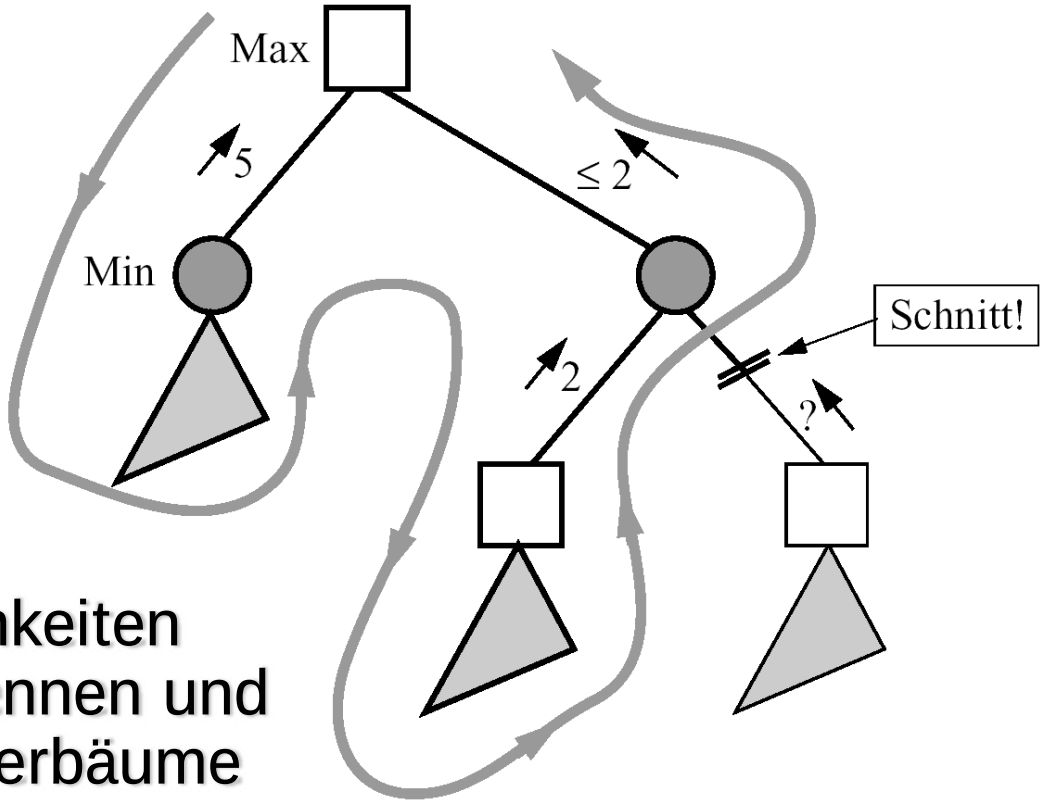
Informatik II für Elektrotechniker FS 2008

Der α-β-Algorithmus

Baumschnitte

- Unterbäume abschneiden, die den Minimax-Wert nicht beeinflussen

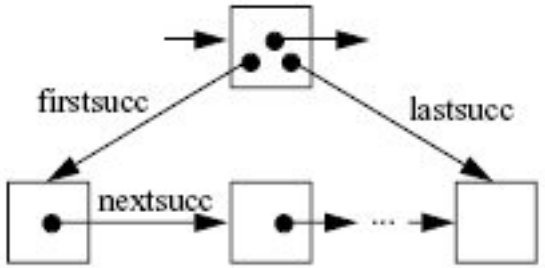


- Ziel: Schnittmöglichkeiten möglichst früh erkennen und möglichst viele Unterbäume abschneiden

Skript S. 182

Implementierung (Beispiel)

```
int maxValue (GameState g, int α, int β) {
    if (cutofftest(g)) return eval(g);
    for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(α, minValue(s, α, β));
        if (α ≥ β) break; // β-Schnitt
    }
    return α;
}
```

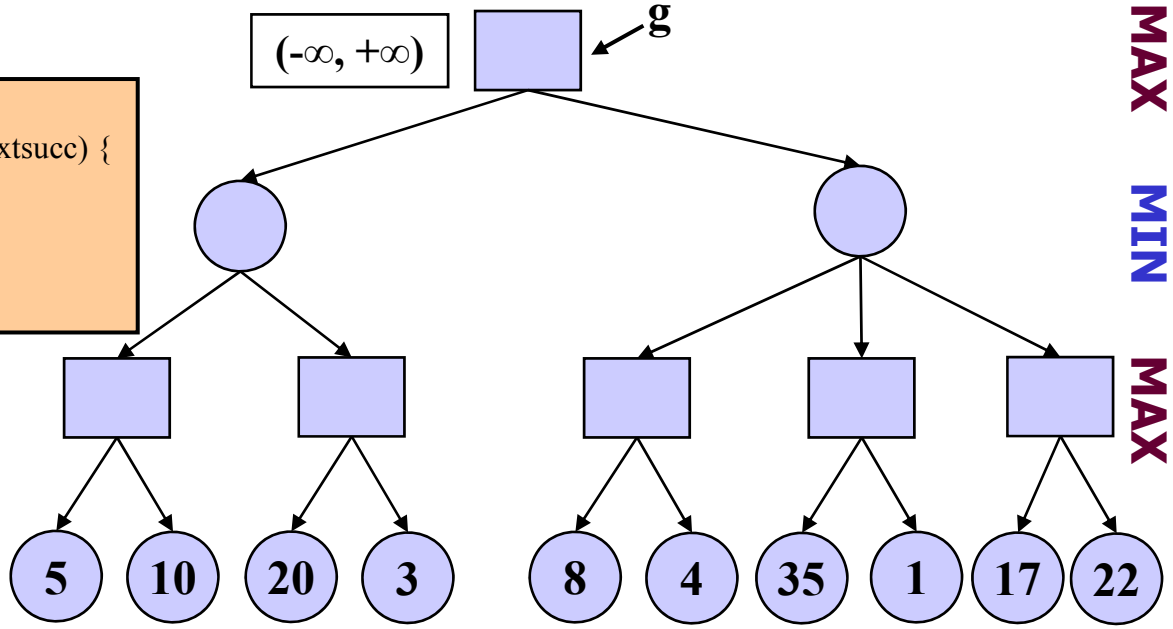



```
int minValue (GameState g, int α, int β) {
    if (cutofftest(g)) return eval(g);
    for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        β = min(β, maxValue(s, α, β));
        if (β ≤ α) break; // α-Schnitt
    }
    return β;
}
```

Skript S. 183

maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(-∞ , minValue(s, -∞, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```

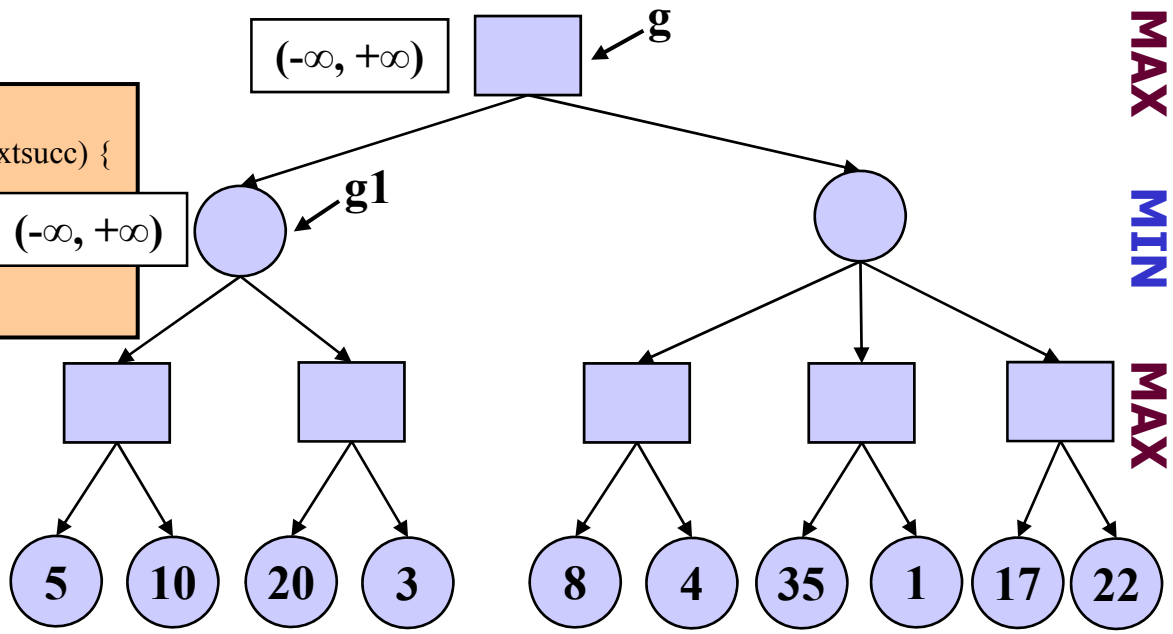


maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(-∞ , minValue(s, -∞, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```

minValue(g1, -∞, +∞)

```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = min(+∞ , maxValue(s, -∞, +∞));
        if ( β ≤ -∞ ) break;
}
return β;
```



maxValue(g, - ∞,+ ∞)

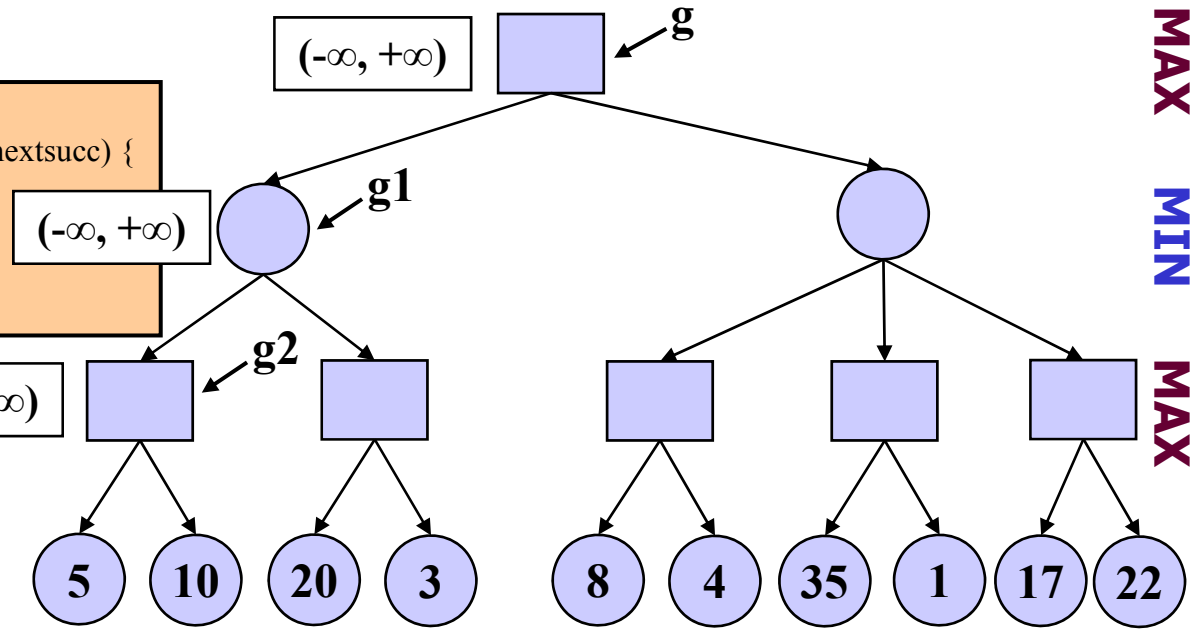
```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(-∞ , minValue(s, -∞, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```

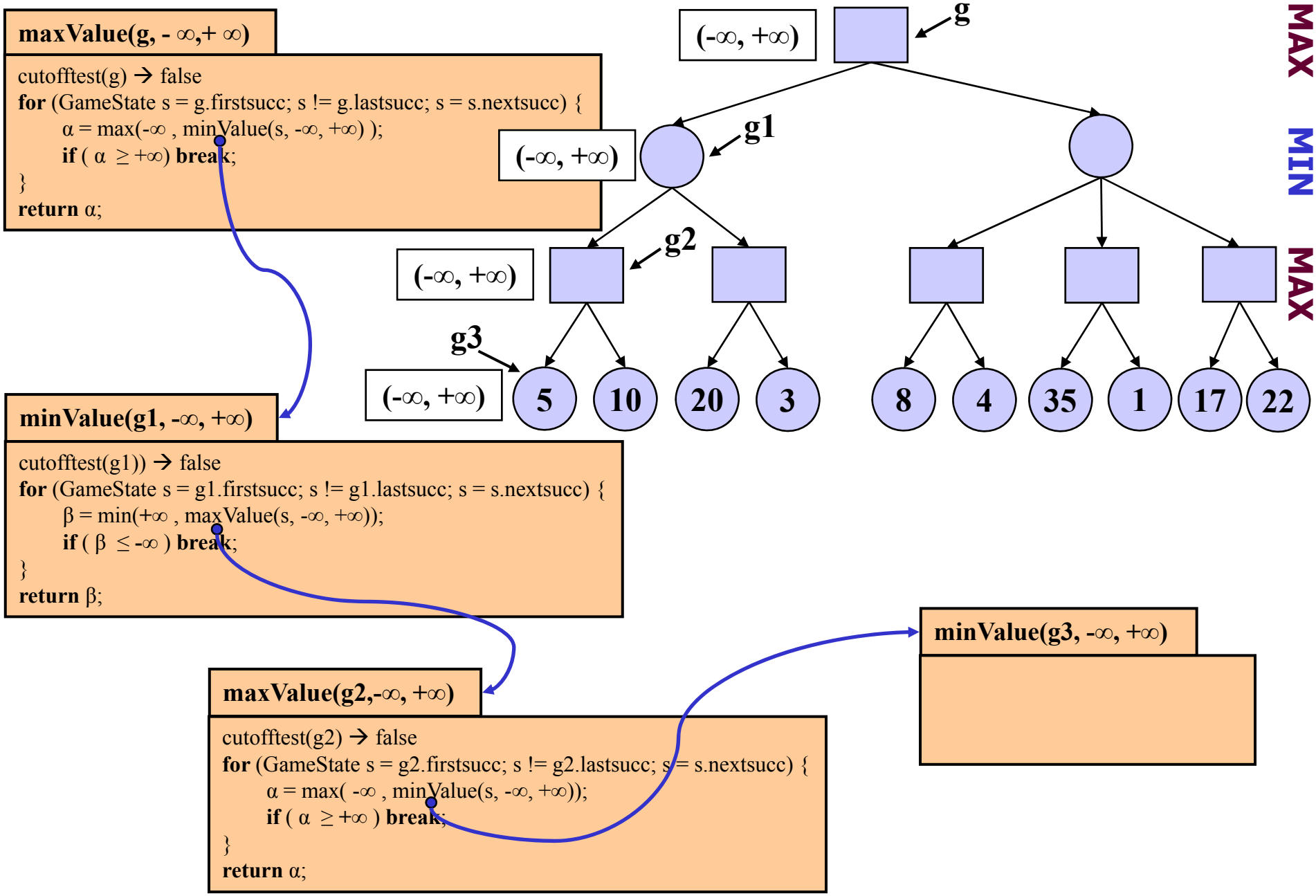
minValue(g1, -∞, +∞)

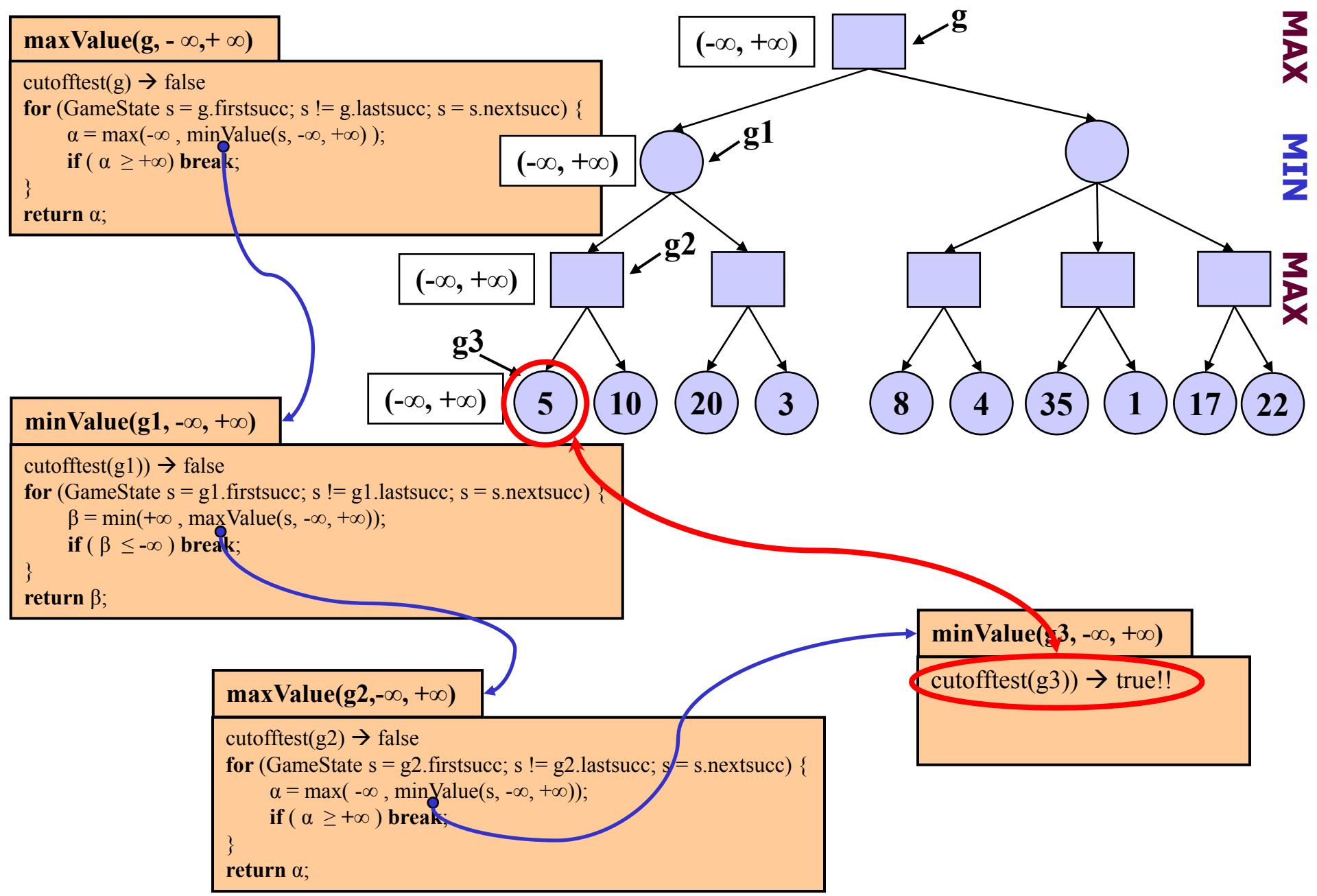
```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(+∞ , maxValue(s, -∞, +∞));
      if ( β ≤ -∞ ) break;
}
return β;
```

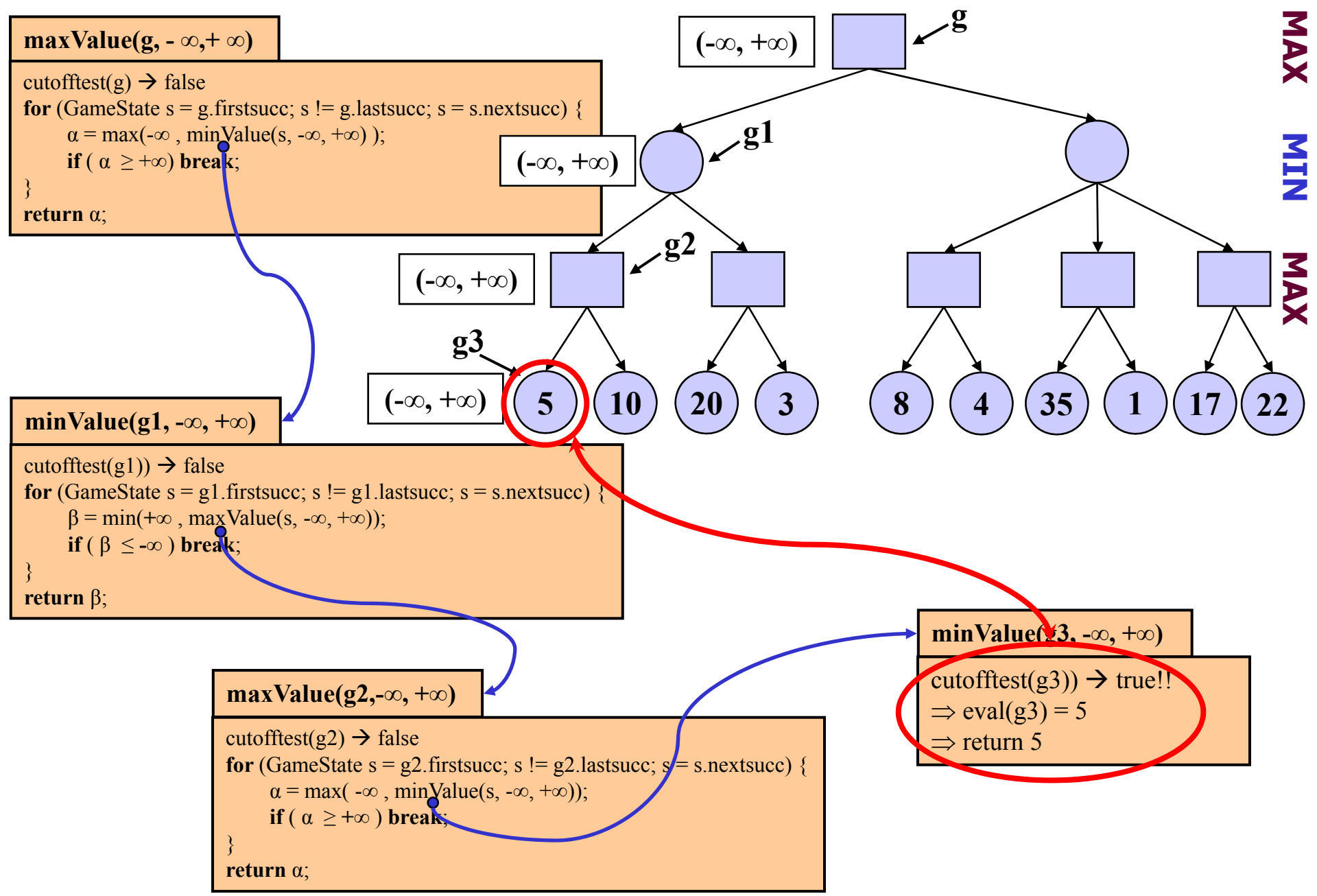
maxValue(g2,-∞, +∞)

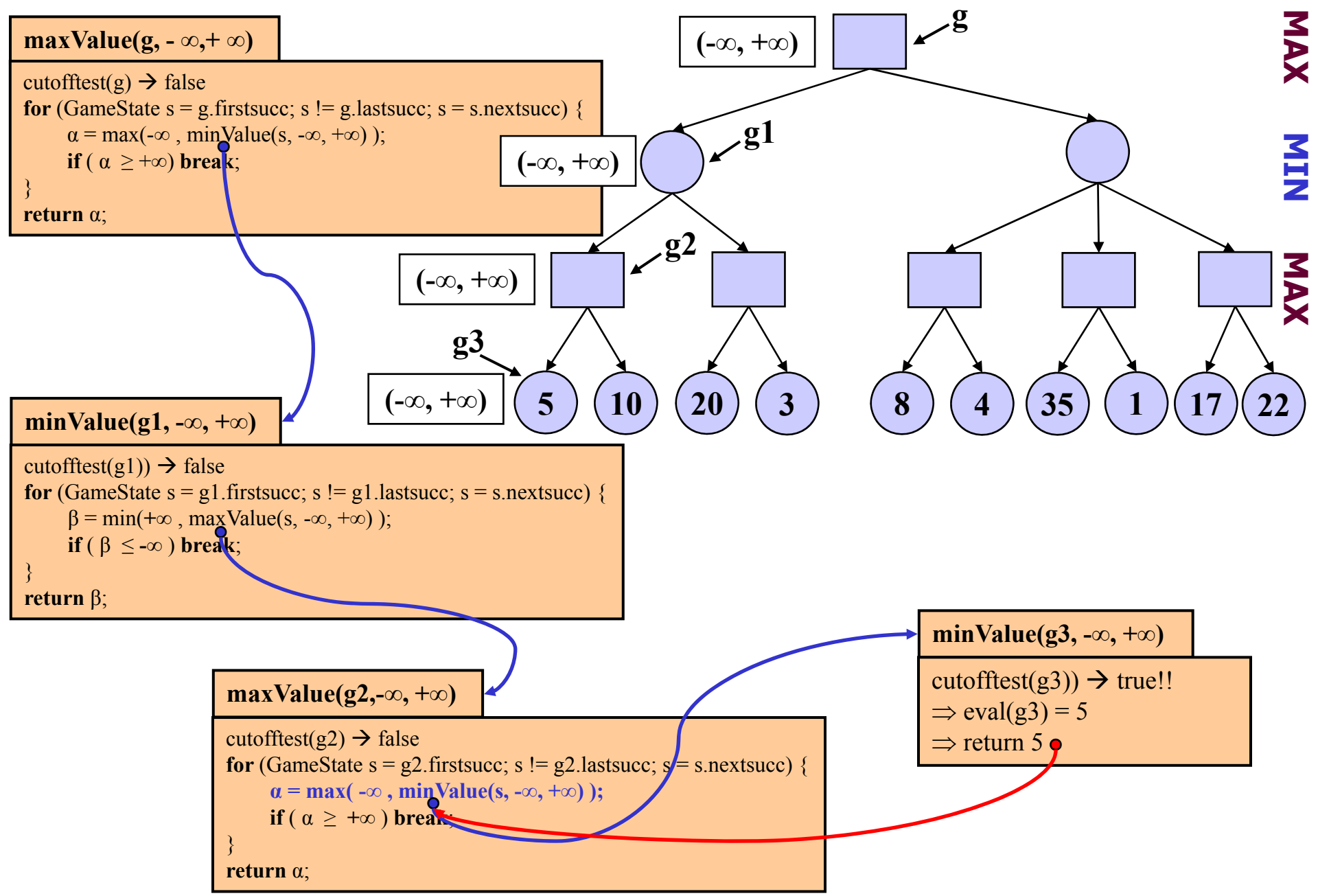
```
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
      α = max( -∞ , minValue(s, -∞, +∞));
      if ( α ≥ +∞ ) break;
}
return α;
```

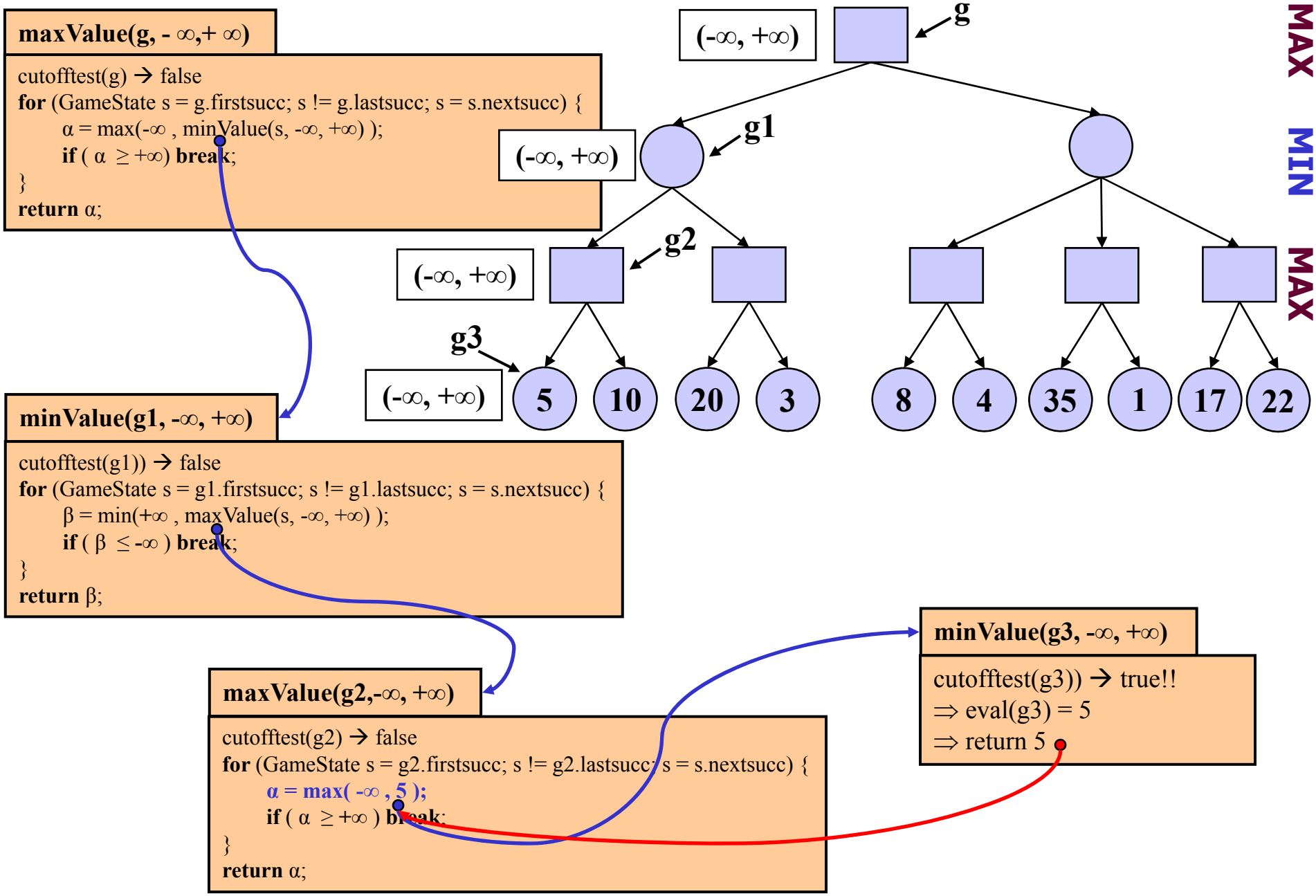



maxValue(g, - ∞,+ ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g3, -∞, +∞)
g
(-∞, +∞)
g1
(-∞, +∞)
g2
(-∞, +∞)
g3
(-∞, +∞)
5
10
20
3
8
4
35
1
17
22
MAX
MIN
MAX


maxValue(g, - ∞,+ ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
(-∞, +∞)
g
g1
(-∞, +∞)
g2
(-∞, +∞)
g3
(-∞, +∞)
5
10
20
3
8
4
35
1
17
22
MAX
MIN
MAX
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g3, -∞, +∞)
cutofftest(g3)) → true!!


maxValue(g, - ∞,+ ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
(-∞, +∞)
g
g1
(-∞, +∞)
g2
(-∞, +∞)
g3
(-∞, +∞)
5
10
20
3
8
4
35
1
17
22
MAX
MIN
MAX
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g3, -∞, +∞)
cutofftest(g3)) → true!!
⇒ eval(g3) = 5
⇒ return 5


maxValue(g, - ∞,+ ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
(-∞, +∞)
g
g1
(-∞, +∞)
g2
(-∞, +∞)
g3
(-∞, +∞)
5
10
20
3
8
4
35
1
17
22
MAX
MIN
MAX
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g3, -∞, +∞)
cutofftest(g3)) → true!!
⇒ eval(g3) = 5
⇒ return 5


maxValue(g, - ∞,+ ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
(-∞, +∞)
g
g1
(-∞, +∞)
g2
(-∞, +∞)
g3
(-∞, +∞)
5
10
20
3
8
4
35
1
17
22
MAX
MIN
MAX
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = max(-∞ , 5);
if (α ≥ +∞) break;
}
return α;
minValue(g3, -∞, +∞)
cutofftest(g3)) → true!!
⇒ eval(g3) = 5
⇒ return 5

maxValue(g, - ∞,+ ∞)

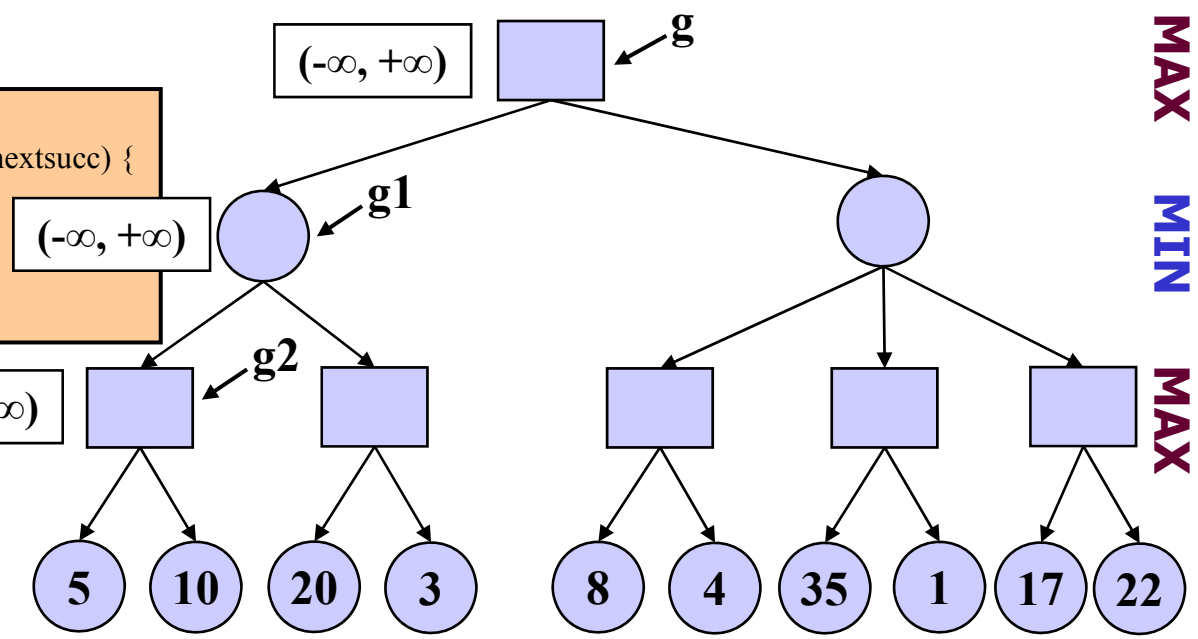
```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(-∞ , minValue(s, -∞, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```

minValue(g1, -∞, +∞)

```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(+∞ , maxValue(s, -∞, +∞) );
      if ( β ≤ -∞ ) break;
}
return β;
```

maxValue(g2,-∞, +∞)

```
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
      α = max( -∞ , 5 );
      if ( α ≥ +∞ ) break;
}
return α;
```



maxValue(g, - ∞,+ ∞)

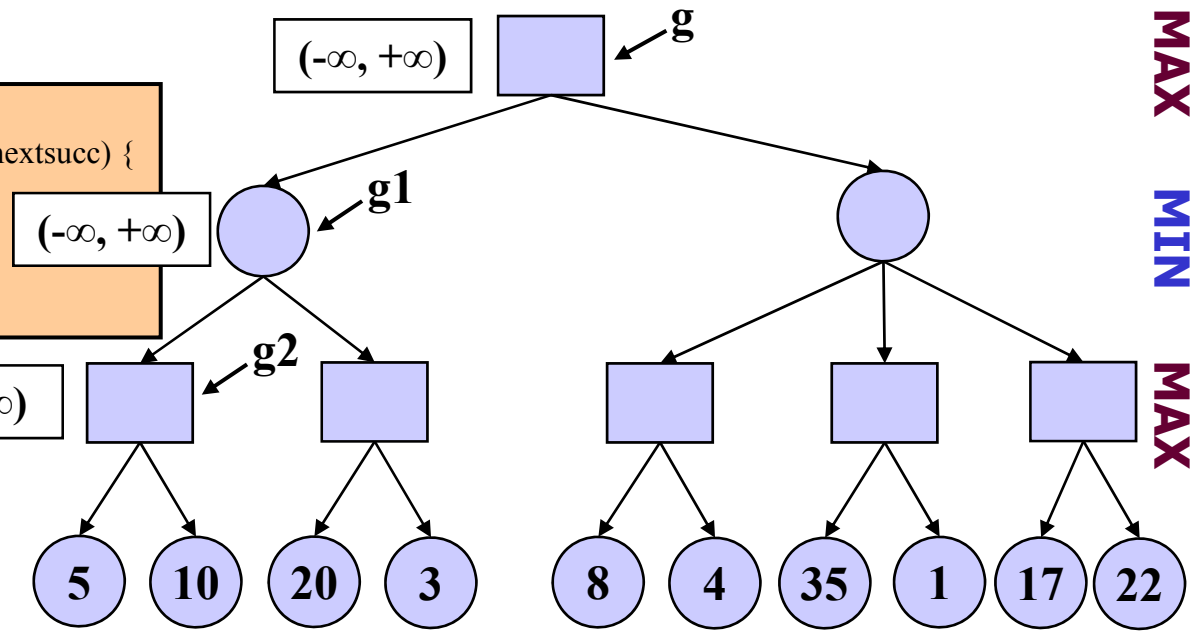
```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(-∞ , minValue(s, -∞, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```

minValue(g1, -∞, +∞)

```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(+∞ , maxValue(s, -∞, +∞) );
      if ( β ≤ -∞ ) break;
}
return β;
```

maxValue(g2,-∞, +∞)

```
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
      α = 5;
      if ( α ≥ +∞ ) break;
}
return α;
```



maxValue(g, - ∞, + ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(-∞ , minValue(s, -∞, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```

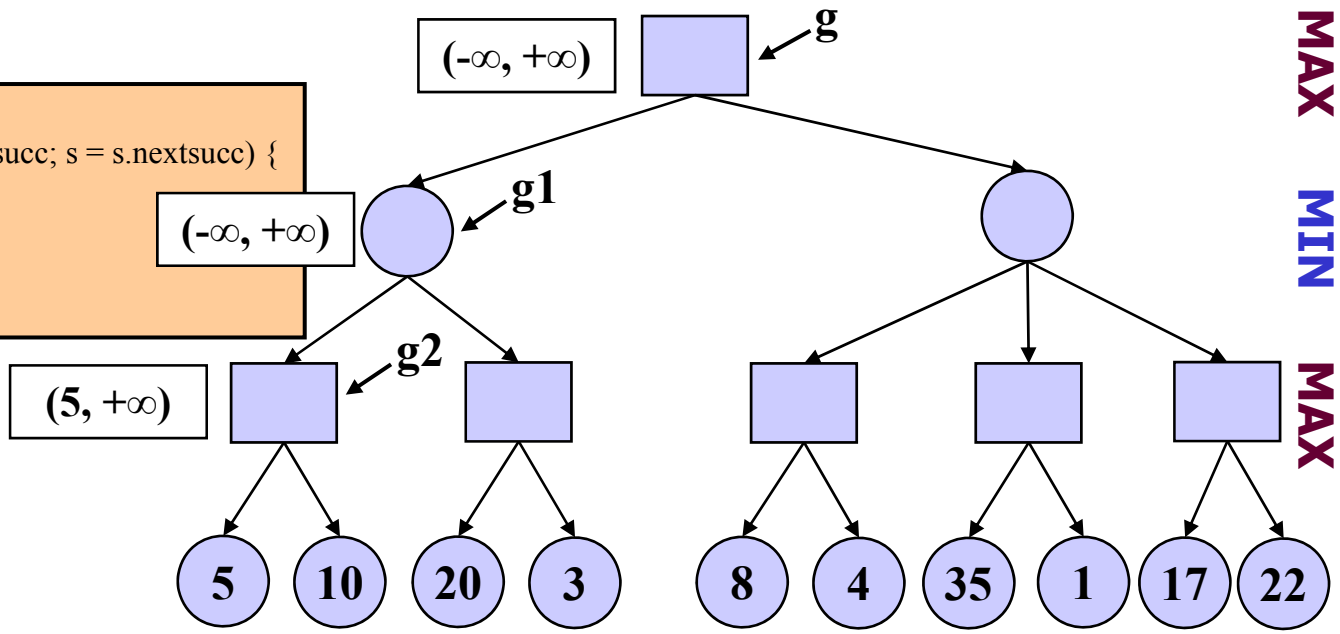
minValue(g1, -∞, +∞)

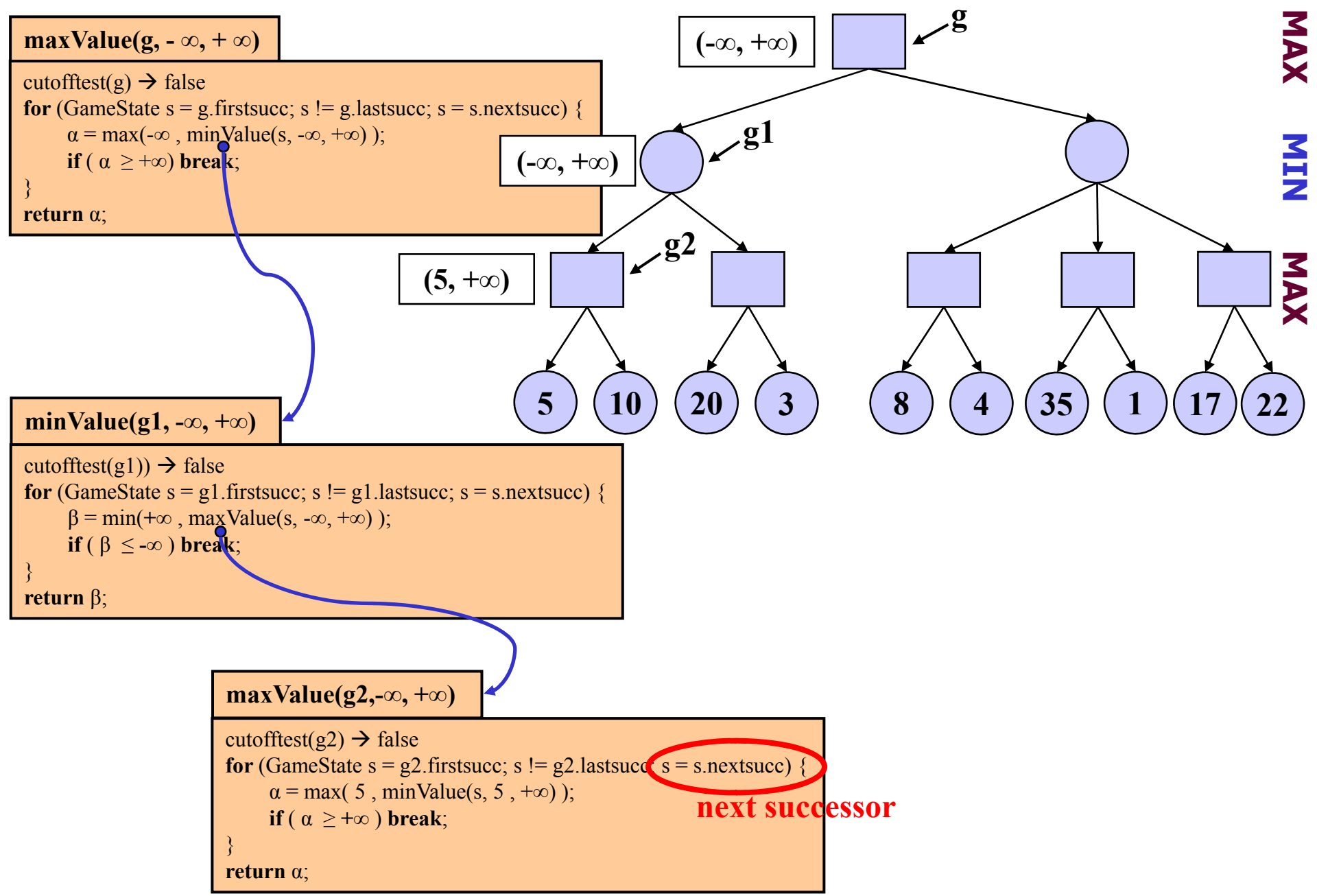
```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(+∞ , maxValue(s, -∞, +∞) );
      if ( β ≤ -∞ ) break;
}
return β;
```

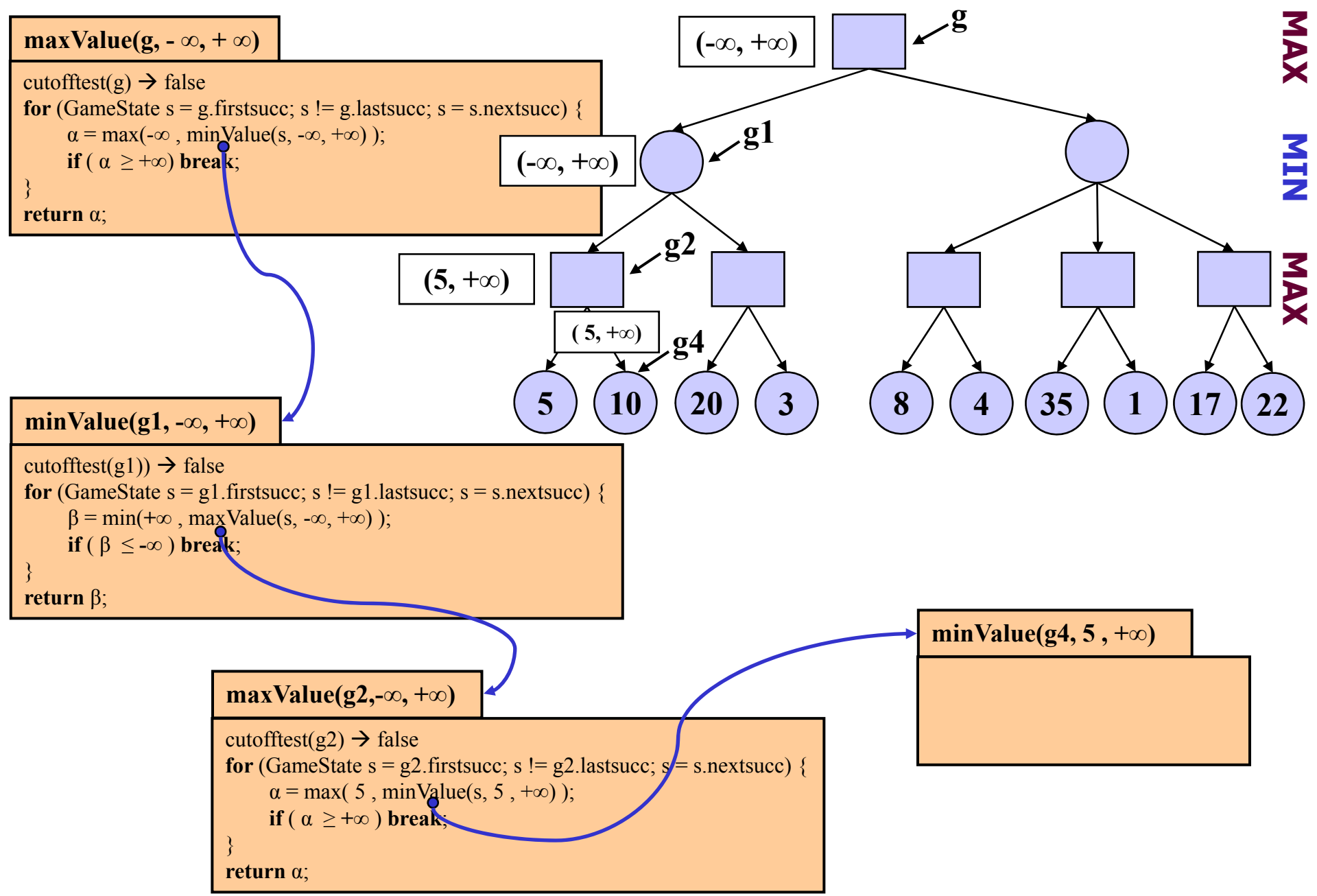
maxValue(g2,-∞, +∞)

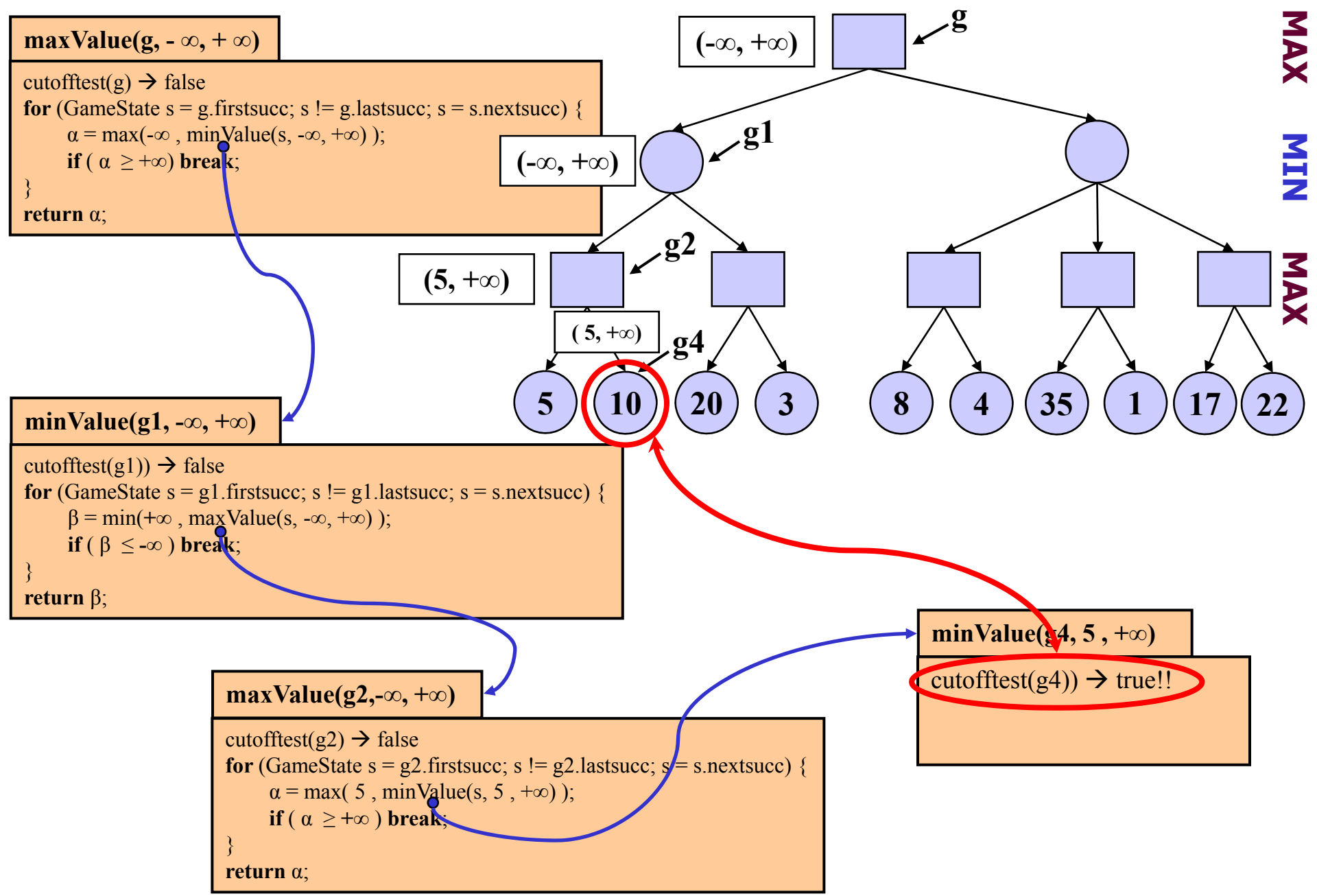
```
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
      α = 5;
      if ( α ≥ +∞ ) break;
}
return α;
```

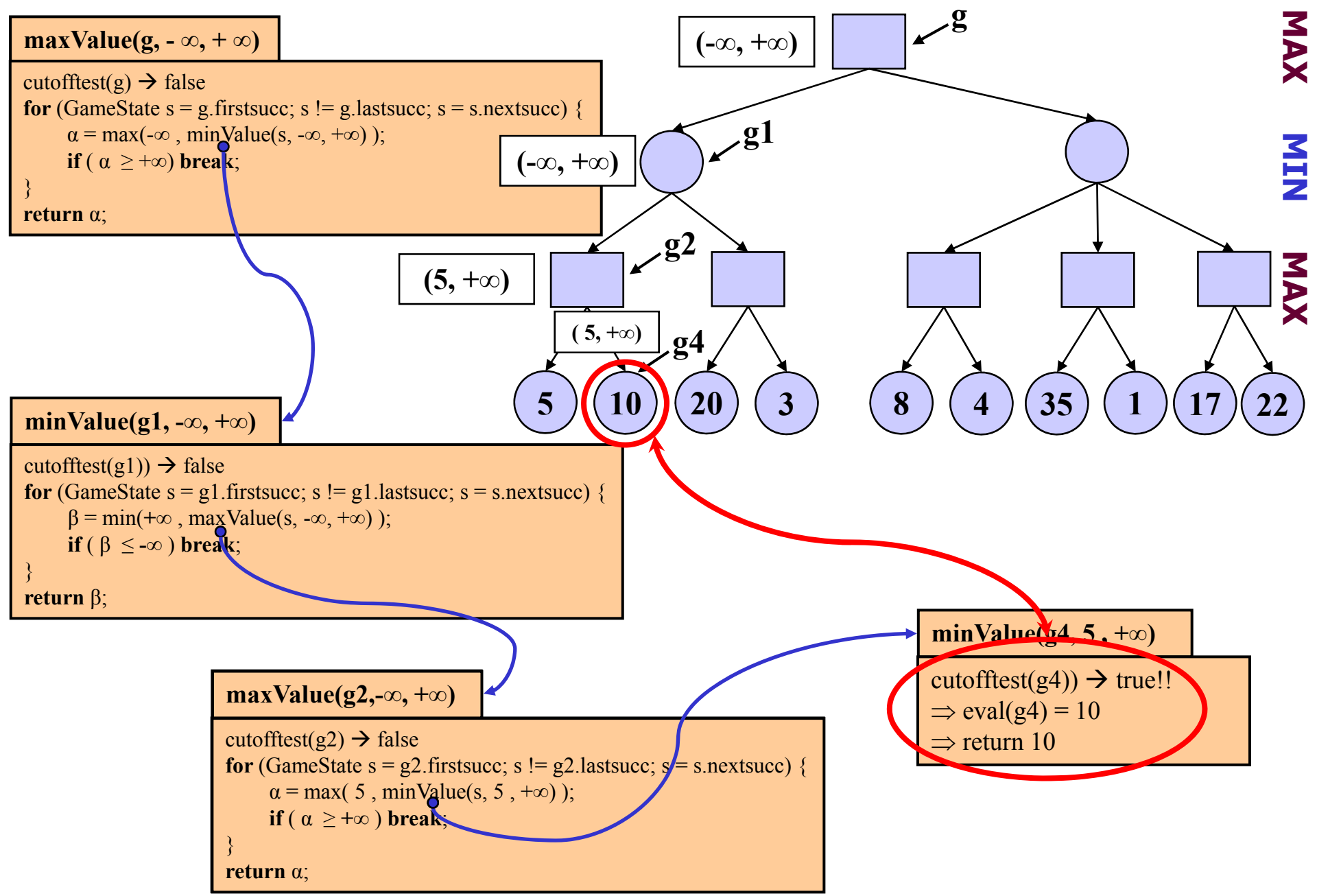
false!

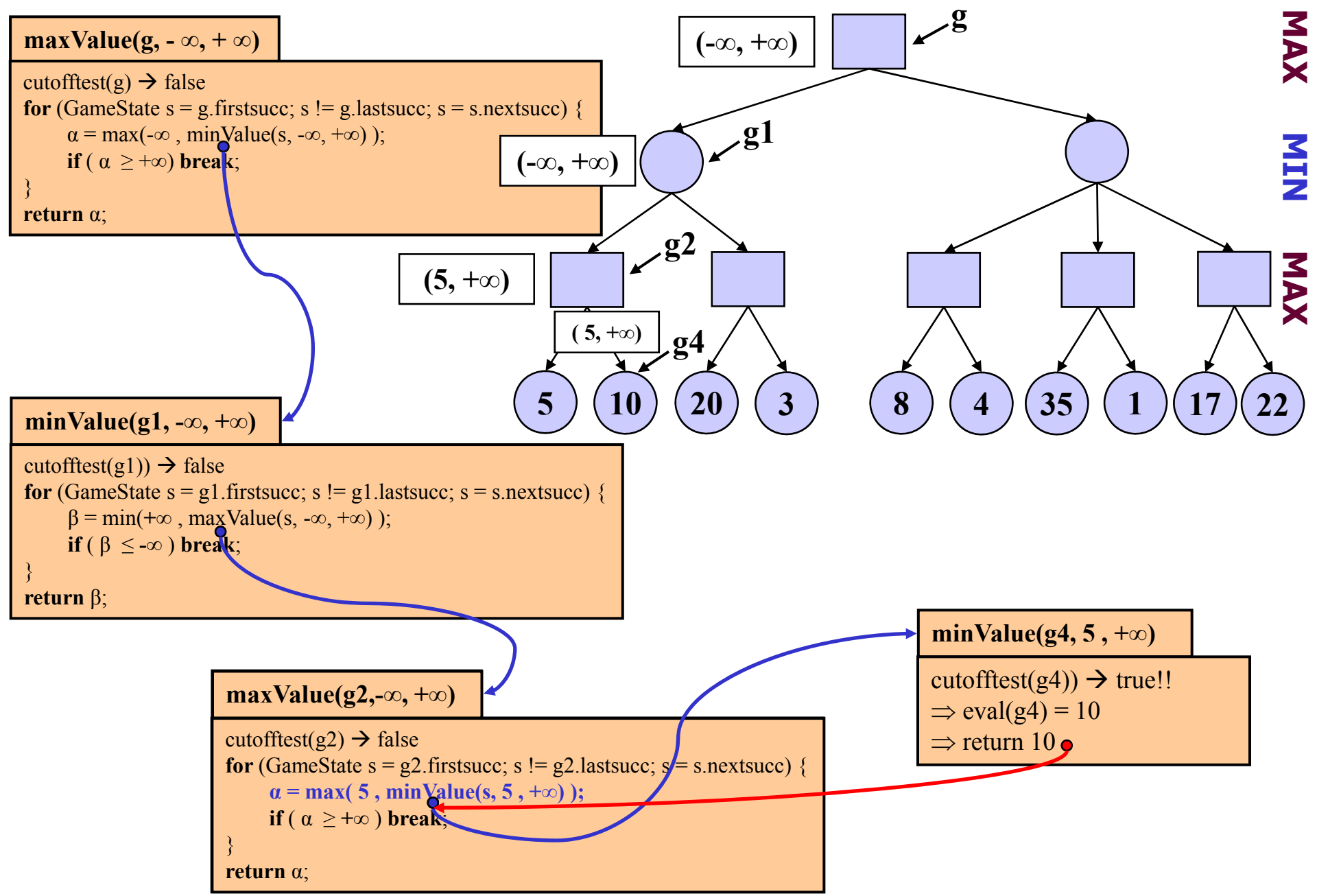


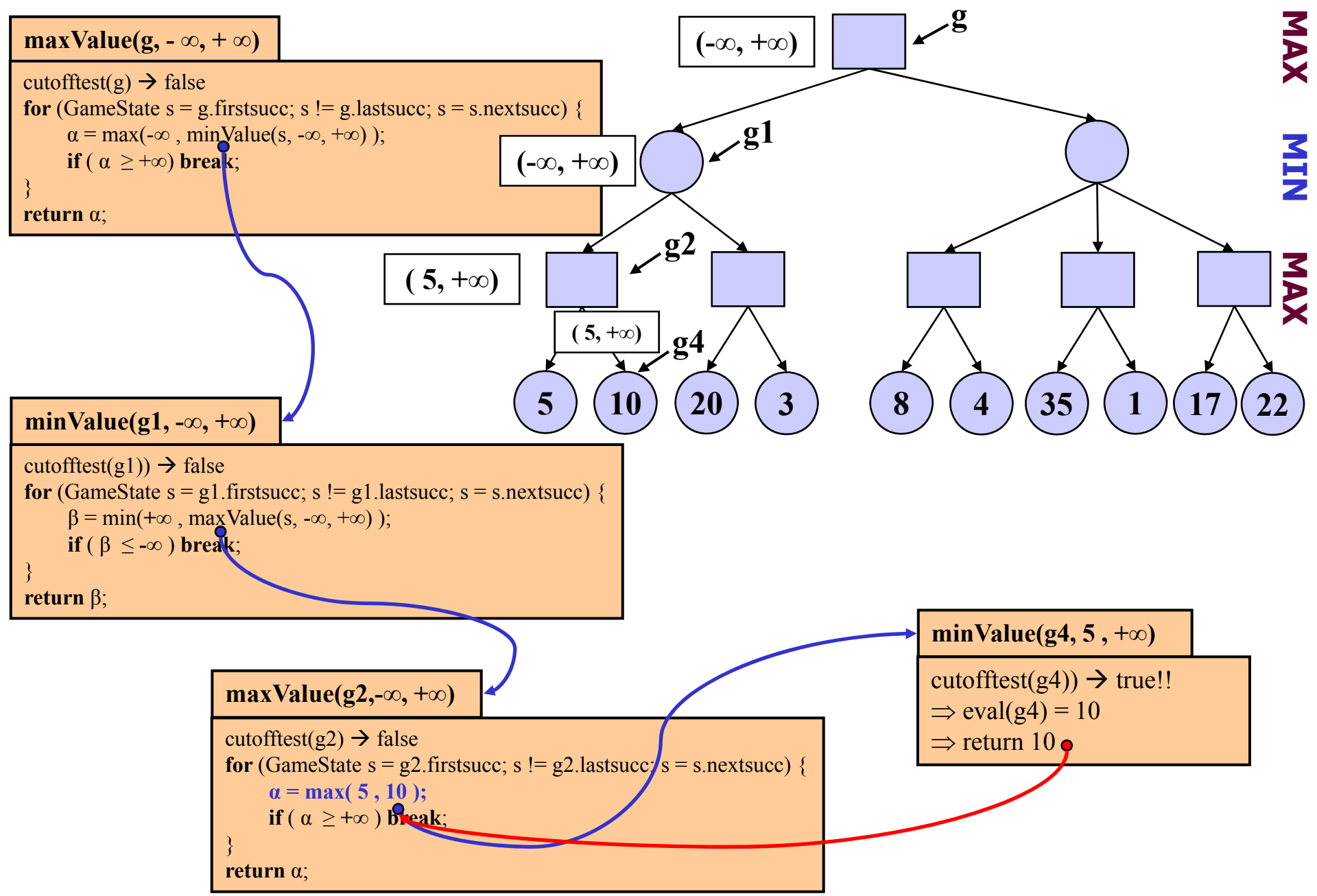

maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = max(5 , minValue(s, 5 , +∞));
if (α ≥ +∞) break;
}
return α;
next successor
(-∞, +∞)
g
g1
(-∞, +∞)
(5, +∞)
g2
5
10
20
3
8
4
35
1
17
22
MAX
MIN
MAX

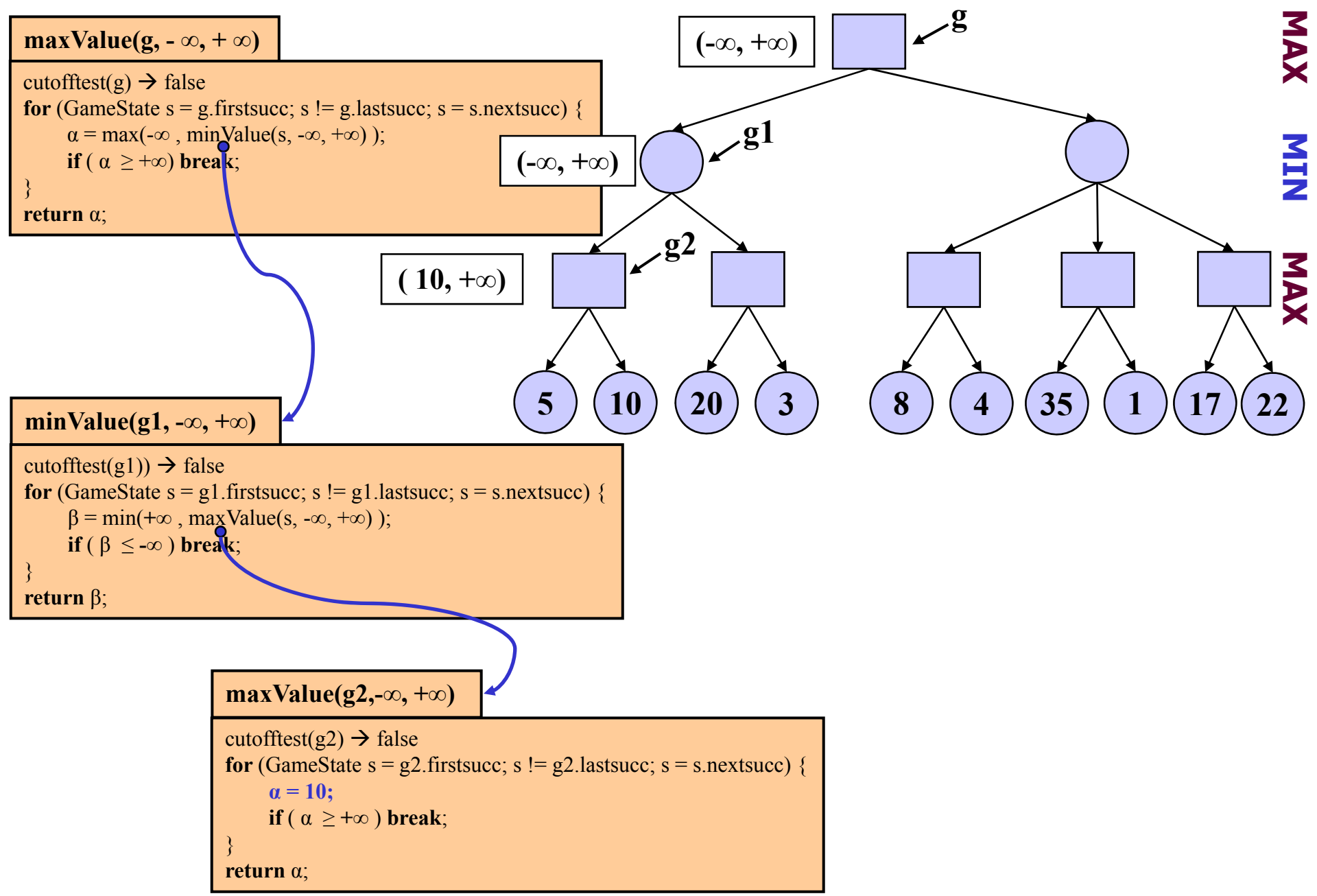

maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
(-∞, +∞)
g
g1
(-∞, +∞)
(5, +∞)
g2
(5, +∞)
g4
MAX
MIN
MAX
5
10
20
3
8
4
35
1
17
22
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = max(5 , minValue(s, 5 , +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g4, 5 , +∞)


maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
(-∞, +∞)
g
g1
(-∞, +∞)
(5, +∞)
g2
(5, +∞)
g4
MAX
MIN
MAX
5
10
20
3
8
4
35
1
17
22
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
minValue(g4, 5 , +∞)
cutofftest(g4)) → true!!
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = max(5 , minValue(s, 5 , +∞));
if (α ≥ +∞) break;
}
return α;


maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
(-∞, +∞)
g
g1
(-∞, +∞)
(5, +∞)
g2
(5, +∞)
g4
5
10
20
3
8
4
35
1
17
22
MAX
MIN
MAX
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = max(5 , minValue(s, 5 , +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g4, 5 , +∞)
cutofftest(g4)) → true!!
⇒ eval(g4) = 10
⇒ return 10


maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
(-∞, +∞)
g
g1
(-∞, +∞)
(5, +∞)
g2
(5, +∞)
g4
5
10
20
3
8
4
35
1
17
22
MAX
MIN
MAX
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = max(5 , minValue(s, 5 , +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g4, 5 , +∞)
cutofftest(g4)) → true!!
⇒ eval(g4) = 10
⇒ return 10


maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
(-∞, +∞)
g
MAX
g1
(-∞, +∞)
MIN
g2
(5, +∞)
MAX
(5, +∞)
g4
5
10
20
3
8
4
35
1
17
22
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = max(5 , 10);
if (α ≥ +∞) break;
}
return α;
minValue(g4, 5 , +∞)
cutofftest(g4)) → true!!
⇒ eval(g4) = 10
⇒ return 10


maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
(-∞, +∞)
g
g1
(-∞, +∞)
g2
(10, +∞)
MAX
MIN
MAX
5
10
20
3
8
4
35
1
17
22
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = 10;
if (α ≥ +∞) break;
}
return α;

maxValue(g, - ∞, + ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(-∞ , minValue(s, -∞, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```

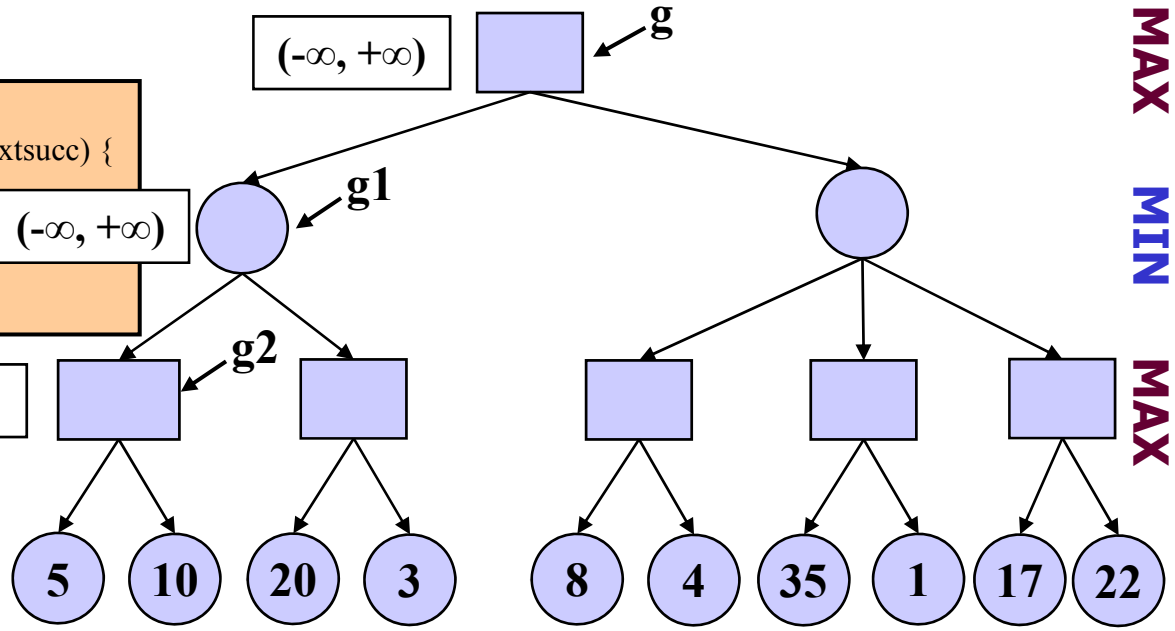
minValue(g1, -∞, +∞)

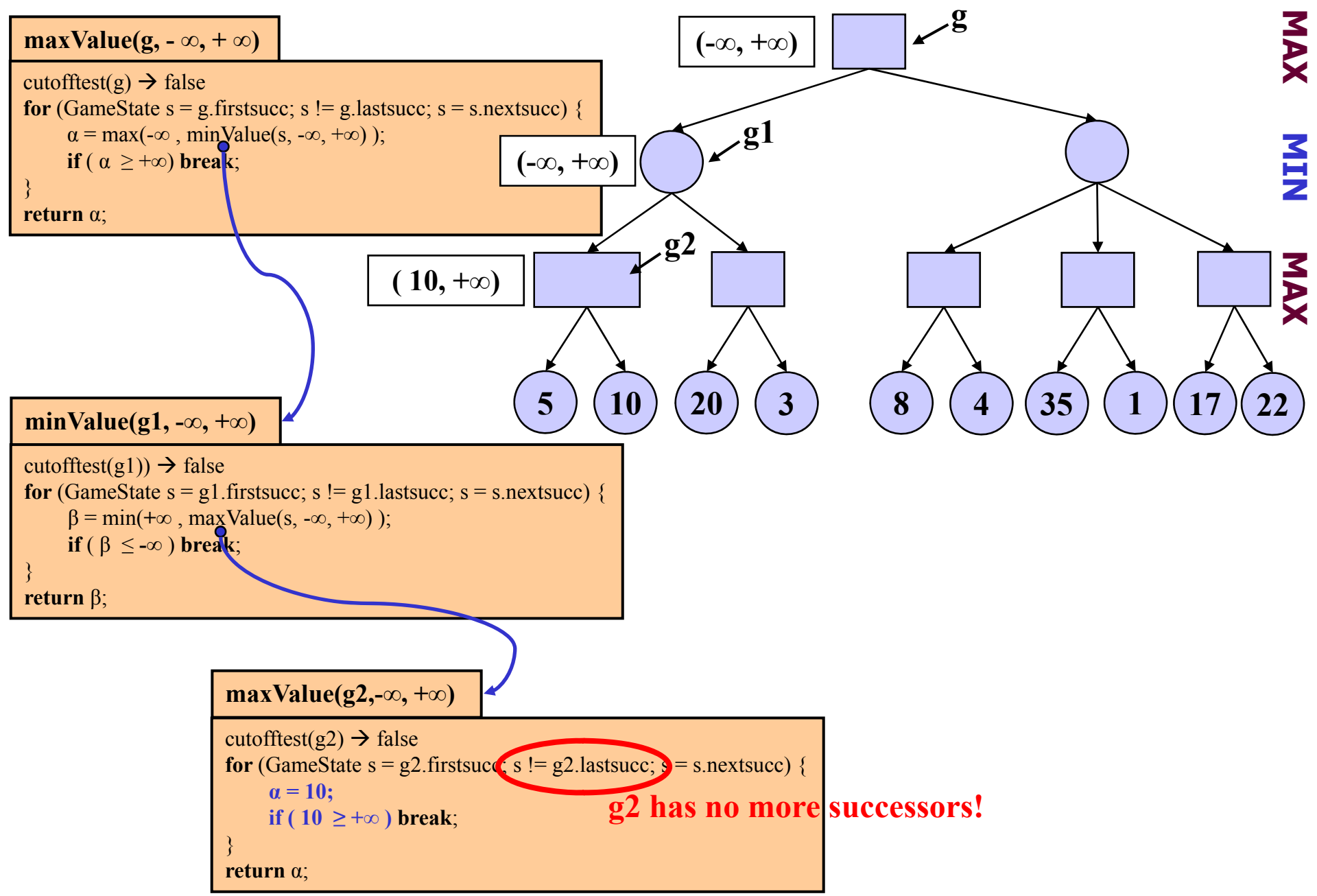
```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(+∞ , maxValue(s, -∞, +∞) );
      if ( β ≤ -∞ ) break;
}
return β;
```

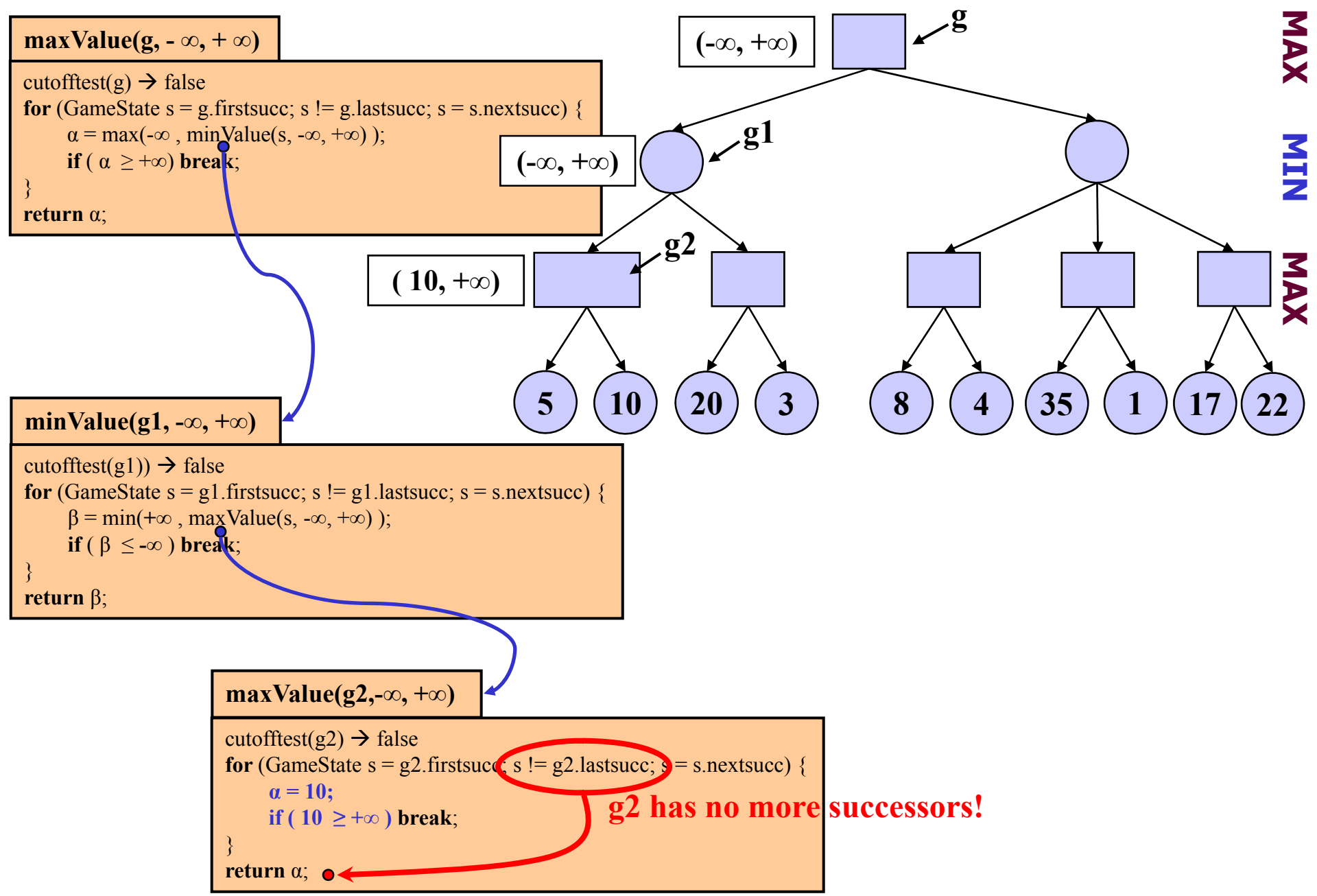
maxValue(g2,-∞, +∞)

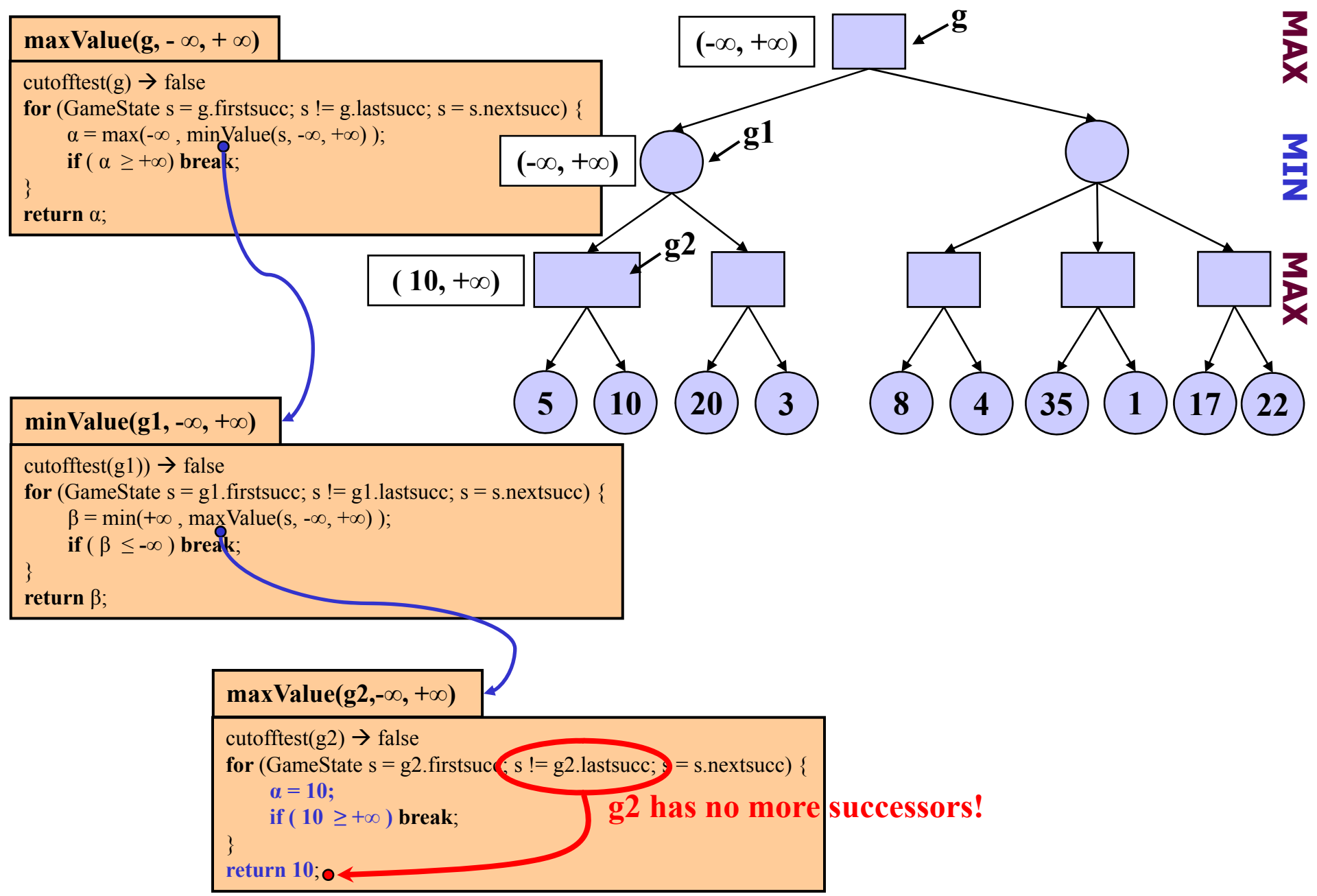
```
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
      α = 10;
      if ( 10 ≥ +∞ ) break;
}
return α;
```

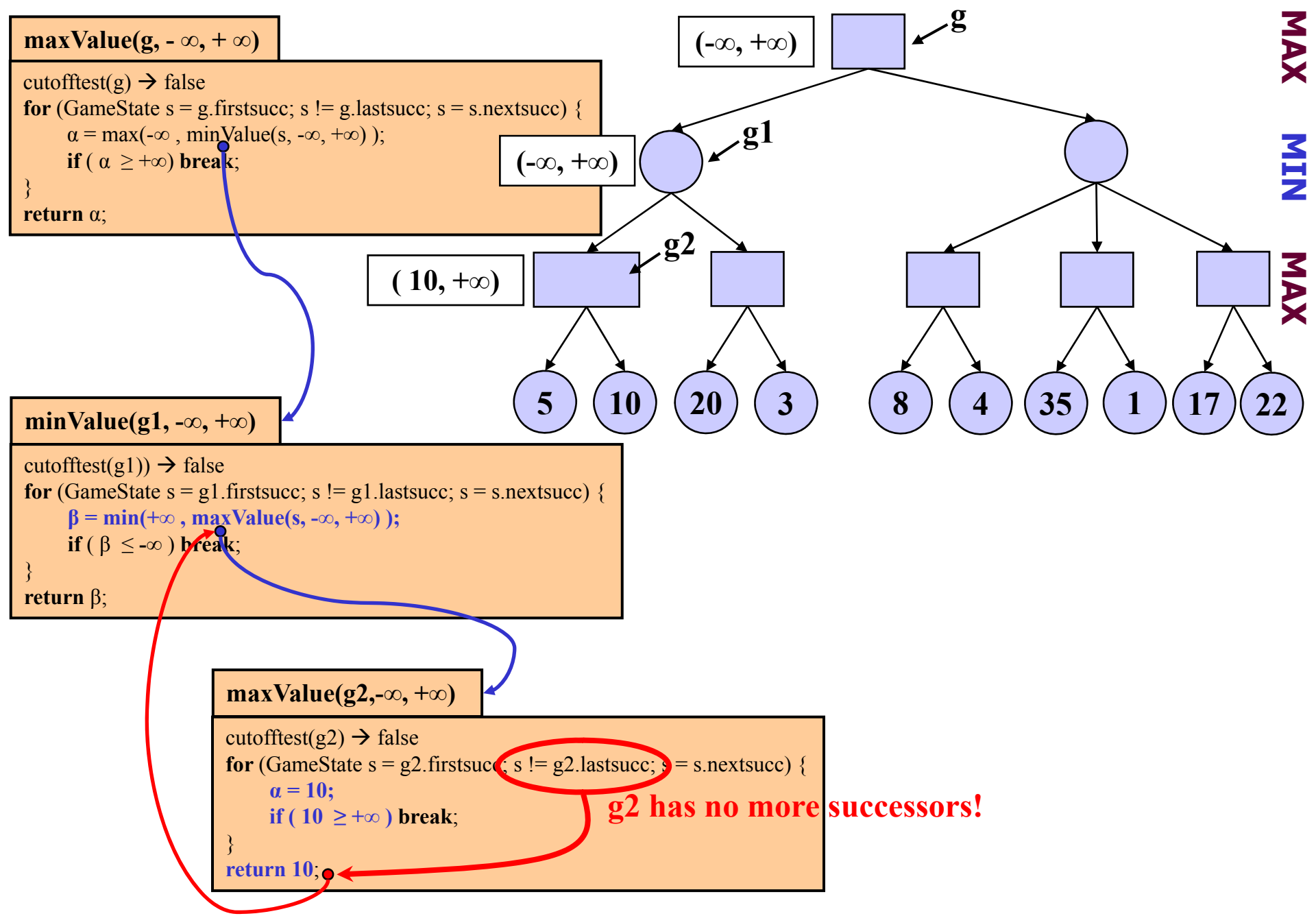
false!

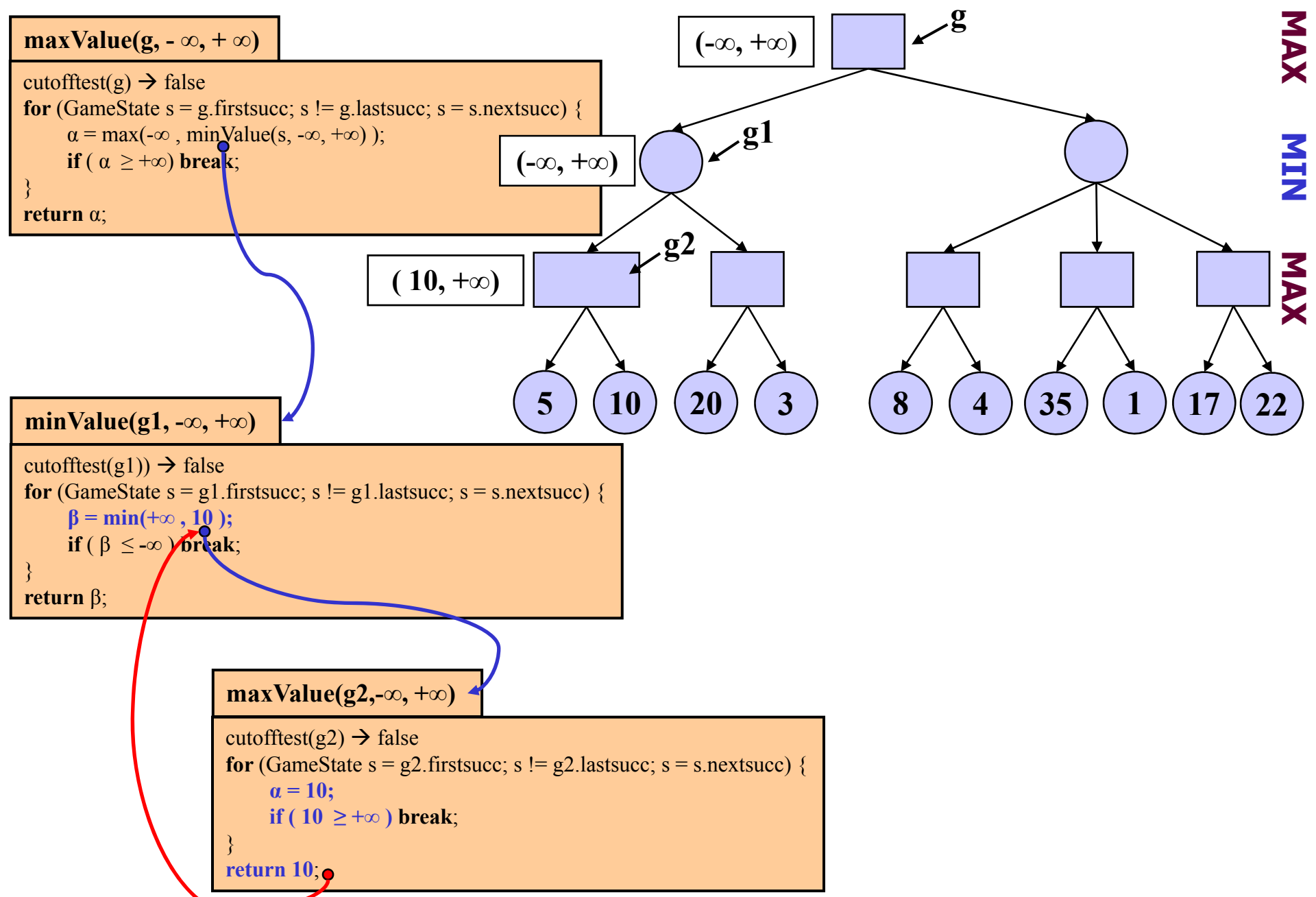



maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = 10;
if (10 ≥ +∞) break;
}
return α;
g2 has no more successors!
(-∞, +∞)
g
g1
(-∞, +∞)
(10, +∞)
g2
MAX
MIN
MAX
5
10
20
3
8
4
35
1
17
22


maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = 10;
if (10 ≥ +∞) break;
}
return α;
g2 has no more successors!
(-∞, +∞)
g
g1
(-∞, +∞)
(10, +∞)
g2
MAX
MIN
MAX
5
10
20
3
8
4
35
1
17
22


maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = 10;
if (10 ≥ +∞) break;
}
return 10;
g2 has no more successors!
(-∞, +∞)
g
g1
(-∞, +∞)
(10, +∞)
g2
MAX
MIN
MAX
5
10
20
3
8
4
35
1
17
22


maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, -∞, +∞));
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = 10;
if (10 ≥ +∞) break;
}
return 10;
g2 has no more successors!
(-∞, +∞)
g
g1
(-∞, +∞)
(10, +∞)
g2
MAX
MIN
MAX
5
10
20
3
8
4
35
1
17
22

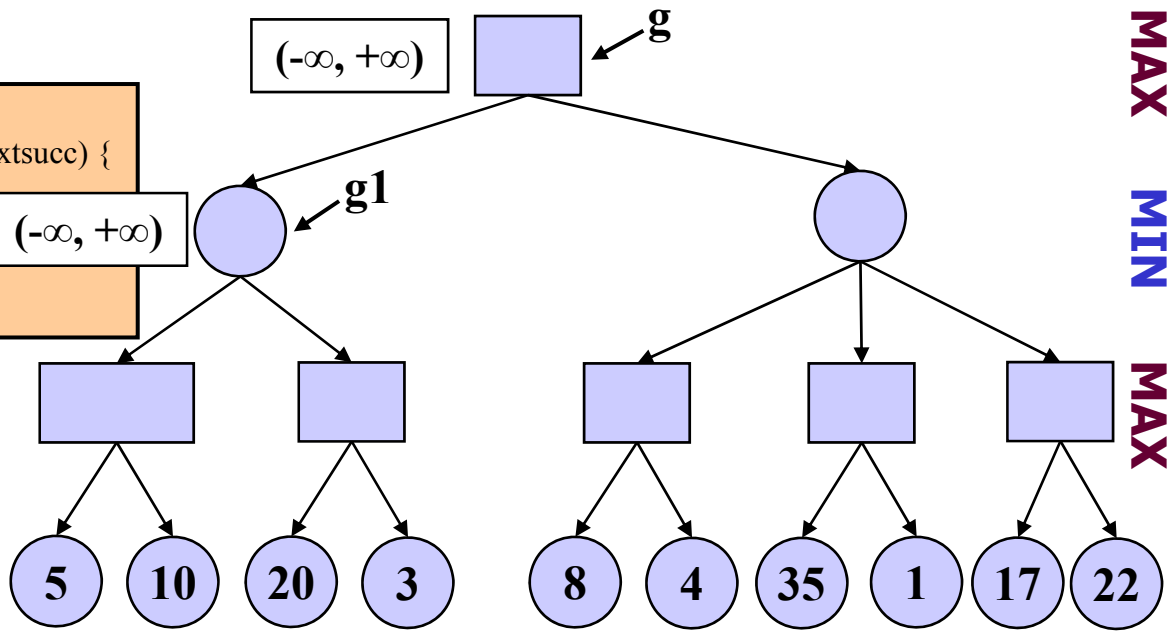

maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , 10);
if (β ≤ -∞) break;
}
return β;
maxValue(g2,-∞, +∞)
cutofftest(g2) → false
for (GameState s = g2.firstsucc; s != g2.lastsucc; s = s.nextsucc) {
α = 10;
if (10 ≥ +∞) break;
}
return 10;
(-∞, +∞)
g
g1
(-∞, +∞)
g2
(10, +∞)
MAX
MIN
MAX
5
10
20
3
8
4
35
1
17
22

maxValue(g, - ∞, + ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(-∞ , minValue(s, -∞, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```

minValue(g1, -∞, +∞)

```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = min(+∞ , 10 );
        if ( β ≤ -∞ ) break;
}
return β;
```

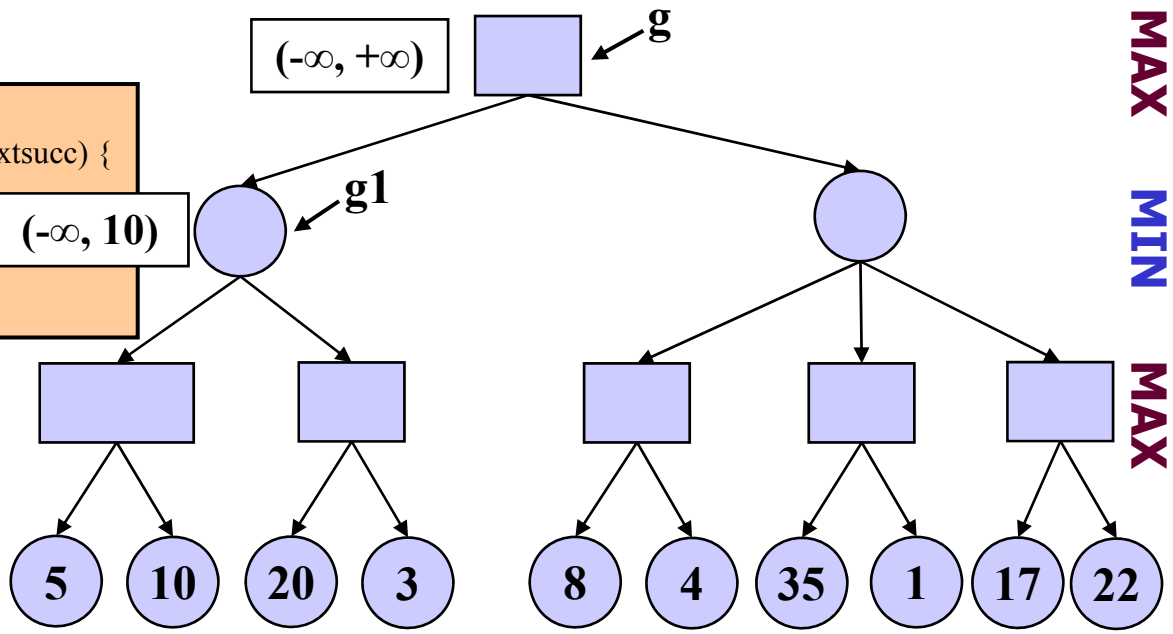


maxValue(g, - ∞, + ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(-∞ , minValue(s, -∞, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```

minValue(g1, -∞, +∞)

```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = 10;
        if ( 10 ≤ -∞ ) break;
}
return β;
```



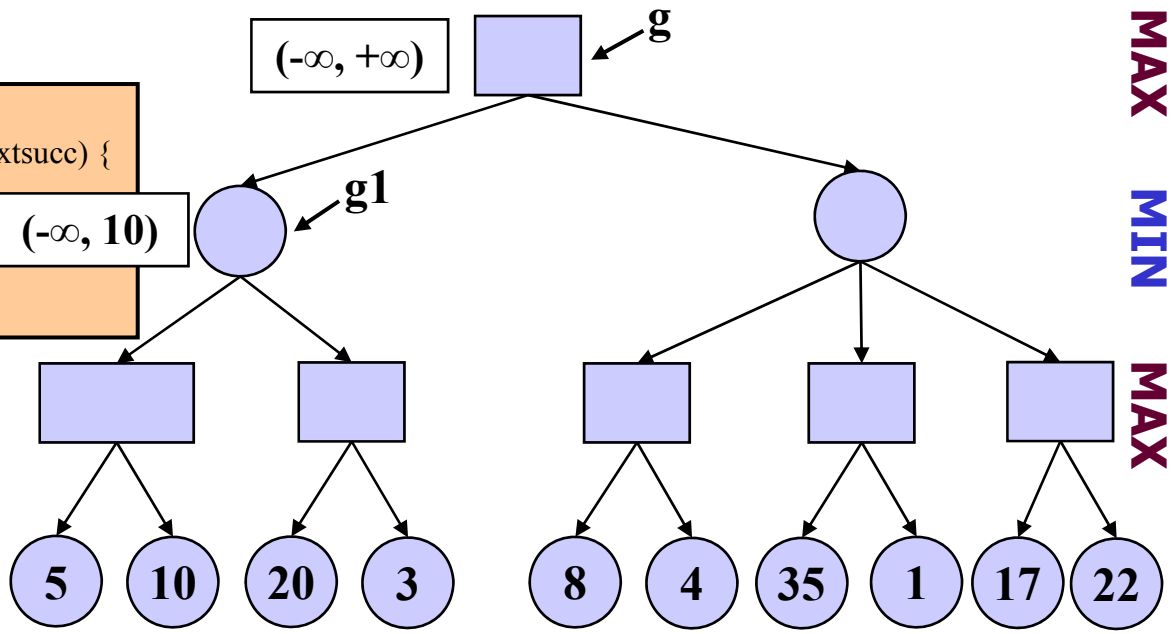
maxValue(g, - ∞, + ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(-∞ , minValue(s, -∞, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```

minValue(g1, -∞, +∞)

```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = 10;
        if ( 10 ≤ -∞ ) break;
}
return β;
```

false!



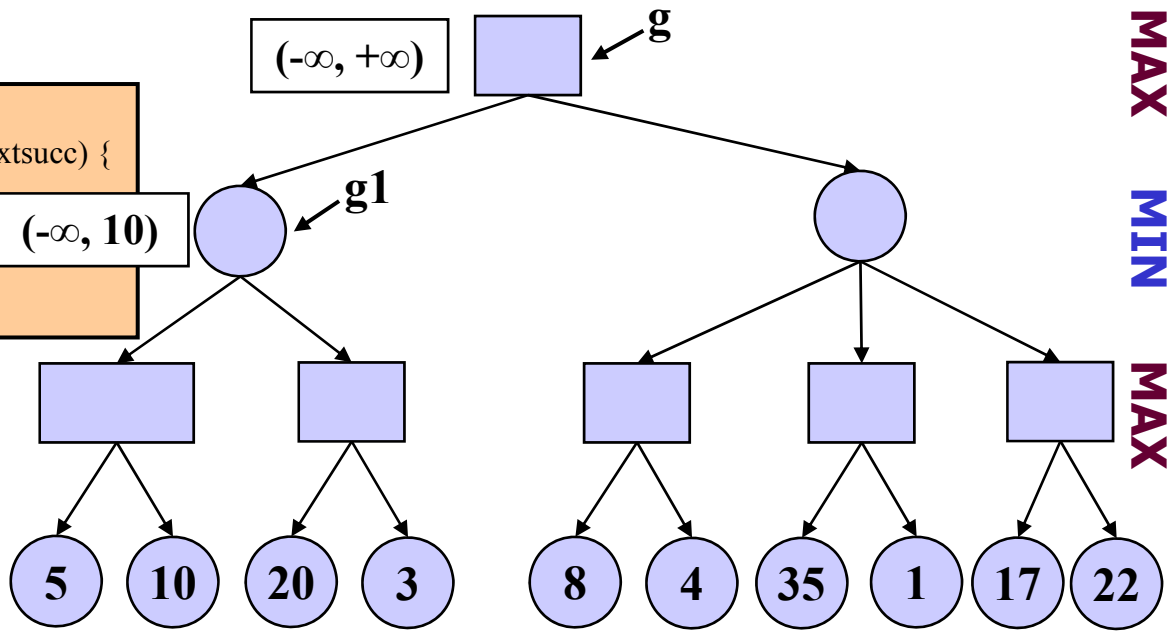
maxValue(g, - ∞, + ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(-∞ , minValue(s, -∞, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```

minValue(g1, -∞, +∞)

```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(10 , maxValue(s, -∞, 10) );
      if ( β ≤ -∞ ) break;
}
return β;
```

next successor



maxValue(g, - ∞, + ∞)

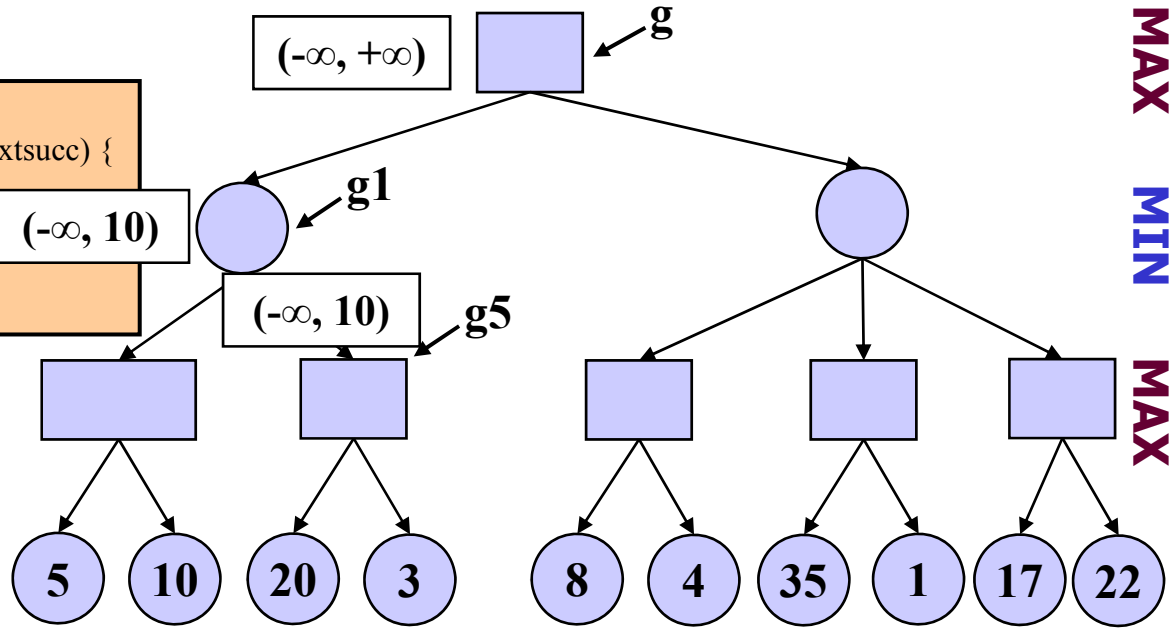
```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(-∞ , minValue(s, -∞, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```

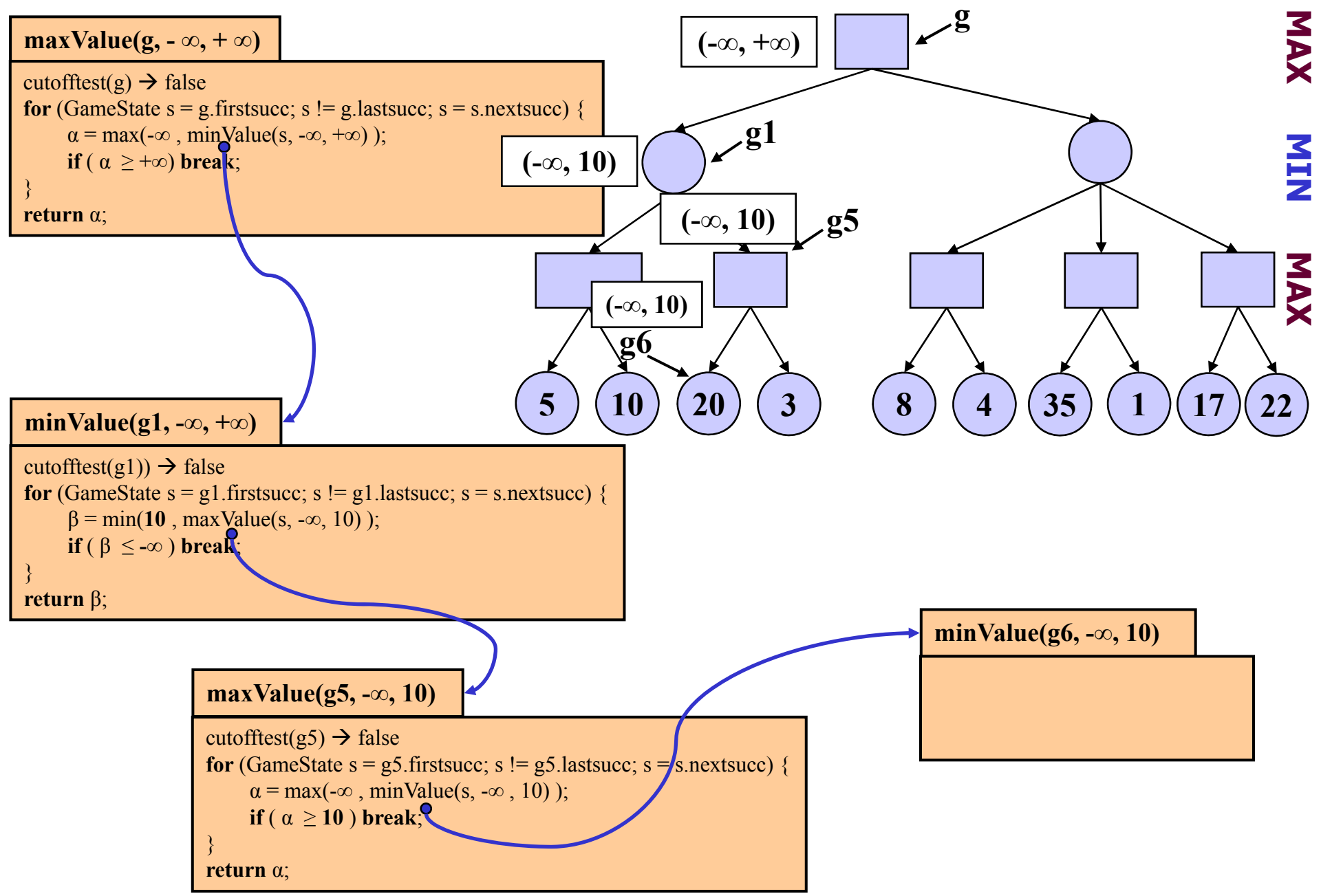
minValue(g1, -∞, +∞)

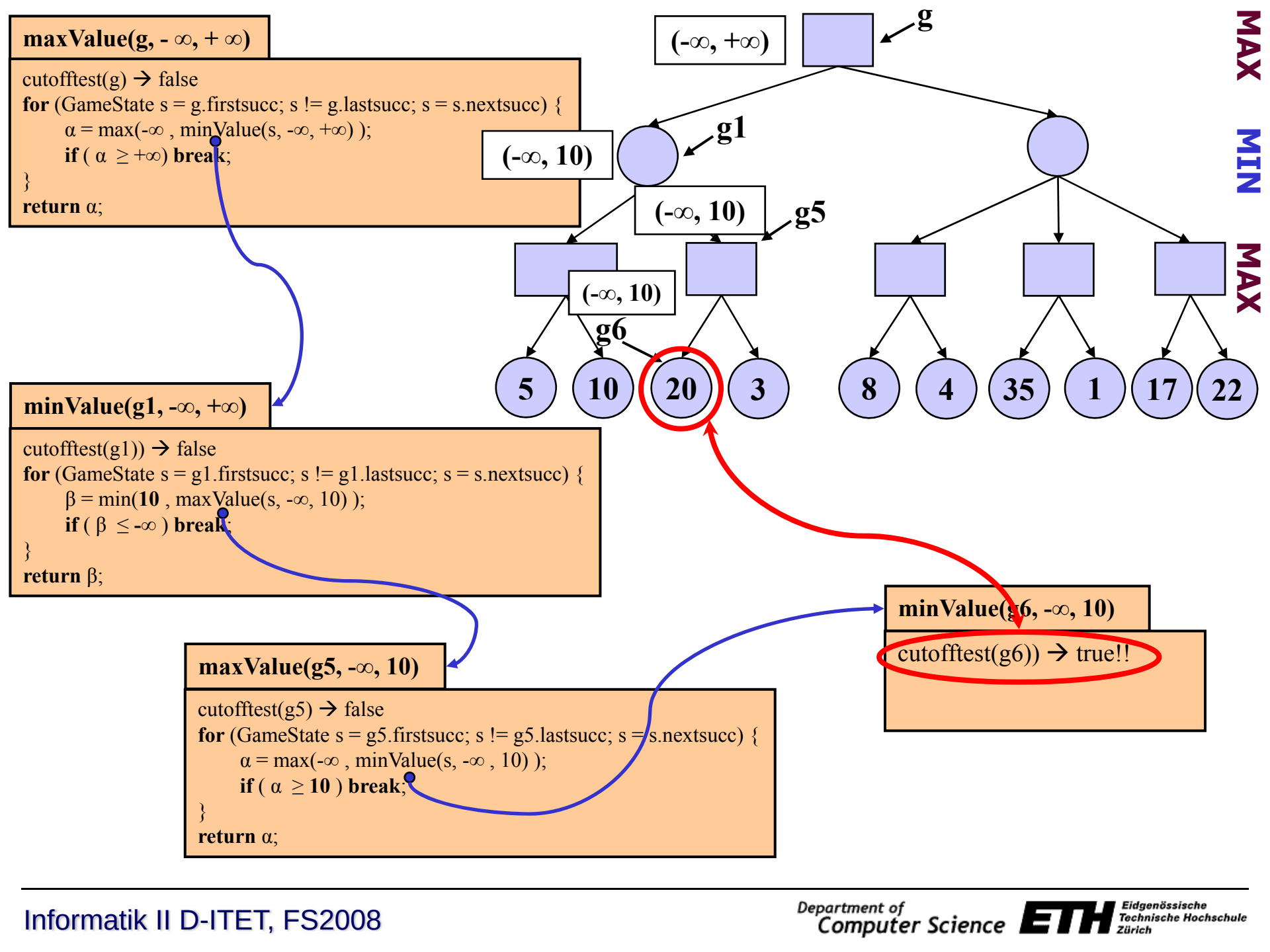
```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(10 , maxValue(s, -∞, 10) );
      if ( β ≤ -∞ ) break;
}
return β;
```

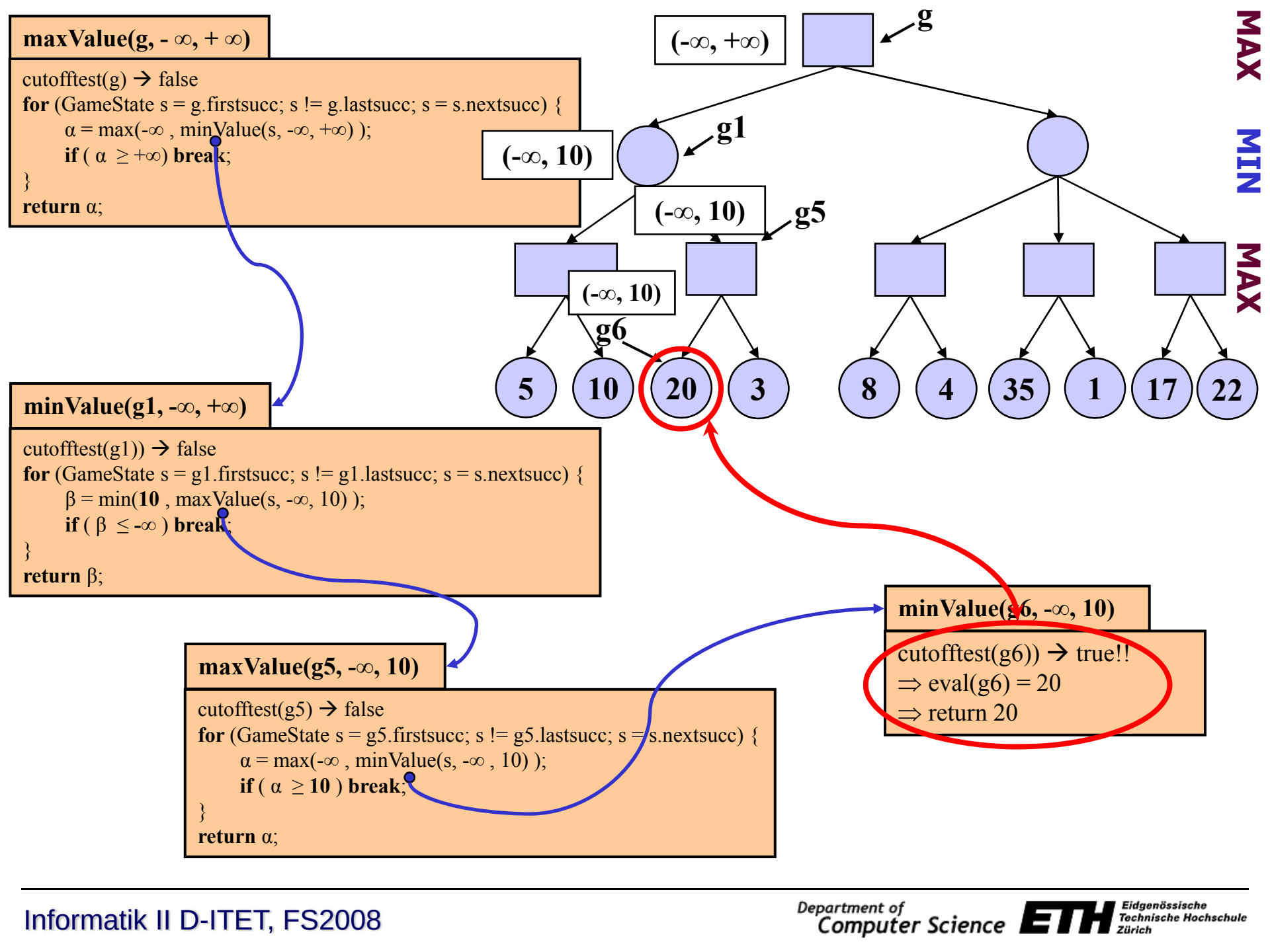
maxValue(g5, -∞, 10)

```
cutofftest(g5) → false
for (GameState s = g5.firstsucc; s != g5.lastsucc; s = s.nextsucc) {
      α = max(-∞ , minValue(s, -∞ , 10) );
      if ( α ≥ 10 ) break;
}
return α;
```



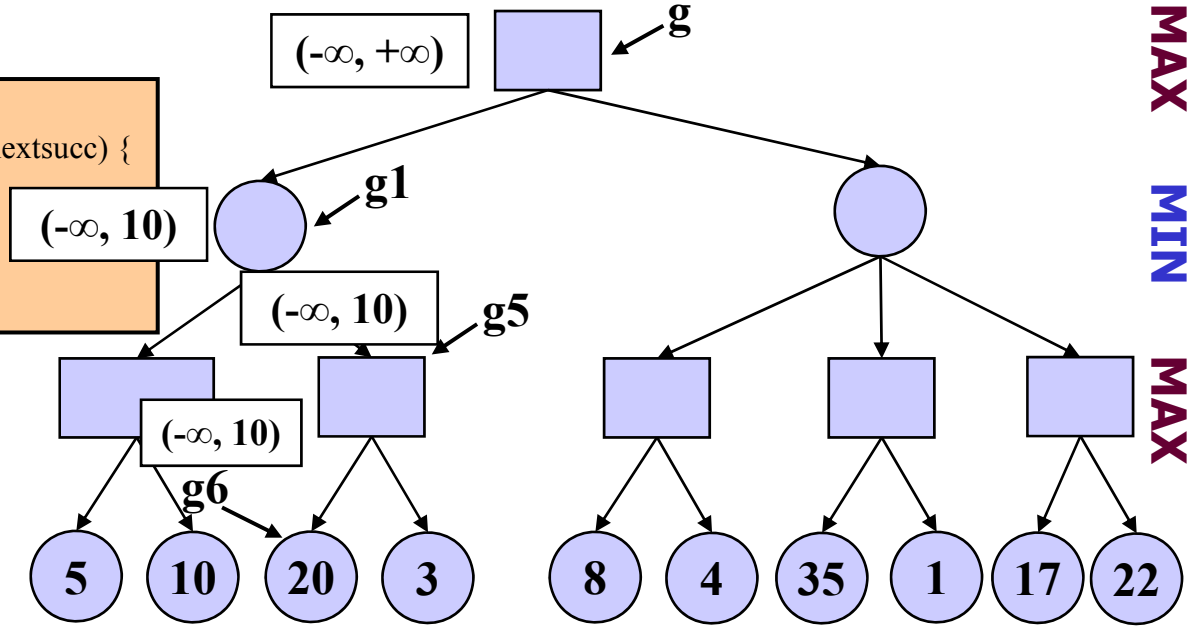

maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
(-∞, +∞)
g
MAX
g1
(-∞, 10)
MIN
(-∞, 10)
g5
MAX
(-∞, 10)
g6
5
10
20
3
8
4
35
1
17
22
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(10 , maxValue(s, -∞, 10));
if (β ≤ -∞) break;
}
return β;
maxValue(g5, -∞, 10)
cutofftest(g5) → false
for (GameState s = g5.firstsucc; s != g5.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞ , 10));
if (α ≥ 10) break;
}
return α;
minValue(g6, -∞, 10)


maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
(-∞, +∞)
g
MAX
(-∞, 10)
g1
MIN
(-∞, 10)
g5
MAX
(-∞, 10)
g6
5
10
20
3
8
4
35
1
17
22
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(10 , maxValue(s, -∞, 10));
if (β ≤ -∞) break;
}
return β;
maxValue(g5, -∞, 10)
cutofftest(g5) → false
for (GameState s = g5.firstsucc; s != g5.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞ , 10));
if (α ≥ 10) break;
}
return α;
minValue(g6, -∞, 10)
cutofftest(g6)) → true!!


maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(10 , maxValue(s, -∞, 10));
if (β ≤ -∞) break;
}
return β;
maxValue(g5, -∞, 10)
cutofftest(g5) → false
for (GameState s = g5.firstsucc; s != g5.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞ , 10));
if (α ≥ 10) break;
}
return α;
minValue(g6, -∞, 10)
cutofftest(g6)) → true!!
⇒ eval(g6) = 20
⇒ return 20
(-∞, +∞)
g
g1
(-∞, 10)
(-∞, 10)
g5
(-∞, 10)
g6
5
10
20
3
8
4
35
1
17
22
MAX
MIN
MAX

maxValue(g, - ∞, + ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(-∞ , minValue(s, -∞, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```



minValue(g1, -∞, +∞)

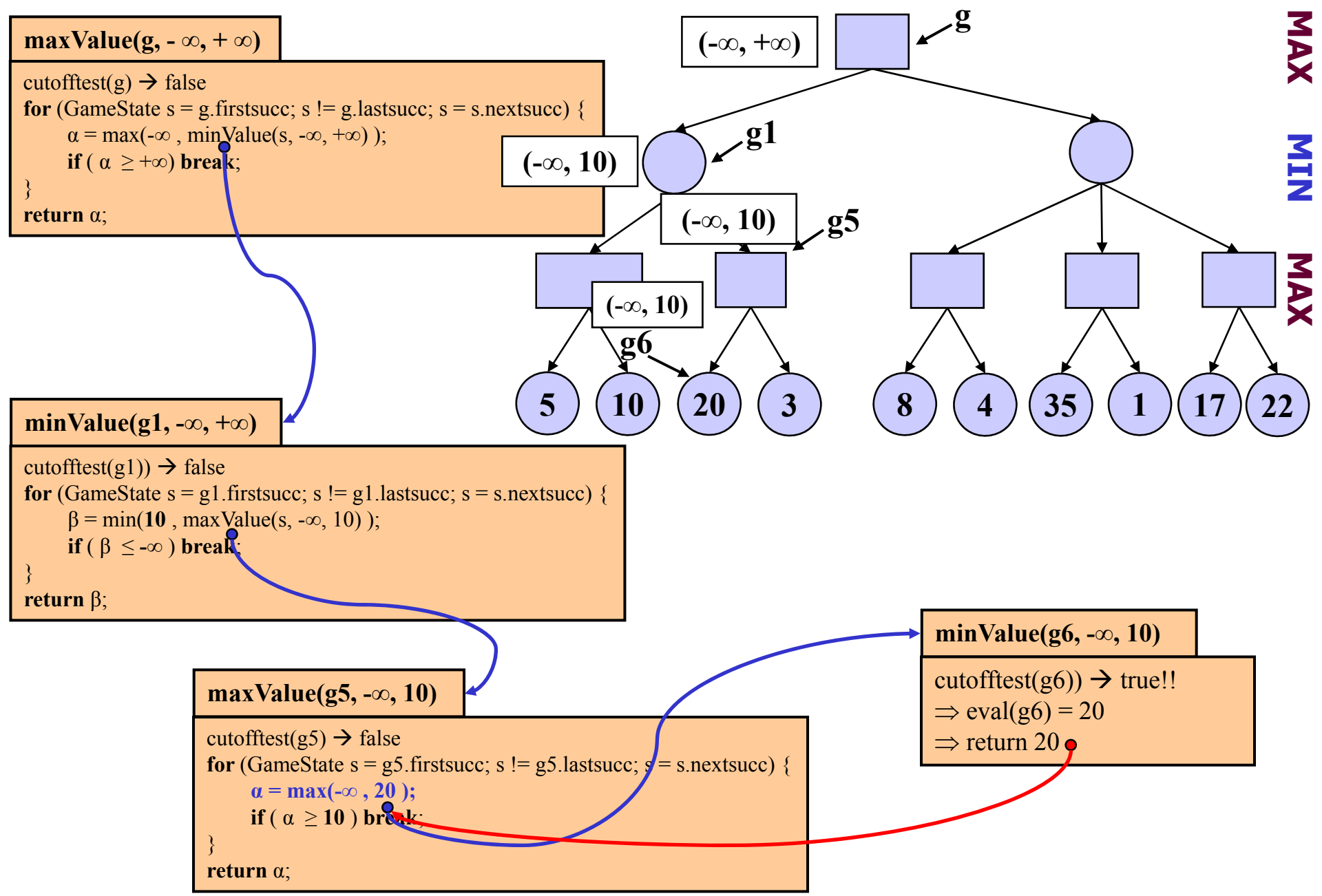
```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(10 , maxValue(s, -∞, 10) );
      if ( β ≤ -∞ ) break;
}
return β;
```

maxValue(g5, -∞, 10)

```
cutofftest(g5) → false
for (GameState s = g5.firstsucc; s != g5.lastsucc; s = s.nextsucc) {
      α = max(-∞, minValue(s, -∞ , 10) );
      if ( α ≥ 10 ) break;
}
return α;
```

minValue(g6, -∞, 10)

```
cutofftest(g6)) → true!!
⇒ eval(g6) = 20
⇒ return 20
```


maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
(-∞, +∞)
g
MAX
(-∞, 10)
g1
MIN
(-∞, 10)
g5
(-∞, 10)
g6
MAX
5
10
20
3
8
4
35
1
17
22
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(10 , maxValue(s, -∞, 10));
if (β ≤ -∞) break;
}
return β;
maxValue(g5, -∞, 10)
cutofftest(g5) → false
for (GameState s = g5.firstsucc; s != g5.lastsucc; s = s.nextsucc) {
α = max(-∞ , 20);
if (α ≥ 10) break;
}
return α;
minValue(g6, -∞, 10)
cutofftest(g6)) → true!!
⇒ eval(g6) = 20
⇒ return 20

maxValue(g, - ∞, + ∞)

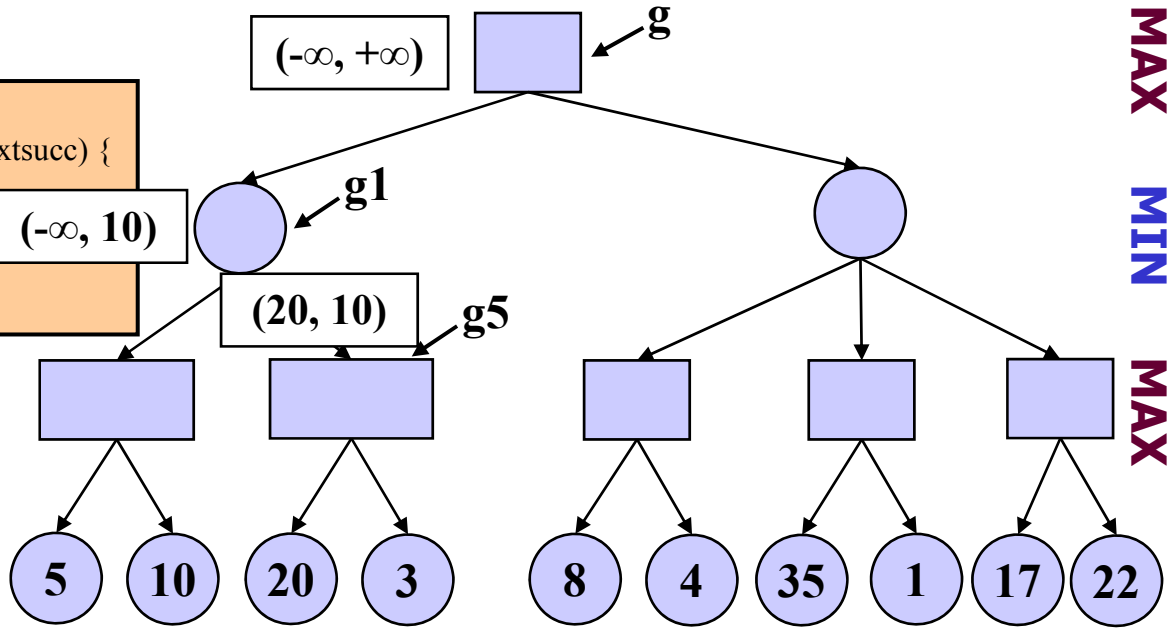
```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(-∞ , minValue(s, -∞, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```

minValue(g1, -∞, +∞)

```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(10 , maxValue(s, -∞, 10) );
      if ( β ≤ -∞ ) break;
}
return β;
```

maxValue(g5, -∞, 10)

```
cutofftest(g5) → false
for (GameState s = g5.firstsucc; s != g5.lastsucc; s = s.nextsucc) {
      α = 20;
      if ( α ≥ 10 ) break;
}
return α;
```



maxValue(g, - ∞, + ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(-∞ , minValue(s, -∞, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```

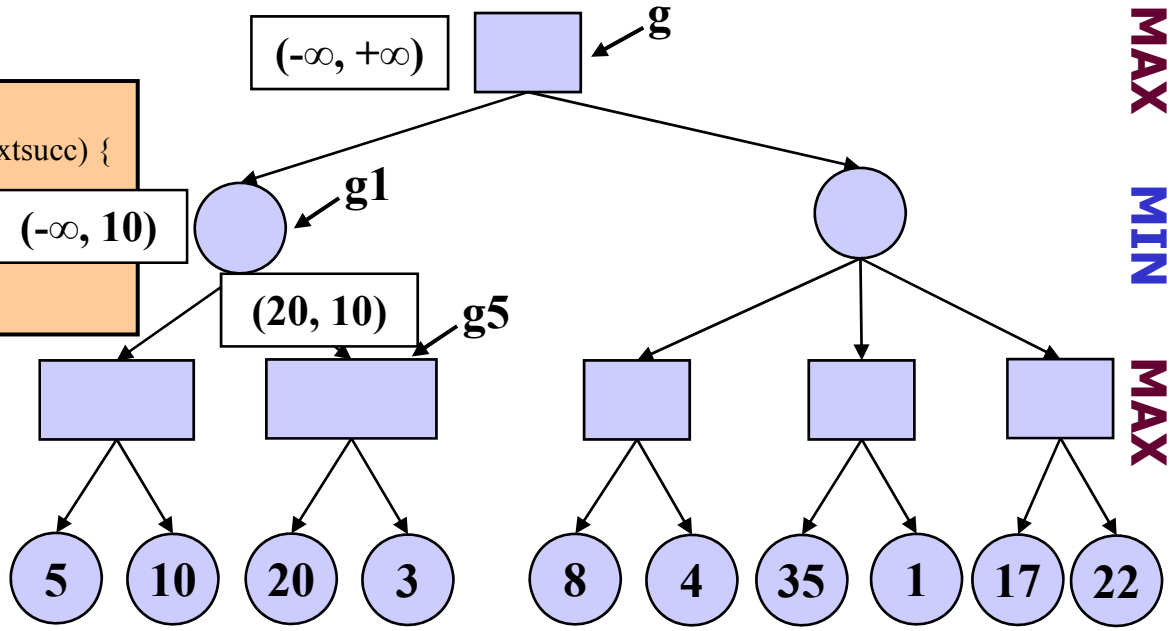
minValue(g1, -∞, +∞)

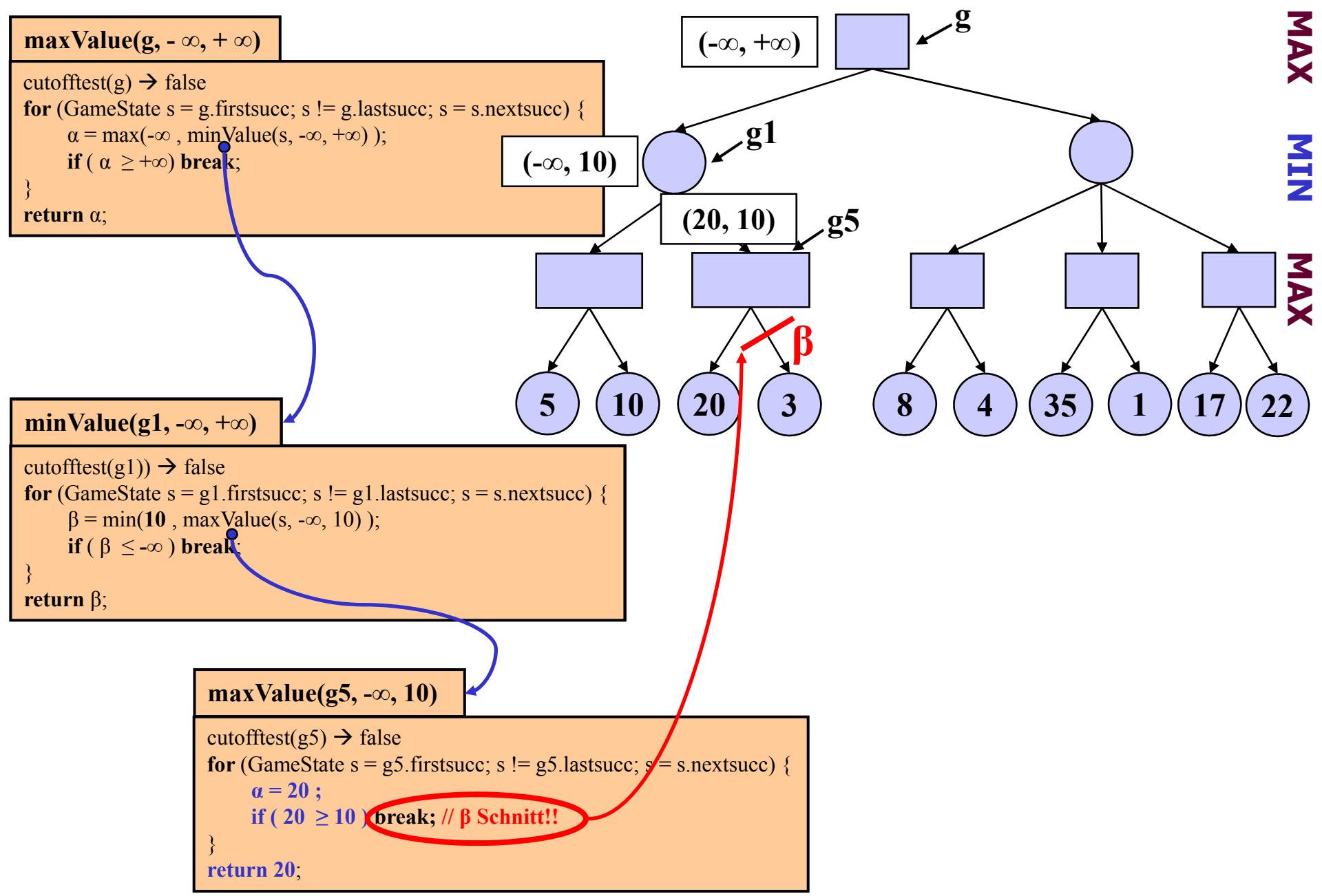
```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(10 , maxValue(s, -∞, 10) );
      if ( β ≤ -∞ ) break;
}
return β;
```

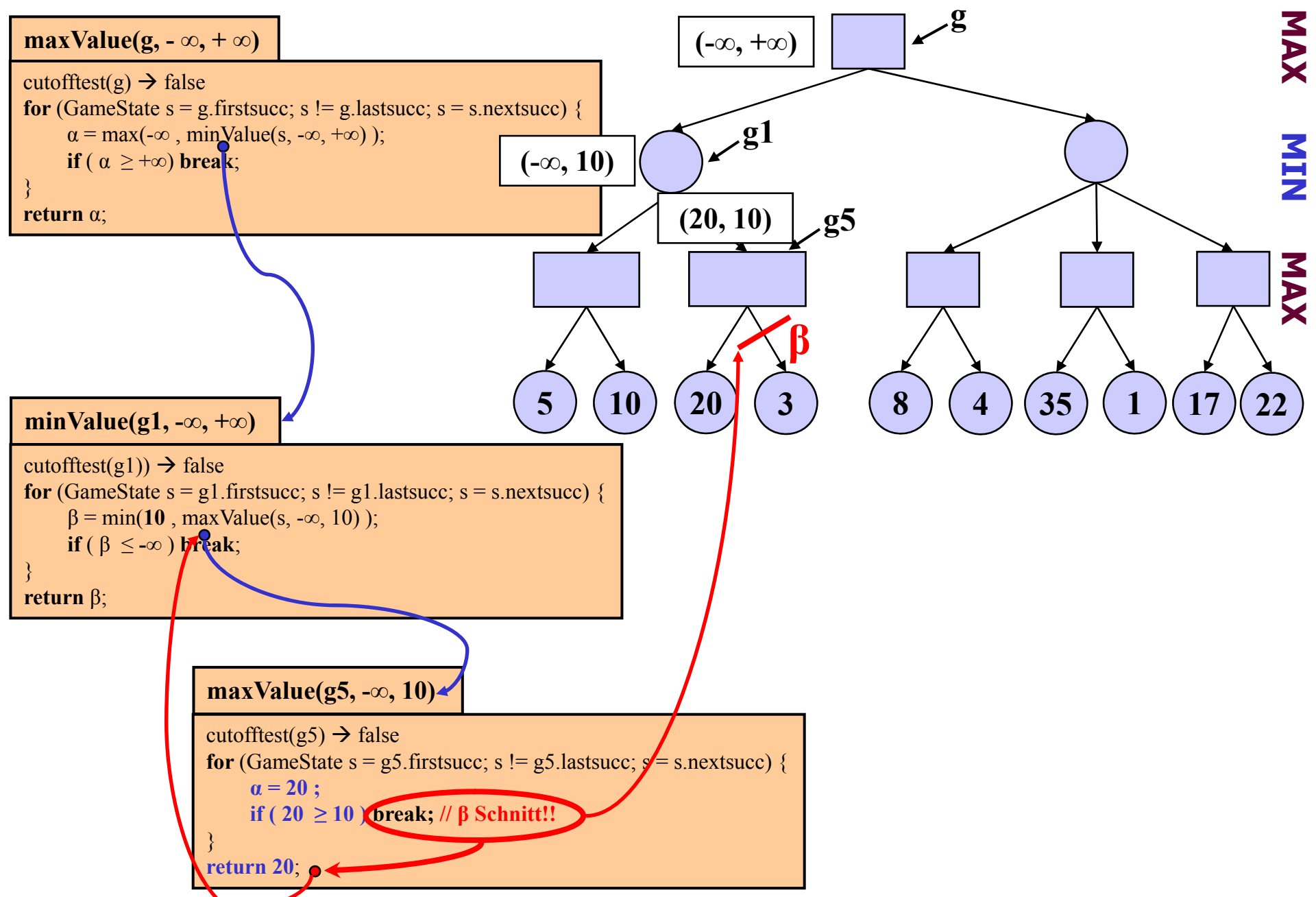
maxValue(g5, -∞, 10)

```
cutofftest(g5) → false
for (GameState s = g5.firstsucc; s != g5.lastsucc; s = s.nextsucc) {
      α = 20 ;
      if ( 20 ≥ 10 ) break;
}
return α;
```

true!




maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(10 , maxValue(s, -∞, 10));
if (β ≤ -∞) break;
}
return β;
maxValue(g5, -∞, 10)
cutofftest(g5) → false
for (GameState s = g5.firstsucc; s != g5.lastsucc; s = s.nextsucc) {
α = 20 ;
if (20 ≥ 10) break; // β Schnitt!!
}
return 20;
(-∞, +∞)
g
(-∞, 10)
g1
(20, 10)
g5
β
MAX
MIN
MAX
5
10
20
3
8
4
35
1
17
22


maxValue(g, - ∞, + ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(-∞ , minValue(s, -∞, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g1, -∞, +∞)
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(10 , maxValue(s, -∞, 10));
if (β ≤ -∞) break;
}
return β;
maxValue(g5, -∞, 10)
cutofftest(g5) → false
for (GameState s = g5.firstsucc; s != g5.lastsucc; s = s.nextsucc) {
α = 20 ;
if (20 ≥ 10) break; // β Schnitt!!
}
return 20;
(-∞, +∞)
g
(-∞, 10)
g1
(20, 10)
g5
β
MAX
MIN
MAX
5
10
20
3
8
4
35
1
17
22

maxValue(g, - ∞, + ∞)

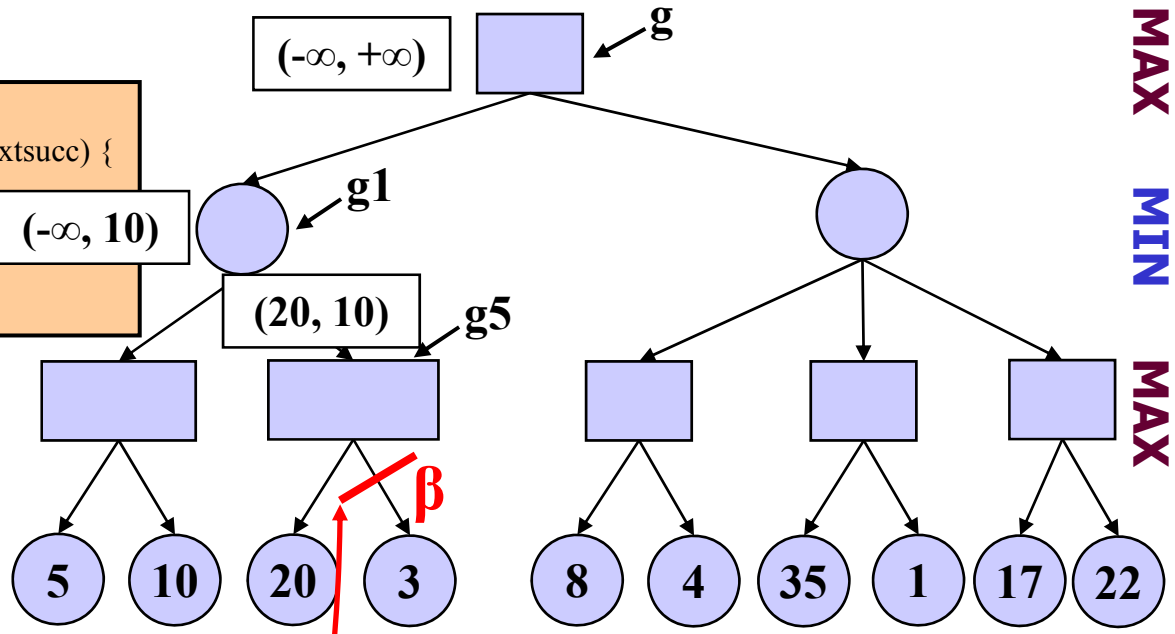
```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(-∞ , minValue(s, -∞, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```

minValue(g1, -∞, +∞)

```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(10 , 20 );
      if ( β ≤ - ∞ ) break;
}
return β;
```

maxValue(g5, -∞, 10)

```
cutofftest(g5) → false
for (GameState s = g5.firstsucc; s != g5.lastsucc; s = s.nextsucc) {
      α = 20 ;
      if ( 20 ≥ 10 ) break; // β Schnitt!!
}
return 20;
```

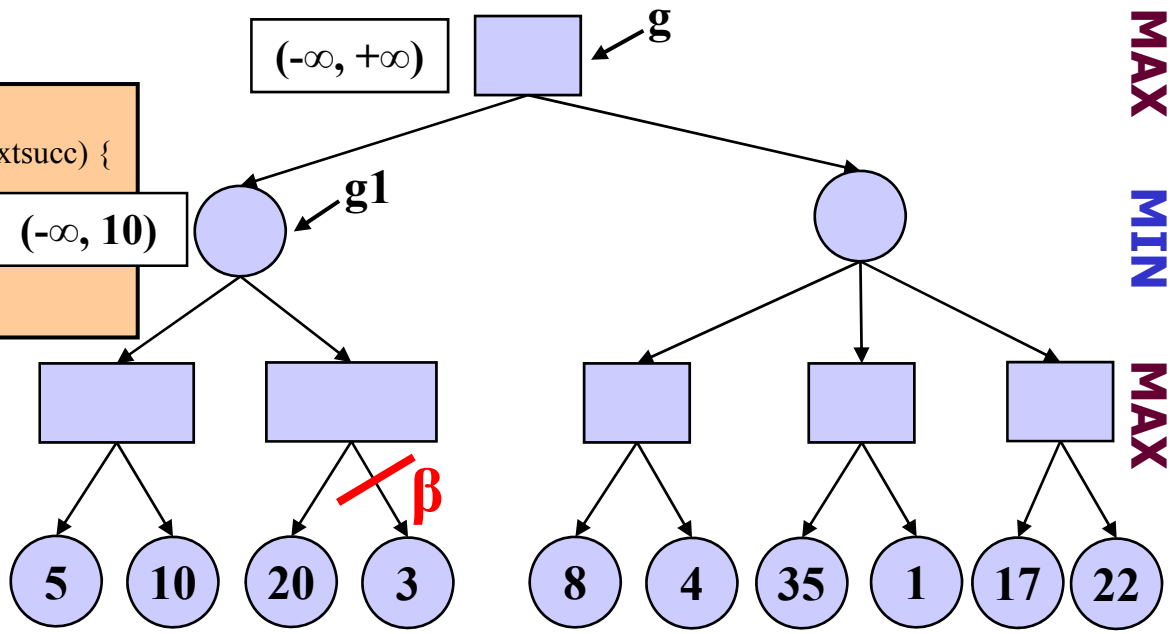


maxValue(g, - ∞, + ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(-∞ , minValue(s, -∞, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```

minValue(g1, -∞, +∞)

```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = 10;
        if ( 10 ≤ - ∞ ) break;
}
return β;
```



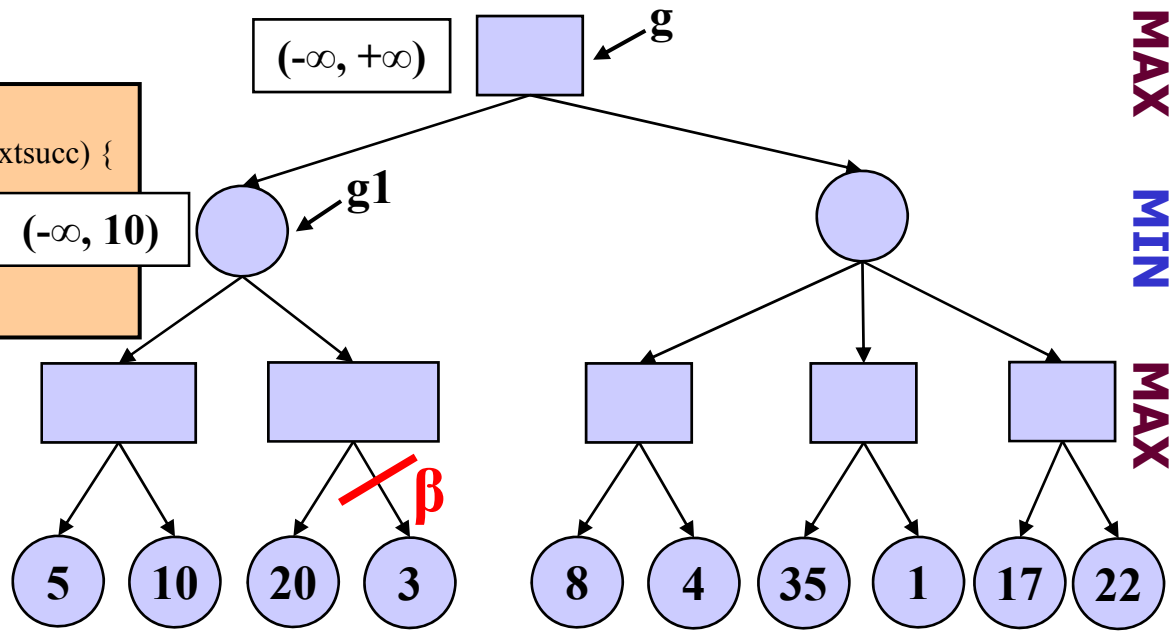
maxValue(g, - ∞, + ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(-∞ , minValue(s, -∞, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```

minValue(g1, -∞, +∞)

```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = 10;
        if ( 10 ≤ - ∞ ) break;
}
return β;
```

false!

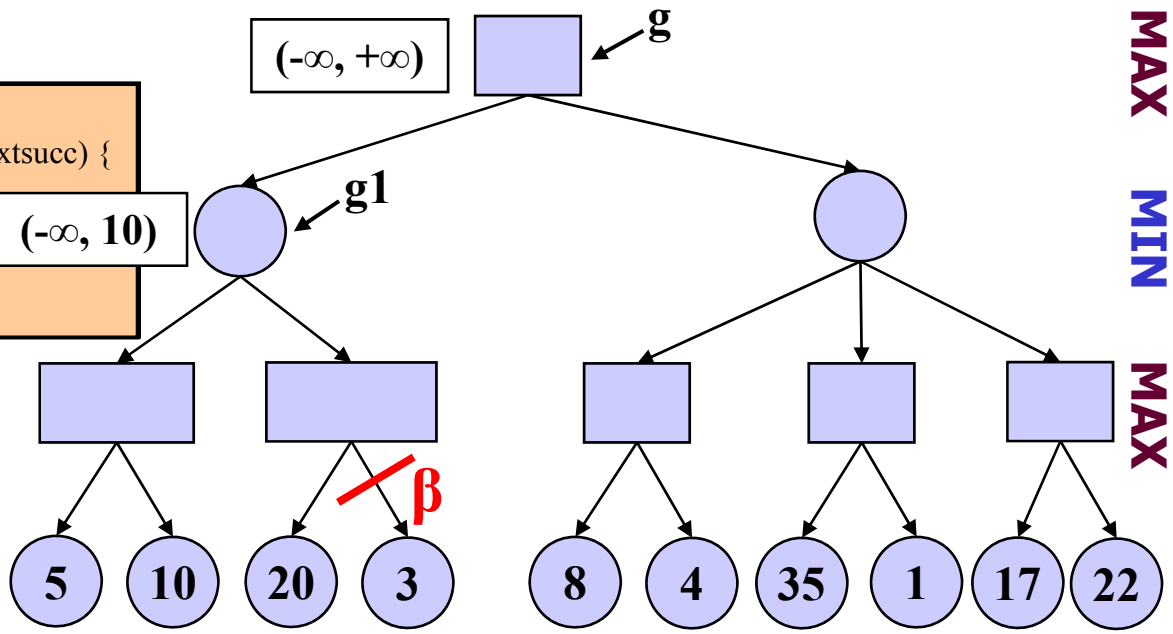


maxValue(g, - ∞, + ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(-∞ , minValue(s, -∞, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```

minValue(g1, -∞, +∞)

```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = 10;
      if ( 10 ≤ - ∞ ) break;
}
return β;
```



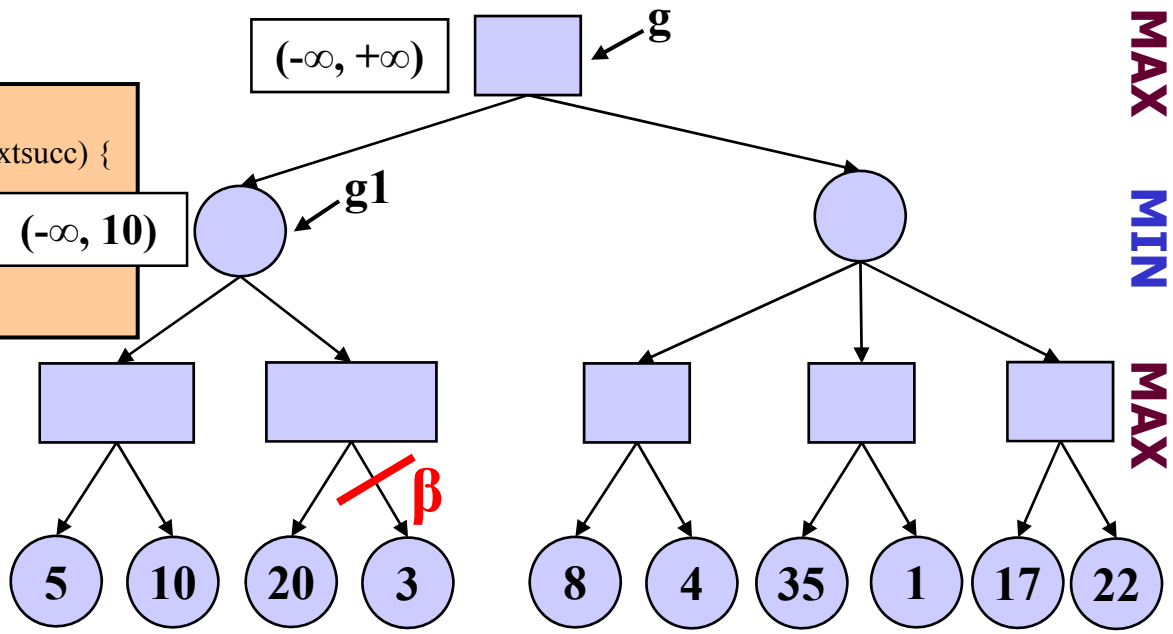
g1 has no more successors!

maxValue(g, - ∞, + ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(-∞ , minValue(s, -∞, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```

minValue(g1, -∞, +∞)

```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = 10;
      if ( 10 ≤ - ∞ ) break;
}
return 10;
```



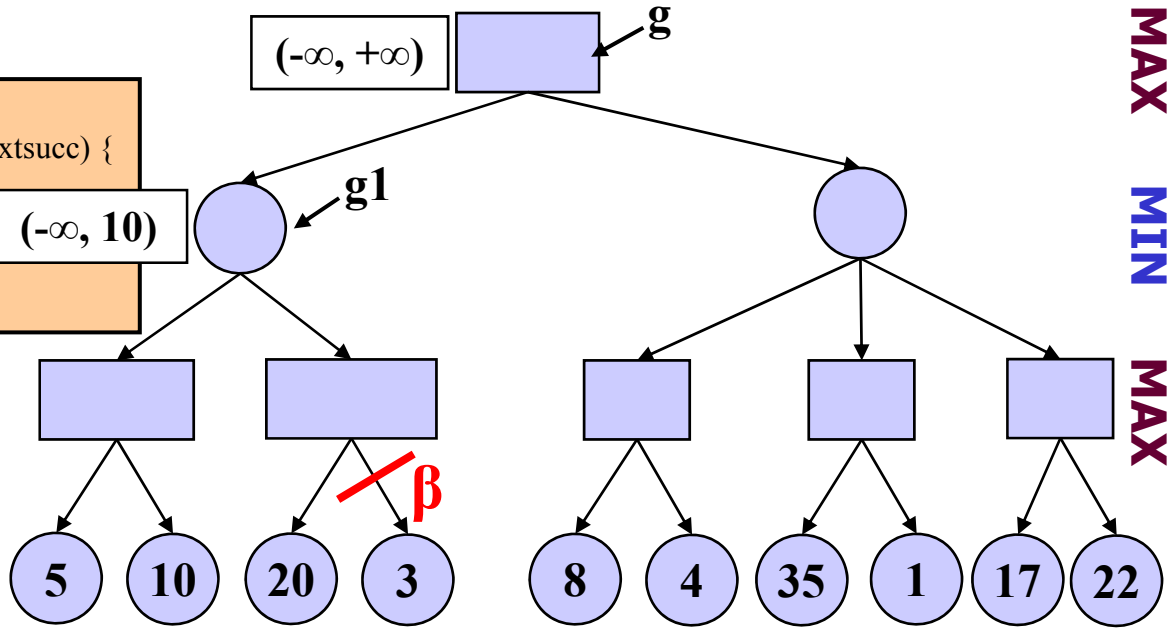
g1 has no more successors!

maxValue(g, - ∞, + ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(-∞ , 10 );
      if ( α ≥ +∞) break;
}
return α;
```

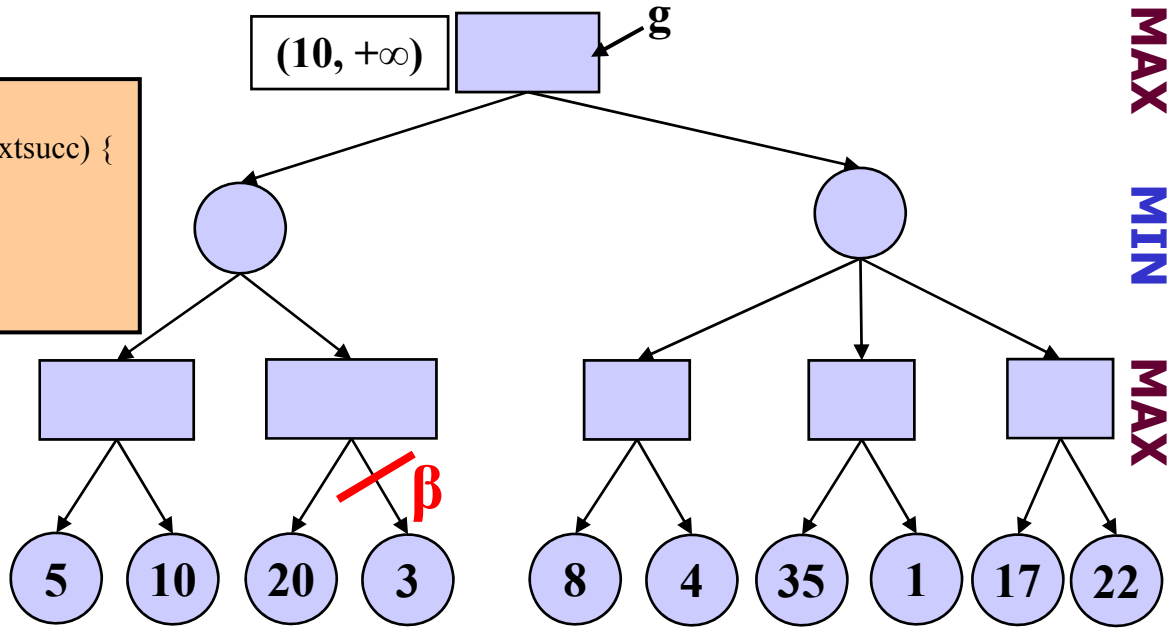
minValue(g1, -∞, +∞)

```
cutofftest(g1)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = 10;
      if ( 10 ≤ - ∞ ) break;
}
return 10;
```



maxValue(g, - ∞, + ∞)

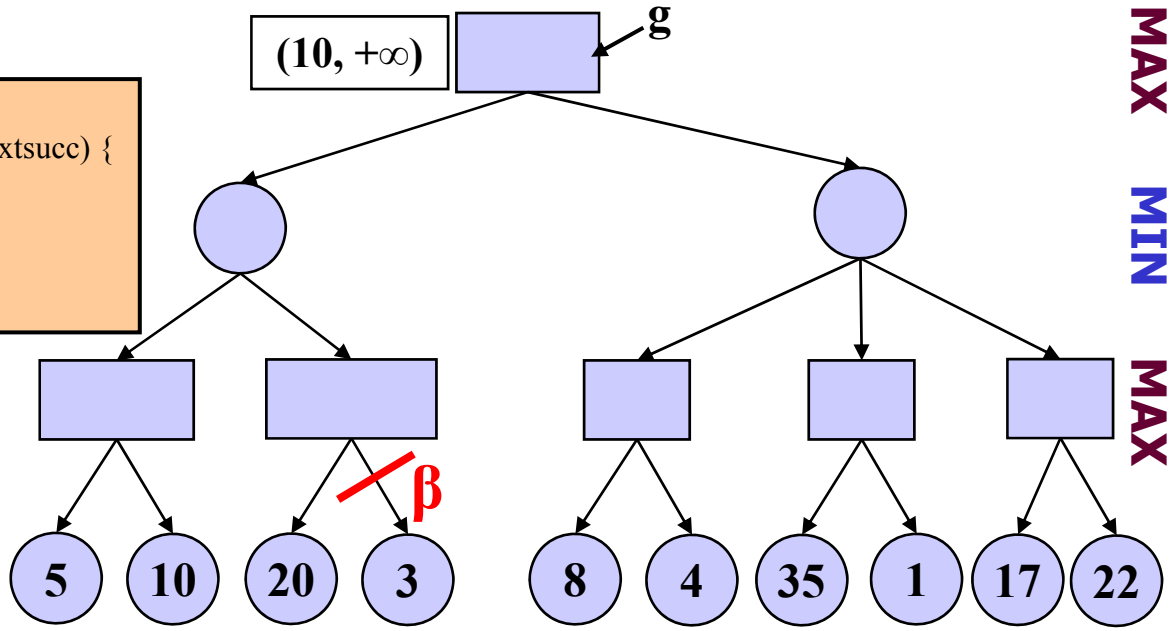
```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = 10;
      if ( α ≥ +∞) break;
}
return α;
```



maxValue(g, - ∞, + ∞)

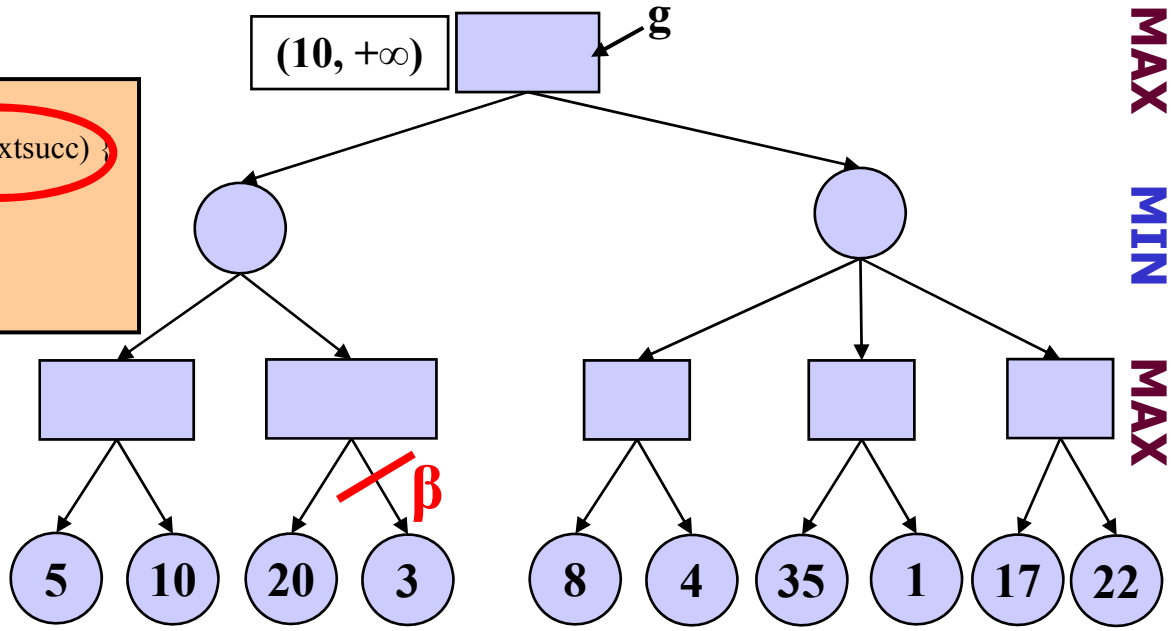
```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
    α = 10;
    if ( 10 ≥ +∞) break;
}
return α;
```

false!



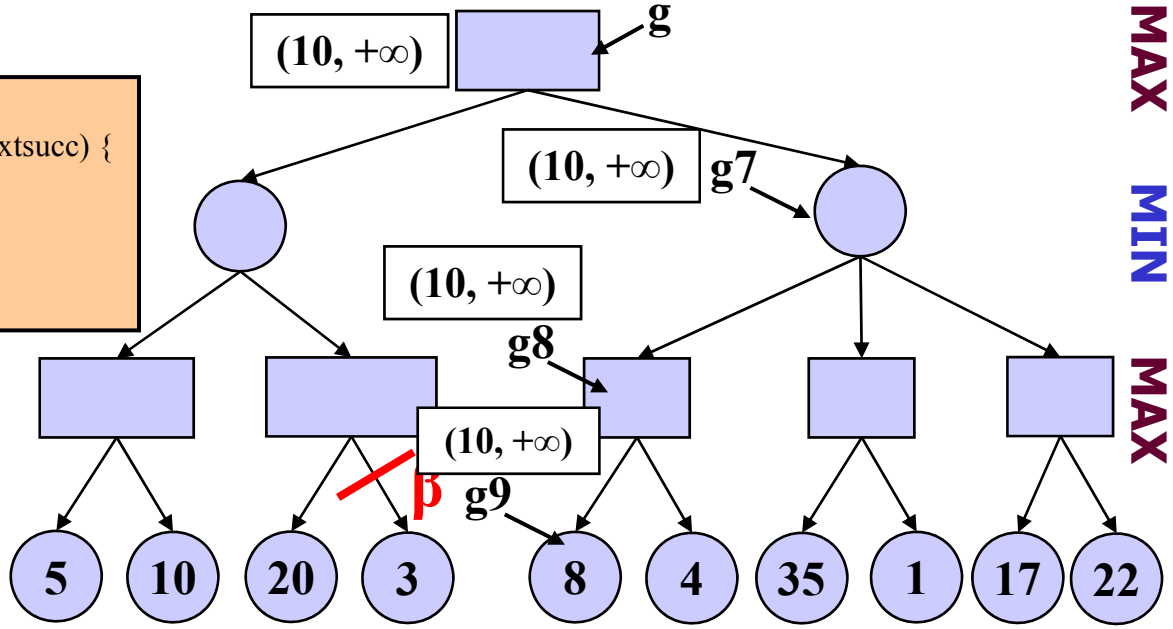
maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(10 , minValue(s, 10, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```



maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(10 , minValue(s, 10, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```



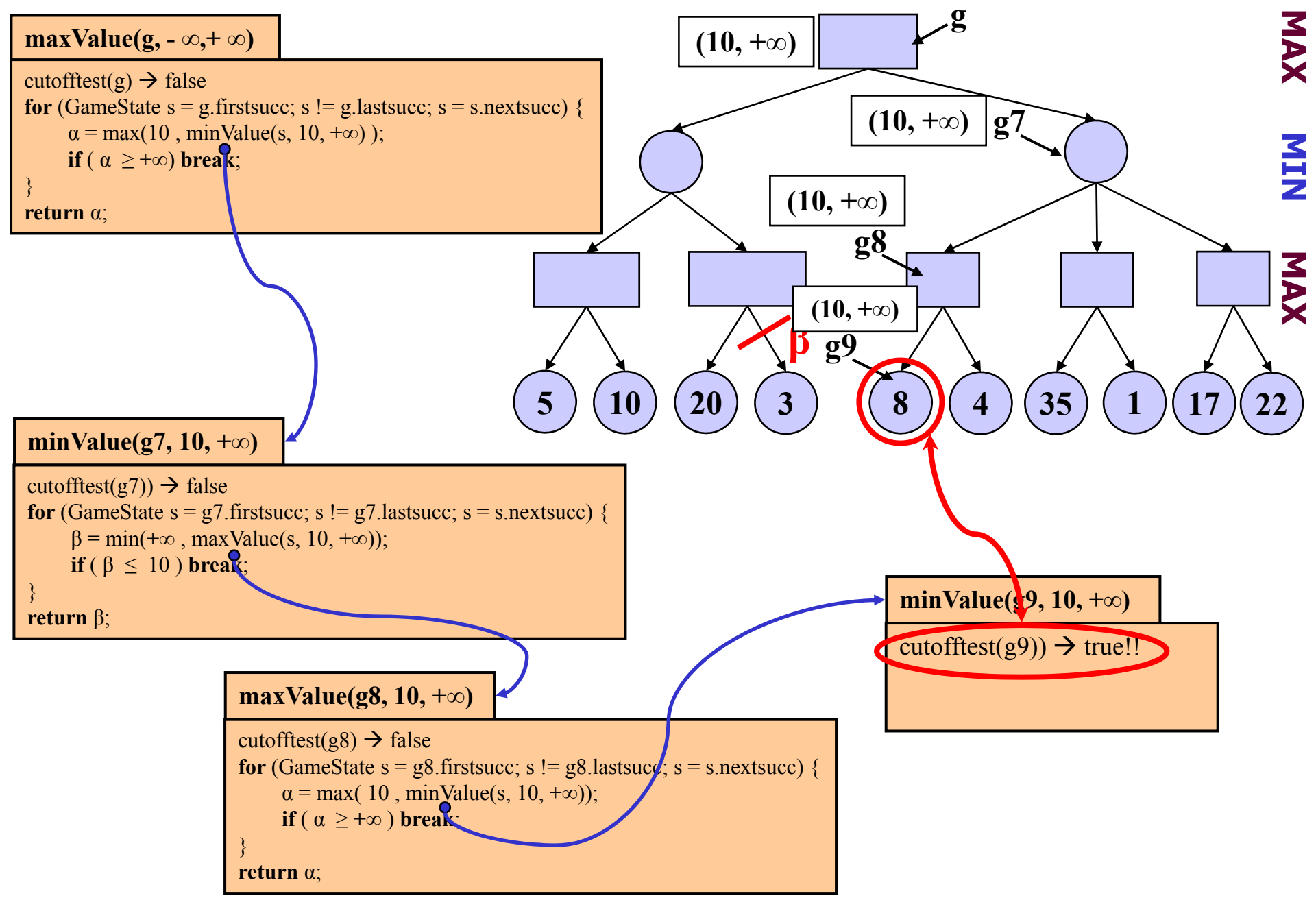
minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g7.firstsucc; s != g7.lastsucc; s = s.nextsucc) {
        β = min(+∞ , maxValue(s, 10, +∞));
        if ( β ≤ 10 ) break;
}
return β;
```

maxValue(g8, 10, +∞)

```
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
        α = max( 10 , minValue(s, 10, +∞));
        if ( α ≥ +∞ ) break;
}
return α;
```

minValue(g9, 10, +∞)


maxValue(g, - ∞,+ ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(10 , minValue(s, 10, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g7, 10, +∞)
cutofftest(g7)) → false
for (GameState s = g7.firstsucc; s != g7.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, 10, +∞));
if (β ≤ 10) break;
}
return β;
maxValue(g8, 10, +∞)
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
α = max(10 , minValue(s, 10, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g9, 10, +∞)
cutofftest(g9)) → true!!
(10, +∞)
g
(10, +∞)
g7
(10, +∞)
g8
(10, +∞)
β
g9
MAX
MIN
MAX
5
10
20
3
8
4
35
1
17
22

maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(10 , minValue(s, 10, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```

minValue(g7, 10, +∞)

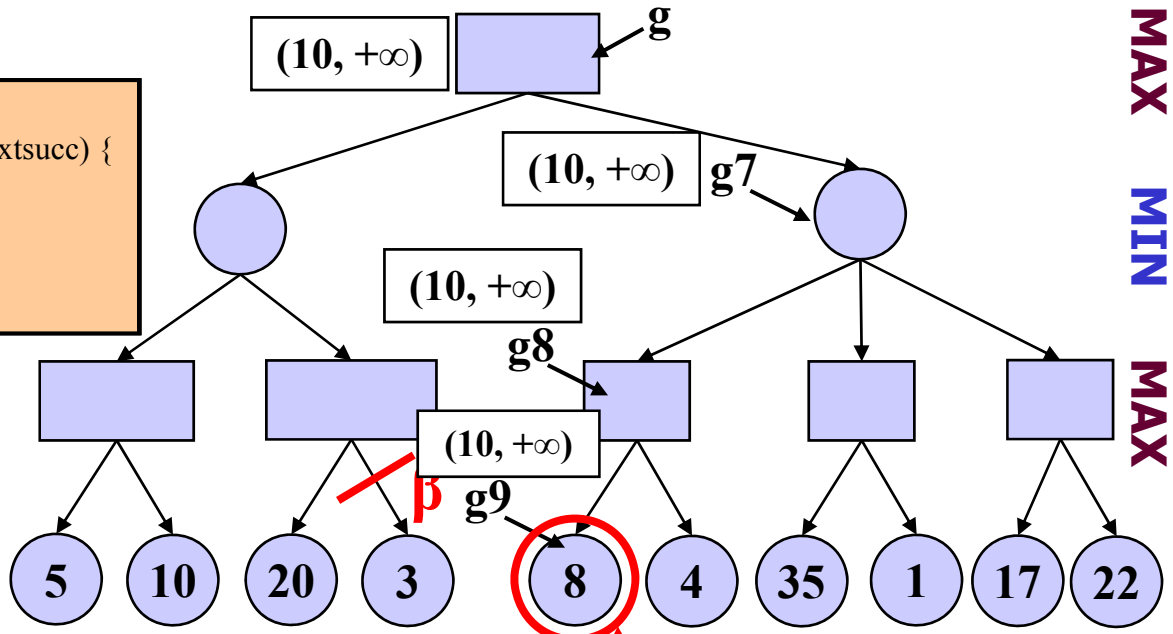
```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(+∞ , maxValue(s, 10, +∞));
      if ( β ≤ 10 ) break;
}
return β;
```

maxValue(g8, 10, +∞)

```
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
      α = max( 10 , minValue(s, 10, +∞));
      if ( α ≥ +∞) break;
}
return α;
```

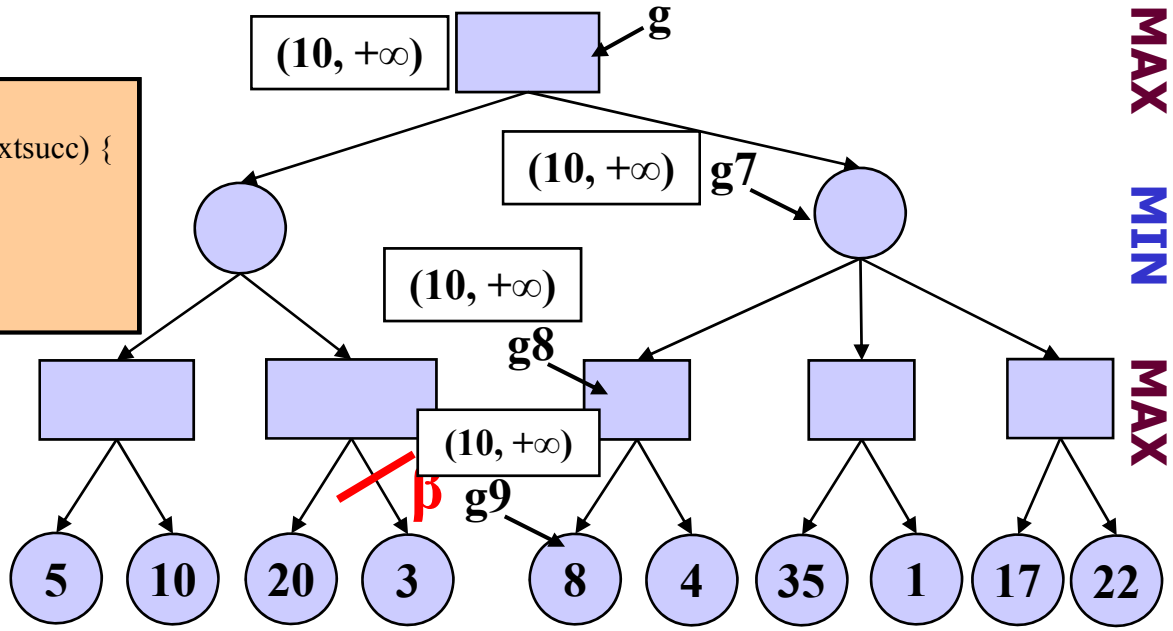
minValue(g9, 10, +∞)

cutofftest(g9)) → true!!
⇒ eval(g9) = 8
⇒ return 8



maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(10 , minValue(s, 10, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```



minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(+∞ , maxValue(s, 10, +∞));
      if ( β ≤ 10 ) break;
}
return β;
```

maxValue(g8, 10, +∞)

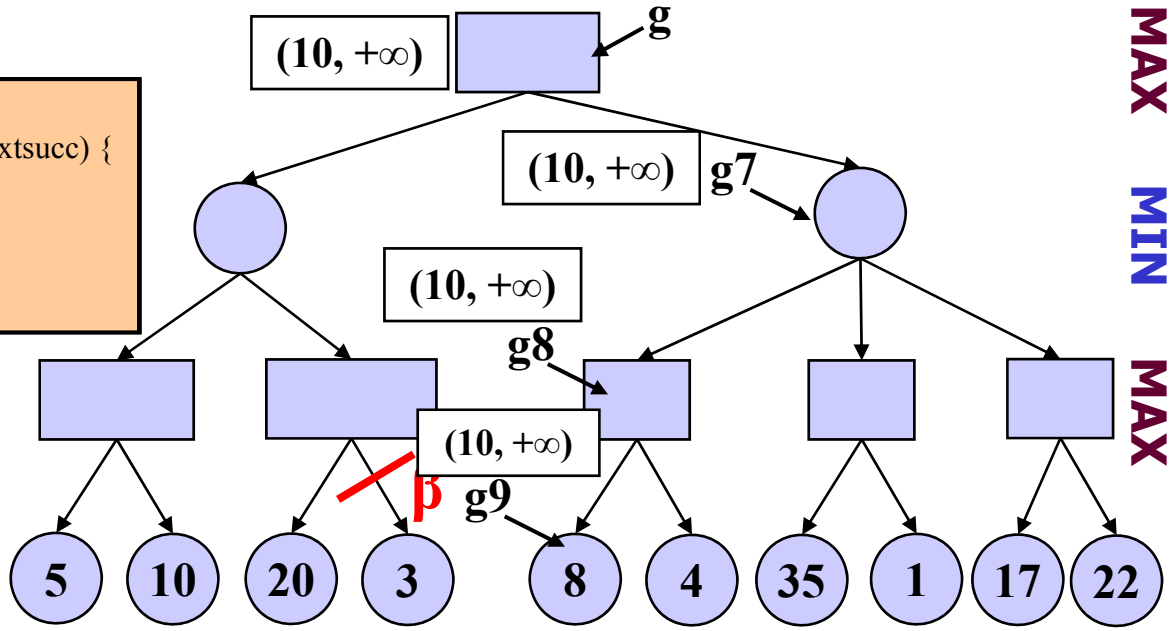
```
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
      α = max( 10 , minValue(s, 10, +∞));
      if ( α ≥ +∞ ) break;
}
return α;
```

minValue(g9, 10, +∞)

```
cutofftest(g9)) → true!!
⇒ eval(g9) = 8
⇒ return 8
```

maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(10 , minValue(s, 10, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```



minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(+∞ , maxValue(s, 10, +∞));
      if ( β ≤ 10 ) break;
}
return β;
```

maxValue(g8, 10, +∞)

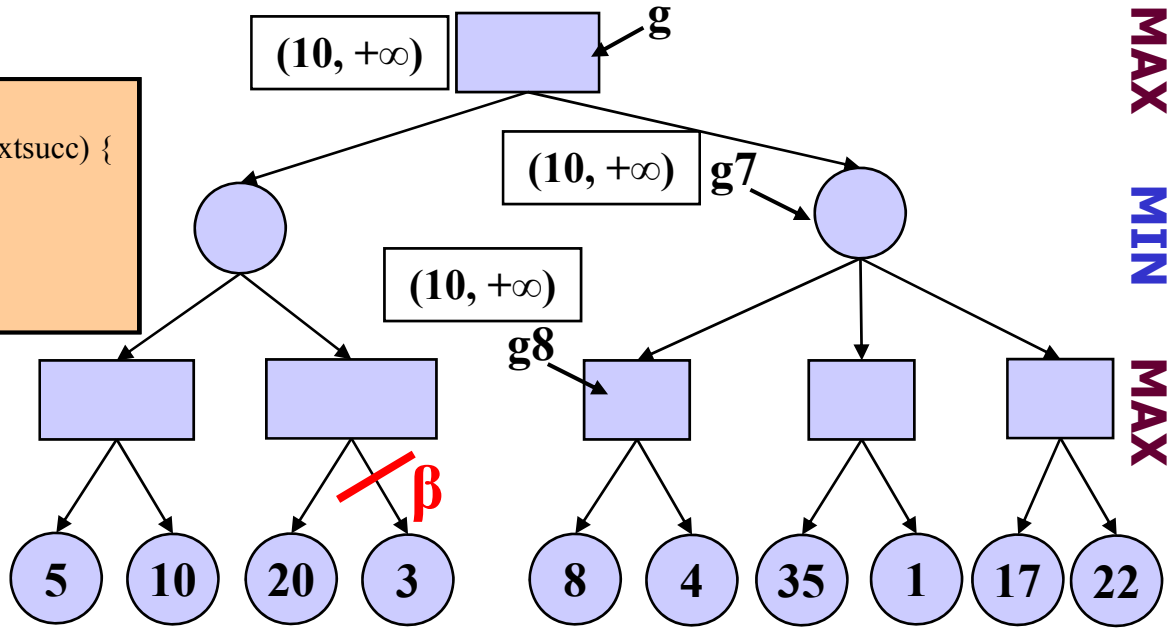
```
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
      α = max( 10 , 8);
      if ( α ≥ +∞ ) break;
}
return α;
```

minValue(g9, 10, +∞)

```
cutofftest(g9)) → true!!
⇒ eval(g9) = 8
⇒ return 8
```

maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(10 , minValue(s, 10, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```



minValue(g7, 10, +∞)

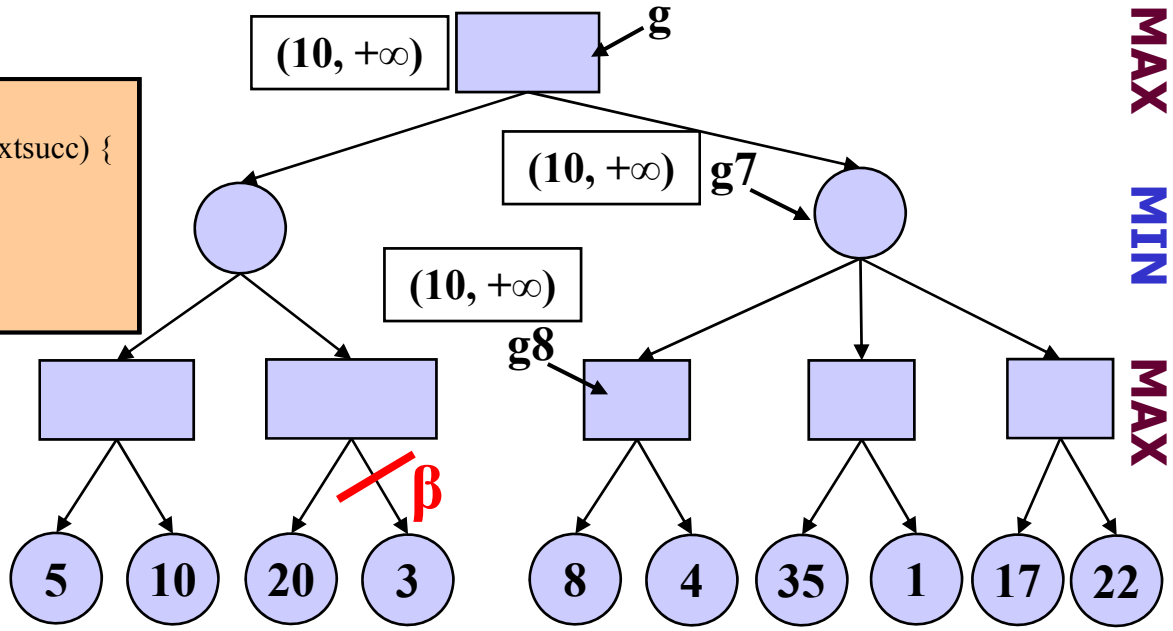
```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = min(+∞ , maxValue(s, 10, +∞));
        if ( β ≤ 10 ) break;
}
return β;
```

maxValue(g8, 10, +∞)

```
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
        α = 10;
        if ( α ≥ +∞ ) break;
}
return α;
```

maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(10 , minValue(s, 10, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```



minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = min(+∞ , maxValue(s, 10, +∞));
        if ( β ≤ 10 ) break;
}
return β;
```

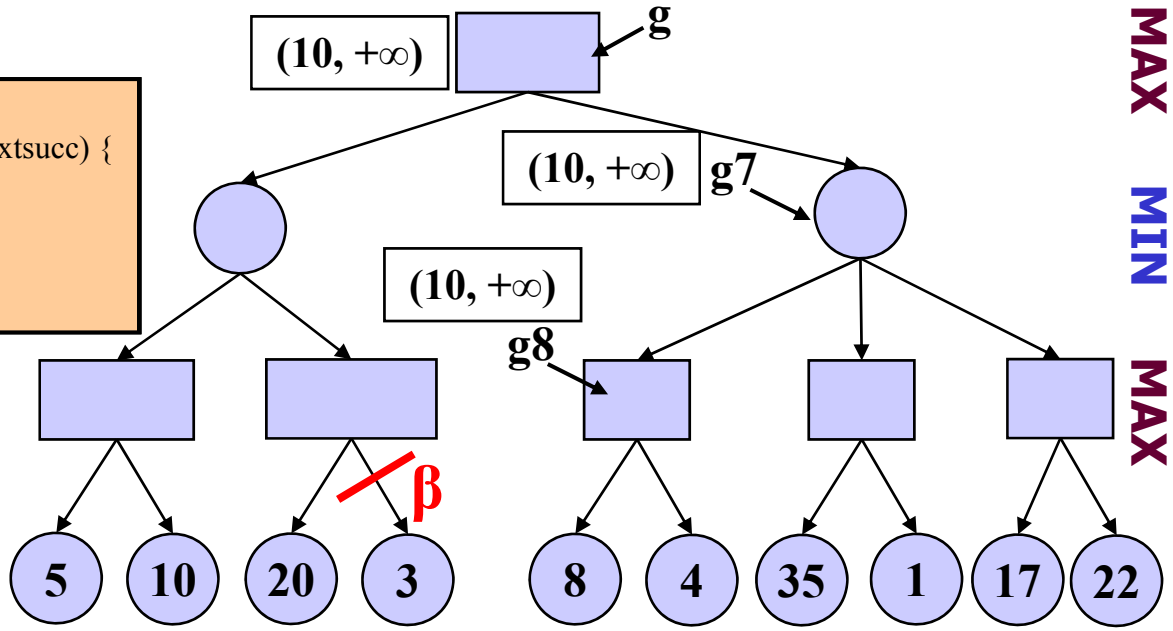
maxValue(g8, 10, +∞)

```
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
        α = 10
        if ( 10 ≥ +∞ ) break;
}
return α;
```

false!

maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(10 , minValue(s, 10, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```



minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = min(+∞ , maxValue(s, 10, +∞));
        if ( β ≤ 10 ) break;
}
return β;
```

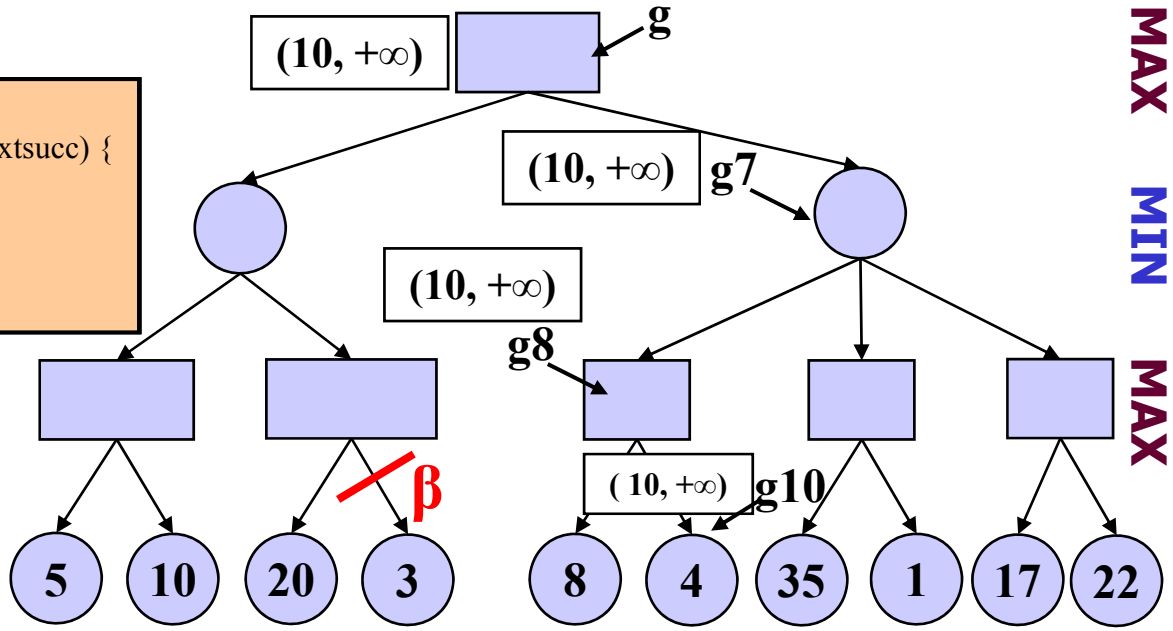
maxValue(g8, 10, +∞)

```
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
        α = max( 10 , minValue(s, 10, +∞));
        if ( α ≥ +∞ ) break;
}
return α;
```

next successor

maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(10 , minValue(s, 10, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```



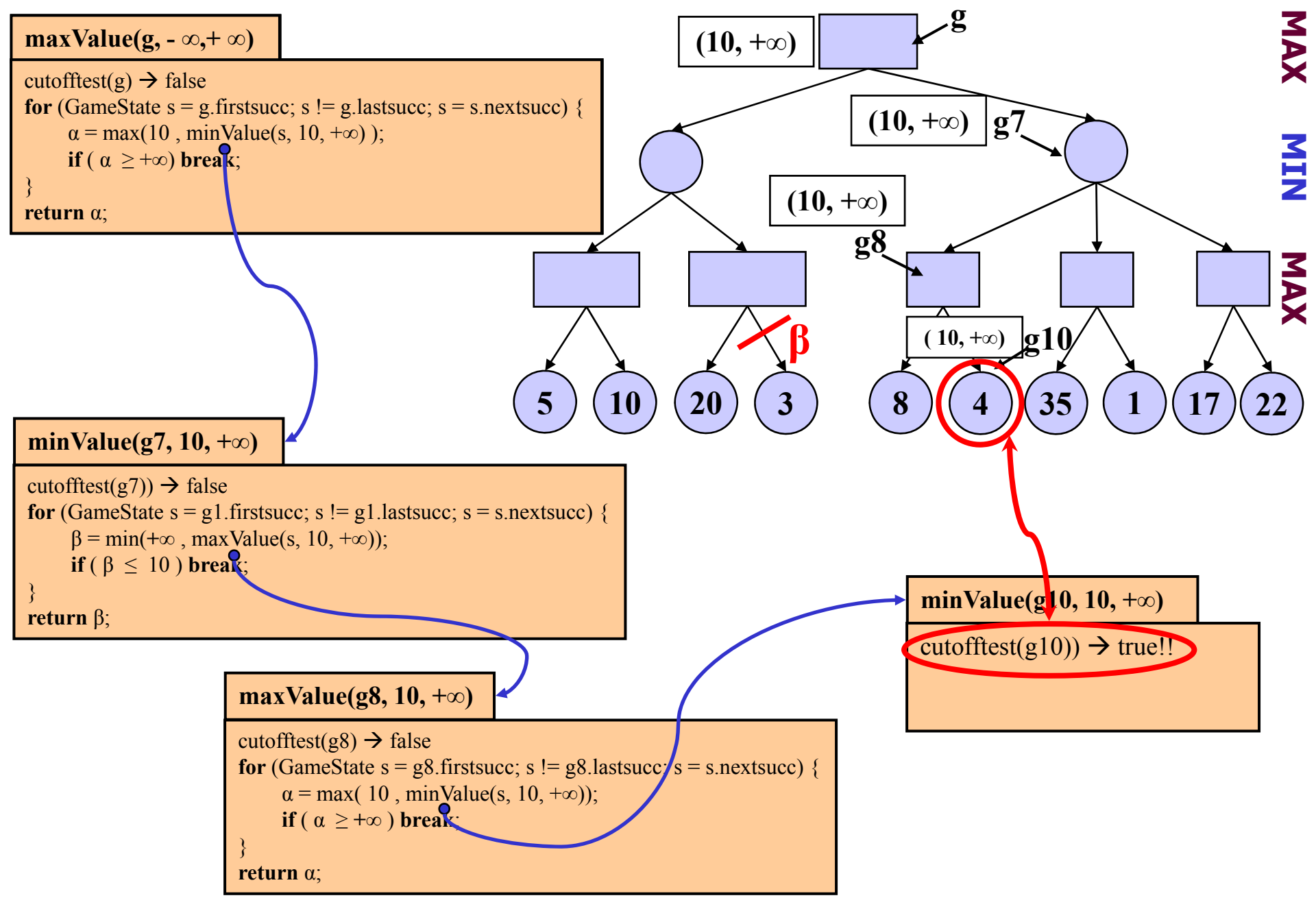
minValue(g7, 10, +∞)

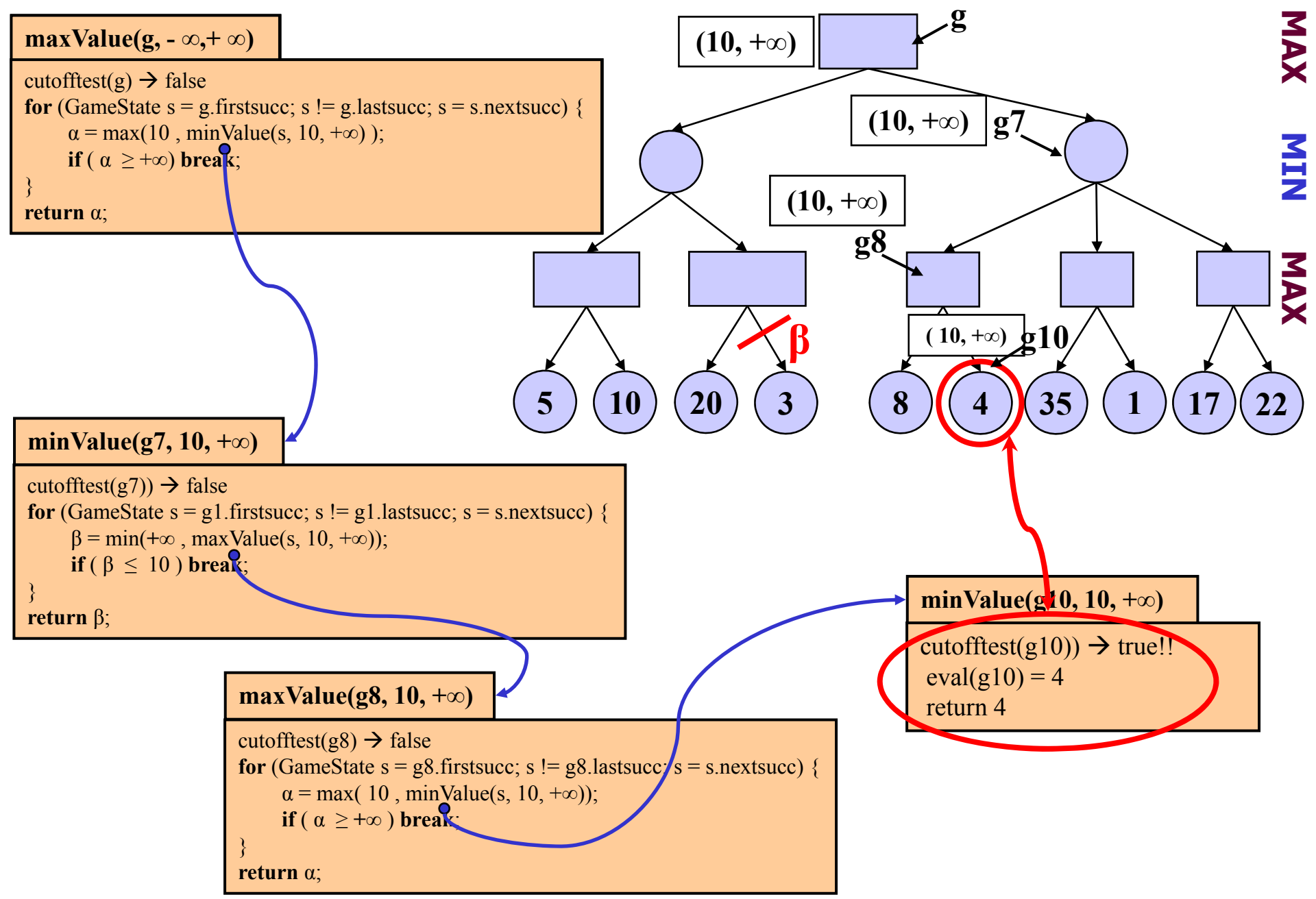
```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = min(+∞ , maxValue(s, 10, +∞));
        if ( β ≤ 10 ) break;
}
return β;
```

maxValue(g8, 10, +∞)

```
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
        α = max( 10 , minValue(s, 10, +∞));
        if ( α ≥ +∞ ) break;
}
return α;
```

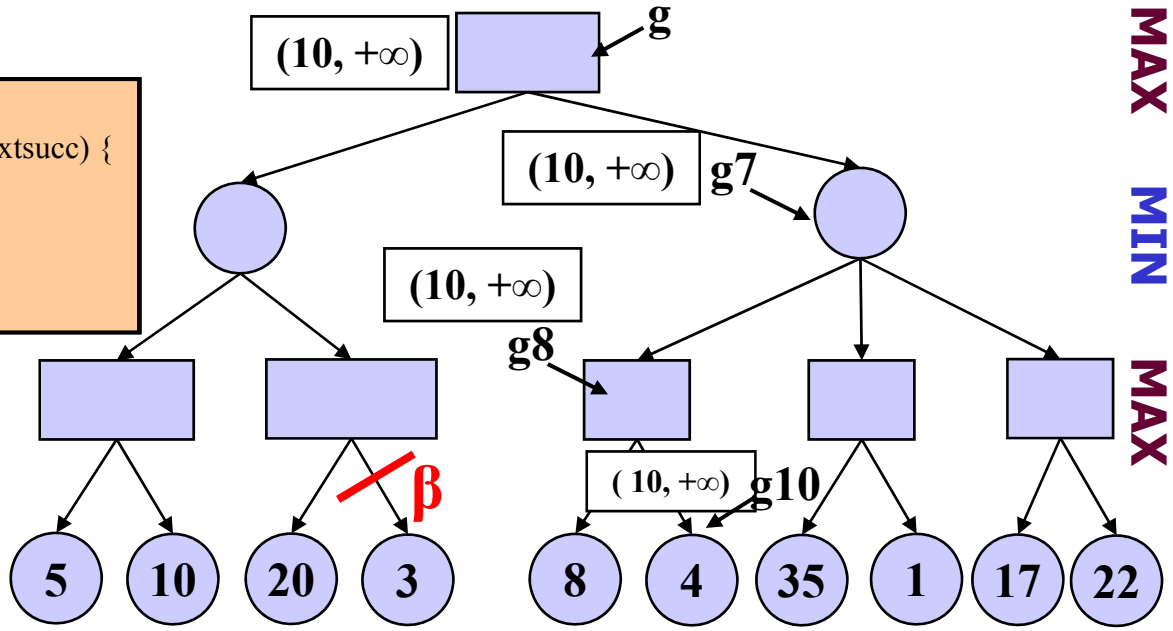
minValue(g10, 10, +∞)


maxValue(g, - ∞,+ ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(10 , minValue(s, 10, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g7, 10, +∞)
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, 10, +∞));
if (β ≤ 10) break;
}
return β;
maxValue(g8, 10, +∞)
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
α = max(10 , minValue(s, 10, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g10, 10, +∞)
cutofftest(g10)) → true!!
(10, +∞)
g
(10, +∞)
g7
(10, +∞)
g8
(10, +∞)
g10
β
MAX
MIN
MAX
5
10
20
3
8
4
35
1
17
22


maxValue(g, - ∞,+ ∞)
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
α = max(10 , minValue(s, 10, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g7, 10, +∞)
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
β = min(+∞ , maxValue(s, 10, +∞));
if (β ≤ 10) break;
}
return β;
maxValue(g8, 10, +∞)
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
α = max(10 , minValue(s, 10, +∞));
if (α ≥ +∞) break;
}
return α;
minValue(g10, 10, +∞)
cutofftest(g10)) → true!!
eval(g10) = 4
return 4
(10, +∞)
g
(10, +∞)
g7
(10, +∞)
g8
(10, +∞)
g10
β
MAX
MIN
MAX
5
10
20
3
8
4
35
1
17
22

maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(10 , minValue(s, 10, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```



minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(+∞ , maxValue(s, 10, +∞));
      if ( β ≤ 10 ) break;
}
return β;
```

maxValue(g8, 10, +∞)

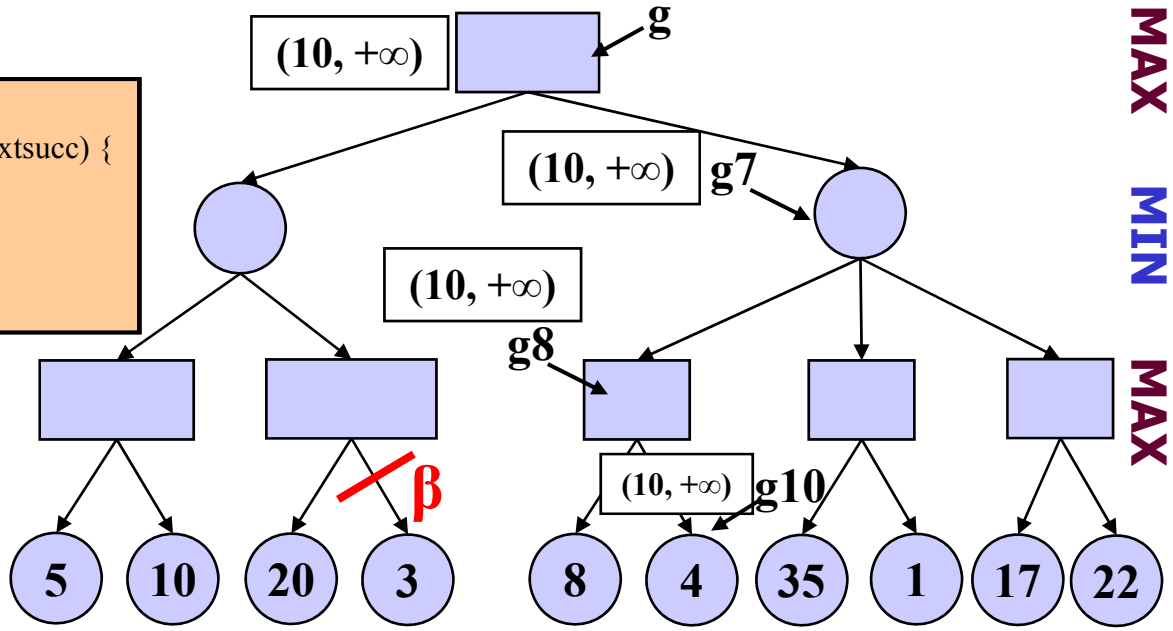
```
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
      α = max( 10 , minValue(s, 10, +∞));
      if ( α ≥ +∞ ) break;
}
return α;
```

minValue(g10, 10, +∞)

```
cutofftest(g10)) → true!!
 eval(g10) = 4
 return 4
```

maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(10 , minValue(s, 10, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```



minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(+∞ , maxValue(s, 10, +∞));
      if ( β ≤ 10 ) break;
}
return β;
```

maxValue(g8, 10, +∞)

```
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
      α = max( 10 , 4 );
      if ( α ≥ +∞ ) break;
}
return α;
```

minValue(g10, 10, +∞)

```
cutofftest(g10)) → true!!
 eval(g10) = 4
 return 4
```

maxValue(g, - ∞,+ ∞)

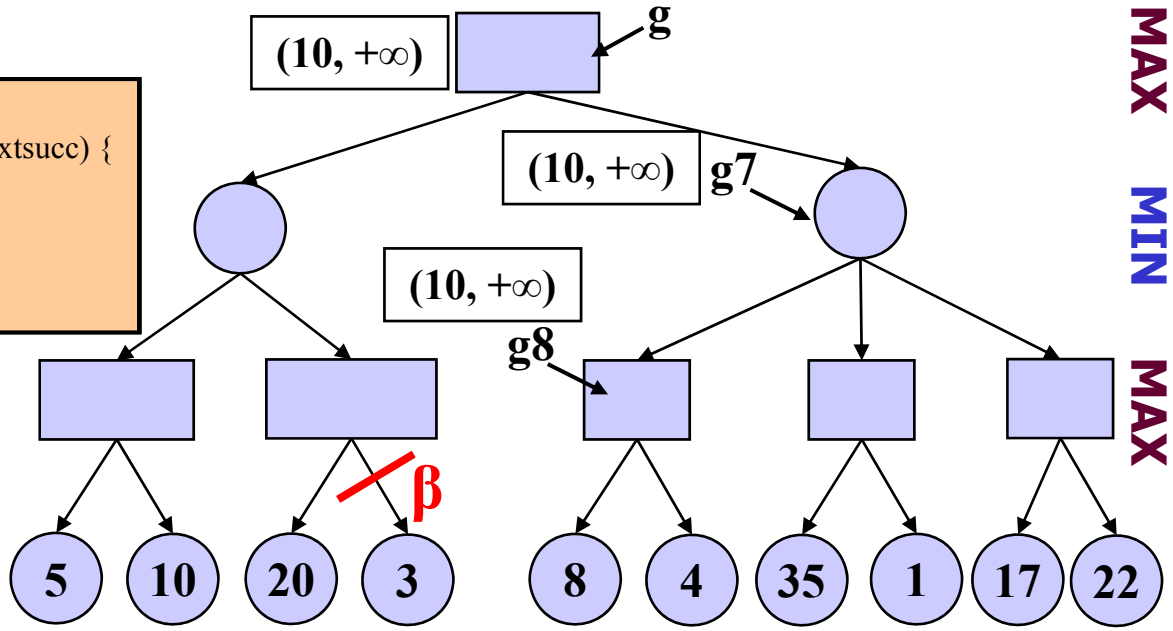
```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(10 , minValue(s, 10, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```

minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = min(+∞ , maxValue(s, 10, +∞));
        if ( β ≤ 10 ) break;
}
return β;
```

maxValue(g8, 10, +∞)

```
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
        α =10;
        if ( α ≥ +∞ ) break;
}
return α;
```



maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(10 , minValue(s, 10, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```

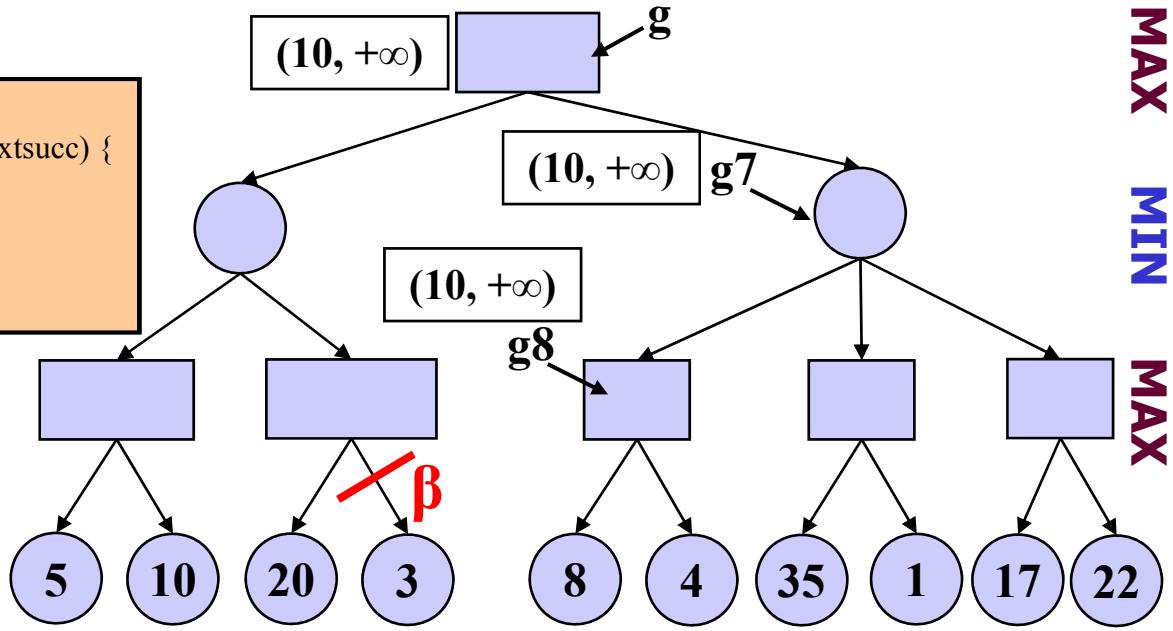
minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = min(+∞ , maxValue(s, 10, +∞));
        if ( β ≤ 10 ) break;
}
return β;
```

maxValue(g8, 10, +∞)

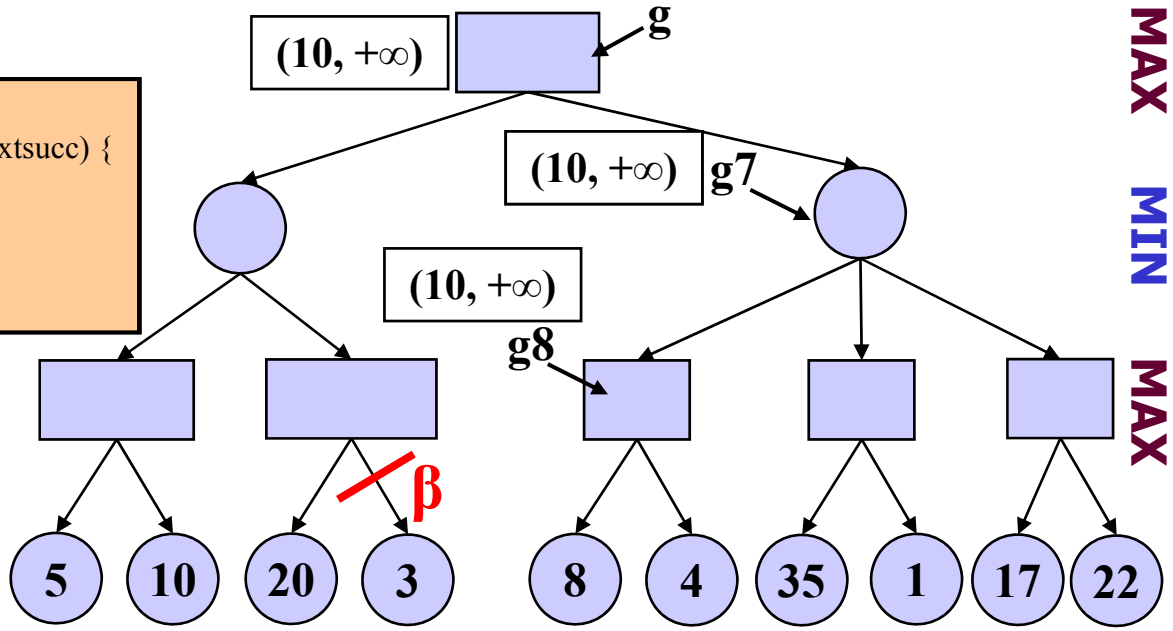
```
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
        α = 10 ;
        if ( 10 ≥ +∞ ) break;
}
return α;
```

false!



maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(10 , minValue(s, 10, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```



minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(+∞ , maxValue(s, 10, +∞));
      if ( β ≤ 10 ) break;
}
return β;
```

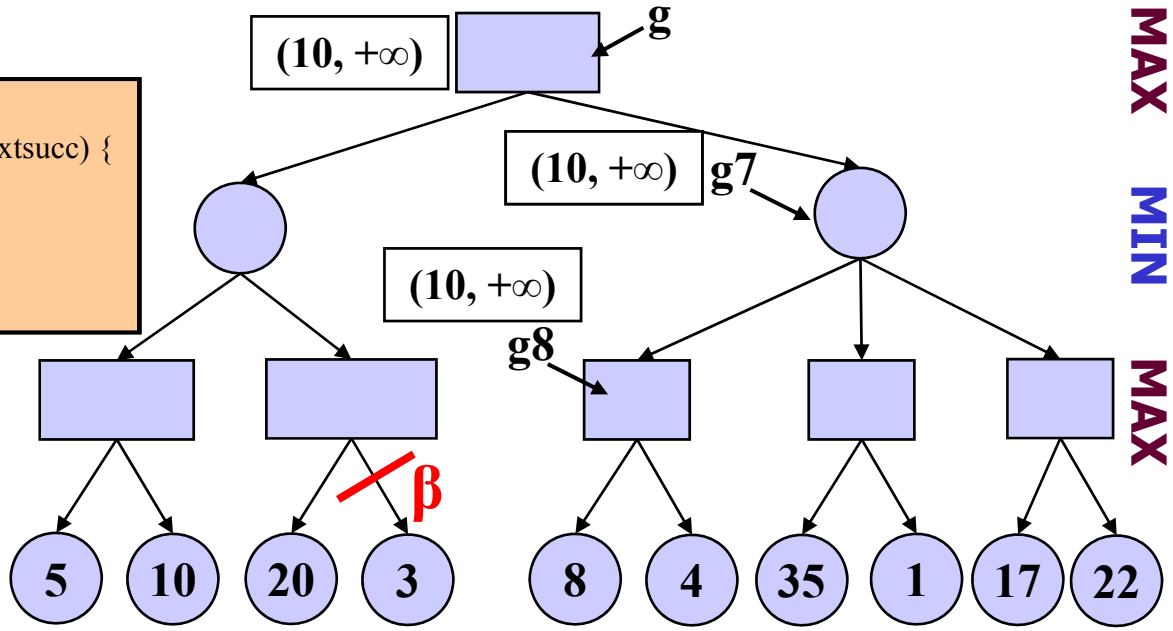
maxValue(g8, 10, +∞)

```
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
      α = max( 10 , minValue(s, 10, +∞));
      if ( α ≥ +∞ ) break;
}
return α;
```

g8 has no more successors!

maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(10 , minValue(s, 10, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```



minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = min(+∞ , maxValue(s, 10, +∞));
      if ( β ≤ 10 ) break;
}
return β;
```

maxValue(g8, 10, +∞)

```
cutofftest(g8) → false
for (GameState s = g8.firstsucc; s != g8.lastsucc; s = s.nextsucc) {
      α = 10 ;
      if ( 10 ≥ +∞ ) break;
}
return 10;
```

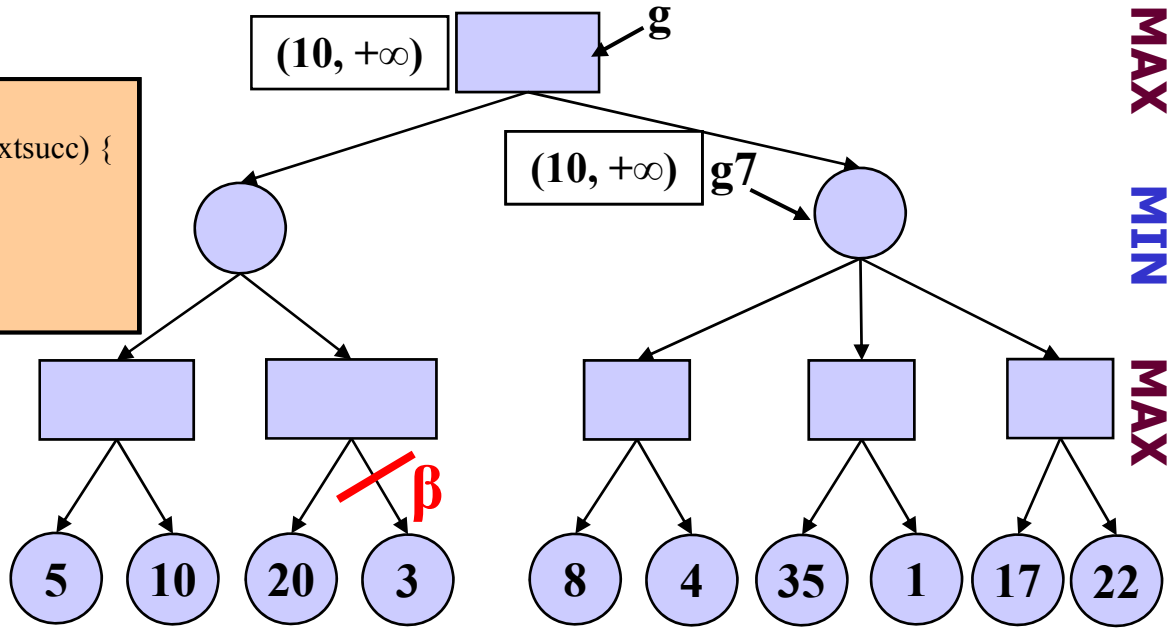
g8 has no more successors!

maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(10 , minValue(s, 10, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```

minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = min(+∞ , 10);
        if ( β ≤ 10 ) break;
}
return β;
```

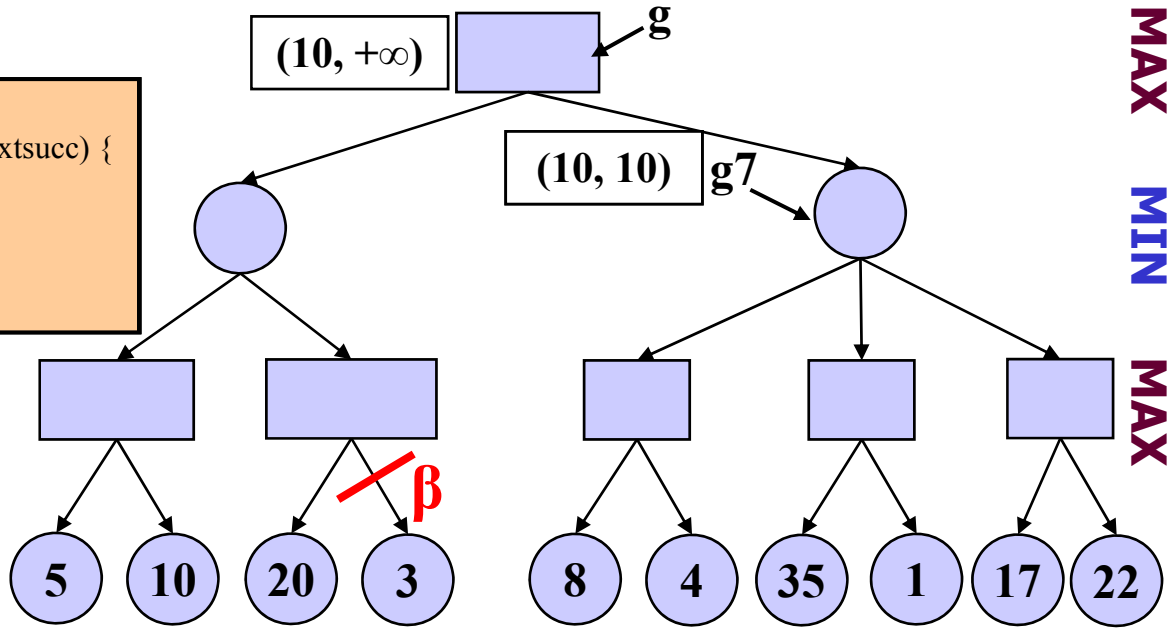


maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(10 , minValue(s, 10, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```

minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = 10;
        if ( β ≤ 10 ) break;
}
return β;
```



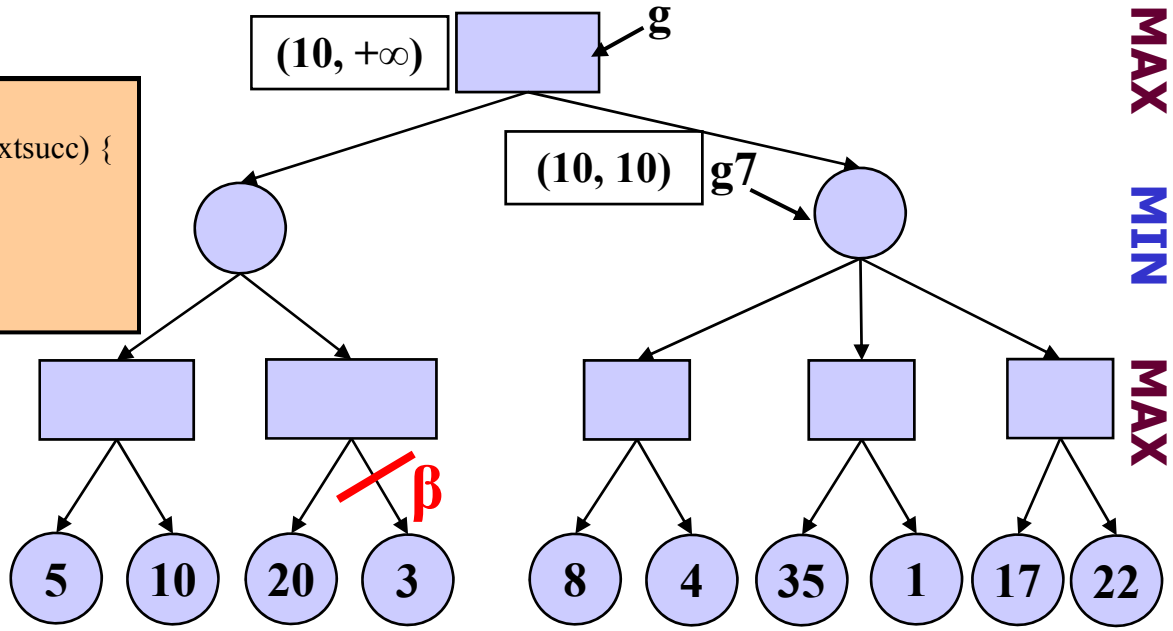
maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(10 , minValue(s, 10, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```

minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = 10;
        if ( 10 ≤ 10 ) break;
}
return β;
```

true!

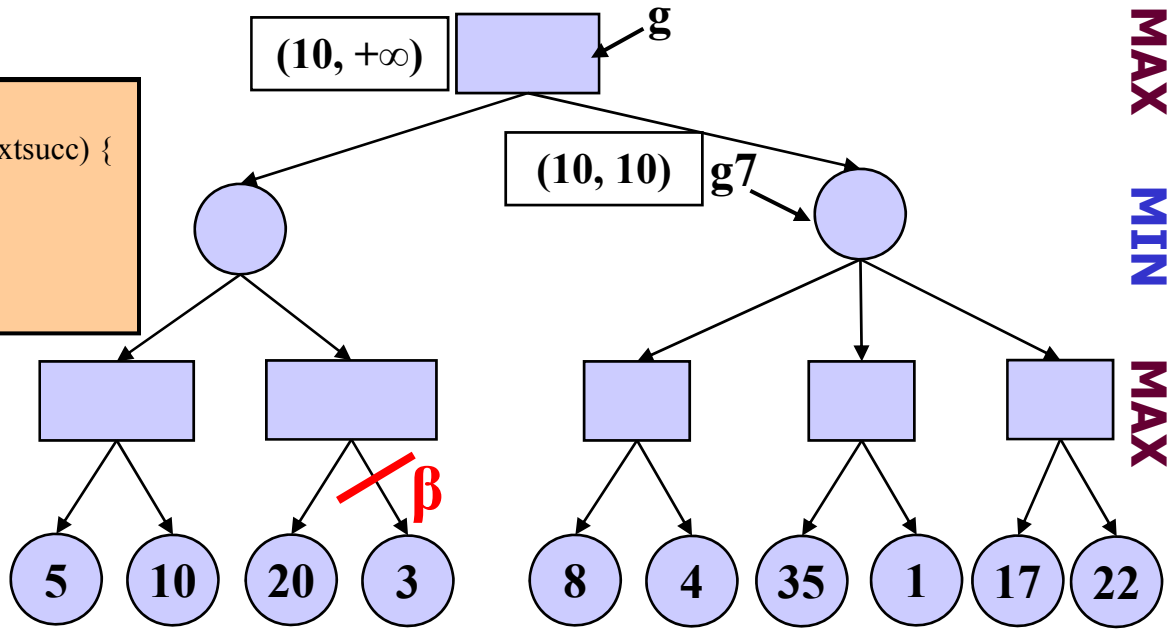


maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(10 , minValue(s, 10, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```

minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = 10;
        if ( 10 ≤ 10) break;
}
return β;
```

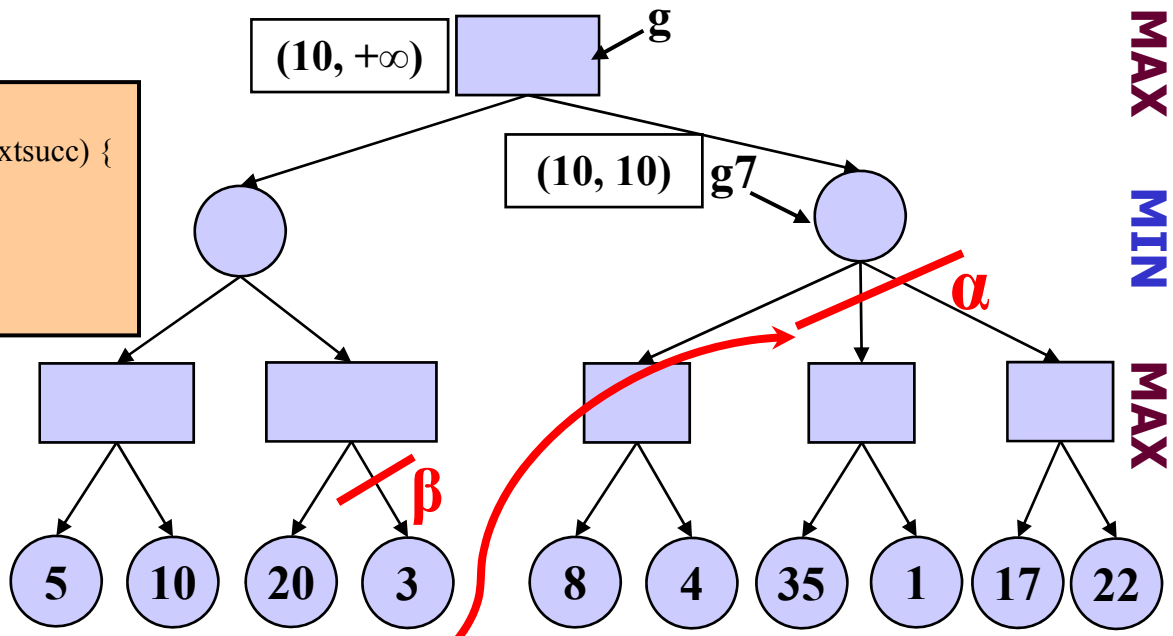


maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = max(10 , minValue(s, 10, +∞) );
        if ( α ≥ +∞) break;
}
return α;
```

minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
        β = 10;
        if ( 10 ≤ 10 ) break;  // α Schnitt!!
}
return 10;
```

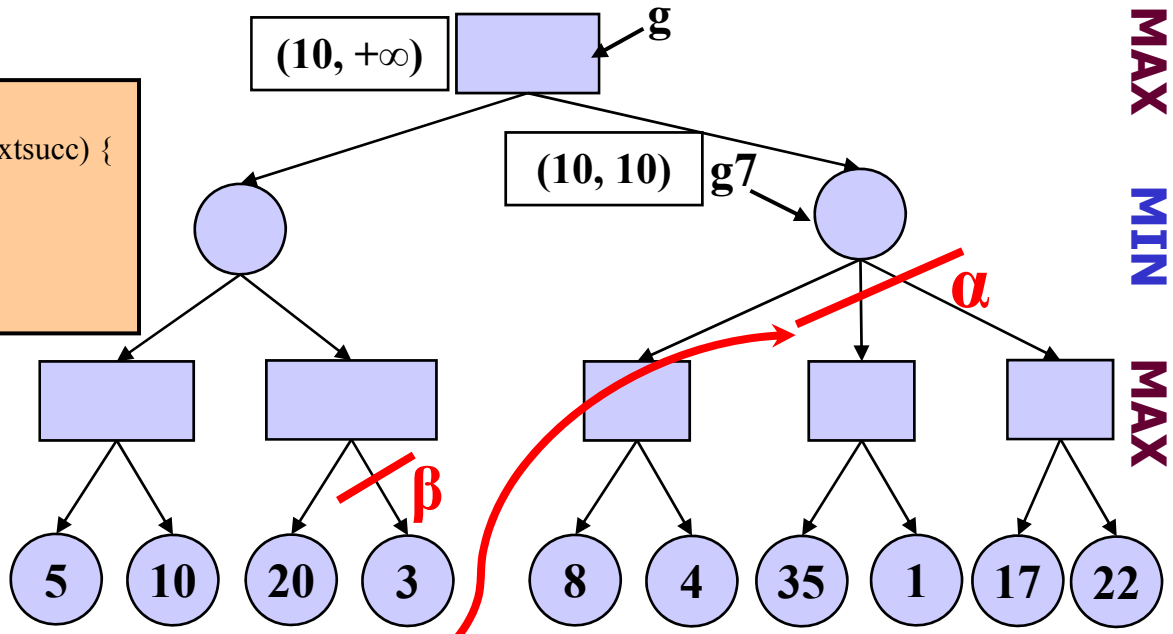


maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(10 , minValue(s, 10, +∞) );
      if ( α ≥ +∞) break;
}
return α;
```

minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = 10;
      if ( 10 ≤ 10 ) break;  // α Schnitt!!
}
return 10;
```

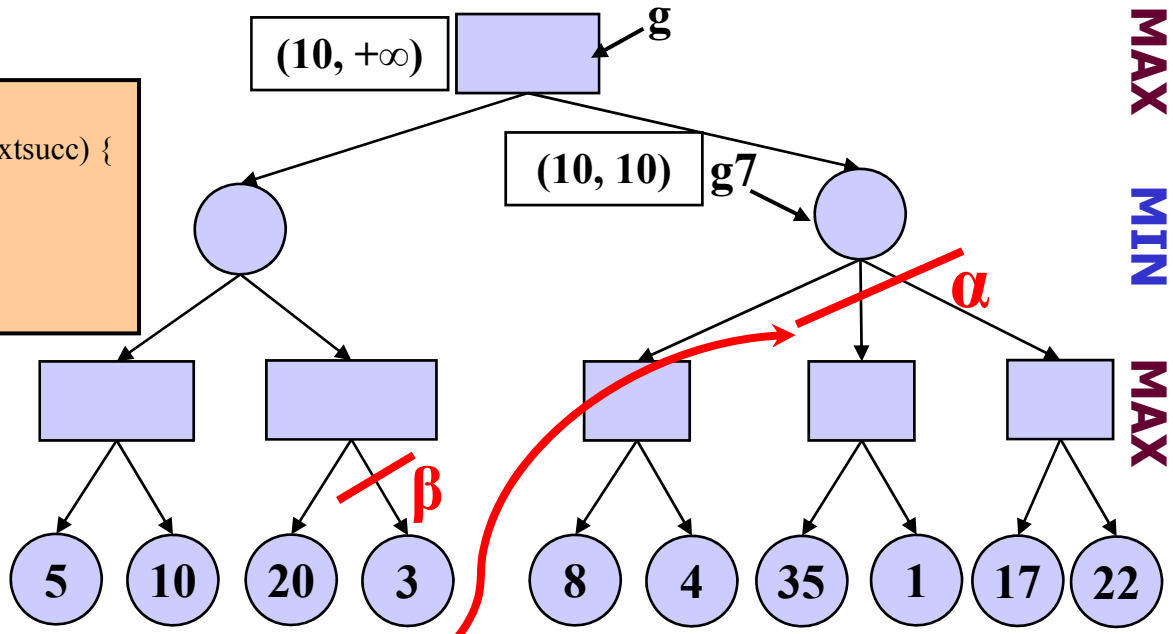


maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = max(10 , 10);
      if ( α ≥ +∞) break;
}
return α;
```

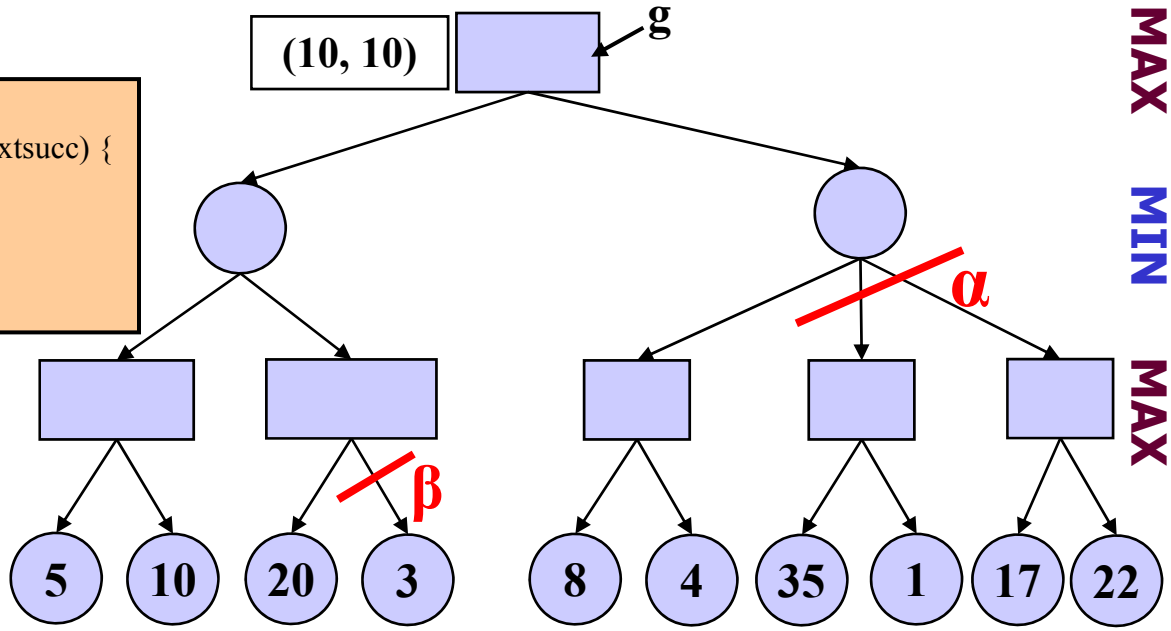
minValue(g7, 10, +∞)

```
cutofftest(g7)) → false
for (GameState s = g1.firstsucc; s != g1.lastsucc; s = s.nextsucc) {
      β = 10;
      if ( 10 ≤ 10 ) break;  // α Schnitt!!
}
return 10;
```



maxValue(g, - ∞,+ ∞)

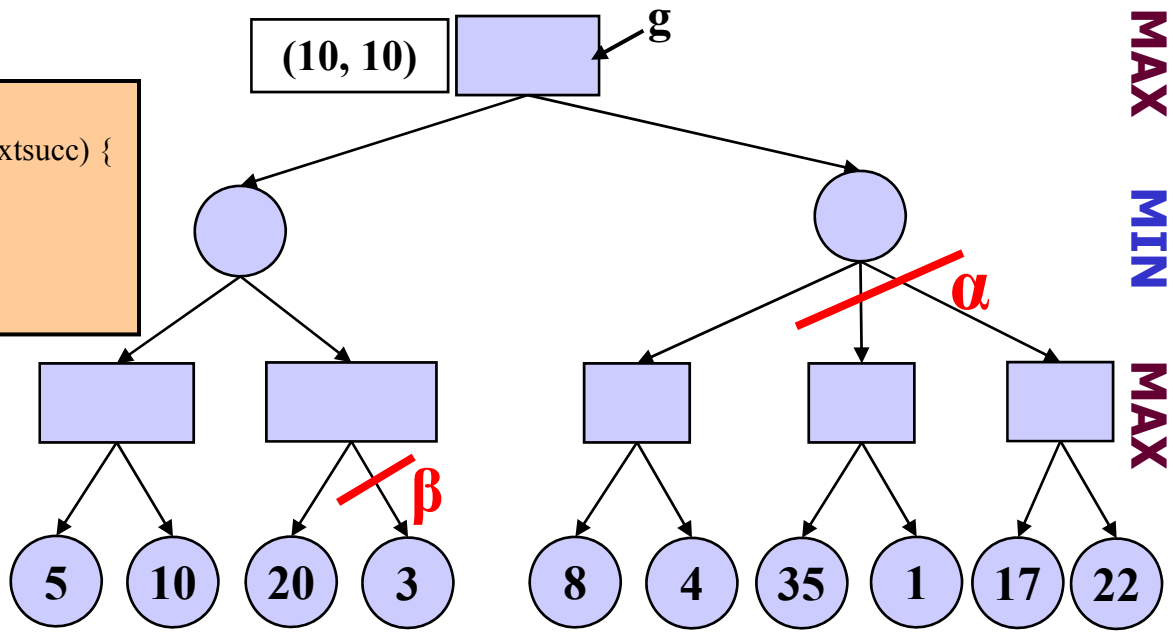
```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
      α = 10
      if ( α ≥ +∞) break;
}
return α;
```



maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
    α = 10;
    if ( 10 ≥ +∞) break;
}
return α;
```

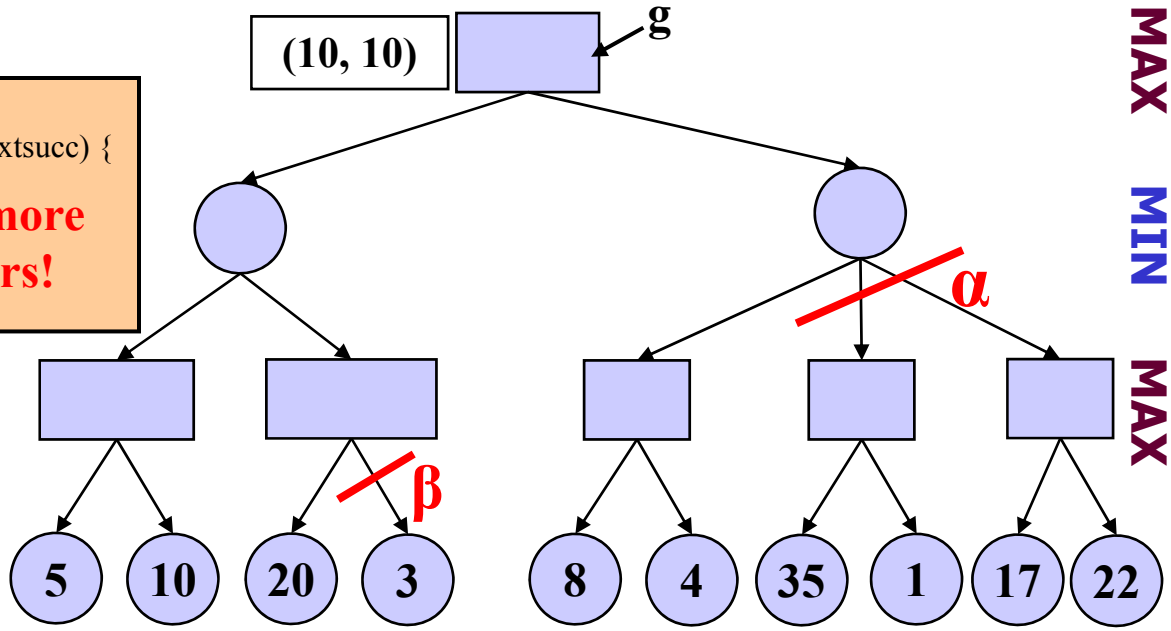
false!



maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
    α = 10;
    if ( 10 ≥ +∞) break;
}
return 10;
```

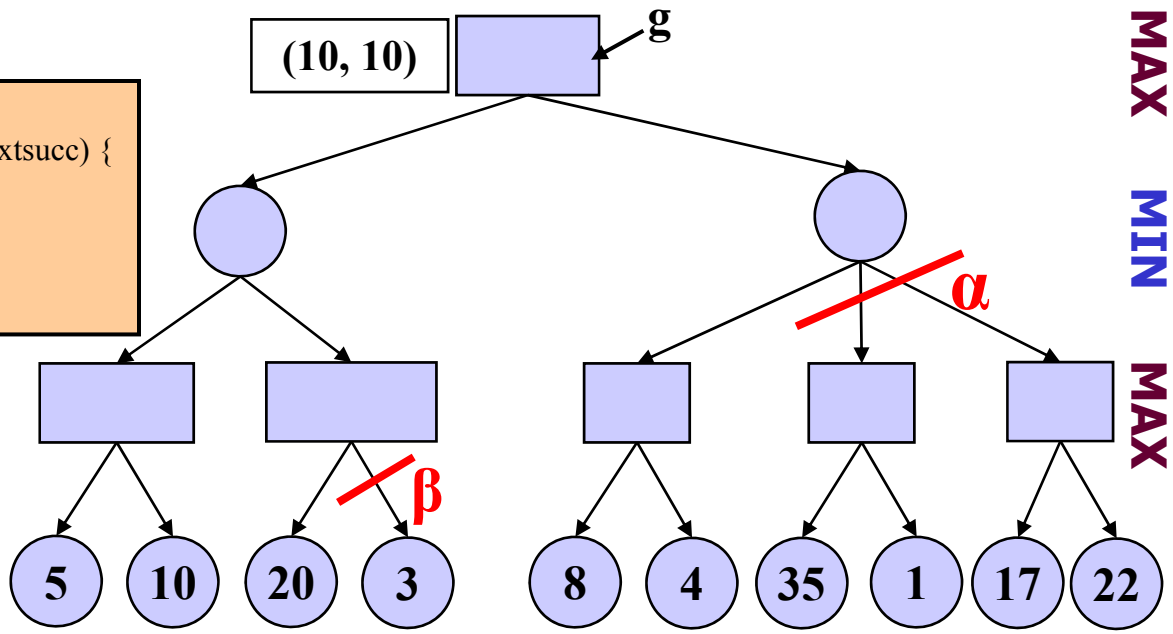
g has no more successors!



Informatik II D-ITET, FS2008

maxValue(g, - ∞,+ ∞)

```
cutofftest(g) → false
for (GameState s = g.firstsucc; s != g.lastsucc; s = s.nextsucc) {
        α = 10;
        if ( 10 ≥ +∞) break;
}
return 10;
```



Maximal yield from current position is 10

- For questions please contact:
  santinis@inf.ethz.ch