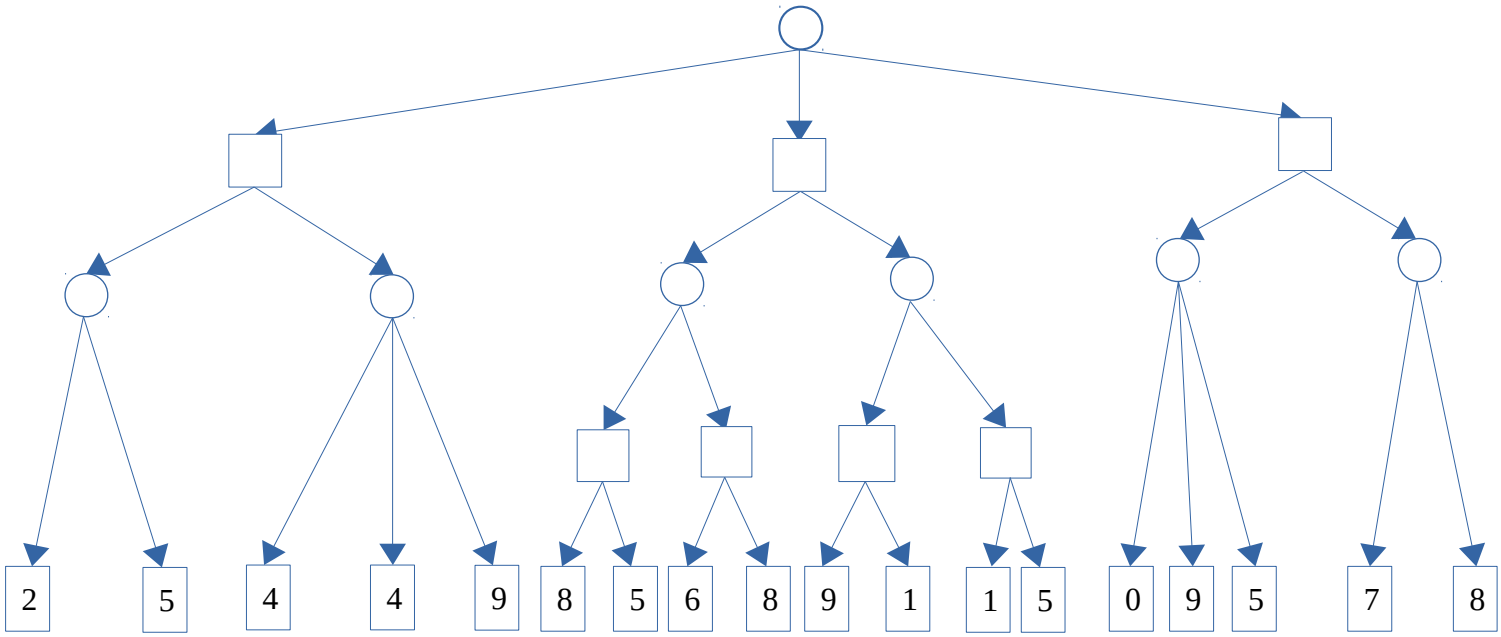
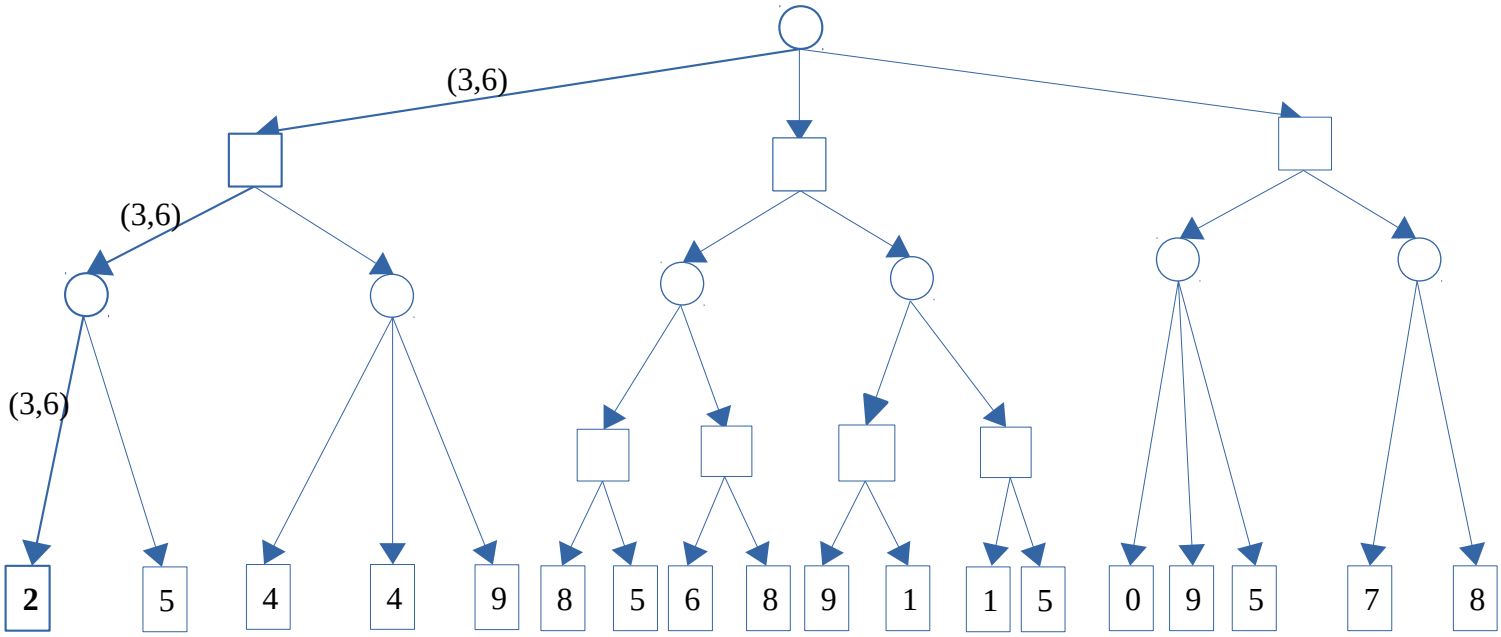
Przyklad: Przesledźmy działanie, wywołanego z oknem (3, 6) algorytmu α- β na następującym grafie gry o możliwych wypłatach od 0 do 9 oraz sumie wypłat 9.



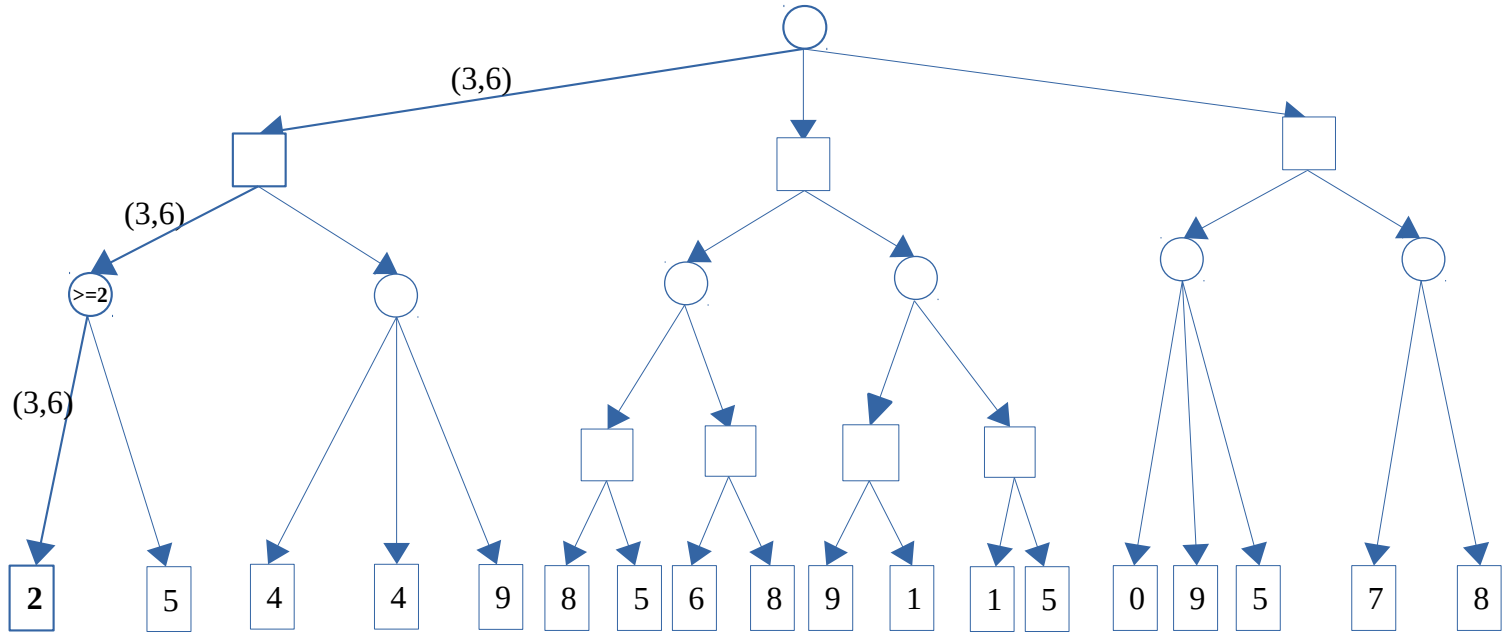
Legenda: ○ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

Przy użyciu początkowego okna (3, 6) następują rekurencyjne wywołania dla kolejnych, pierwszych (od lewej) następników, aż do osiągnięcia pozycji końcowej o wartosci 2.



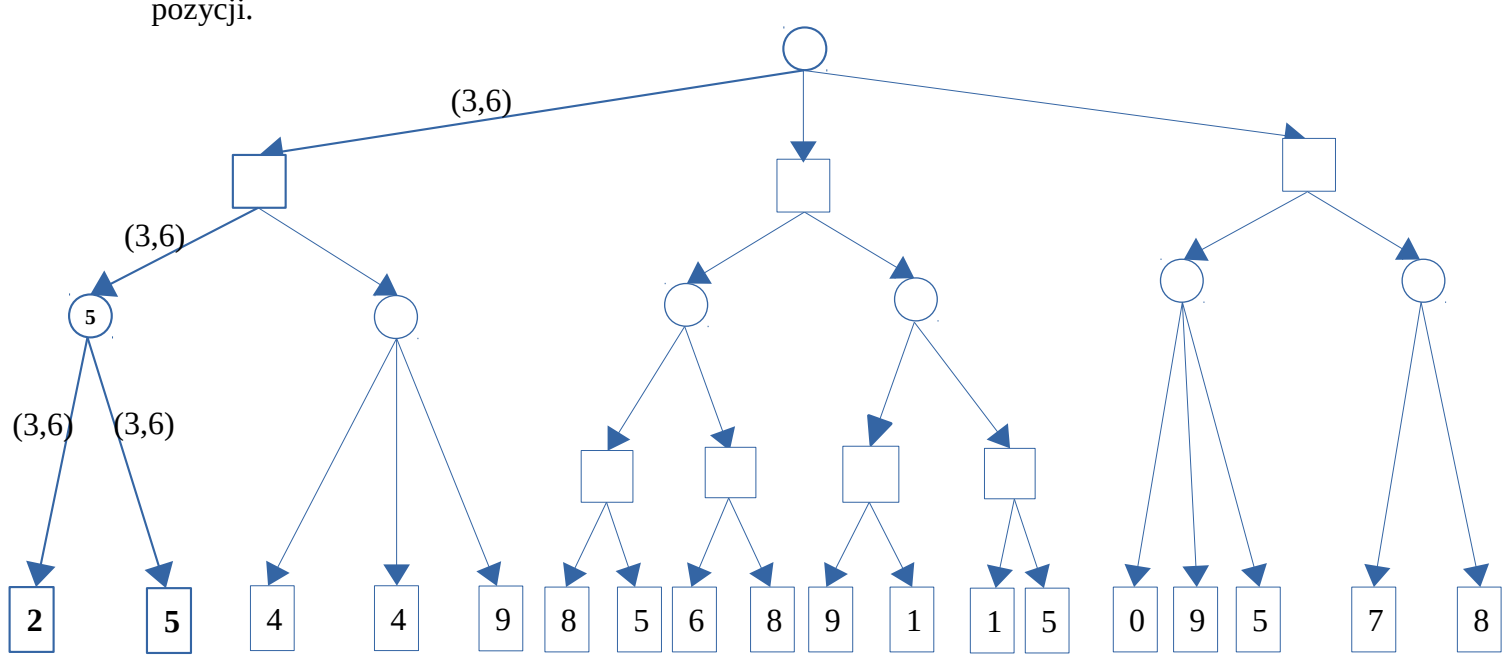
Legenda: ○ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

2 staje się bieżącą wartością poprzednika osiągniętej pozycji końcowej. Poprzednik ten jest typu MAX, więc jego ostateczna wartość wyniesie 2 lub więcej. Nie można odciąć więc jego drugiego następnika, bo wartość >=2 może się zmieścić w oknie (3, 6), tj. [2, +∞] ∩ (3, 6) != 0.



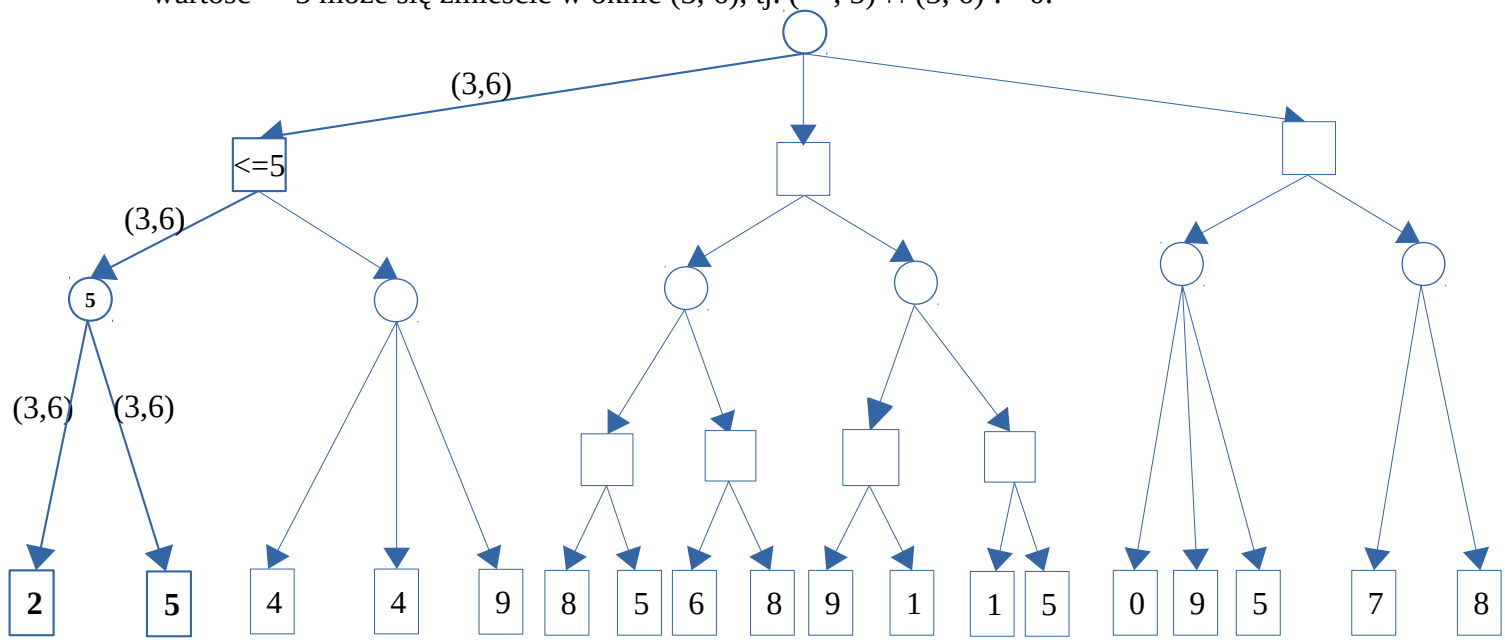
Legenda: ○ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

Dla drugiego następnika algorytm wywołuje się w takim samym oknie jak poprzednik, czyli (3, 6). 5 poprawia bieżace MAKSIMUM. Nie ma już więcej następników, więc 5 jest ostateczna wartością pozycji.



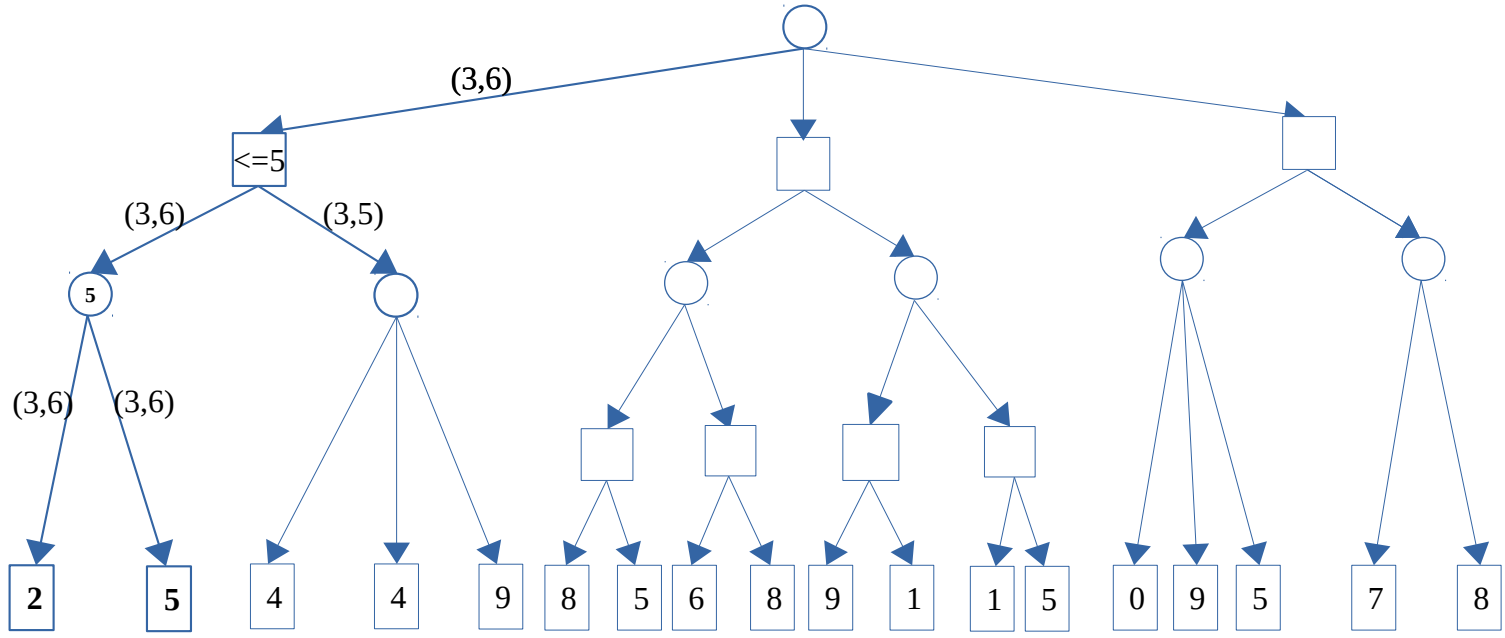
Legenda: ○ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

5 staje się bieżącą wartością poprzednika zbadanej pozycji. Poprzednik ten jest typu MIN, więc jego ostateczna wartość wyniesie 5 lub mniej. Nie można odciąć więc jego drugiego następnika, bo wartość <=5 może się zmieścić w oknie (3, 6), tj. (-∞, 5) ∩ (3, 6) != 0.



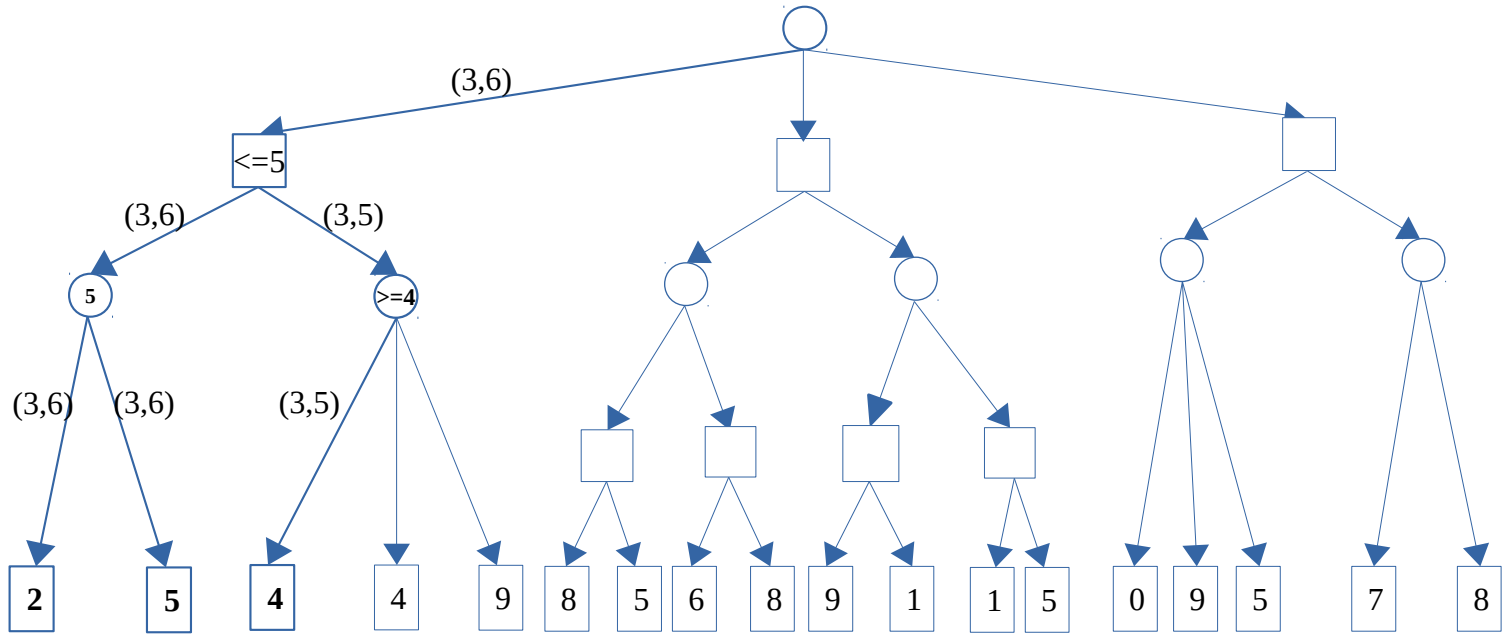
Legenda: ○ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

Dla drugiego następnika algorytm wywołuje się w oknie (-∞, 5) ∩ (3, 6) = (3, 5), bo wartości >=5 nie mogłyby poprawić bieżacego MINIMUM 5, nie są więc interesujące.



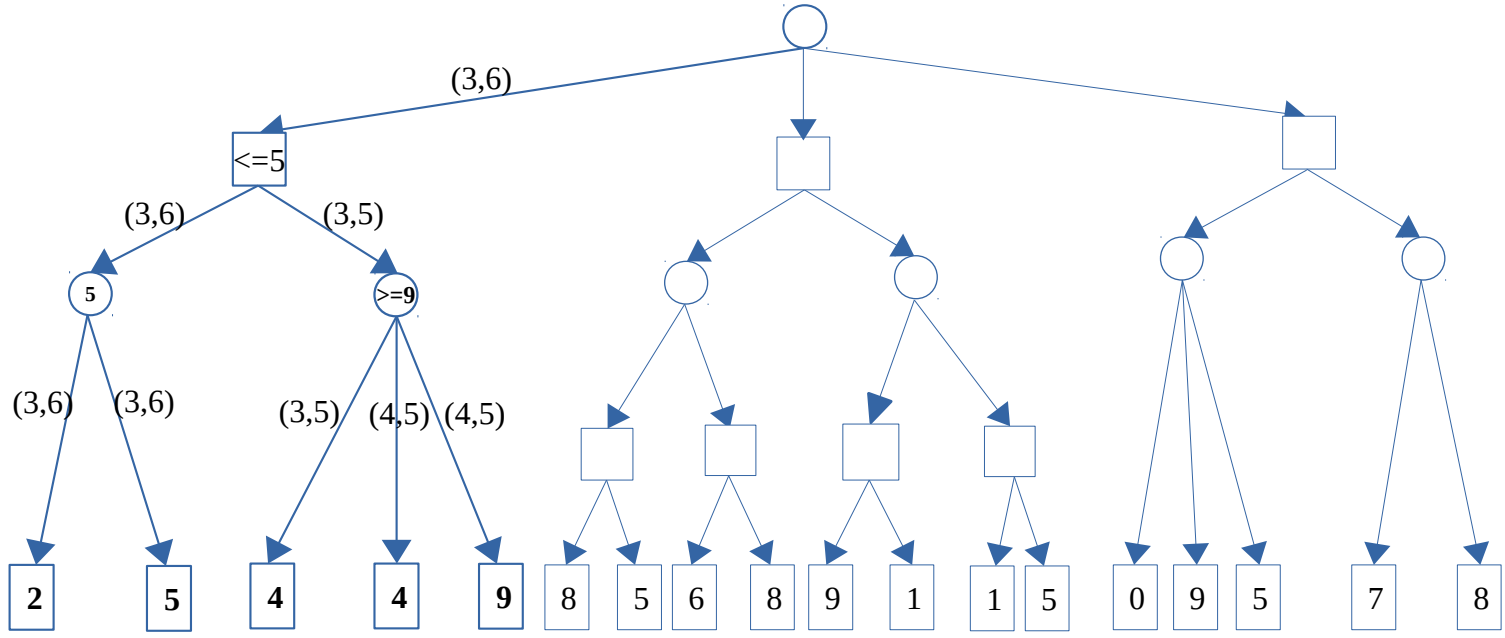
Legenda: ○ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

Pierwszy następnik badany jest w tym samym oknie co jego poprzednik, czyli (3, 5). Jego wartość 4 staje się bieżacym MAKSIMUM poprzednika.



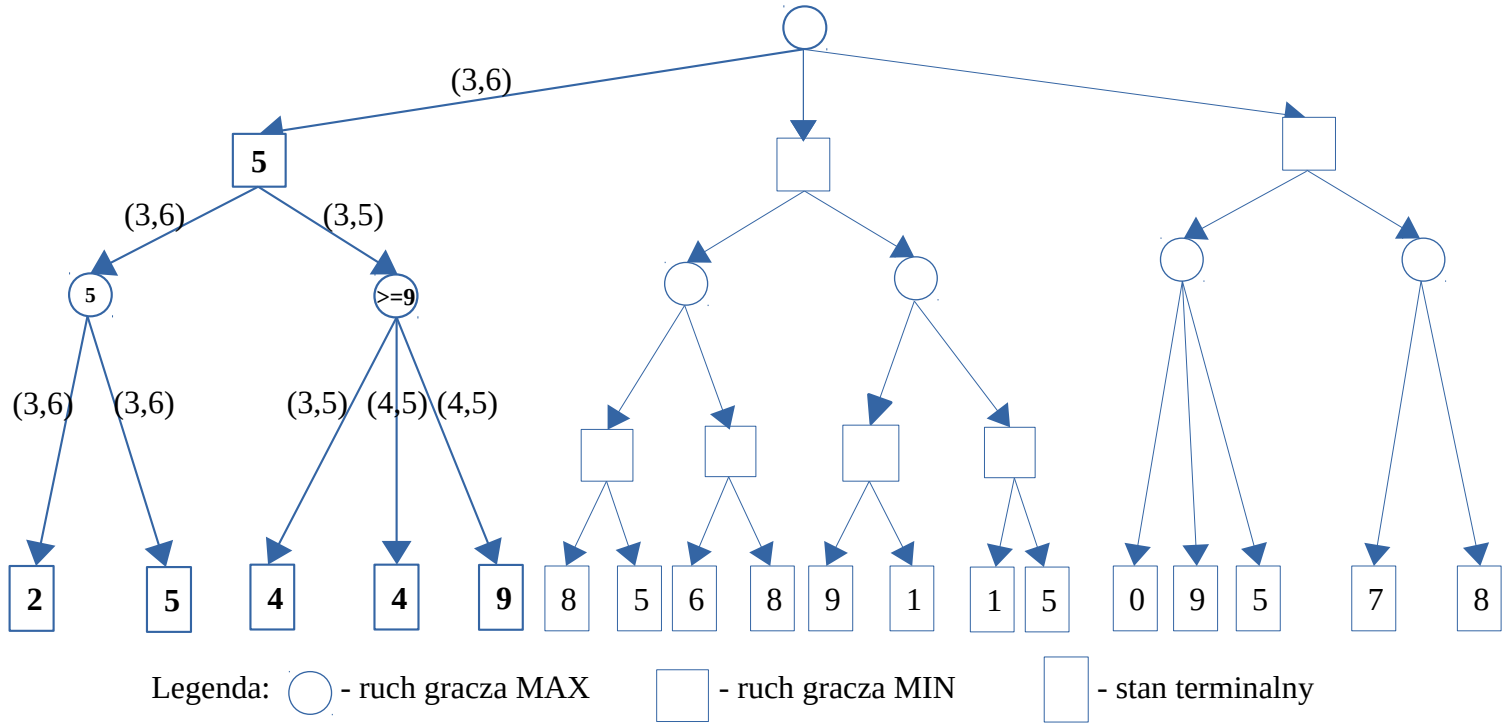
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

Ta wartość >=4 mieści się w oknie (3, 5), tj. [4, +∞) ∩ (3, 5) != 0. Następuje więc badanie kolejnych następników, ale już z oknem (4, 5), bo wartości <=4 nie mogłyby poprawić bieżacego MAXIMUM 4, nie są więc interesujące. 4 nie poprawia bieżacego MAKSIMUM. Natomiast 9 poprawia bieżace MAKSIMUM 4. Nie ma już więcej następników, więc 9 staje się ostateczną wartością pozycji (zwróconą z) badanego węzła.



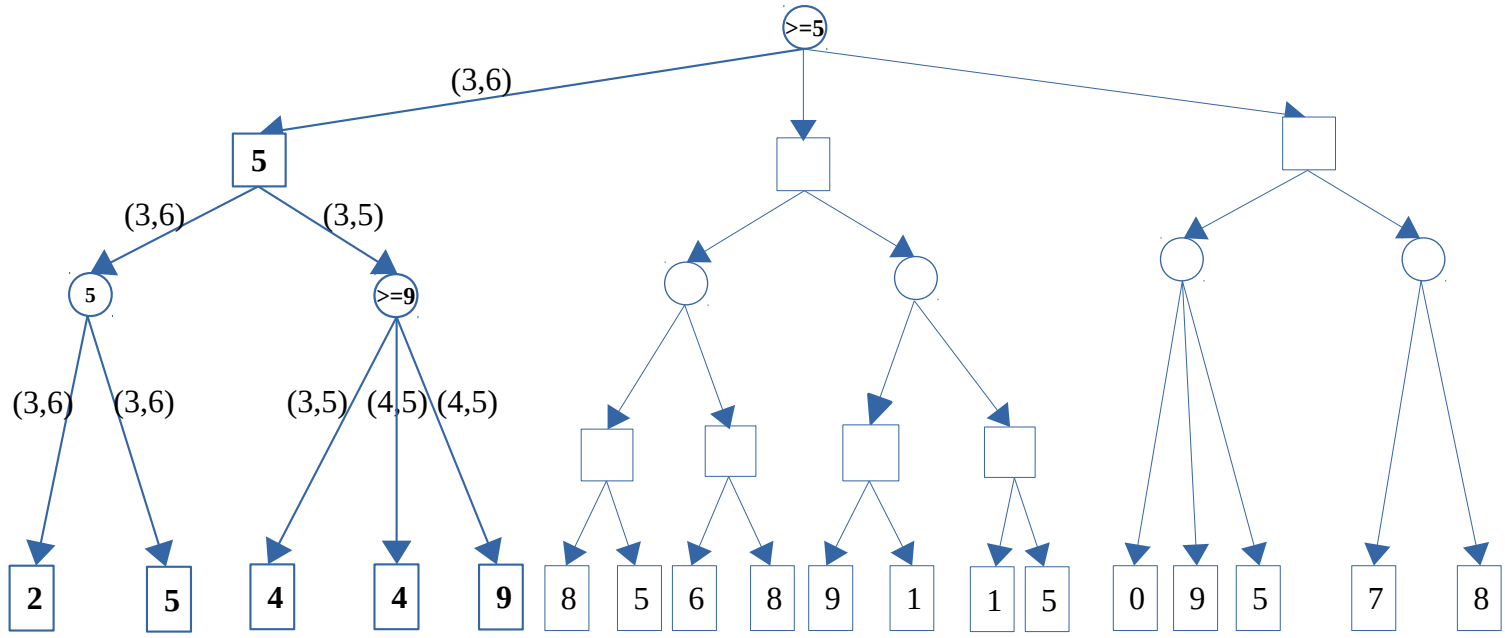
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

Ta 9 nie poprawia bieżącego MINIMUM 5 lewego następnika korzenia. 5 staje się zatem jego ostateczną wartością, bo nie ma on więcej dzieci.



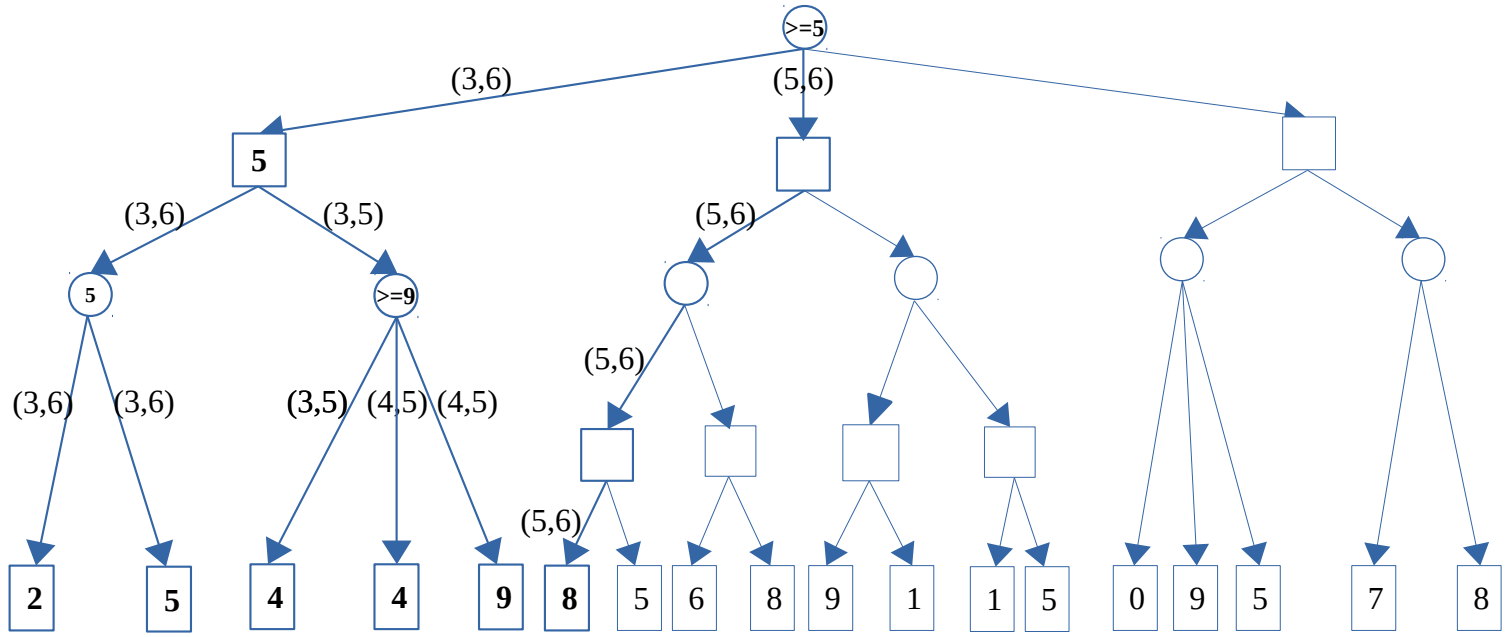
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

5 staje się bieżącą wartością korzenia. Korzeń jest typu MAX, więc jego ostateczna wartość wyniesie 5 lub więcej. Nie można odciąć jego środkowego następnika, bo wartość >= 5 może się zmieścić w oknie (3, 6), tj. [5, +∞) ∩ (3, 6) != 0.



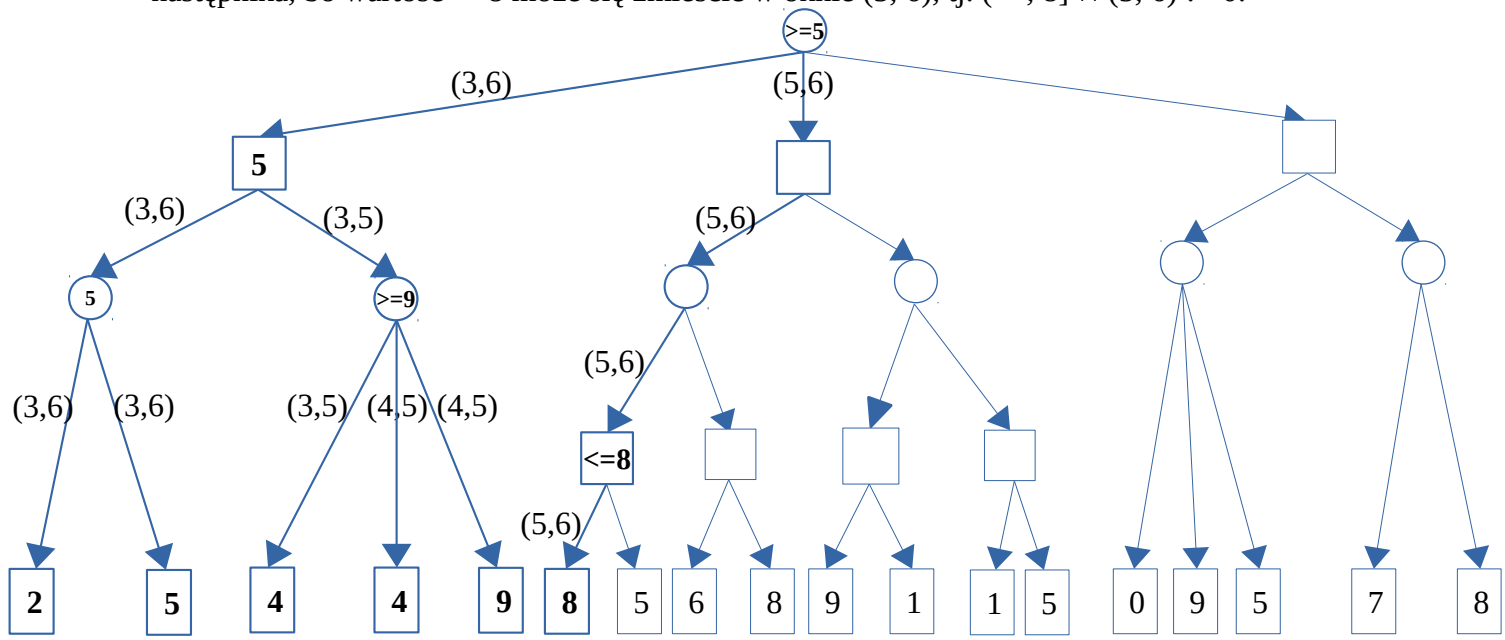
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

Dla środkowego następnika korzenia algorytm wywołuje się w oknie (5, +∞) ∩ (3, 6) = (5, 6). Ponieważ wartości <= 5 nie mogłyby poprawić bieżacego maksimum 5, nie są więc interesujące. Dalej następują rekurencyjne wywołania dla kolejnych pierwszych (od lewej) następników, aż do osiągnięcia pozycji końcowej o wartości 8.



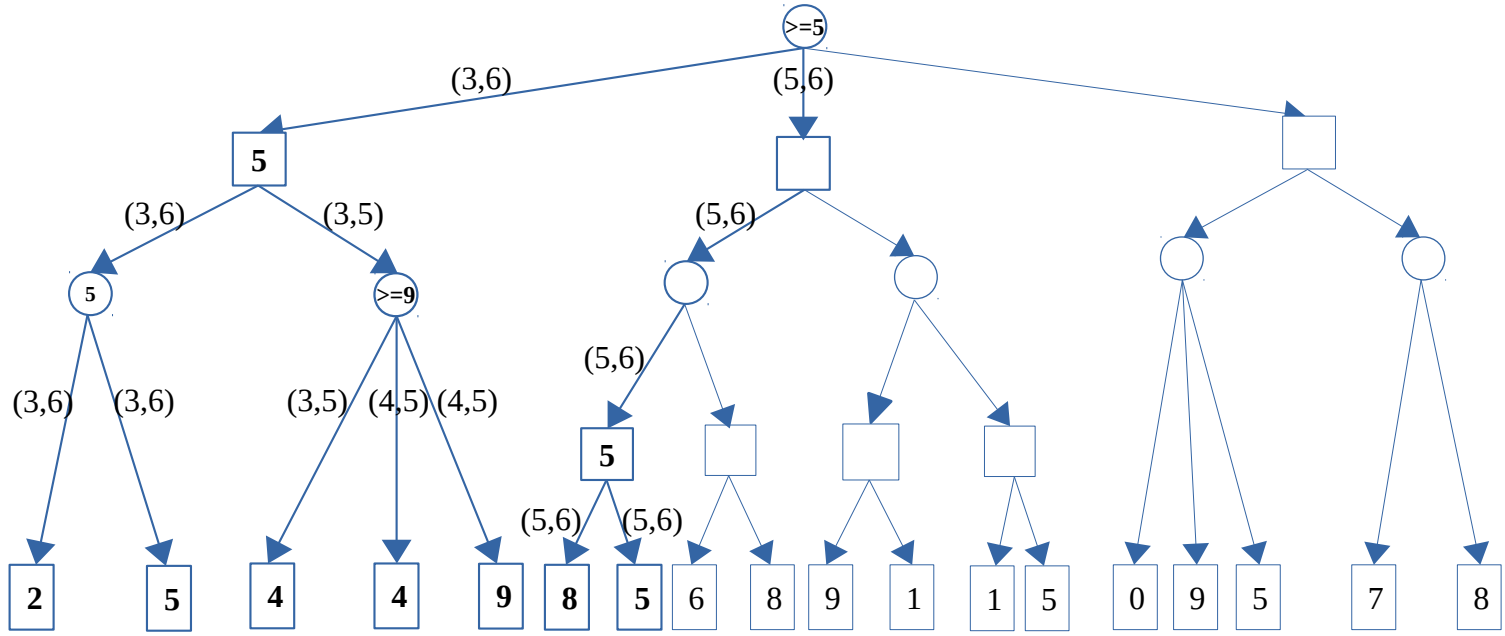
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

8 staje się bieżącą wartością poprzednika osiągnietej pozycji końcowej. Poprzednik ten jet typu MIN, więc jego ostateczna wartość wyniesie 8 lub mniej. Nie można odciąć więc jego drugiego następnika, bo wartość <=8 może się zmiescic w oknie (5, 6), tj. (-∞, 8] ∩ (5, 6) != 0.



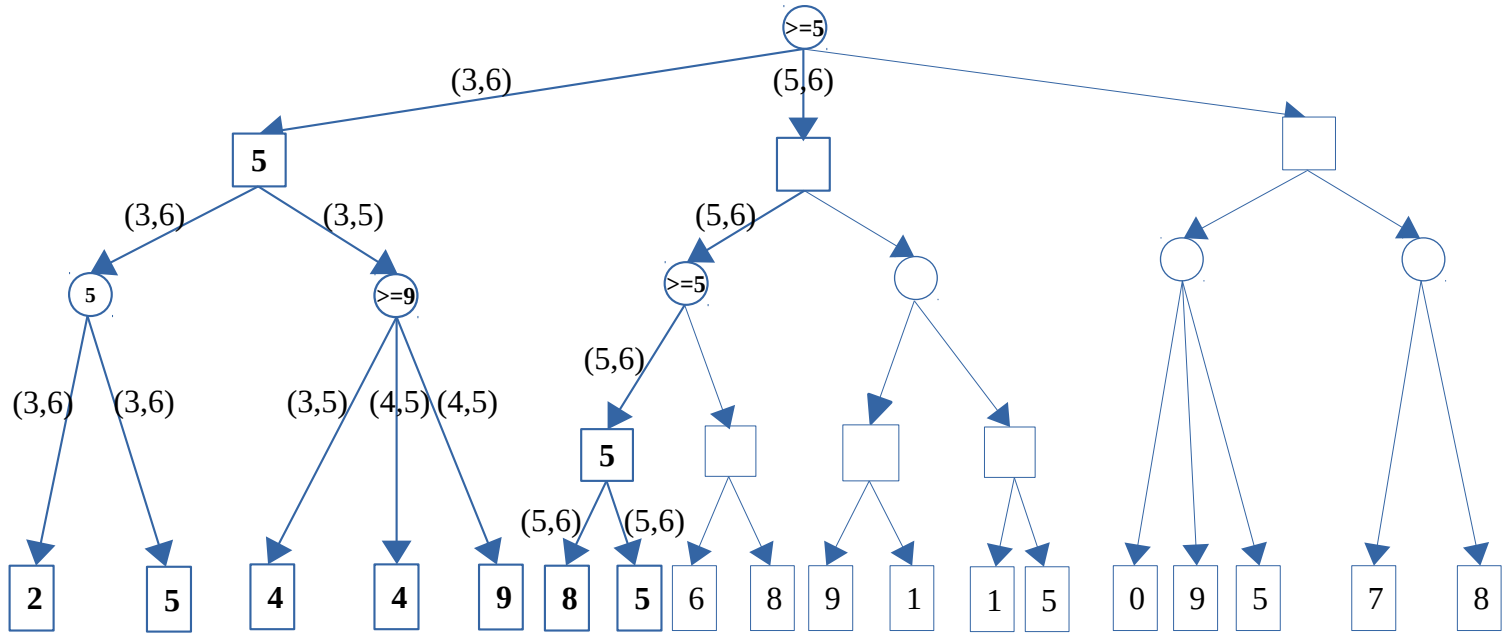
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

Dla drugiego następnika algorytm wywołuje się w oknie (-∞, 8) ∩ (5, 6) = (5, 6). Ponieważ wartości <=8 mogą zmieścić się w przekazanym z góry oknie, więc są interesujące. 5 poprawia bieżace minimum. Nie ma już więcej następników, więc 5 jest ostateczną wartością pozycji.



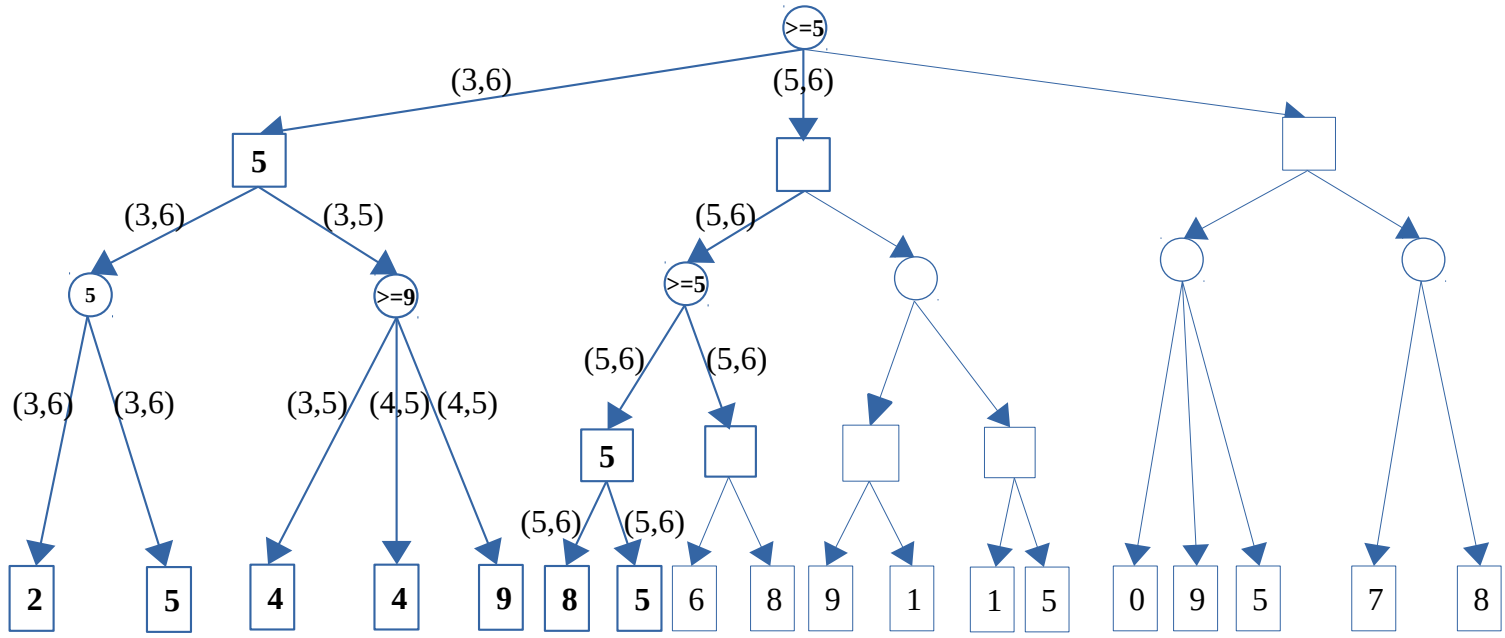
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN ▯ - stan terminalny

5 staje się bieżącą wartością poprzednika zbadanej pozycji. Poprzednik ten jest typu MAX, więc jego ostateczna wartość wyniesie 5 lub więcej. Nie można odciac więc jego drugiego następnika, bo wartości >=5 mogą się zmieścić w oknie (5, 6), tj. [5, +∞) ∩ (5, 6) != 0.



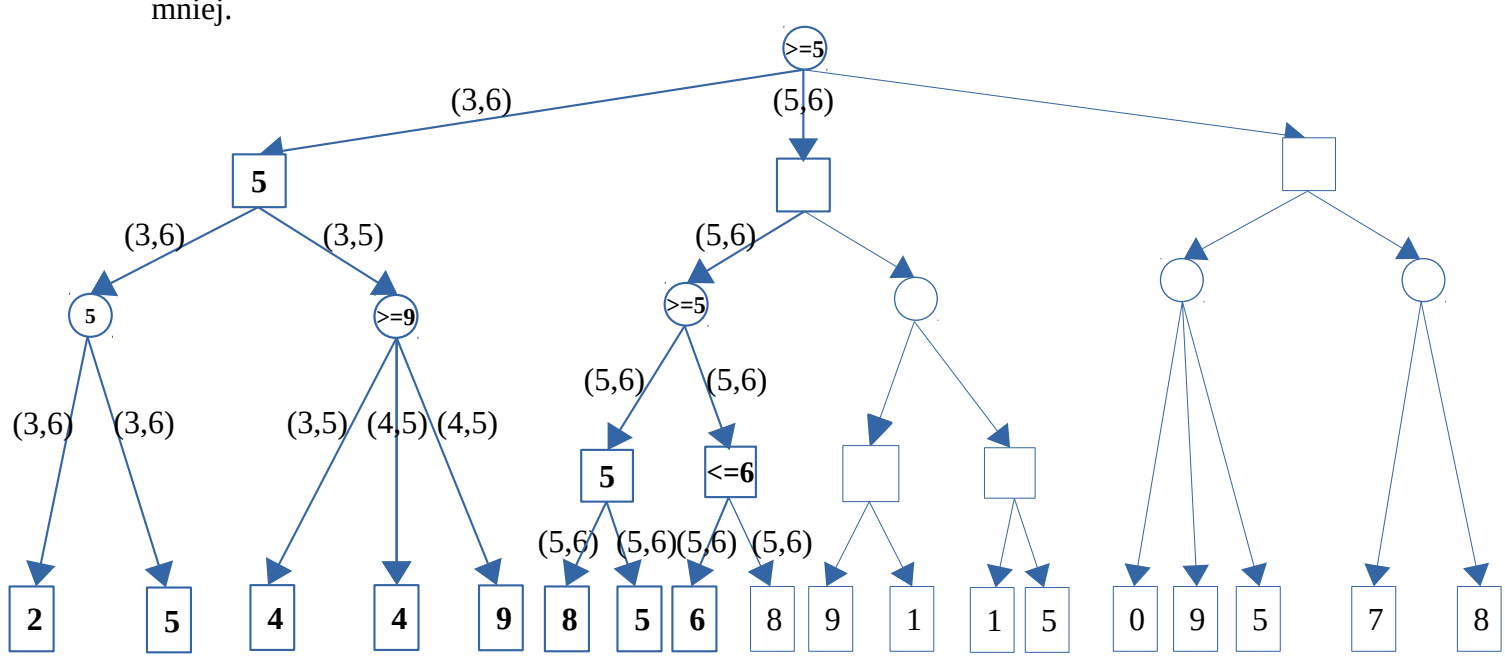
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN ▯ - stan terminalny

Dla drugiego następnika algorytm wywołuje się w oknie [5, +∞) ∩ (5, 6) = (5, 6), czyli tym samym co jego poprzednik.



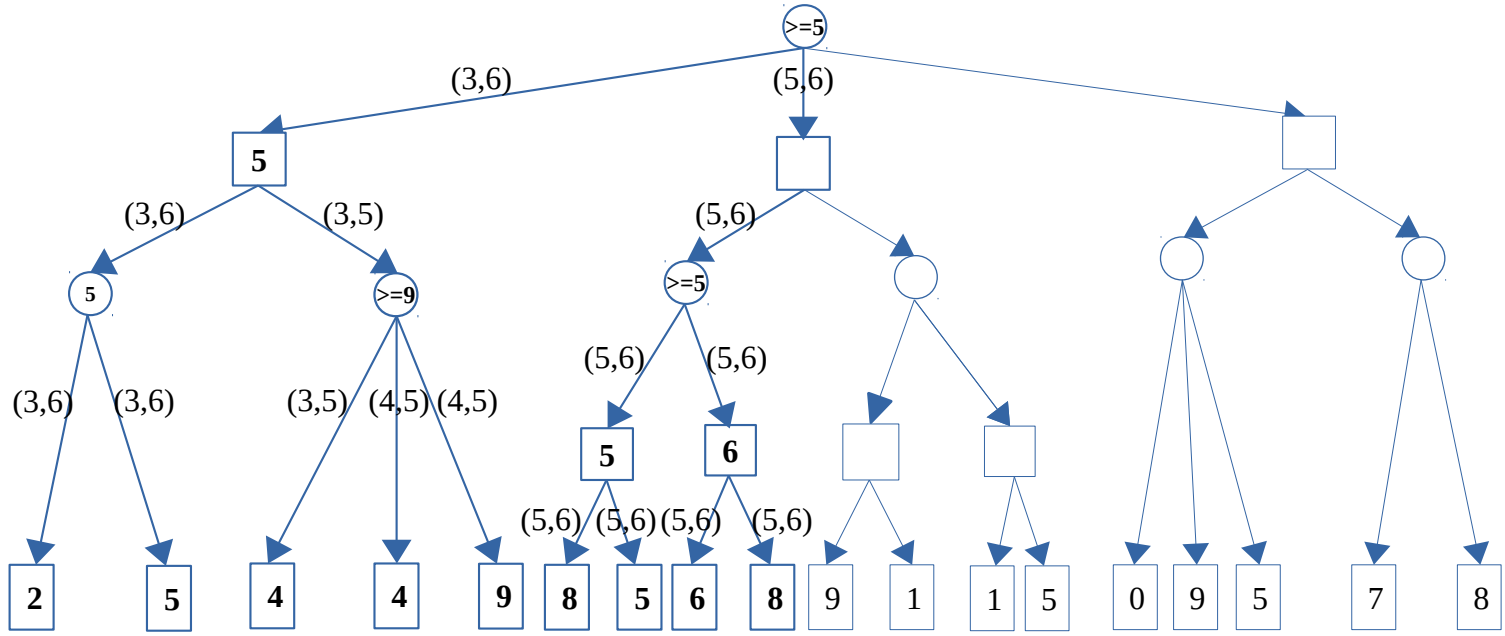
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

Pierwszy następnik badany jest w tym samym oknie co jego poprzednik, czyli (5, 6). Jego wartość 6 staje się bieżacym MINIMUM poprzednika, którego ostateczna wartość wyniesie zatem 6 lub mniej.



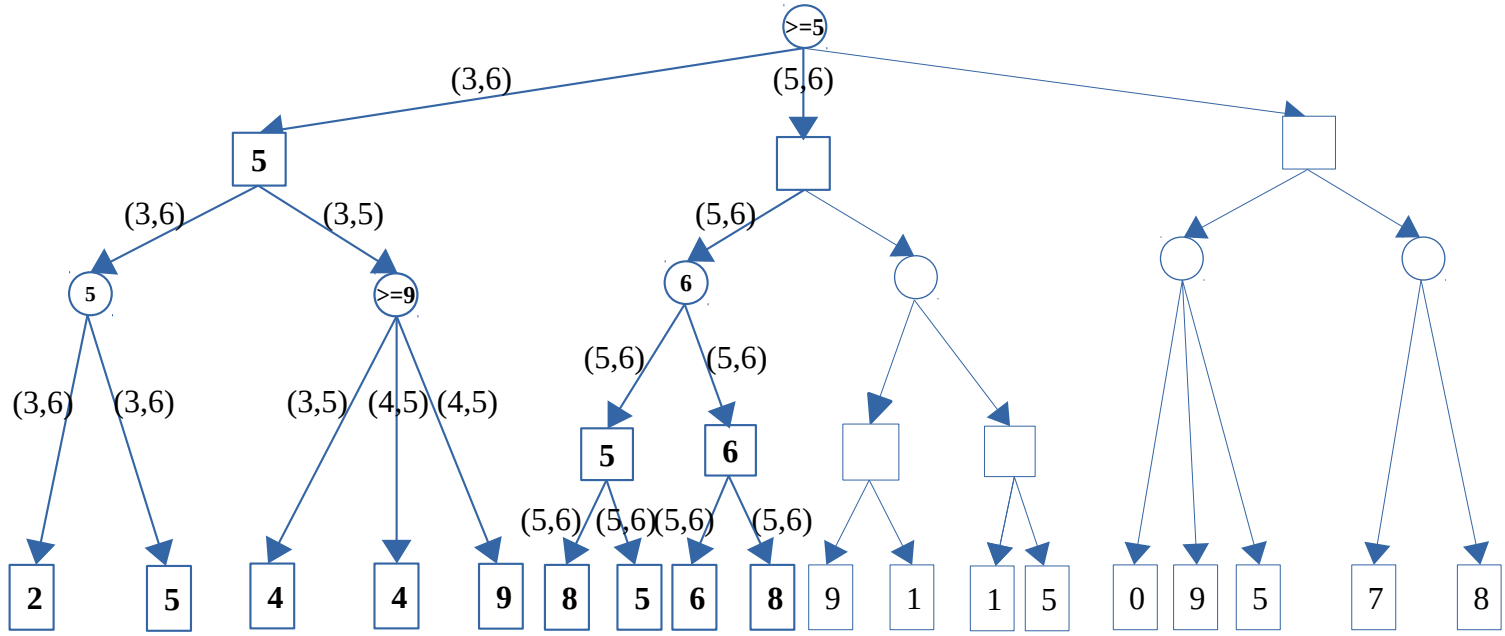
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

Dla drugiego następnika algorytm wywołuje się w tym samym oknie co jego poprzednik, czyli (5,6). 8 nie poprawia bieżacego MINIMUM 6 poprzednika. Nie ma już więcej następników, więc 6 staje się zatem jego ostateczną wartością.



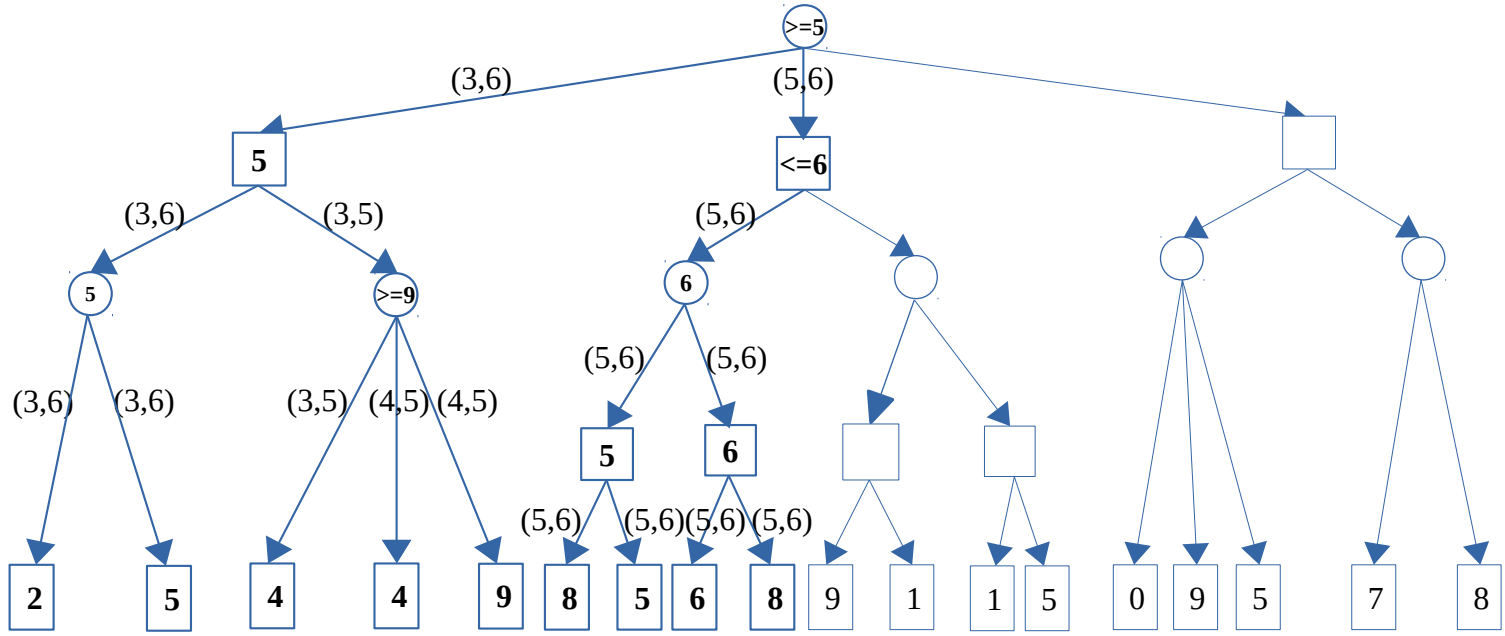
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

6 poprawia bieżace MAXIMUM 5 poprzednika, więc staje się jego ostateczną wartością, bo nie ma on więcej dzieci.



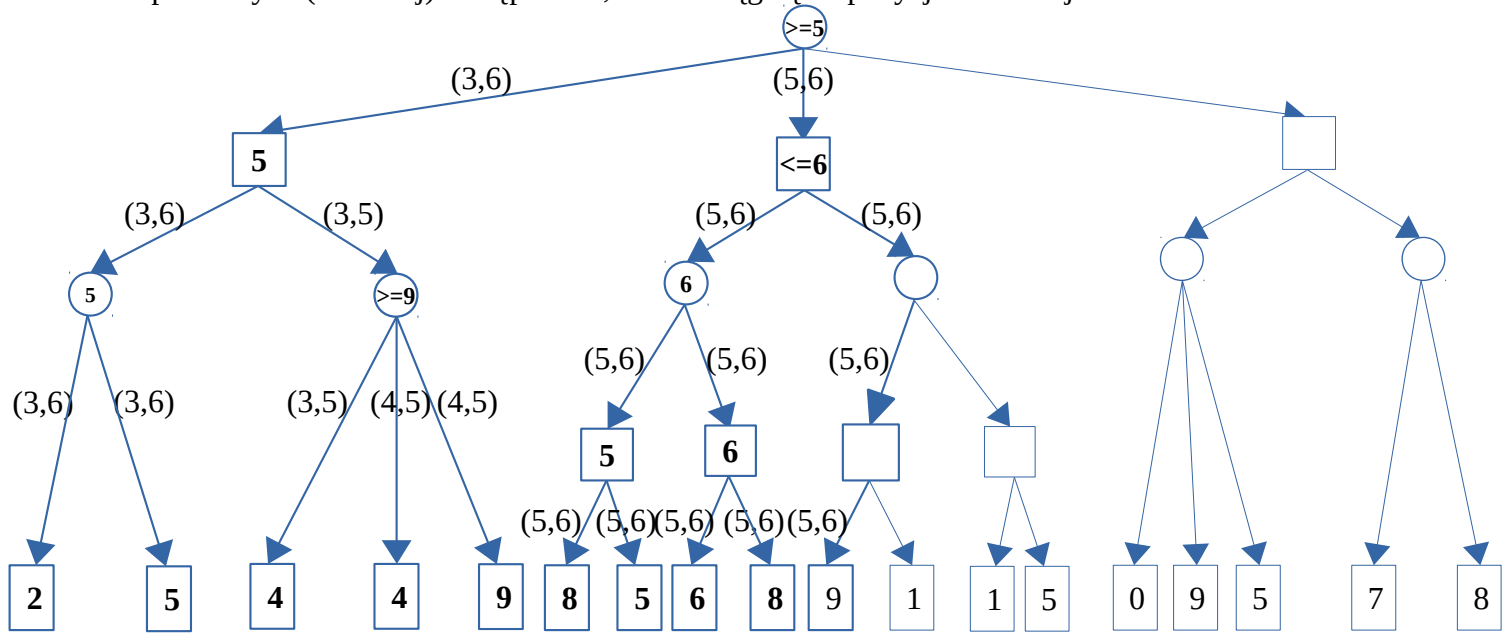
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

6 staje się bieżącą wartością poprzednika zbadanej pozycji. Poprzednik ten jest typu MIN, więc jego ostateczna wartość wyniesie 6 lub mniej. Nie można odciąc więc jego prawego następnika, bo wartość <=6 może się zmieścić w oknie (5, 6), tj. (-∞, 6) ∩ (5, 6) != 0.



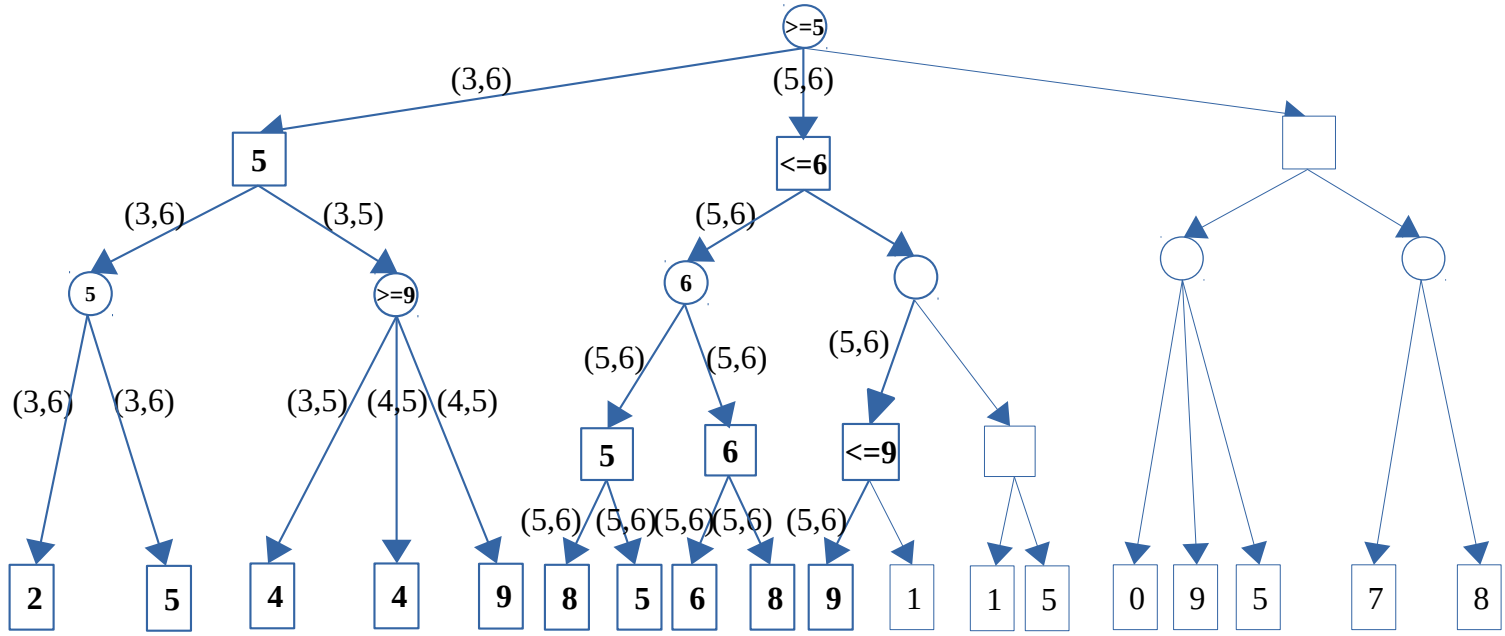
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN ▯ - stan terminalny

Dla drugiego następnika algorytm wywołuje się w oknie (-∞, 6] ∩ (5, 6) = (5, 6), czyli w takim samym oknie jak dla jego poprzednika. Dalej następują rekurencyjne wywołania dla kolejnych, pierwszych (od lewej) następników, aż do osiągnięcia pozycji końcowej 9.



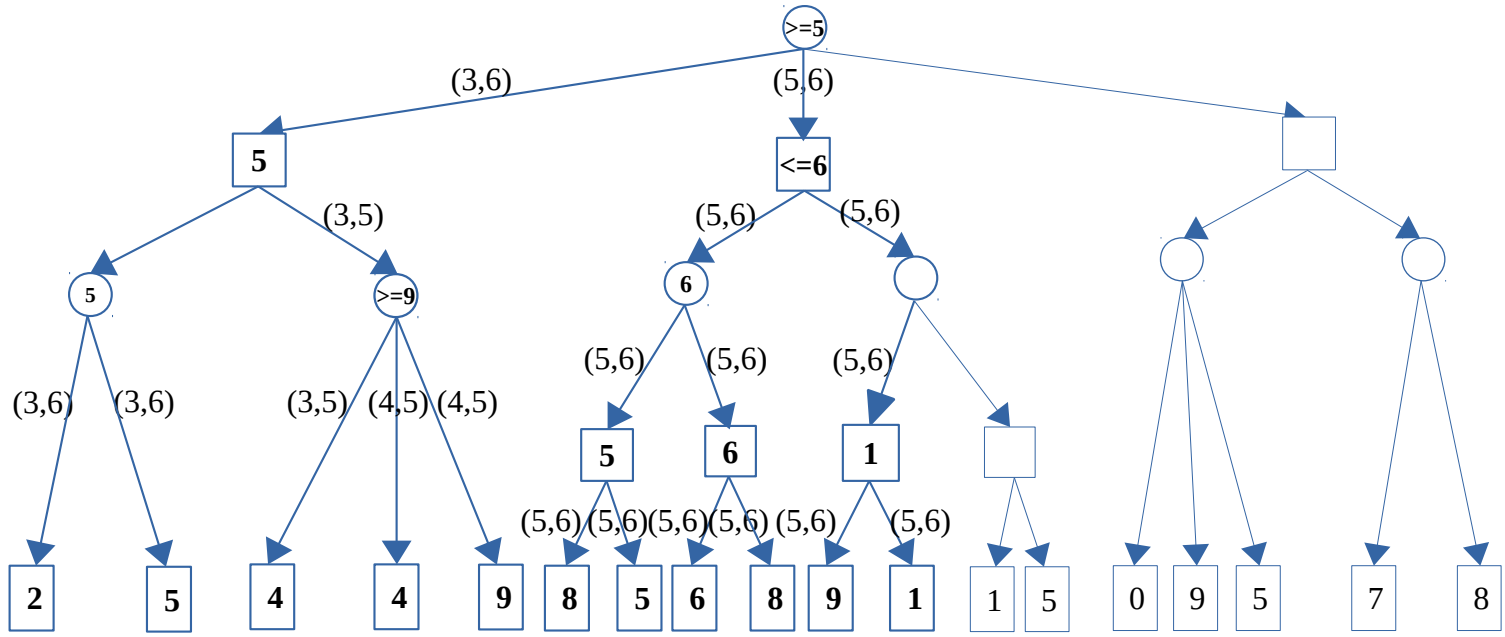
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN ▯ - stan terminalny

9 staje się bieżącą wartością poprzednika osiągniętej pozycji końcowej. Poprzednik ten jest typu MIN, więc jego ostateczna wartość wyniesie 9 lub mniej. Nie można odciąć więc jego drugiego następnika, bo wartość <=9 może się zmieścić w oknie (5, 6), tj. (-∞, 9] ∩ (5, 6) != 0.



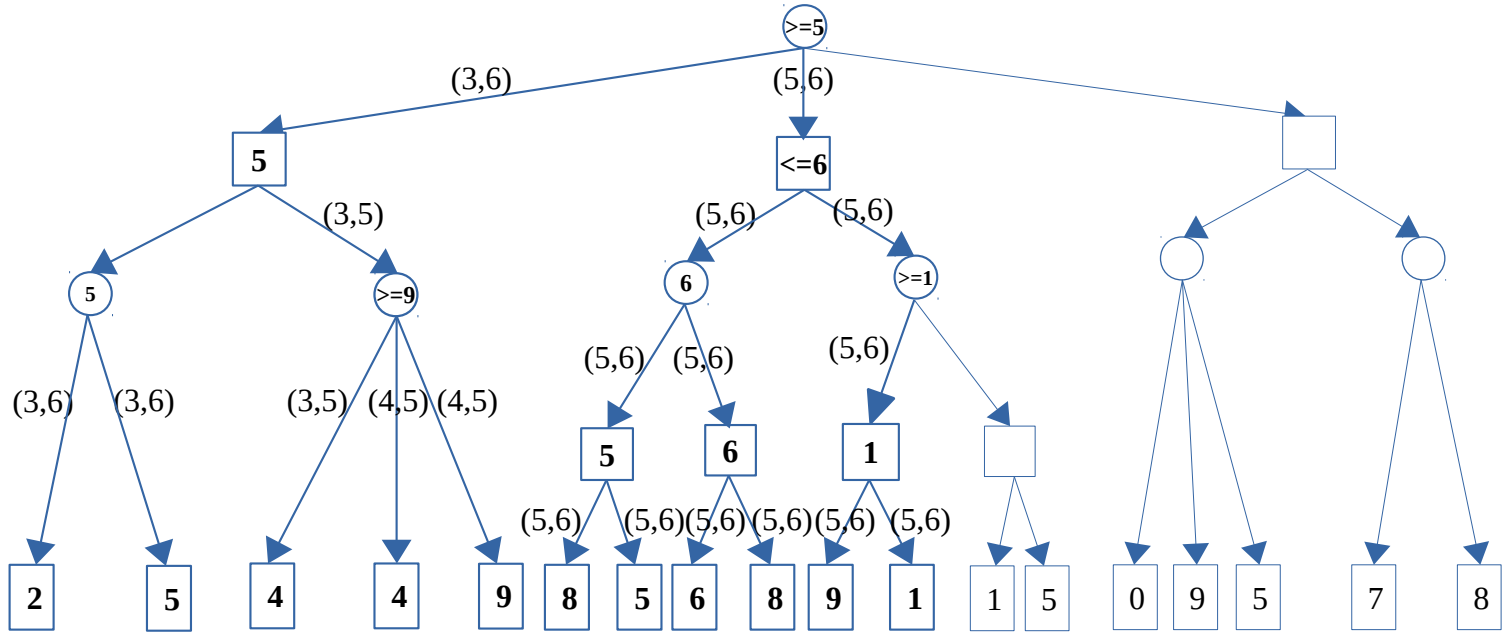
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

Dla drugiego następnika algorytm wywołuje się w oknie (-∞, 9] ∩ (5, 6) = (5, 6). 1 poprawia bieżące MINIMUM. Nie ma już więcej następników, więc 1 jest ostateczną wartością pozycji.



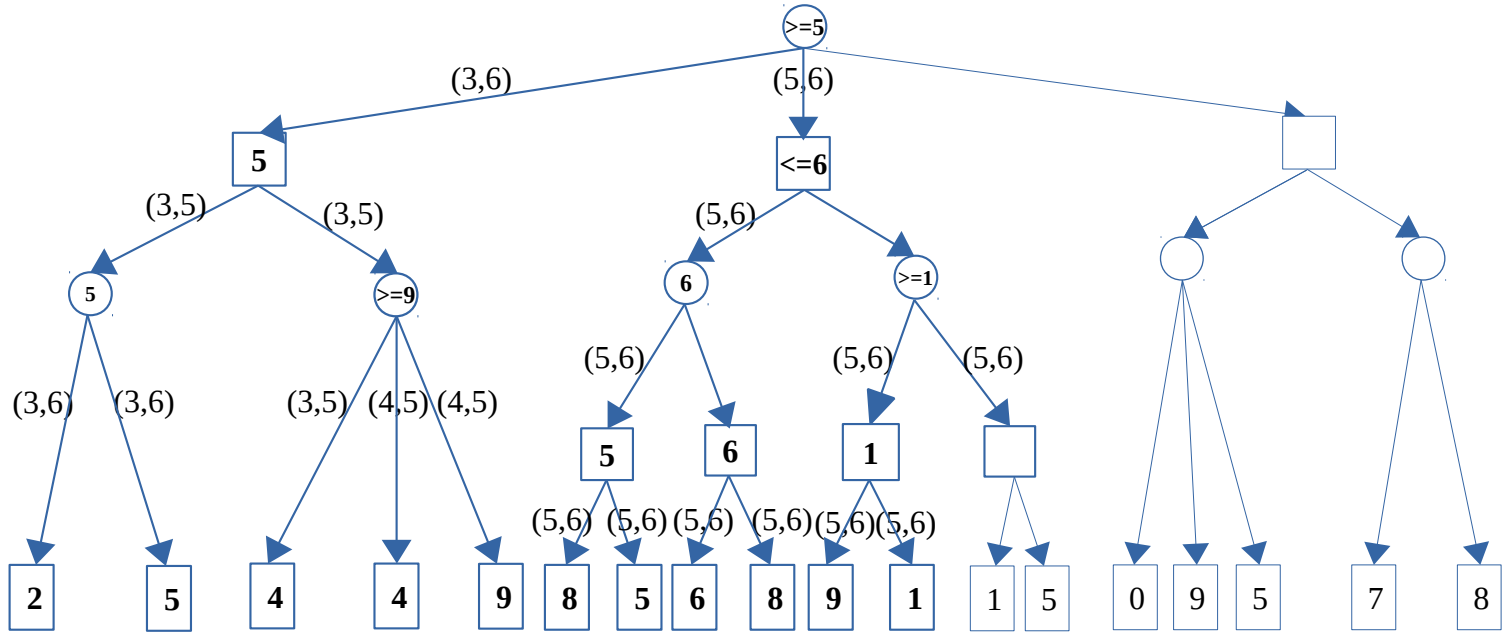
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

1 staje się bieżącą wartością poprzednika. Poprzednik ten jest typu MAX, więc jego ostateczna wartość wyniesie 1 lub więcej. Nie można odciąć więc jego drugiego następnika, bo wartość >=1 może się zmieścić w oknie (5, 6), tj. [1, +∞) ∩ (5, 6) != 0.



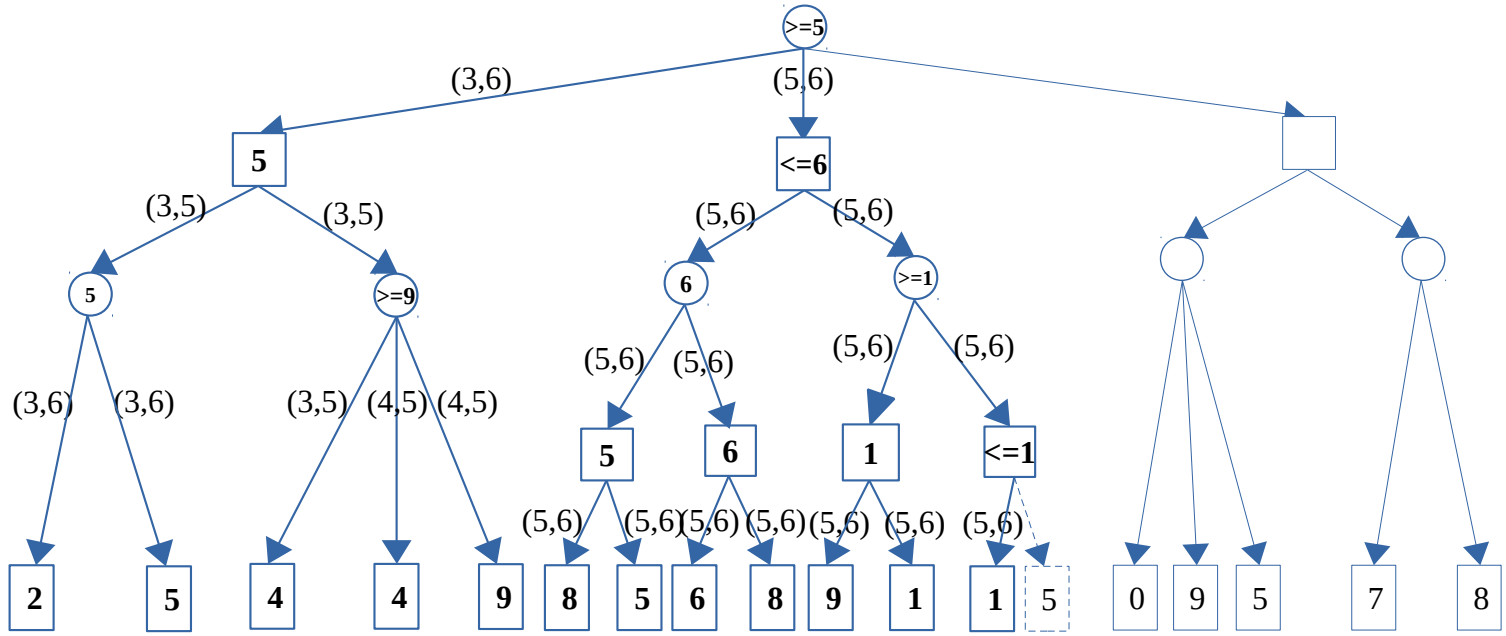
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN ▯ - stan terminalny

Dla drugiego następnika algorytm wywołuje się w oknie [1, +∞) ∩ (5, 6) = (5, 6).



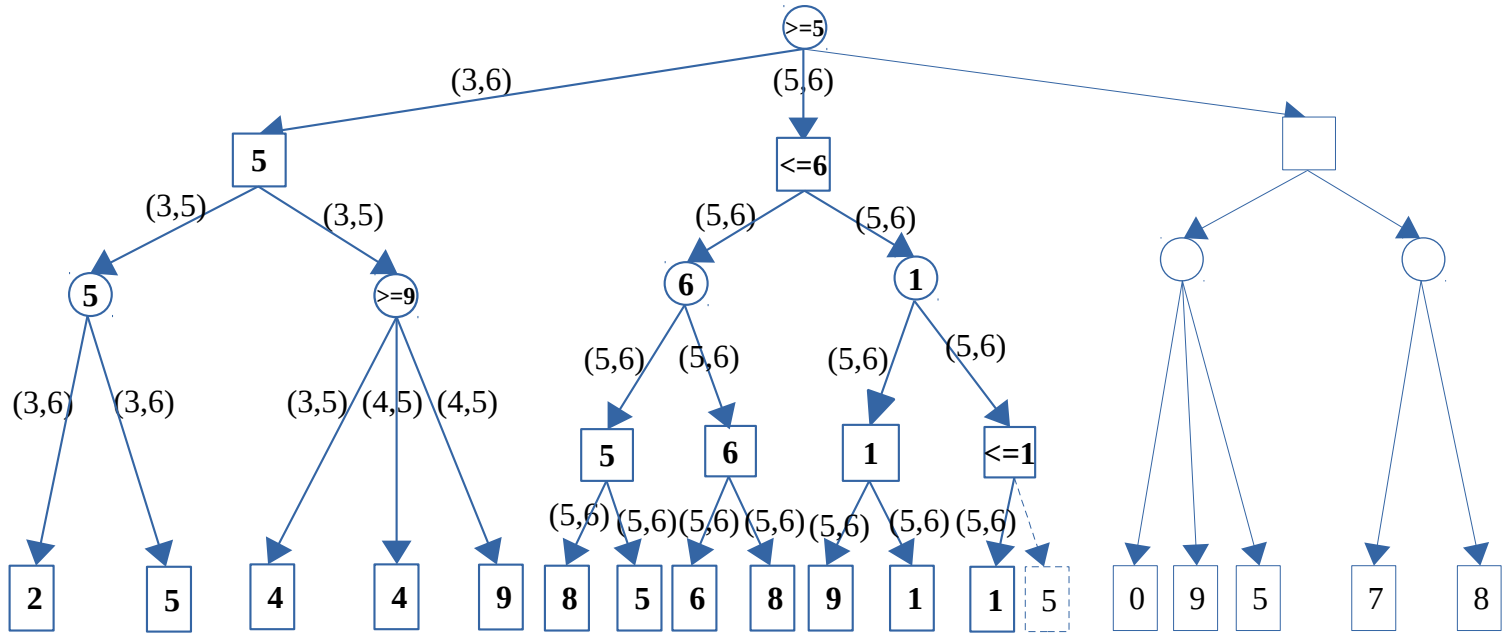
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN ▯ - stan terminalny

Pierwszy następnik badany jest w tym samym oknie co jego poprzednik, czyli (5, 6). Jego wartość 1 staje się bieżącym MINIMUM poprzednika, którego ostateczna wartość wyniesie zatem 1 lub mniej. Ta ostateczna wartość <=1 nie zmieści się w oknie (5, 6), tj. (-∞, 1] ∩ (5, 6) = 0. Następuje więc odciecie kolejnego następnika. Nie jest on w ogóle badany (jego wartość nie wpłynie na wartość korzenia). Bieżące MINIMIM, czyli 1 staje się ostateczną wartością (zwróconą z) badanego węzła.



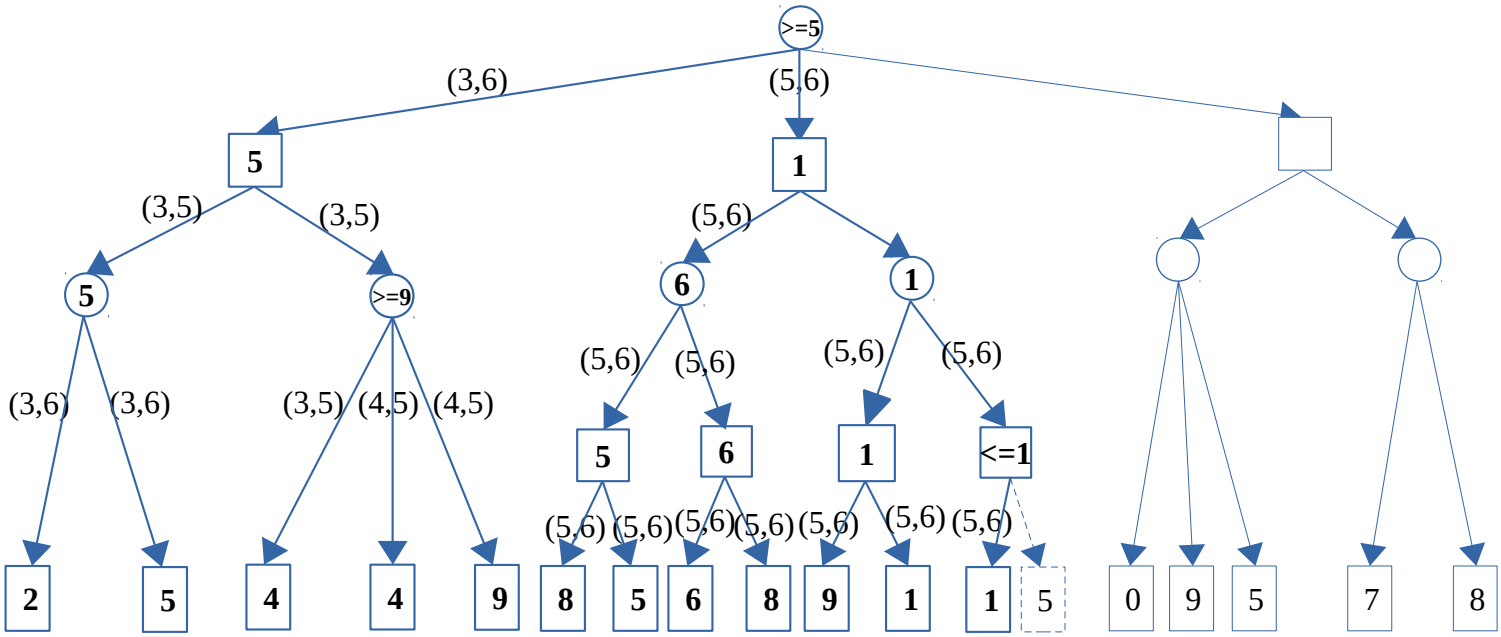
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN ▯ - stan terminalny

Ta 1 nie poprawia bieżącego maksimum poprzednika. 1 staje się zatem jego ostateczną wartością, bo nie ma on więcej dzieci.



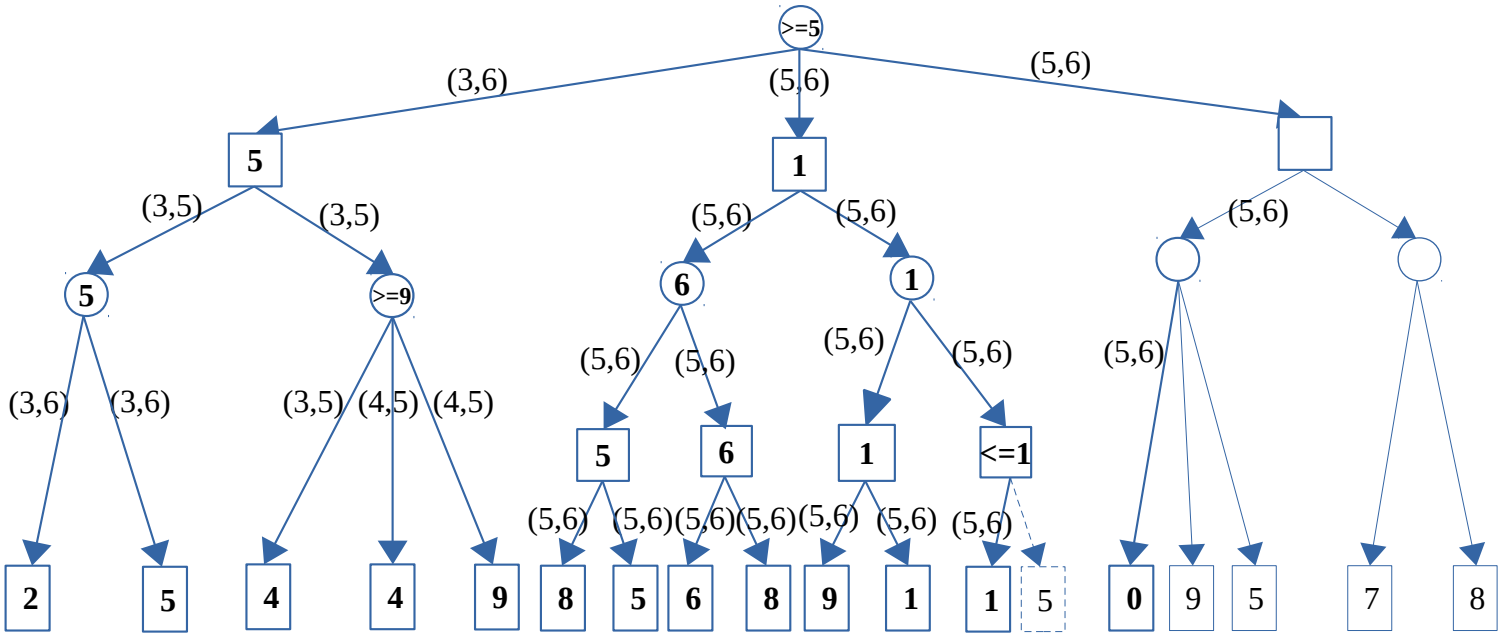
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN ▯ - stan terminalny

Ta 1 poprawia bieżące MINIMUM środkowego następnika korzenia. 1 staje się zatem jego ostateczną wartością, bo nie ma on więcej dzieci.



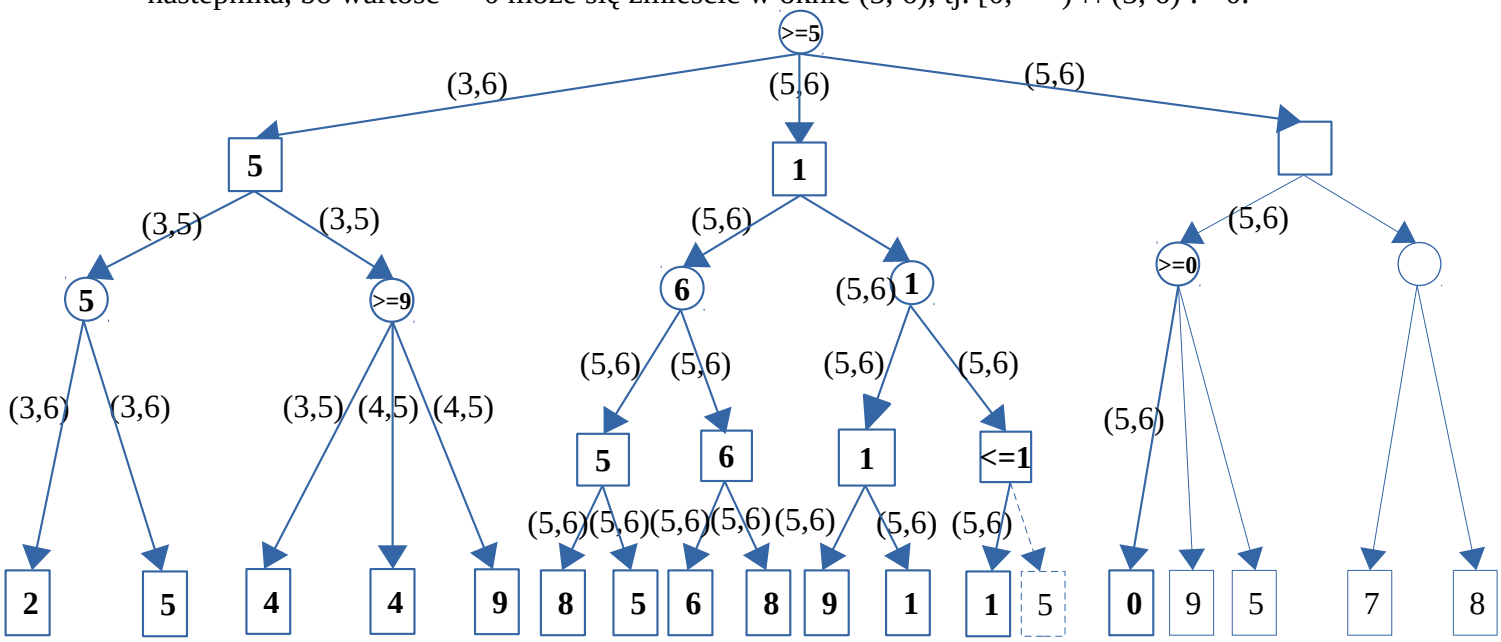
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN ▯ - stan terminalny

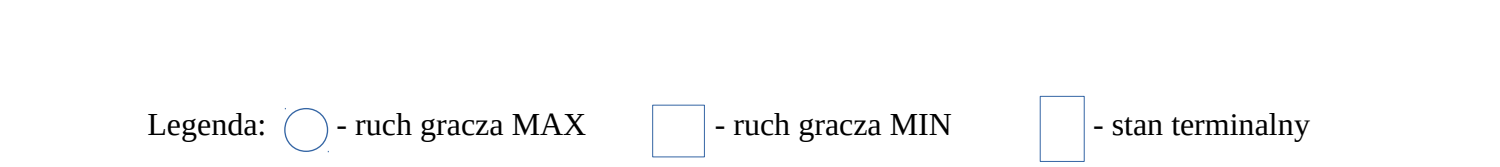
Dla prawego następnika korzenia algorytm wywołuje się w oknie $[5, +\infty) \cap (3, 6) = (5, 6)$. Dalej następują rekurencyjne wywołania dla kolejnych pierwszych (od lewej) następników, aż do osiągnięcia pozycji końcowej o wartości 0.



Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN ▯ - stan terminalny

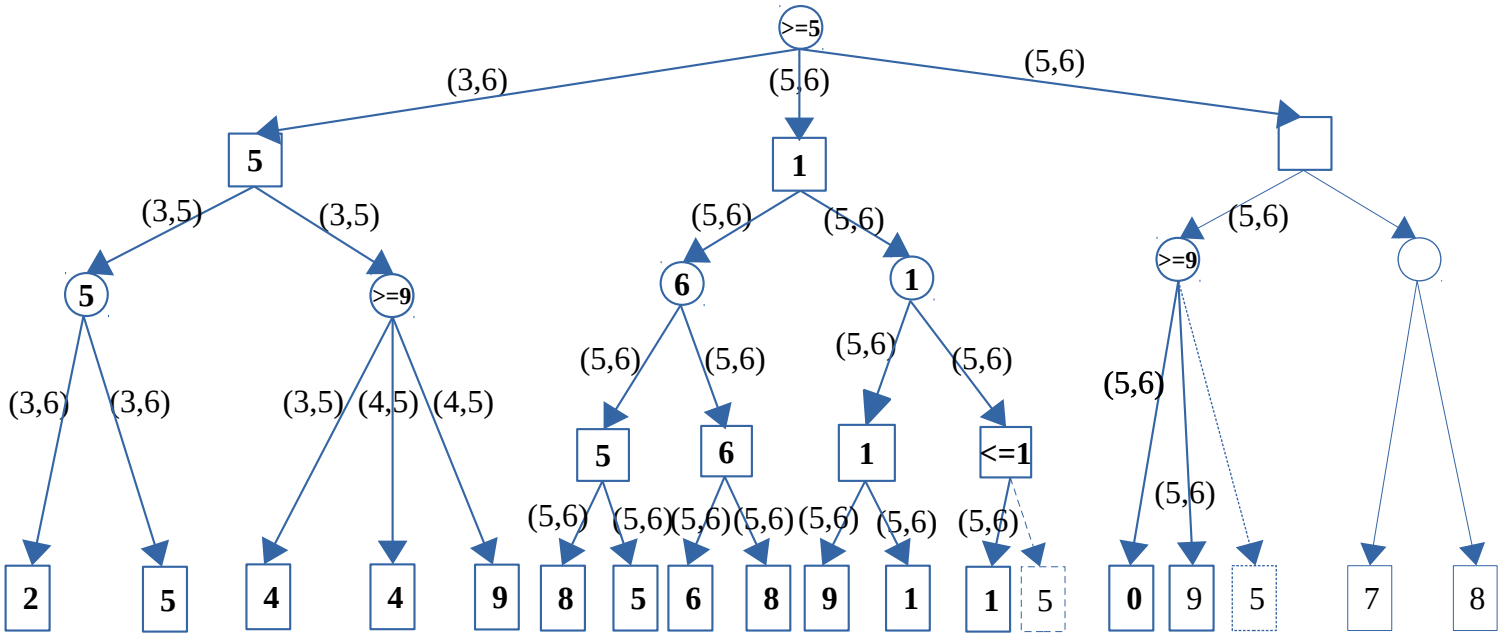
0 staje się biezaca wartoscia poprzednika osiagnietej pozycji koncowej. Poprzednik ten jest typu MAX, więc jego ostateczna wartosc wyniesie 0 lub więcej. Nie można odciac więc jego drugiego nastepnika, bo wartosc >=0 może się zmiescic w oknie (5, 6), tj. [0, +∞) ∩ (5, 6) != 0.

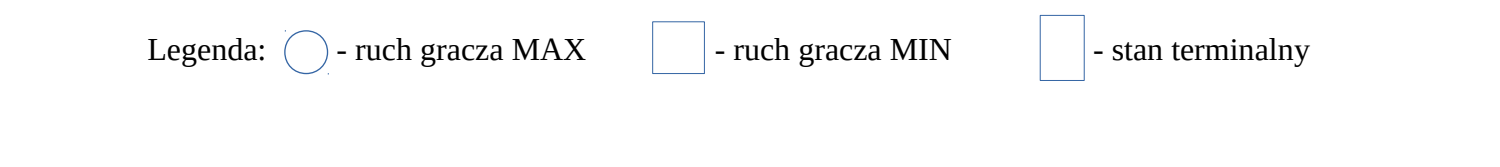




Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

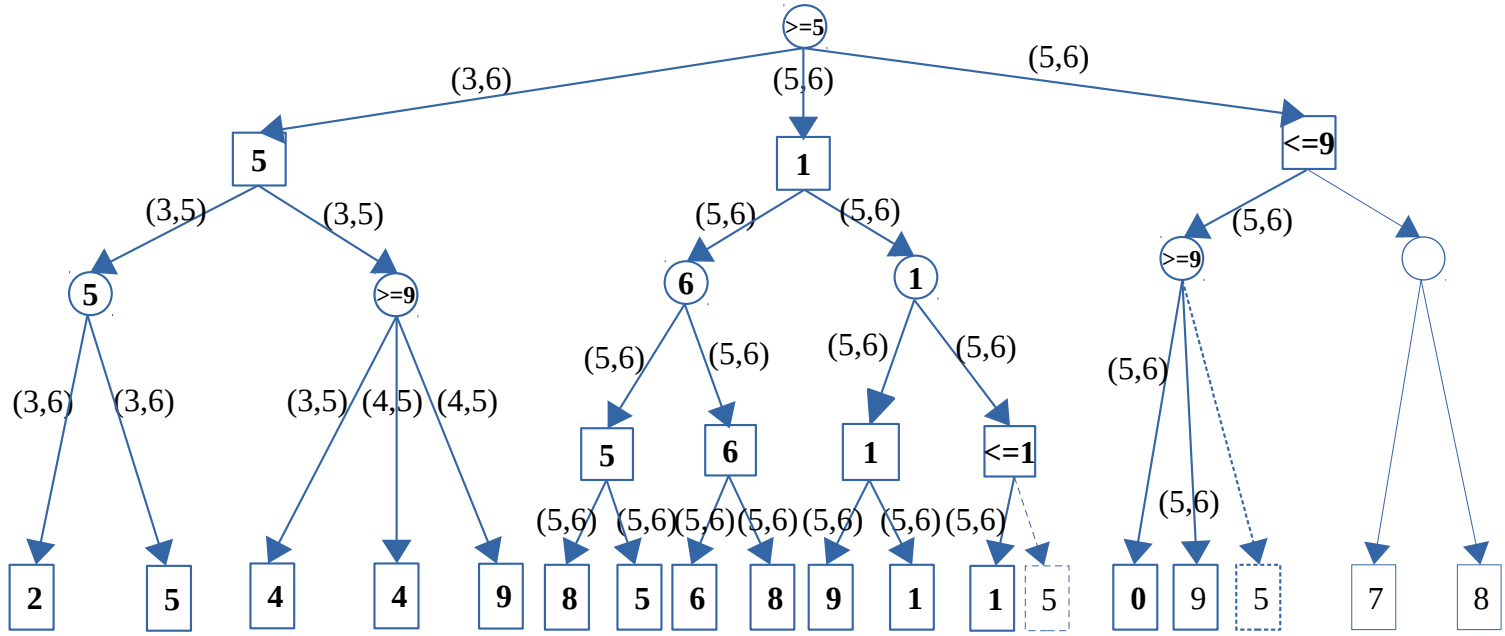
Dla drugiego następnika algorytm wywołuje się w oknie [0, +∞) ∩ (5, 6) = (5, 6). 9 poprawia bieżace MAKSIMUM 0. Ta ostateczna wartość nie zmieści się w oknie (5, 6), tj [9, +∞) ∩ (5, 6) =0. Następuje więc odciecie kolejnego następnika. Nie jest on w ogóle badany (jego wartość nie wpłynie na wartość korzenia). Bieżace MAKSIMUM, czyli 9 staje się ostateczną wartością (zwróconą z) badanego węzła.





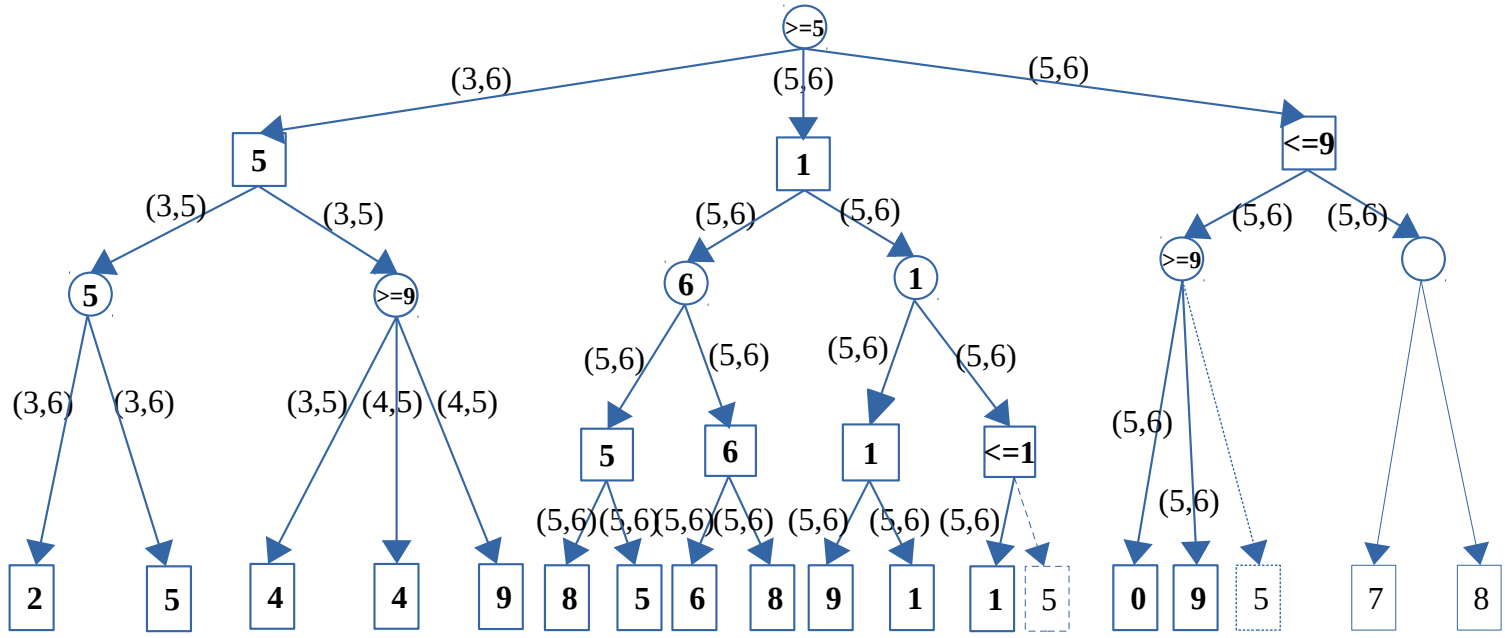
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

9 staje się bieżącą wartością poprzednika zbadanej pozycji. Poprzednik ten jest typu MIN, więc jego ostateczna wartość wyniesie 9 lub mniej. Nie można odciąć więc jego drugiego następnika, bo wartość <=9 może się zmieścić w oknie (5, 6), tj. (-∞, 9] ∩ (5, 6) != 0.



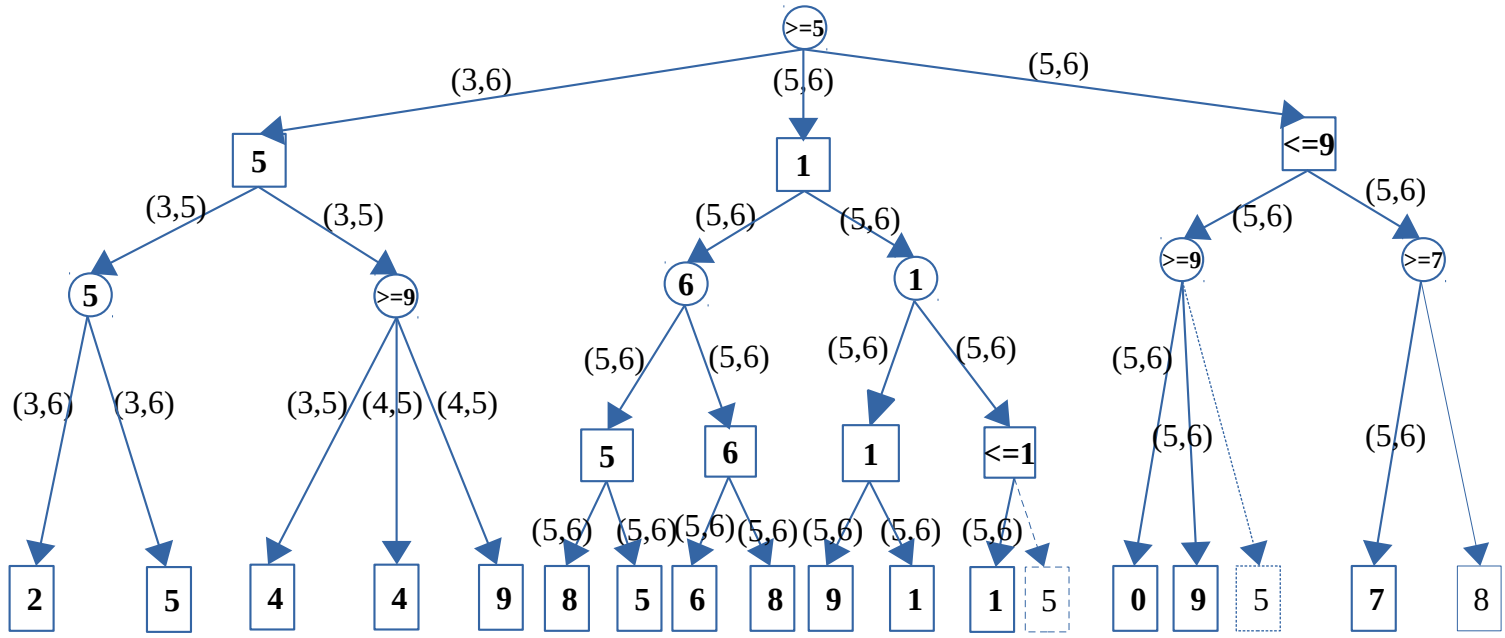
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN ▯ - stan terminalny

Dla drugiego następnika algorytm wywołuje się w oknie (-∞, 9] ∩ (5, 6) = (5, 6).



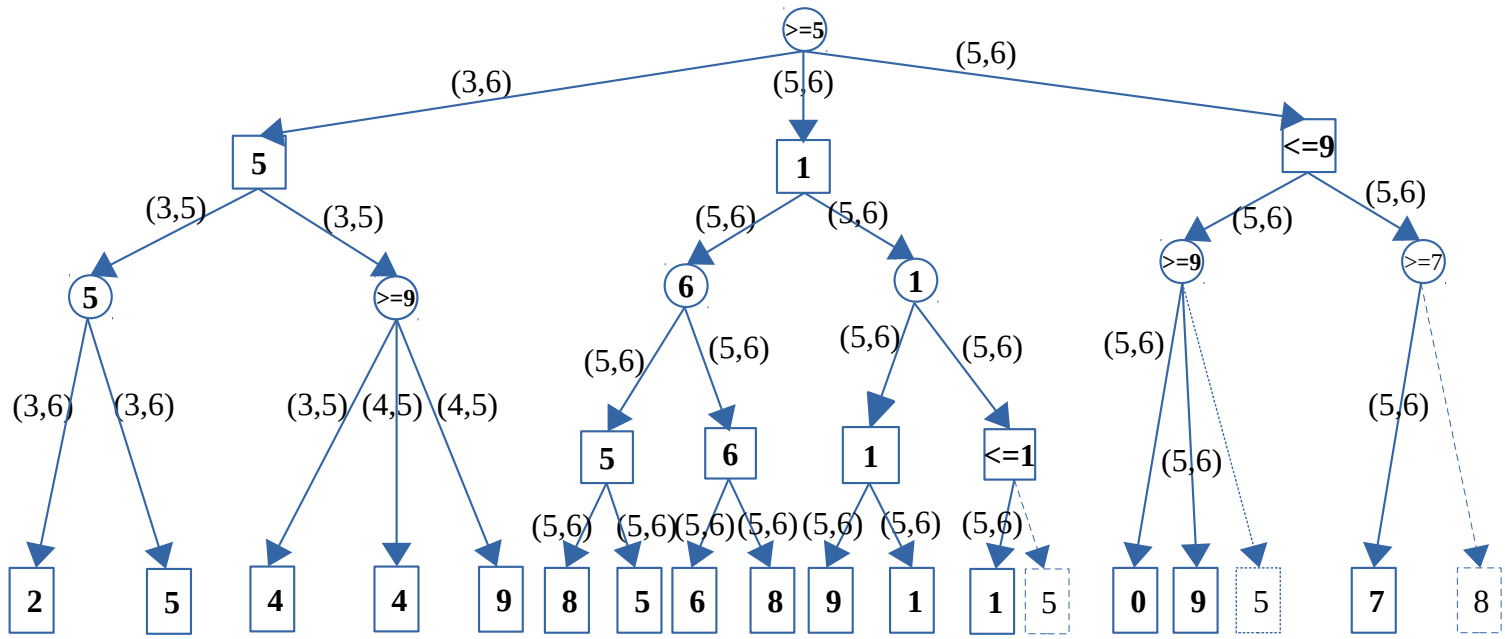
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN ▯ - stan terminalny

Pierwszy następnik badany jest w tym samym oknie co jego poprzednik, czyli (5, 6). Jego wartość 7 staje się bieżącym MAKSIMUM poprzednika, którego ostateczna wartość wyniesie zatem >=7.



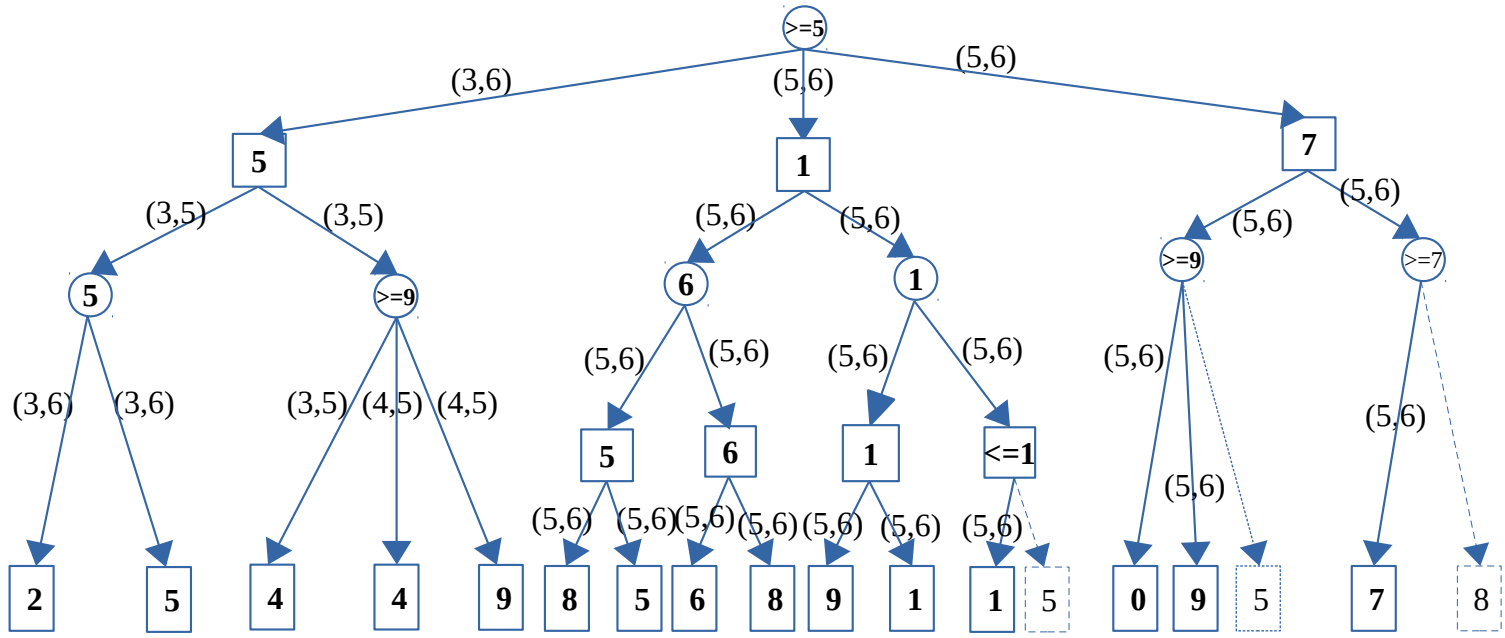
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

Ta ostateczna wartość >=7 nie zmieści się w oknie (5, 6), tj. [7, +∞) ∩ (5, 6) = 0. Następuje więc odcięcie kolejnego nastepnika. Nie jest on w ogóle badany (jego wartość nie wpłynie na wartość korzenia). Bieżace MAKSIMUM, czyli 7 staje się ostateczną wartością (zwróconą z) badanego węzła.



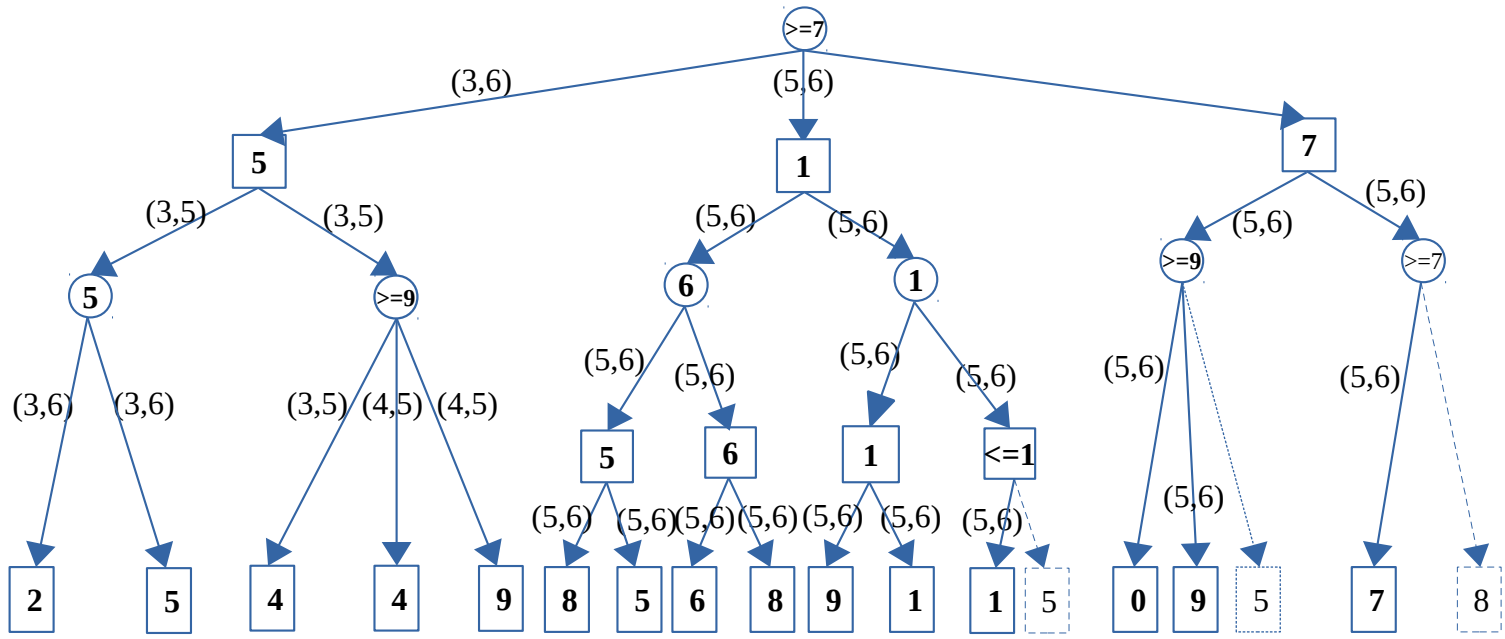
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

Ta 7 poprawia biezące MINIMUM 9 prawego następnika korzenia. 7 staje się więc jego ostateczną wartością, bo nie ma on więcej dzieci.



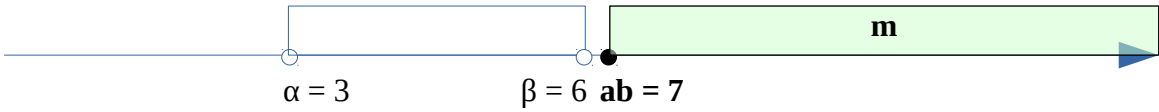
Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

7 staje się ostateczna wartoscia korzenia. Korzen ten jest typu MAX więc jego ostateczna wartosc wyniesie 7 lub więcej.



Legenda: ◯ - ruch gracza MAX □ - ruch gracza MIN □ - stan terminalny

MINMAX(s) >= 7



Wartość minimaksowa wynosi **co najmniej** 7.

Na podstawie twierdzenia o wynikach algorytmu α-β:

ab = AlfaBeta(s, α, β) oraz **m** = MinMax(s).

ab – wartość obliczona dla pozycji algorytmem alfa-beta w oknie (3, 6).

m – wartość minimaksowa tej samej pozycji.

Wtedy możemy wyróżnić trzy sytuacje:

α >= ab ⇒ ab >= m,

α < ab < β ⇒ ab = m,

β <= ab ⇒ ab <= m.

Autor: Piotr Włodarczyk