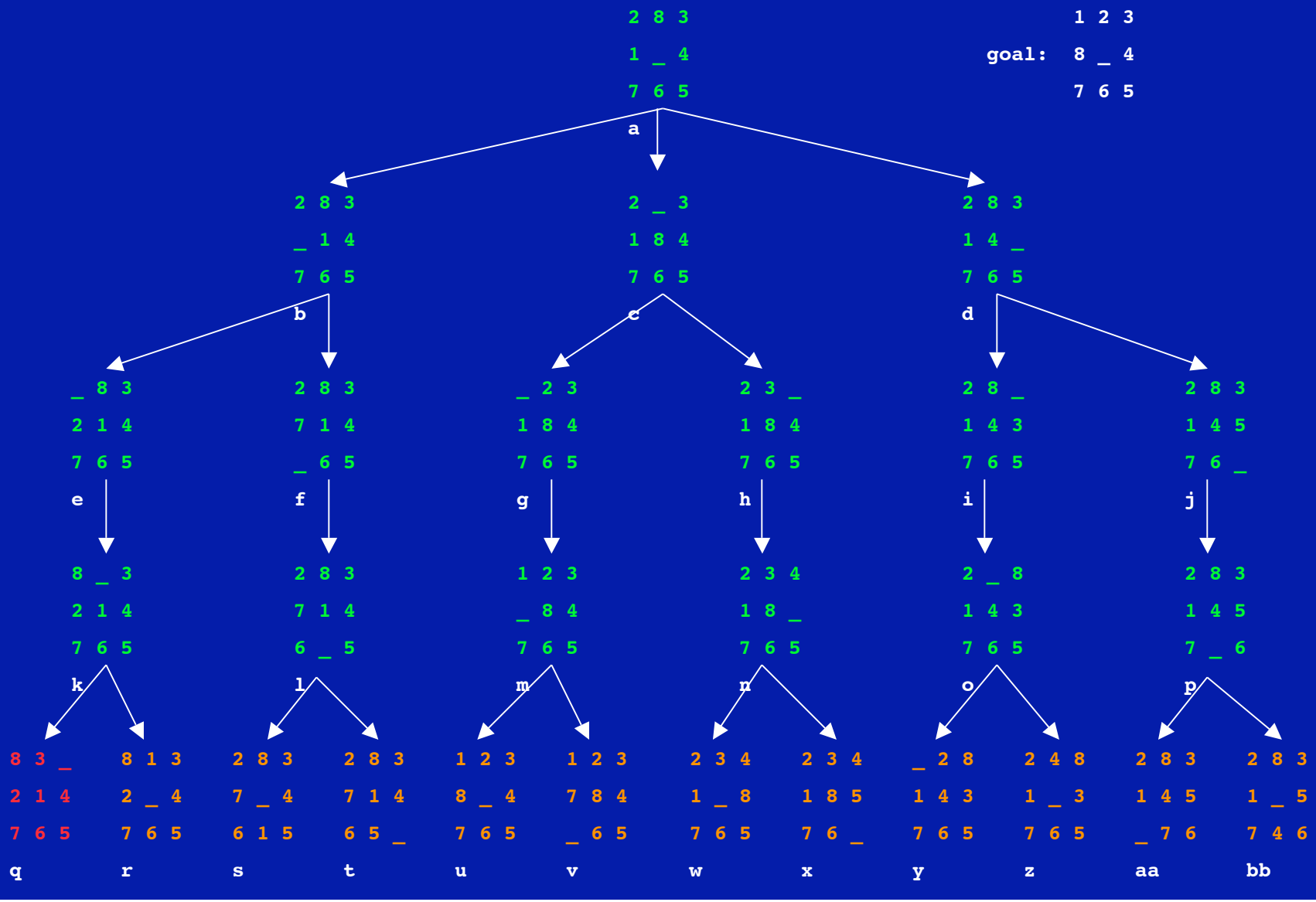
frontier: [q,r,s,t,u,v,w,x,y,z]



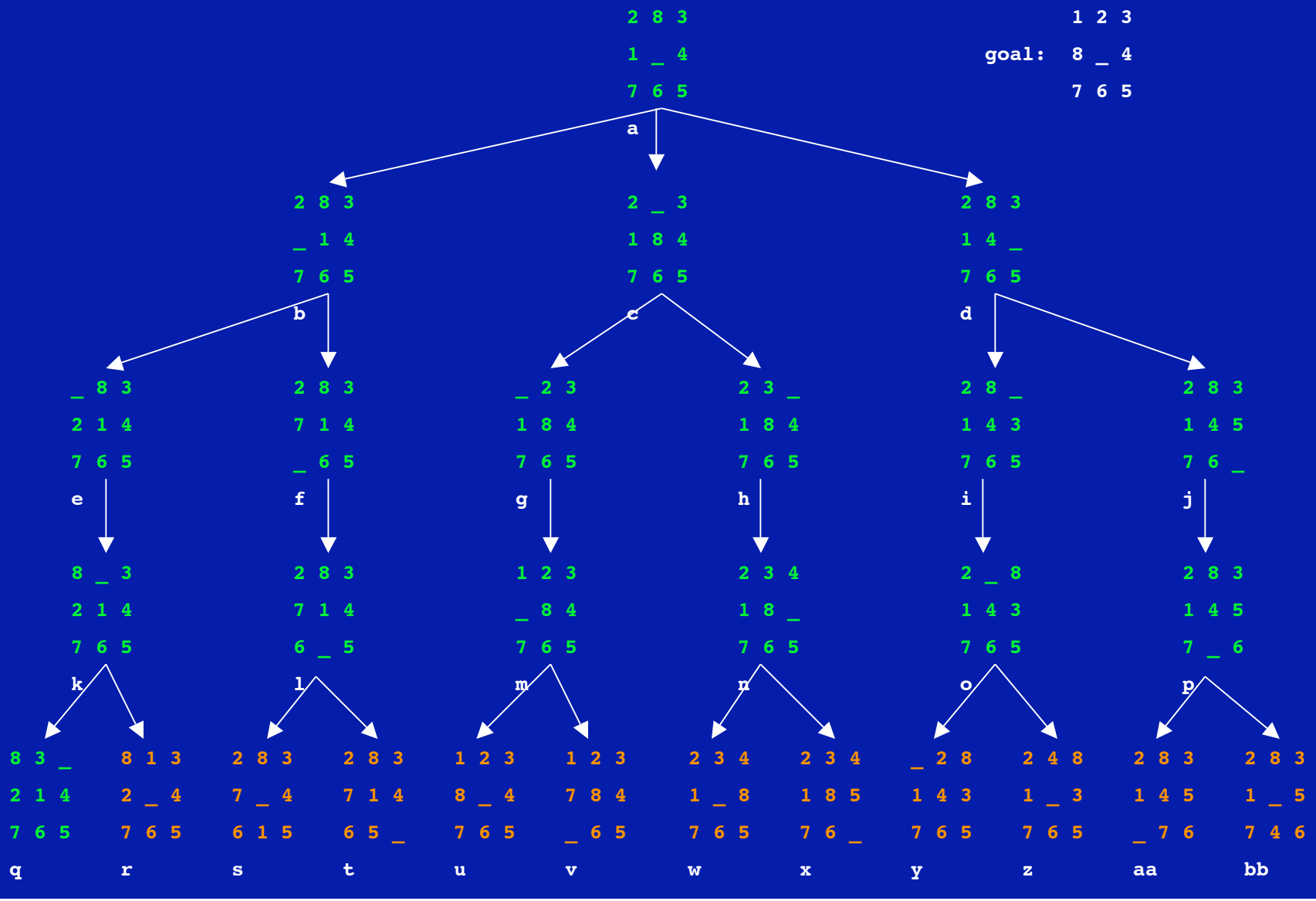
frontier: [q,r,s,t,u,v,w,x,y,z,aa,bb]



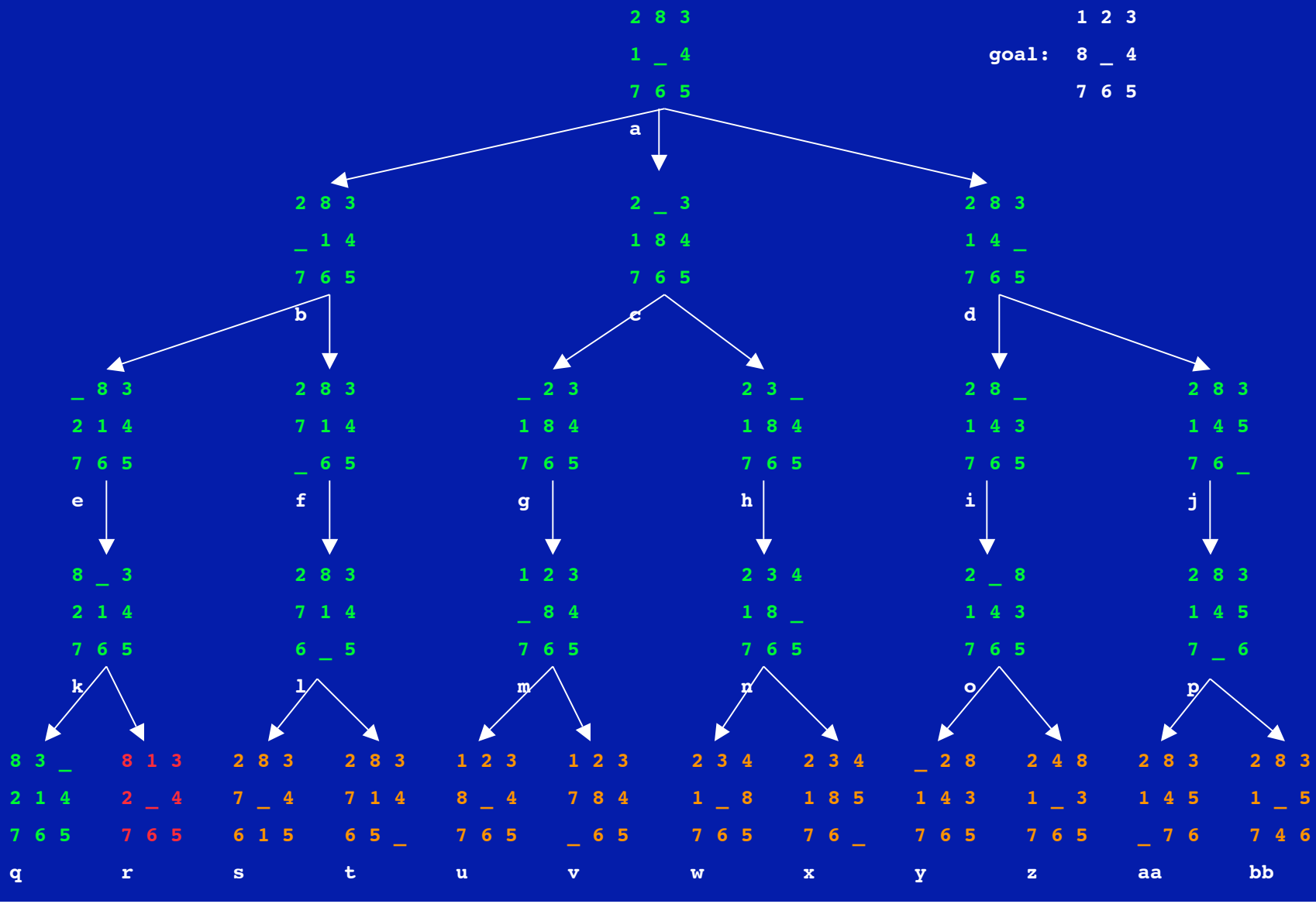
frontier: [r,s,t,u,v,w,x,y,z,aa,bb]



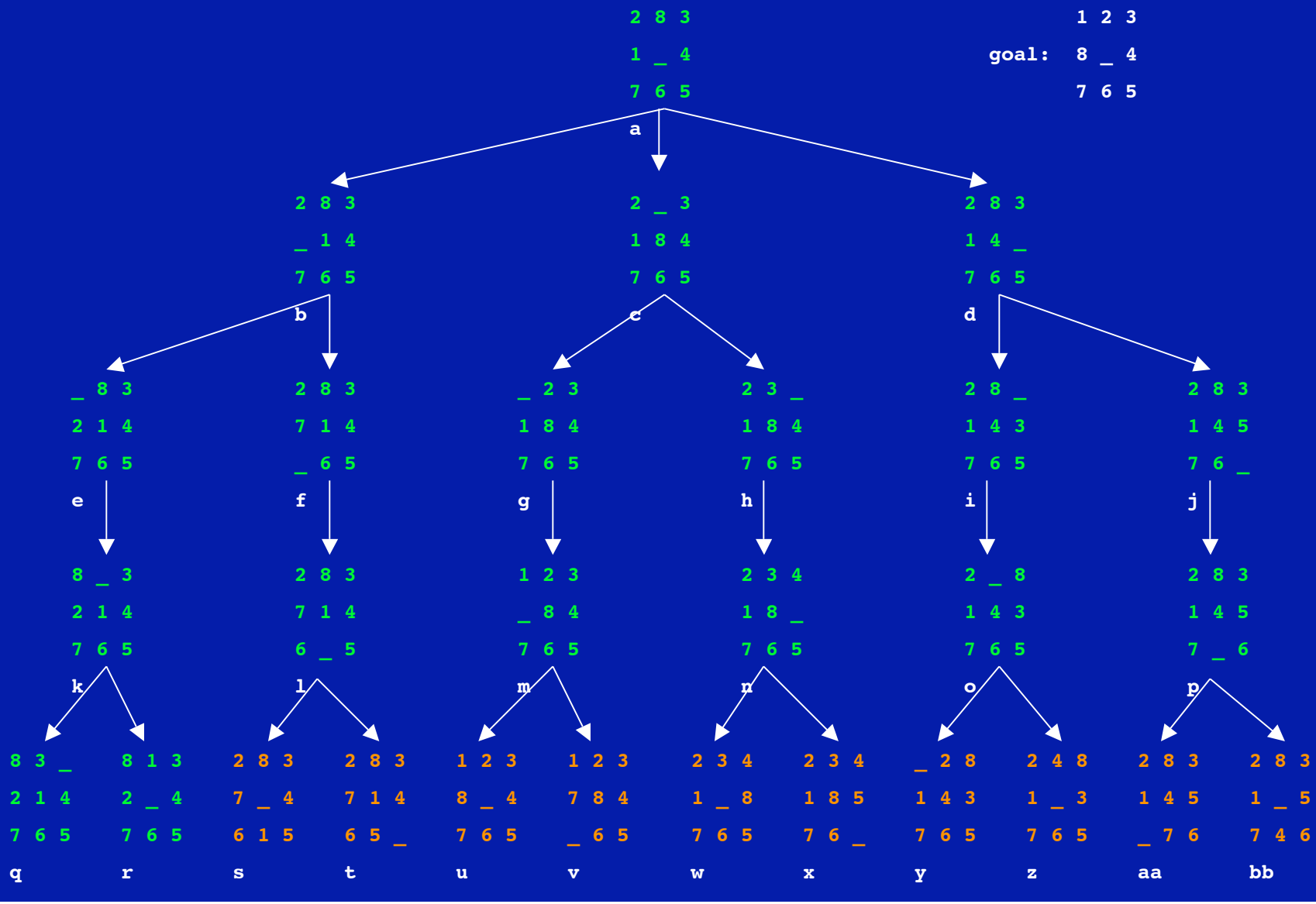
frontier: [r,s,t,u,v,w,x,y,z,aa,bb]



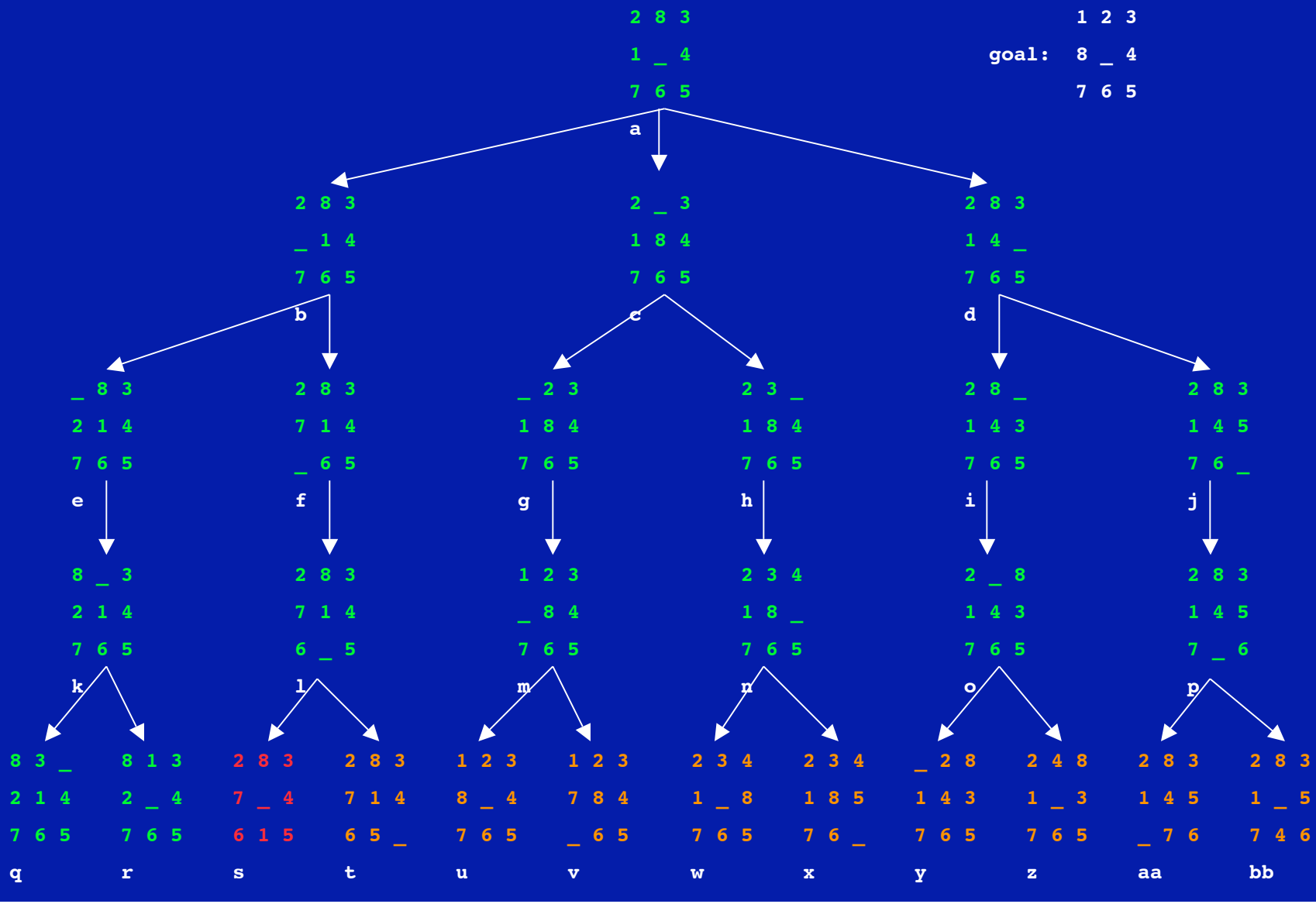
frontier: [s,t,u,v,w,x,y,z,aa,bb]



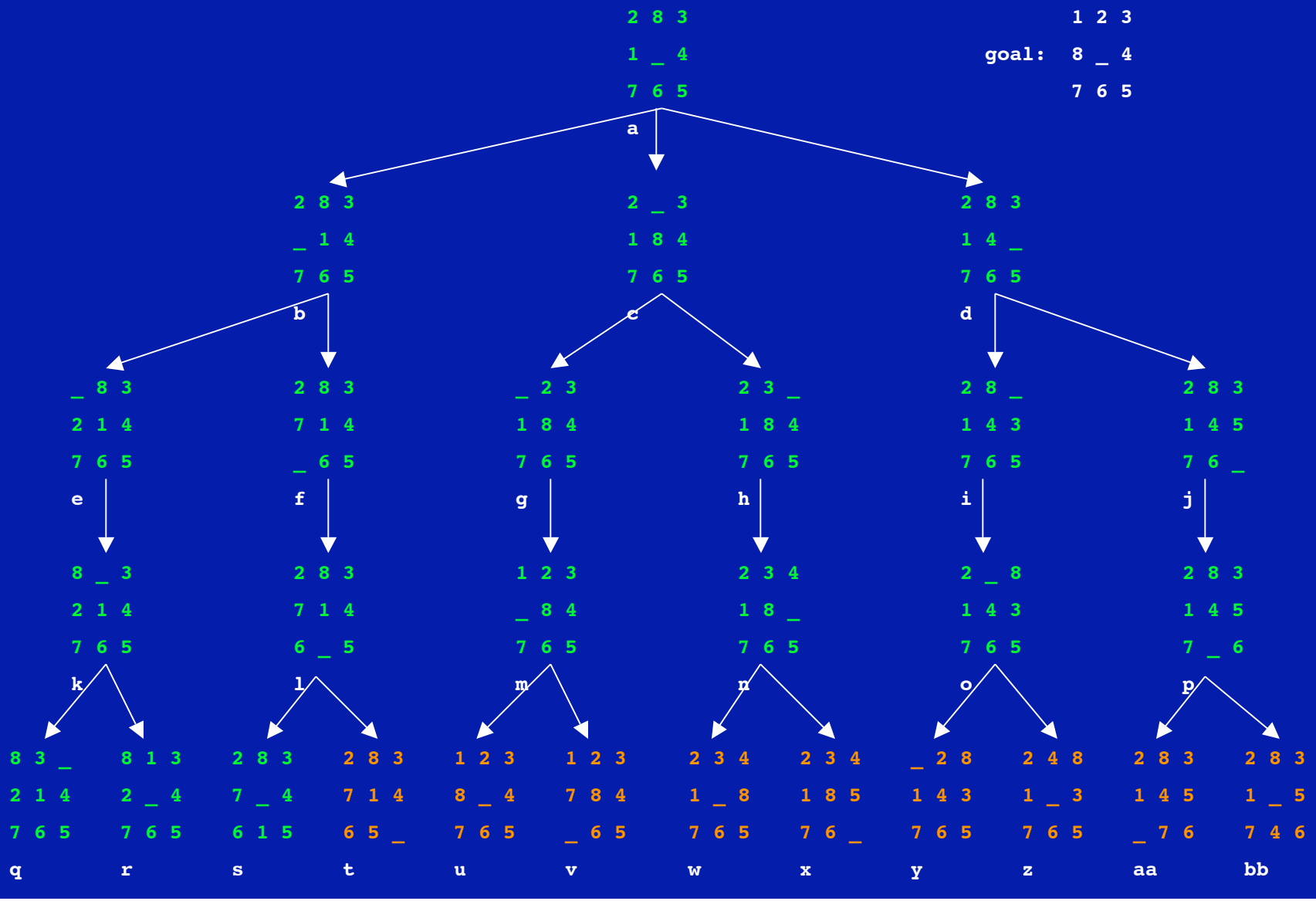
frontier: [s,t,u,v,w,x,y,z,aa,bb]



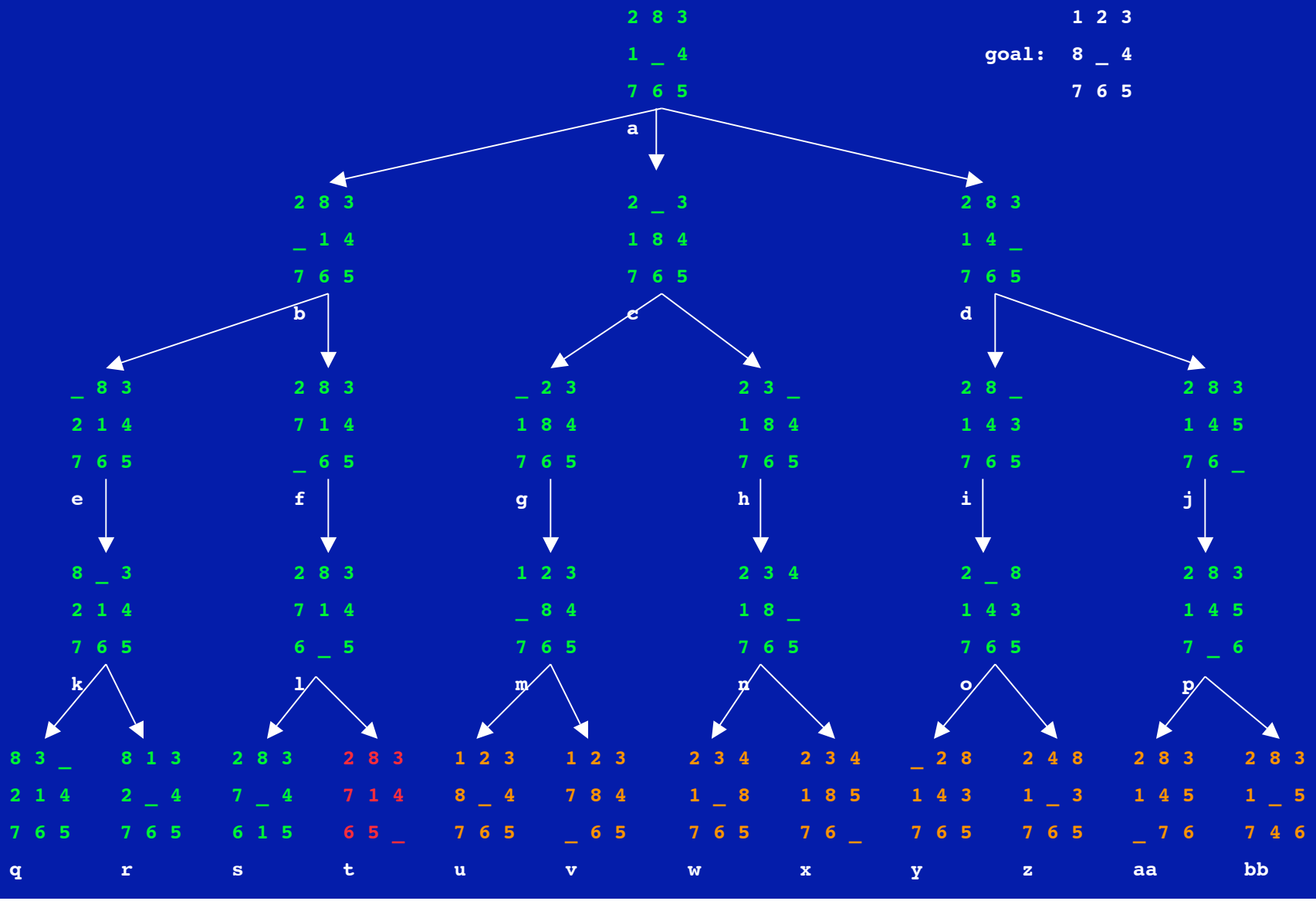
frontier: [t,u,v,w,x,y,z,aa,bb]



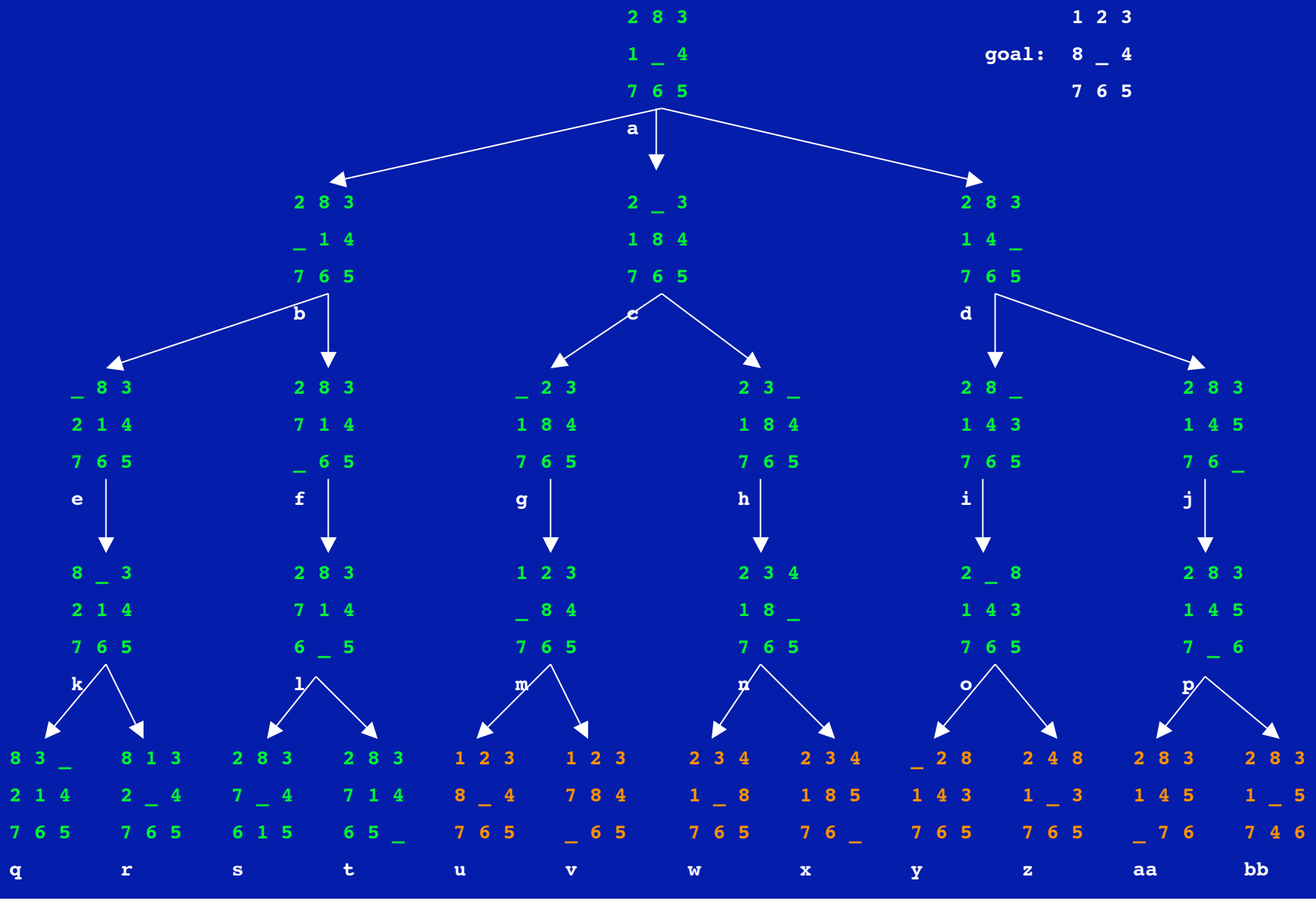
frontier: [t,u,v,w,x,y,z,aa,bb]



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



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



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