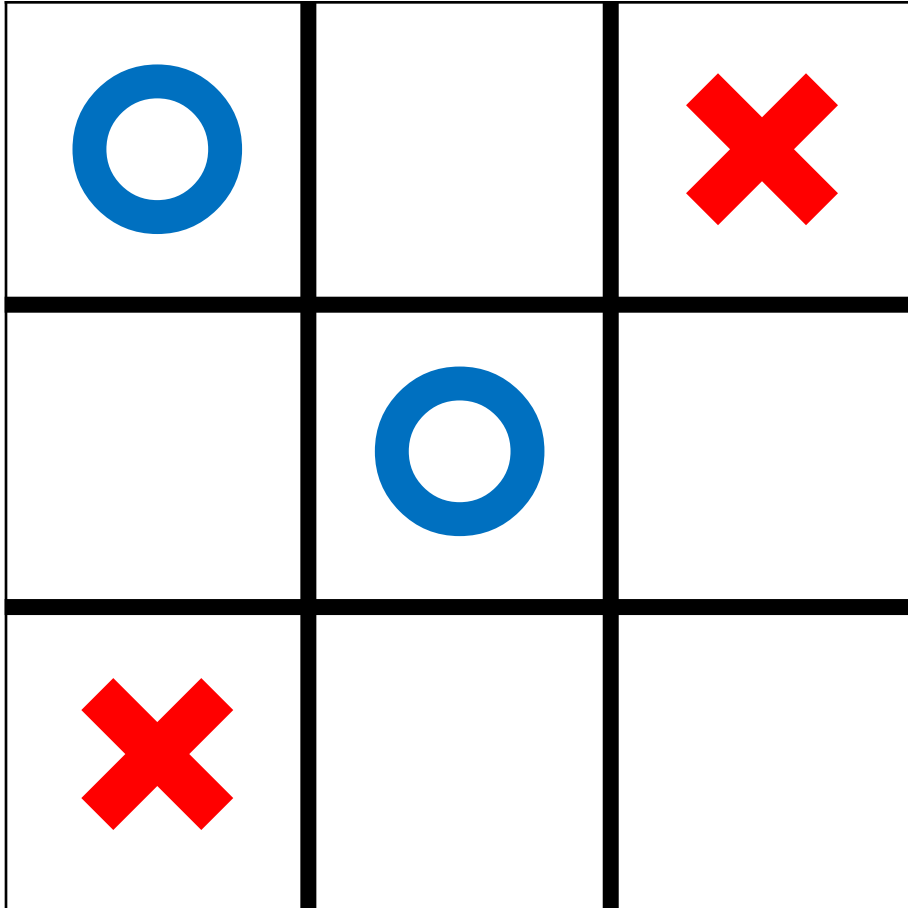


KI-Algorithmen für 2-Personen-Spiele: Minimax

X ist am Zug. Wer gewinnt?



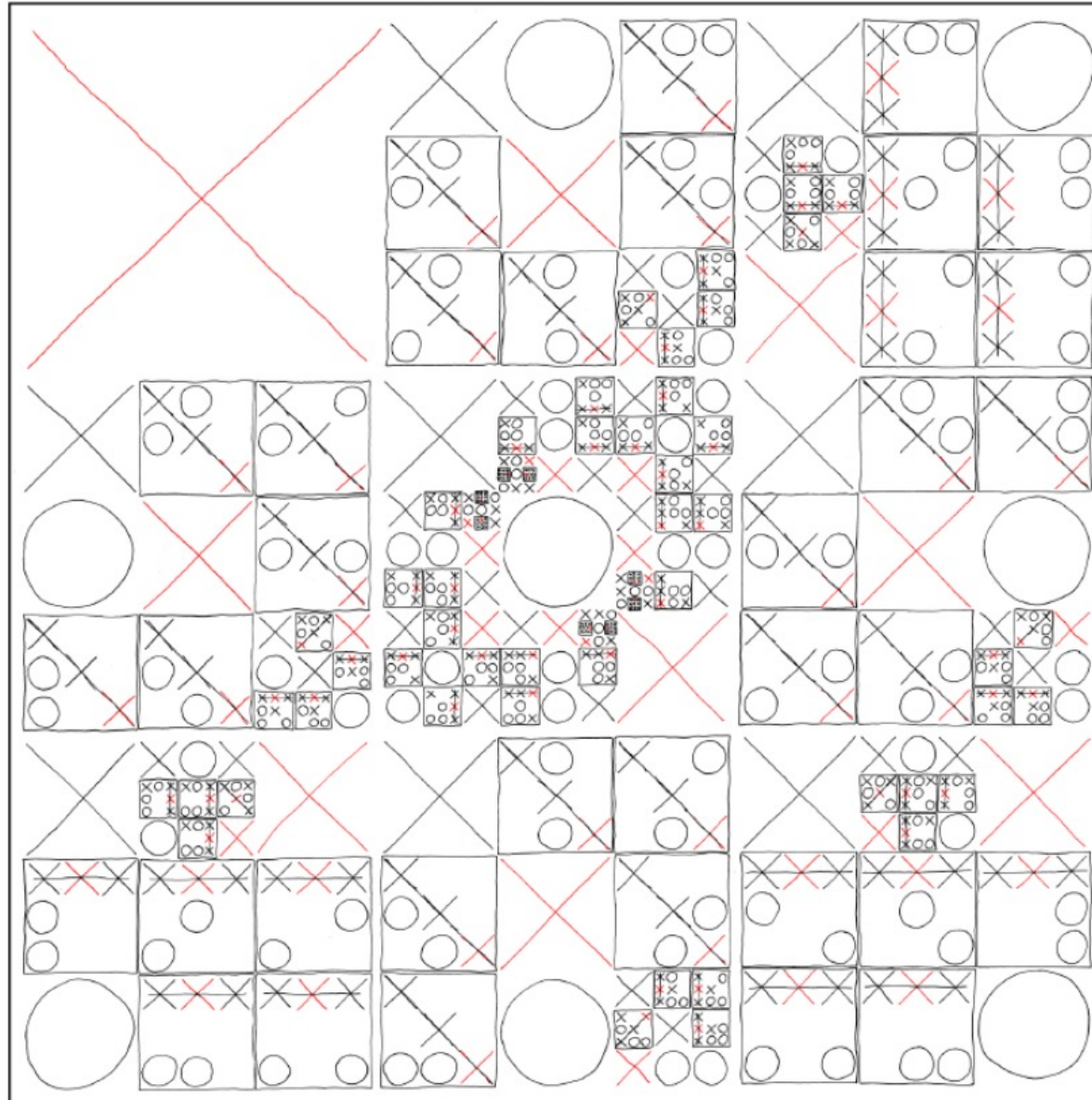
- X kann einen Sieg erzwingen
- muss dafür aber optimal spielen
- sonst gewinnt O

- Um zu erkennen, dass X sicher gewinnt, muss man 3 Züge in die Zukunft **planen**
- Die Zukunft zu antizipieren und zu beeinflussen ist ein zentraler Aspekt von Intelligenz

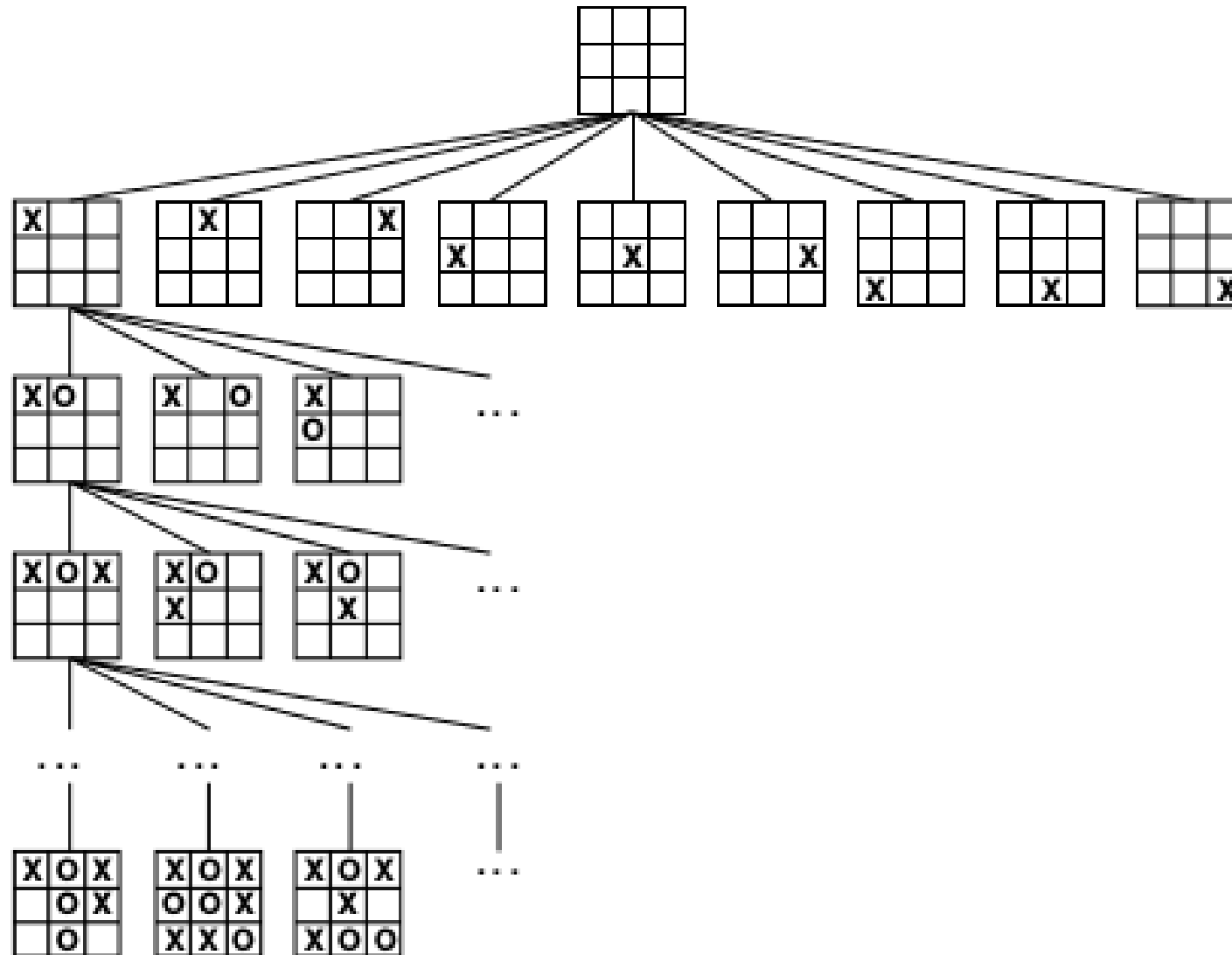
COMPLETE MAP OF OPTIMAL TIC-TAC-TOE MOVES

YOUR MOVE IS GIVEN BY THE POSITION OF THE LARGEST RED SYMBOL ON THE GRID. WHEN YOUR OPPONENT PICKS A MOVE, ZOOM IN ON THE REGION OF THE GRID WHERE THEY WENT. REPEAT.

MAP FOR X:

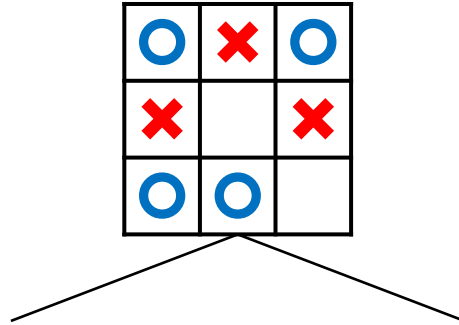


Der Spielbaum von Tic Tac Toe (Ausschnitt!)



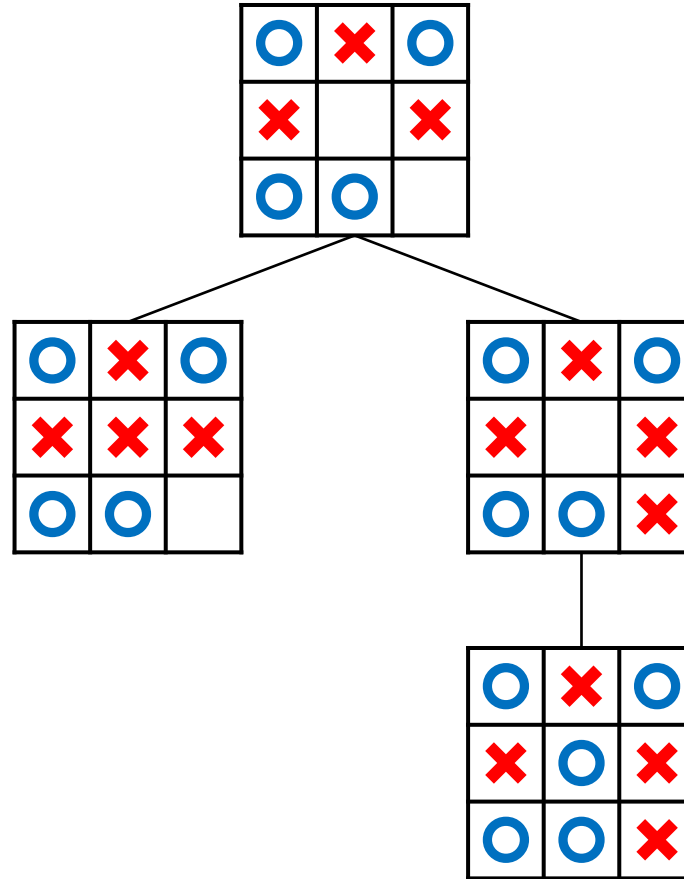
Vervollständige den Spielbaum.

(X ist am Zug)

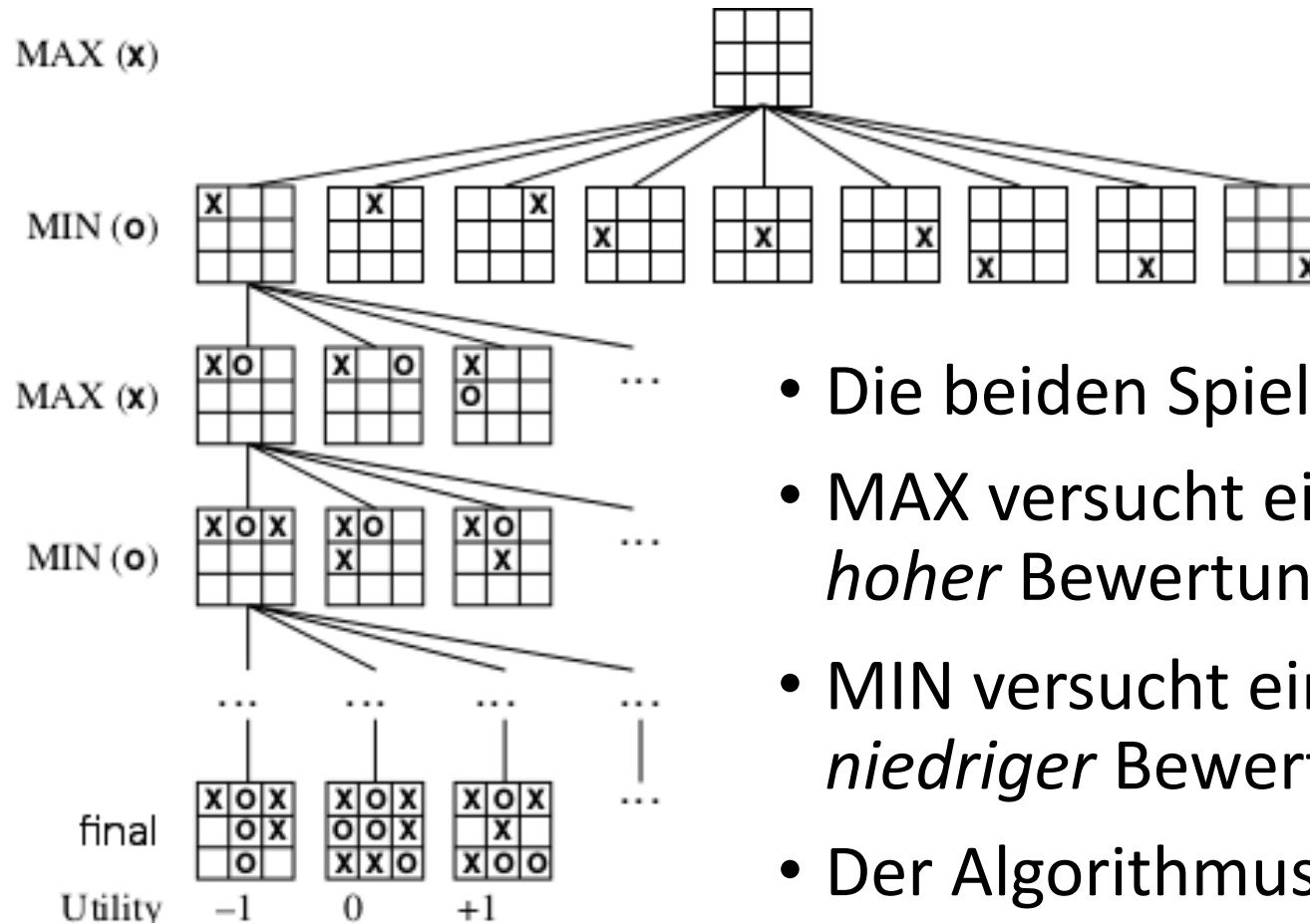


Vervollständige den Spielbaum.

(X ist am Zug)



Warum heißt es „Minimax“?

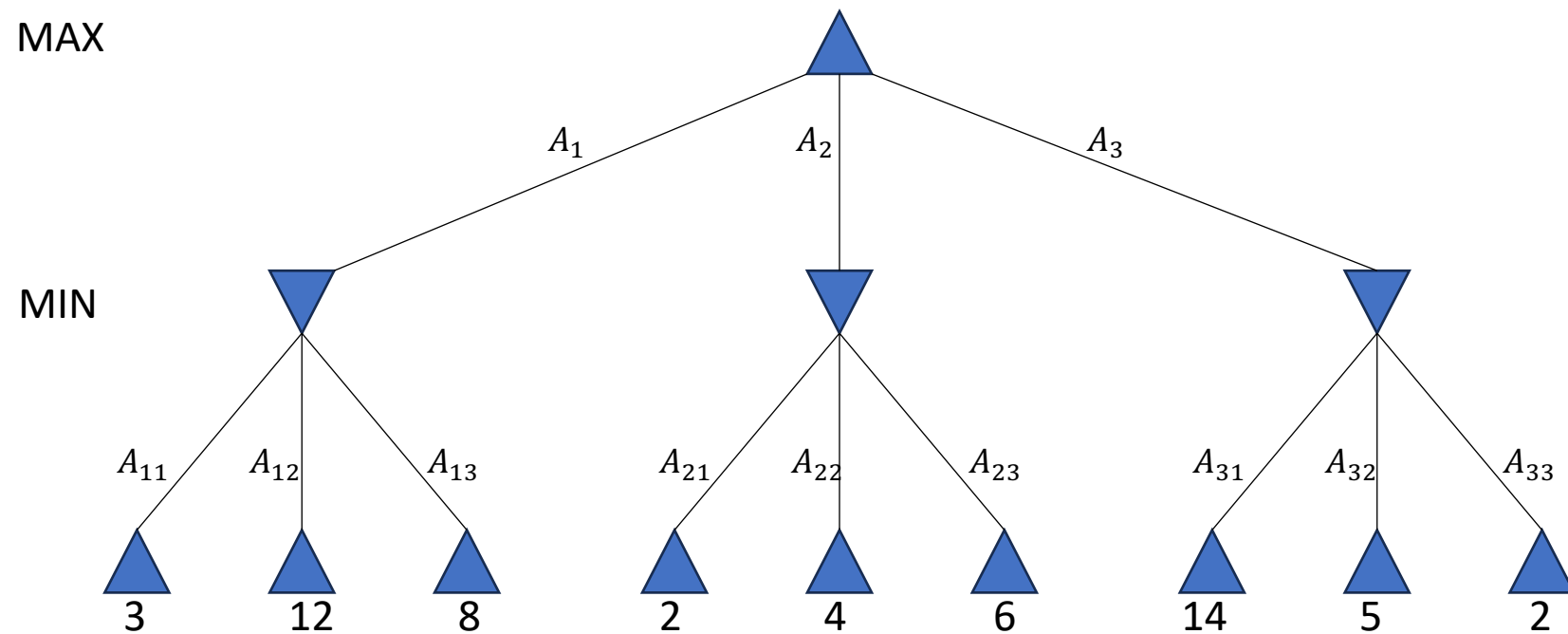


- Die beiden Spieler werden MAX und MIN genannt
- MAX versucht einen Zielzustand mit möglichst *hoher* Bewertung zu erreichen
- MIN versucht einen Zielzustand mit möglichst *niedriger* Bewertung zu
- Der Algorithmus betrachtet sich standardmäßig als MAX-Spieler

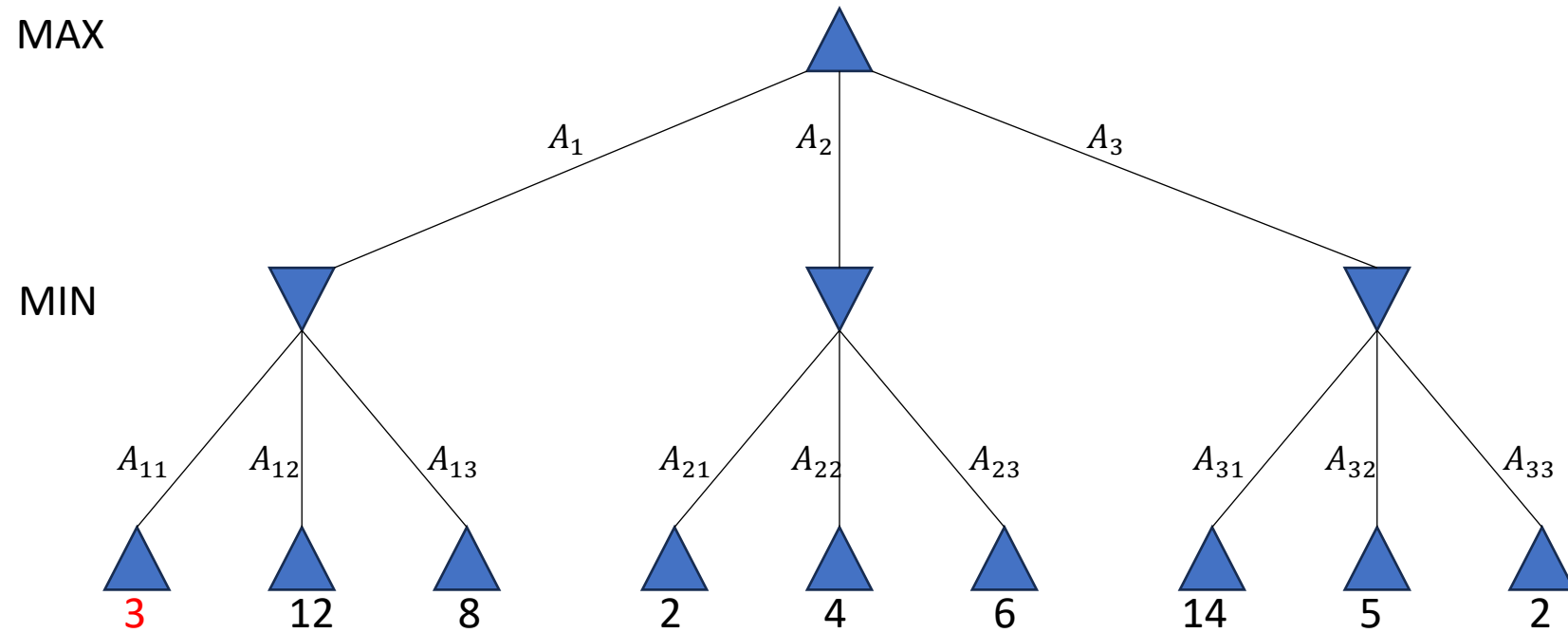
Der Minimax-Algorithmus

- Durchlaufe den Baum in **Tiefensuche** und berechne dabei jeweils den Wert dieses Zustands und den besten Folgezug
- Falls du in einem Endzustand e bist, ergibt sich der Wert direkt durch die Bewertungsfunktion als $b(e)$. Gib diesen Wert an den vorhergehenden Knoten zurück
- Falls du in keinem Endzustand bist, dann unterscheide:
 - MIN ist am Zug: Bewertung ist das **Minimum** der Bewertungen der Nachfolgezustände
 - MAX ist am Zug: Bewertung ist das **Maximum** der Bewertungen der Nachfolgezustände
- Im Wurzelknoten: MAX wählt den Zug, der zur maximalen Bewertung führt

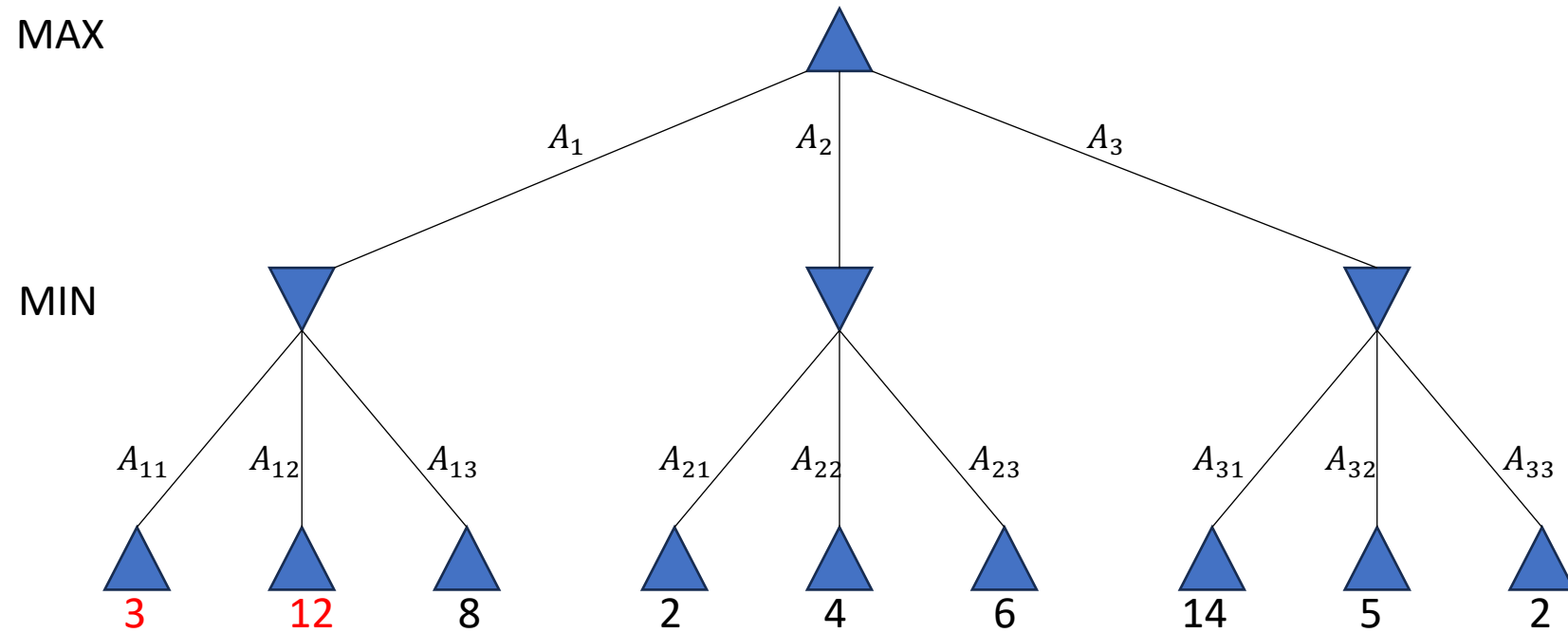
Beispiel: Welchen Zug sollte MAX wählen?



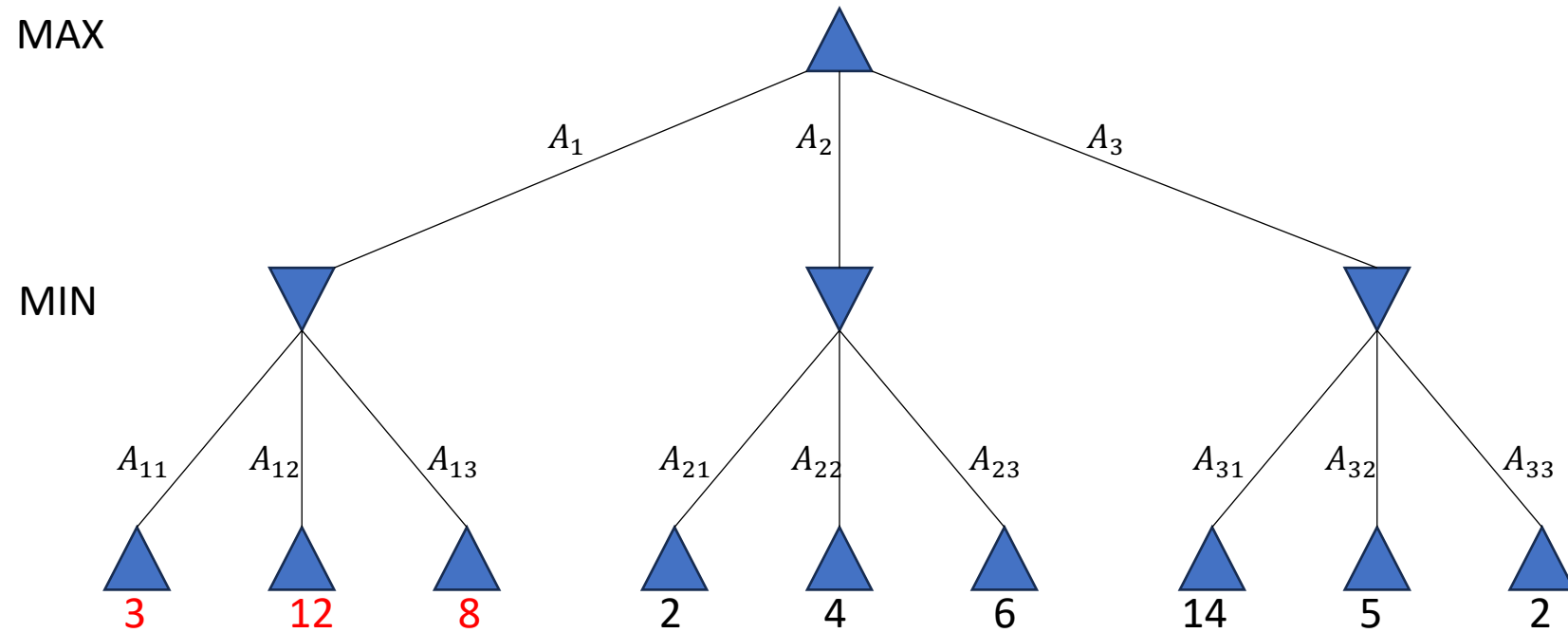
Beispiel



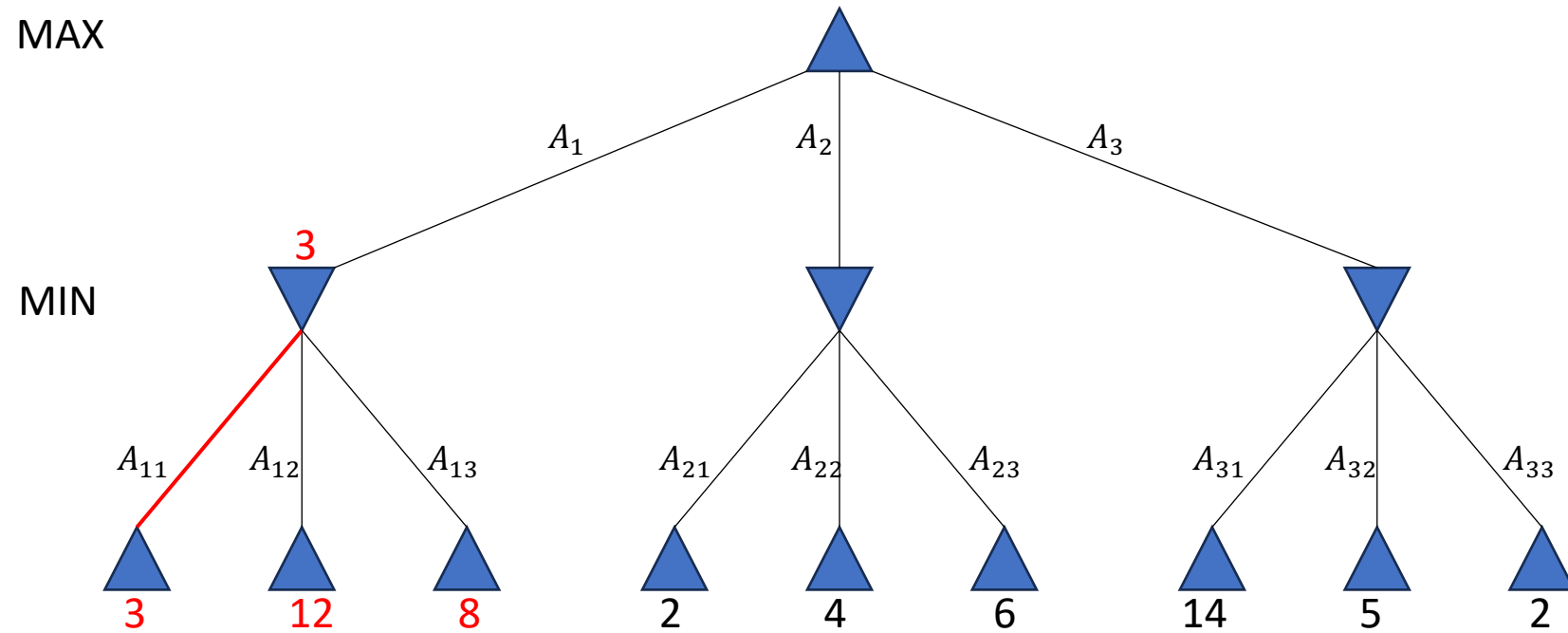
Beispiel



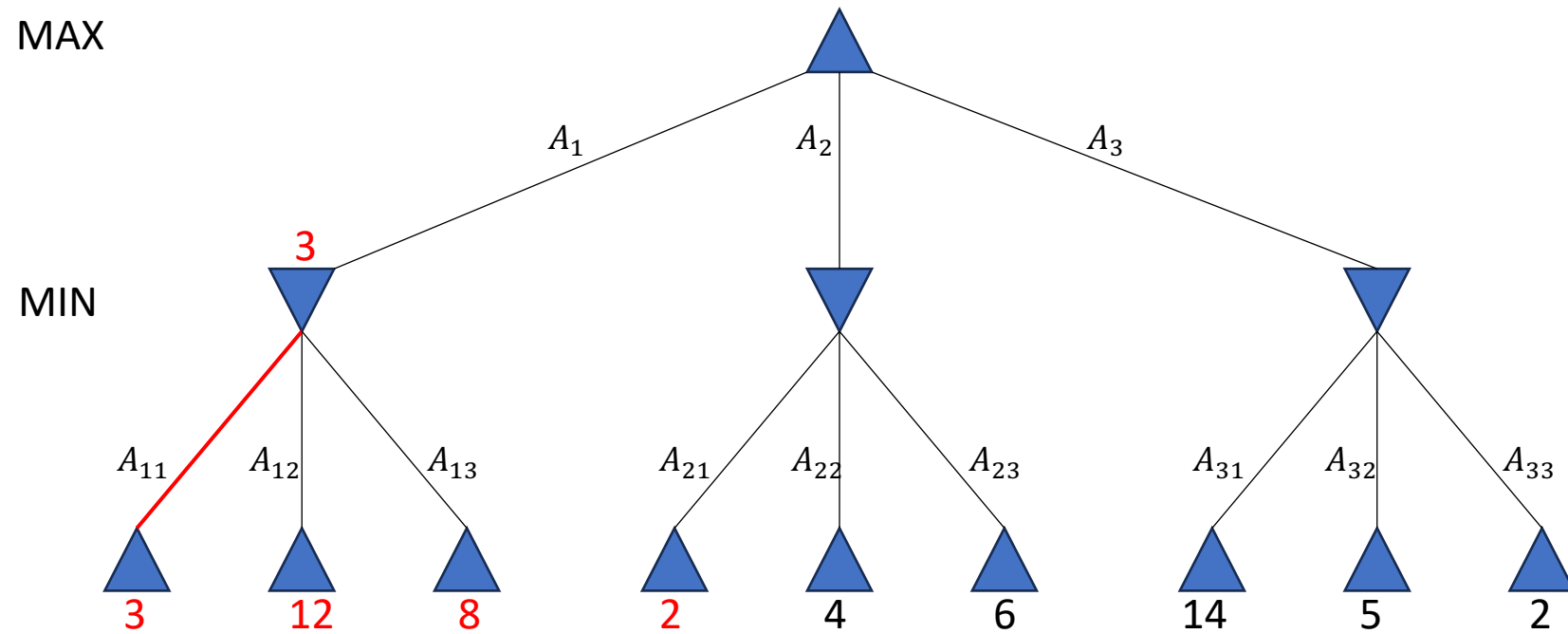
Beispiel



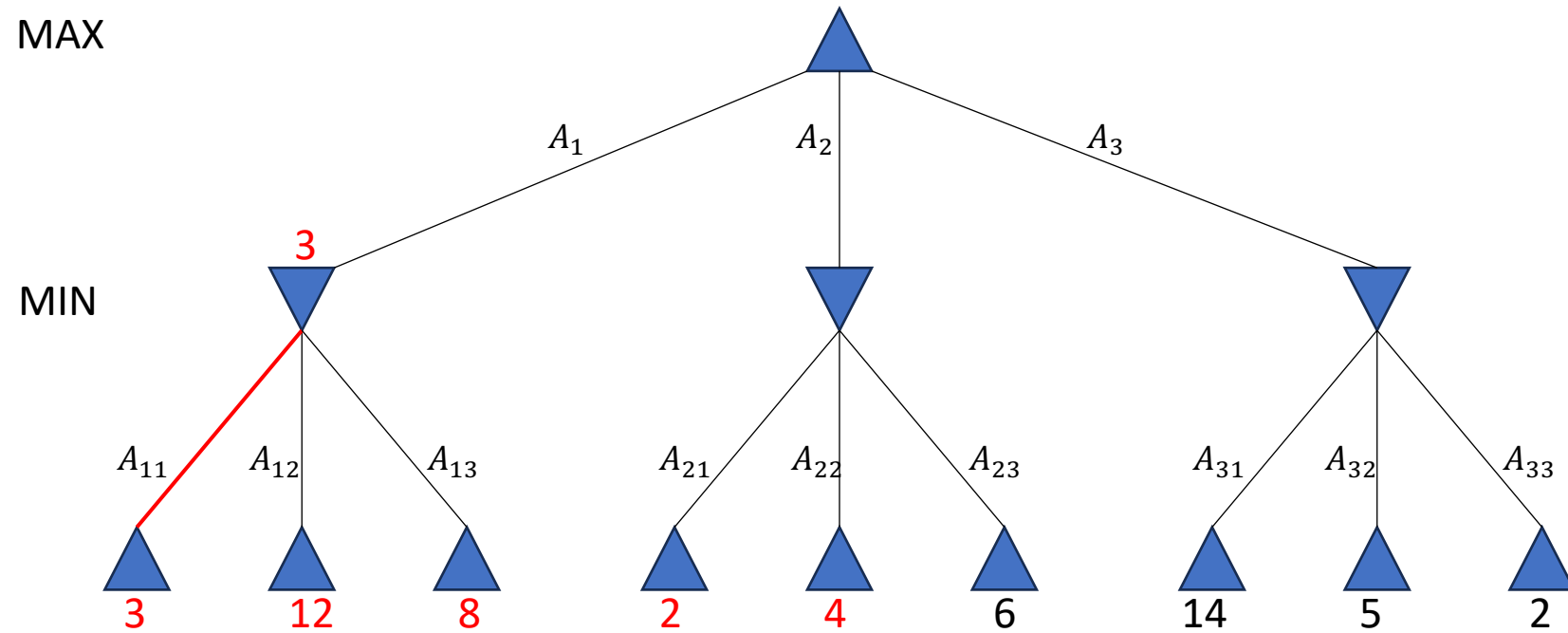
Beispiel



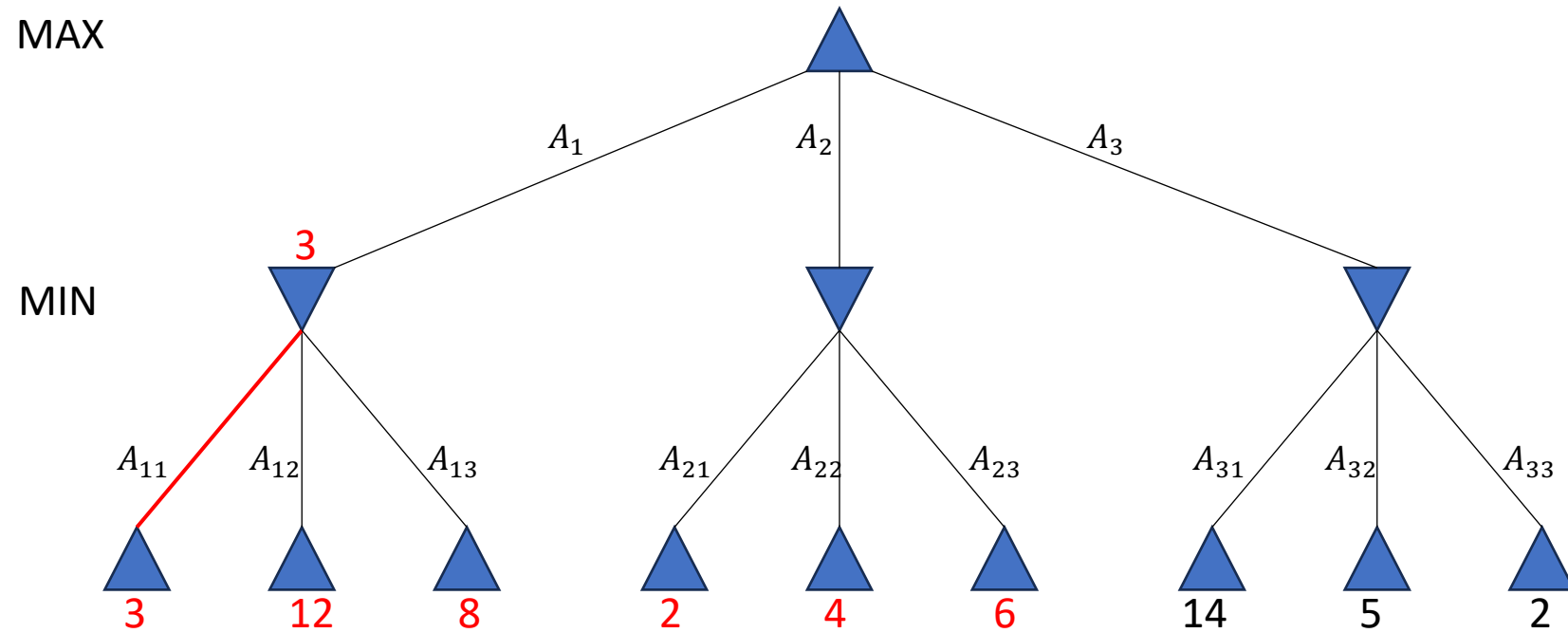
Beispiel



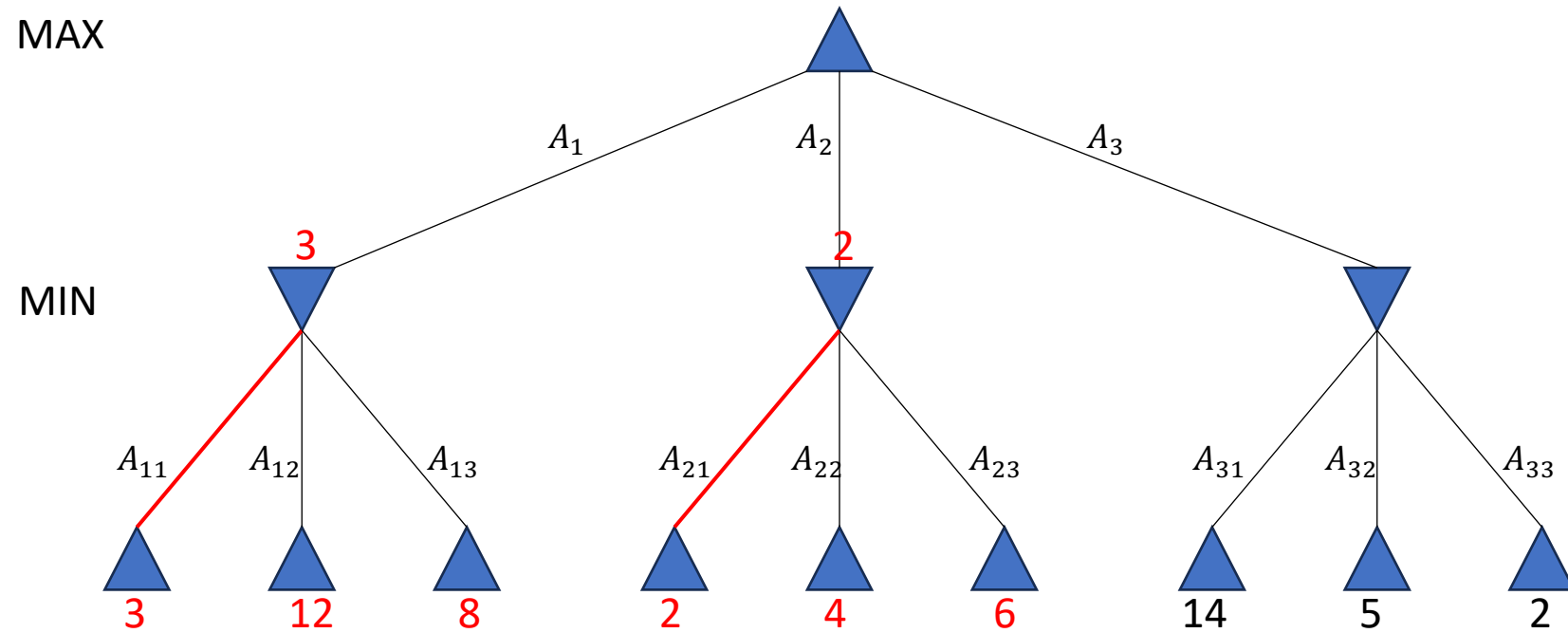
Beispiel



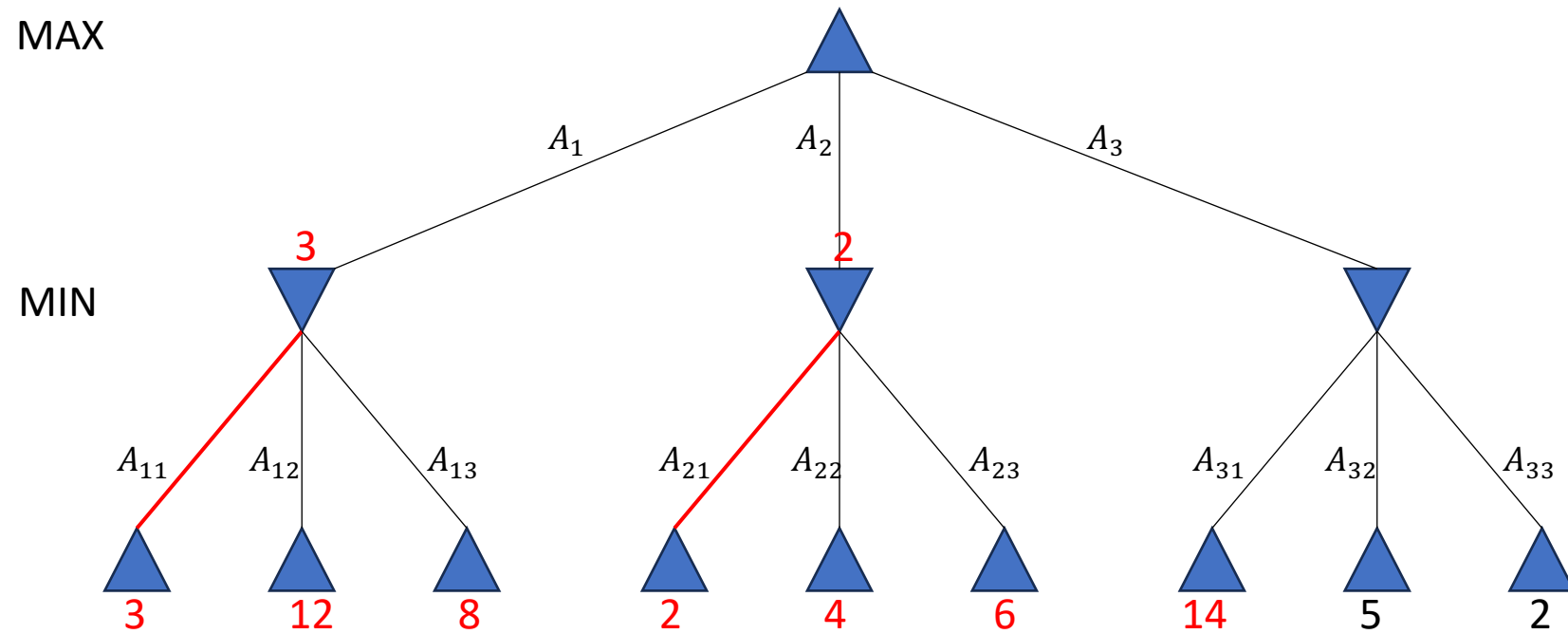
Beispiel



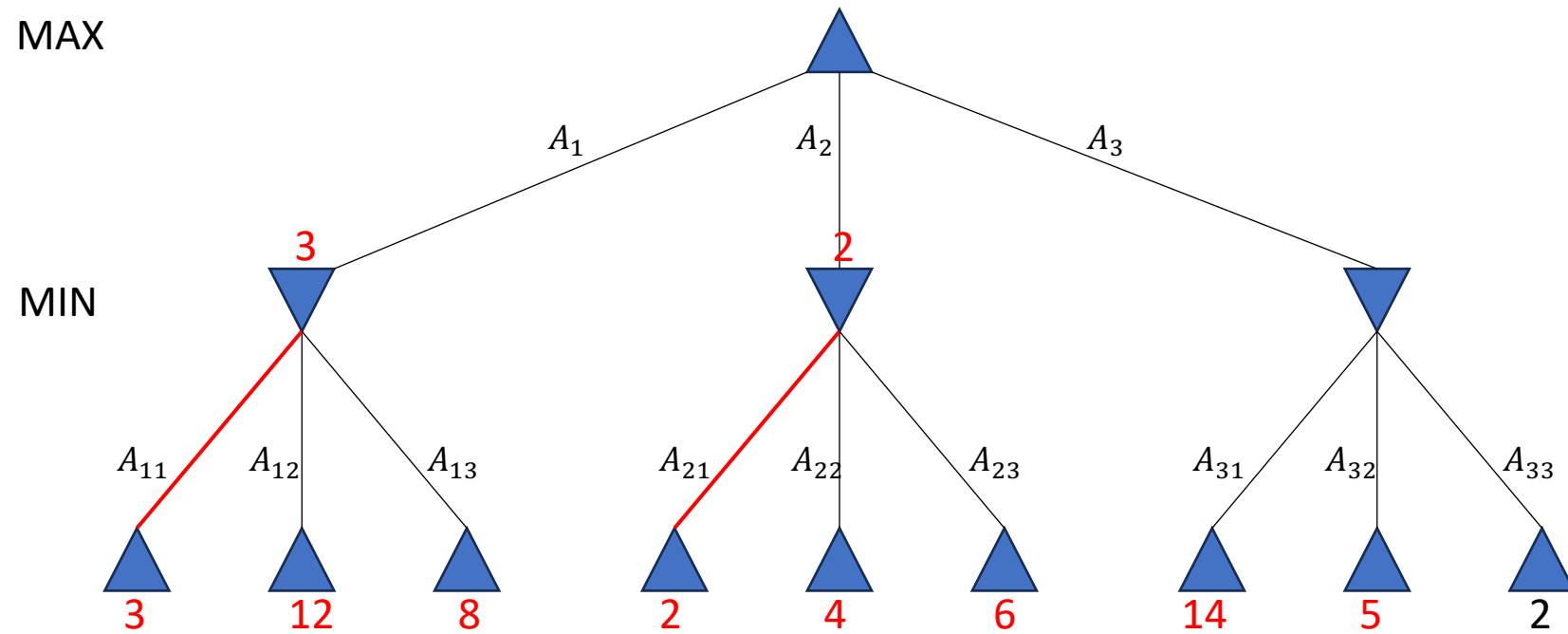
Beispiel



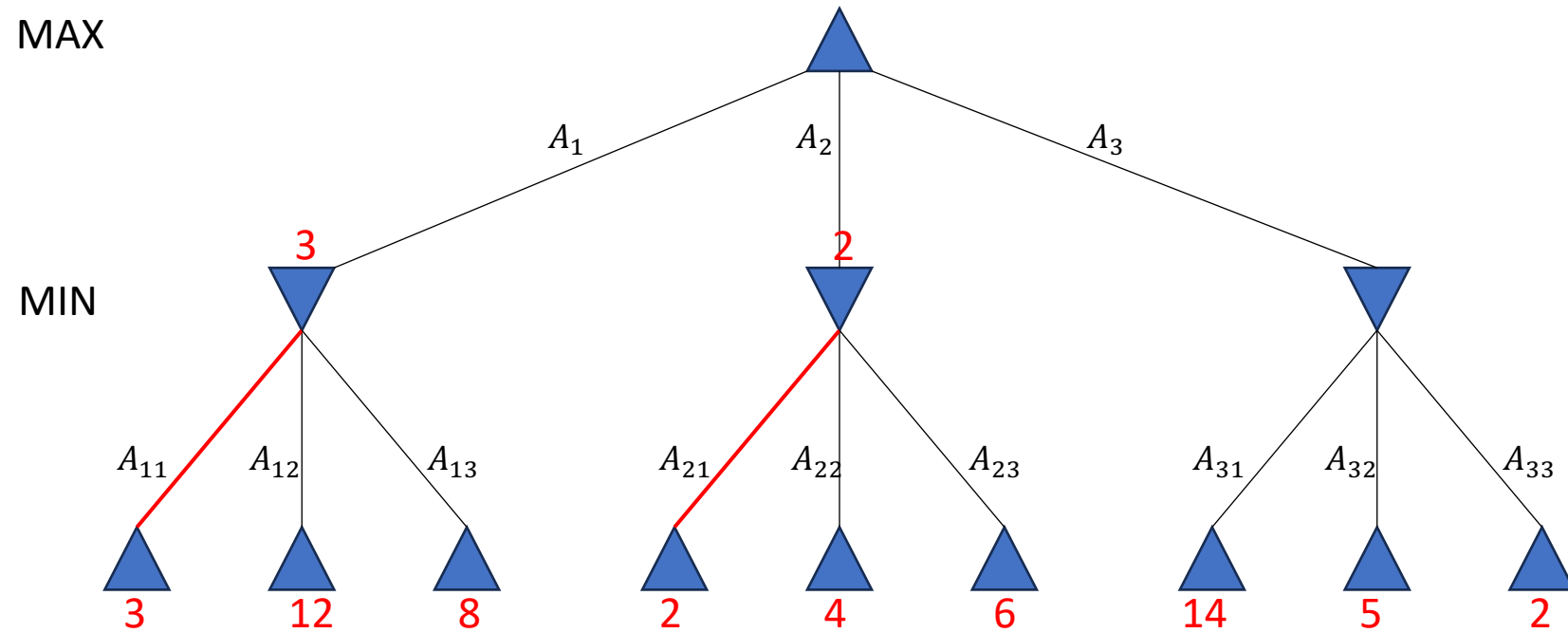
Beispiel



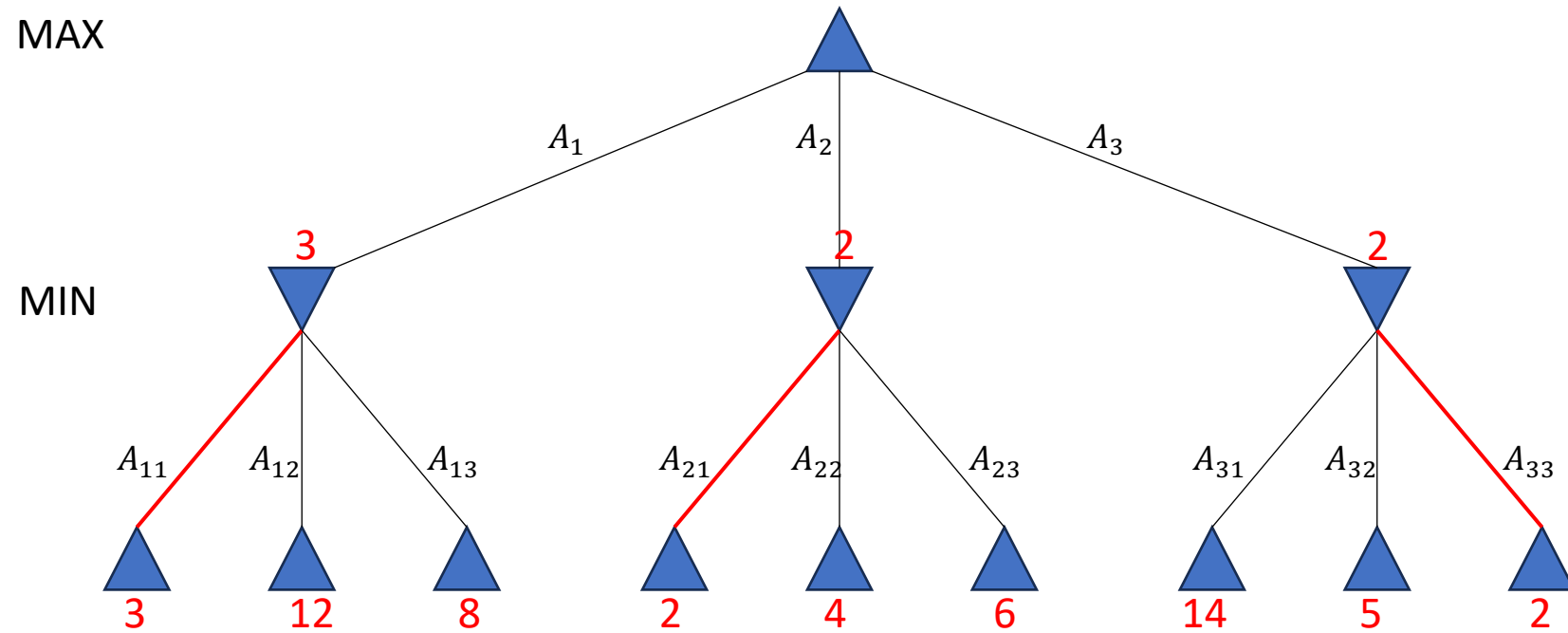
Beispiel



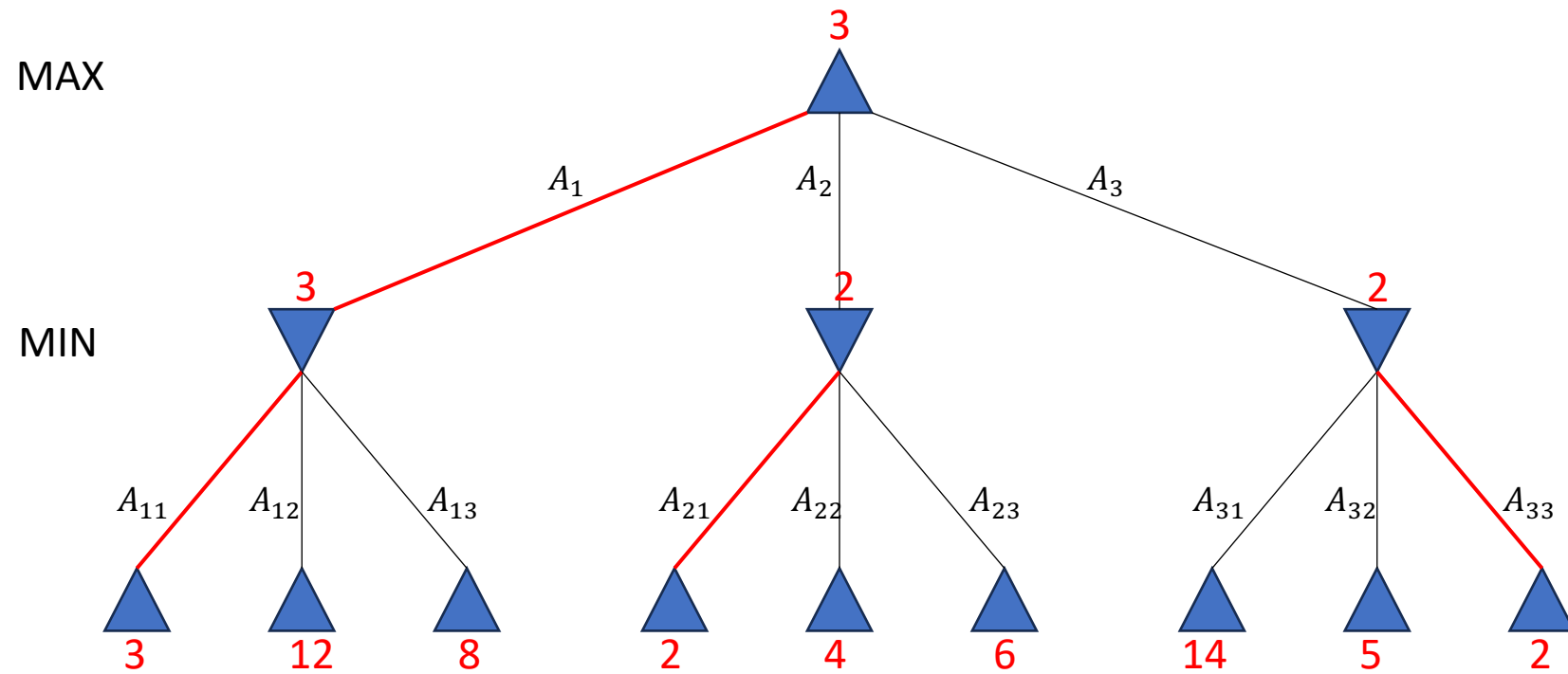
Beispiel



Beispiel



Beispiel



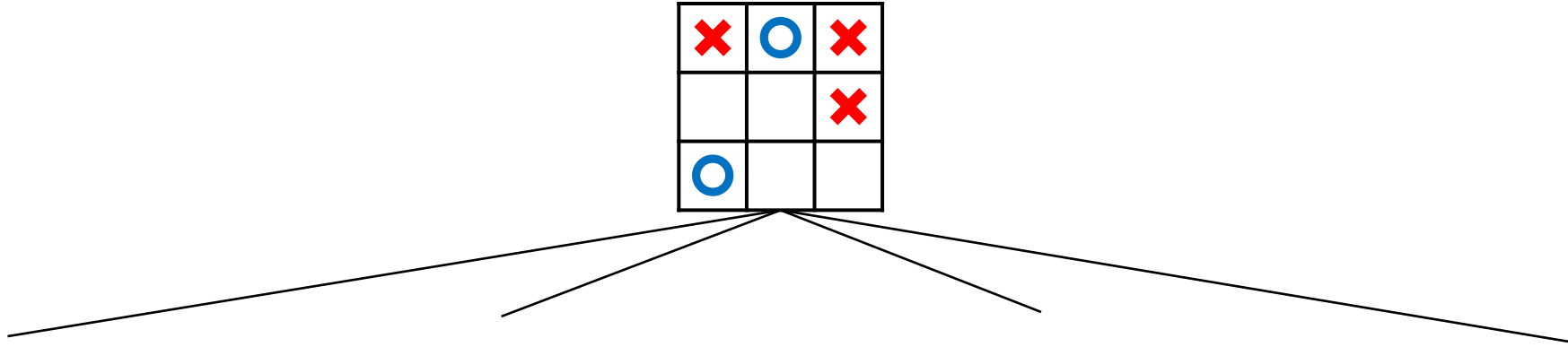
Minimax in Pseudocode

```
function minimax( $p$ )  
if  $p$  is terminal position:  
    return  $\langle u(p), \text{none} \rangle$   
 $best\_move := \text{none}$   
if  $player(p) = \text{MAX}$ :  
     $v := -\infty$   
else:  
     $v := \infty$   
for each  $\langle move, p' \rangle \in succ(p)$ :  
     $\langle v', best\_move' \rangle := minimax(p')$   
    if ( $player(p) = \text{MAX}$  and  $v' > v$ ) or  
        ( $player(p) = \text{MIN}$  and  $v' < v$ ):  
         $v := v'$   
         $best\_move := move$   
return  $\langle v, best\_move \rangle$ 
```

Minimax in Python

```
def minimax(spielfeld: Zustand, ist_maximierer: bool) -> tuple[Zug, int]:  
  
    if spielfeld.ist_endzustand():  
        # Basisfall: Spiel ist zu Ende, daher keine weiteren Züge möglich  
        return DUMMY_ZUG, spielfeld.bewertung()  
  
    bester_zug = DUMMY_ZUG # Initialisierung mit Dummy-Zug und Dummy-Wert  
    bester_wert = -UNENDLICH if ist_maximierer else UNENDLICH  
  
    for zug in spielfeld.moegliche_zuege():  
        neues_spielfeld = spielfeld.führe_zug_aus(zug)  
        _, wert = minimax(neues_spielfeld, ist_maximierer=not ist_maximierer)  
        if (ist_maximierer == True and wert > bester_wert) or (  
            ist_maximierer == False and wert < bester_wert):  
            bester_zug = zug # besserer Zug gefunden  
            bester_wert = wert  
  
    return bester_zug, bester_wert
```

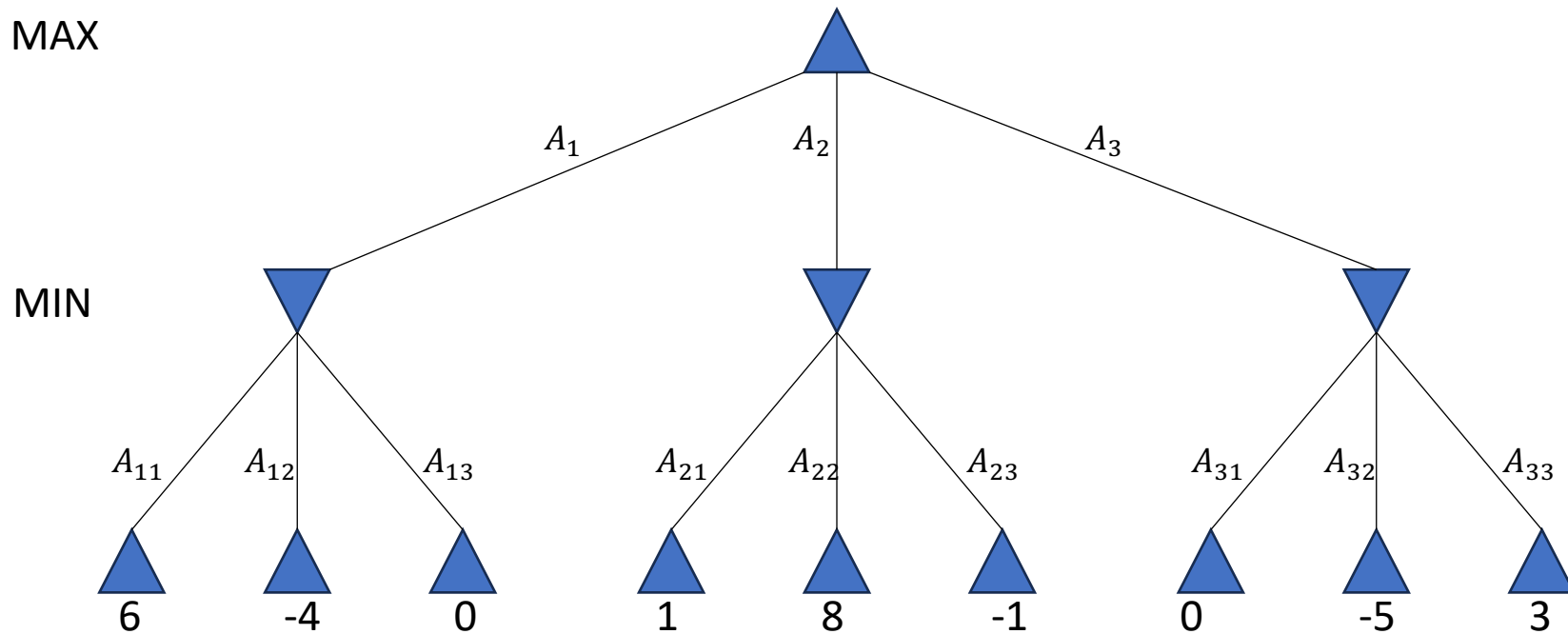

O ist am Zug. Vervollständige den Spielbaum, bewerte den aktuellen Zustand mit dem Minimax-Algorithmus.



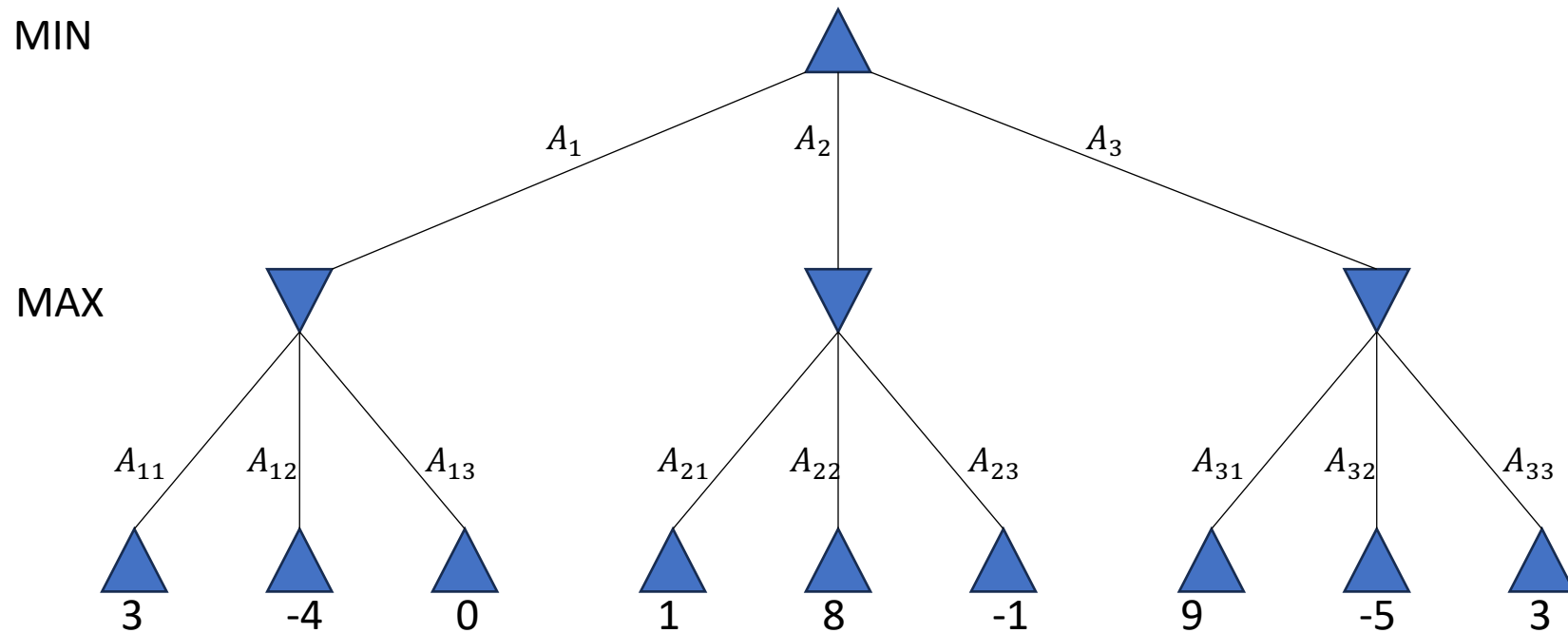
Was tun, wenn der Spielbaum zu groß ist?

- Minimax besucht *alle* Blätter des Spielbaums → exponentiell viele Spielzustände
- Idee: Echte Bewertungsfunktion b approximieren durch Evaluationsfunktion e , die die „Güte“ eines Zustands abschätzt, auch wenn dieser noch kein Endzustand ist
- Beispiel Schach:
 - Figuren haben einen Wert (Dame = 9, Turm = 5, ..., Bauer = 1)
 - Zustandsevaluation: Summe Figuren weiß – Summe Figuren schwarz

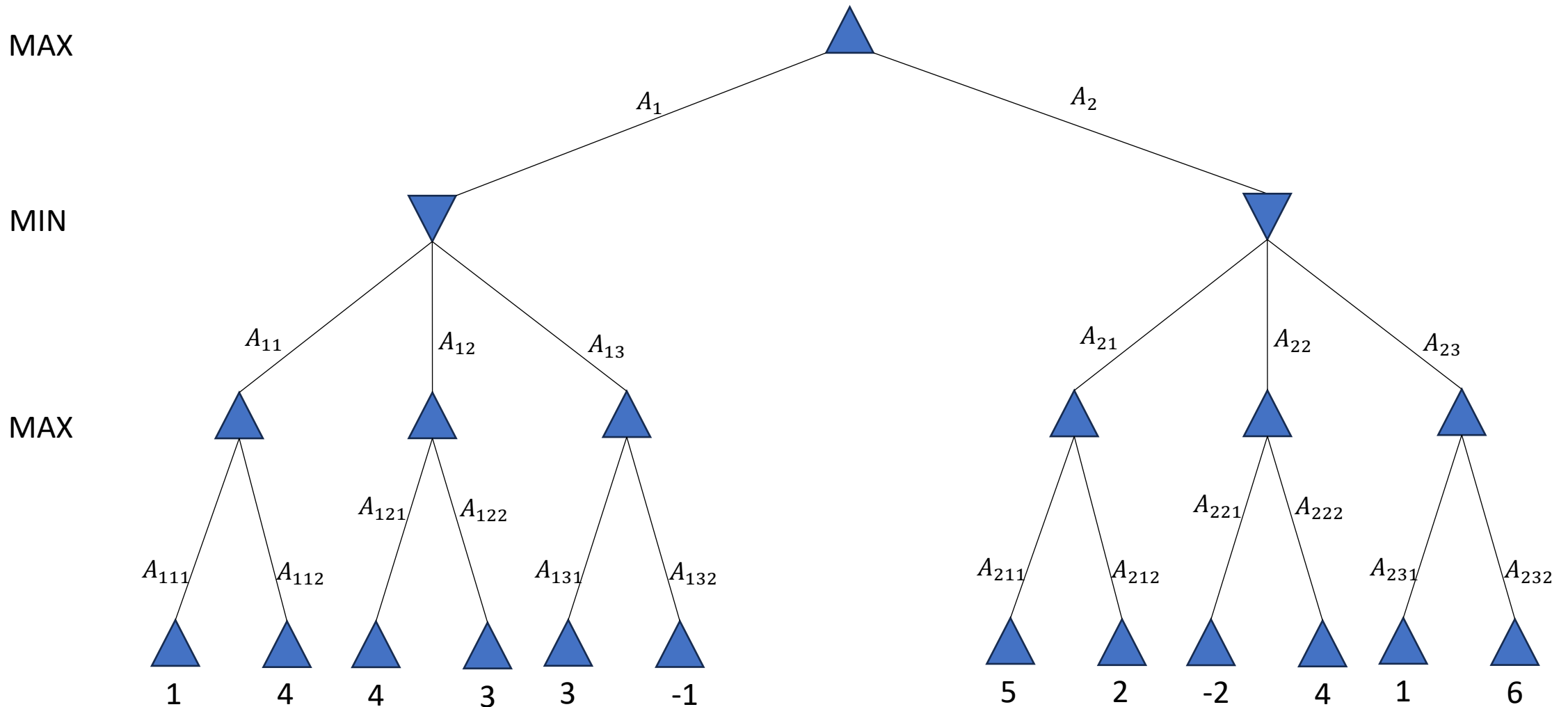
Aufgabe: Welchen Zug sollte MAX wählen und wie viele Punkte kann er erreichen?



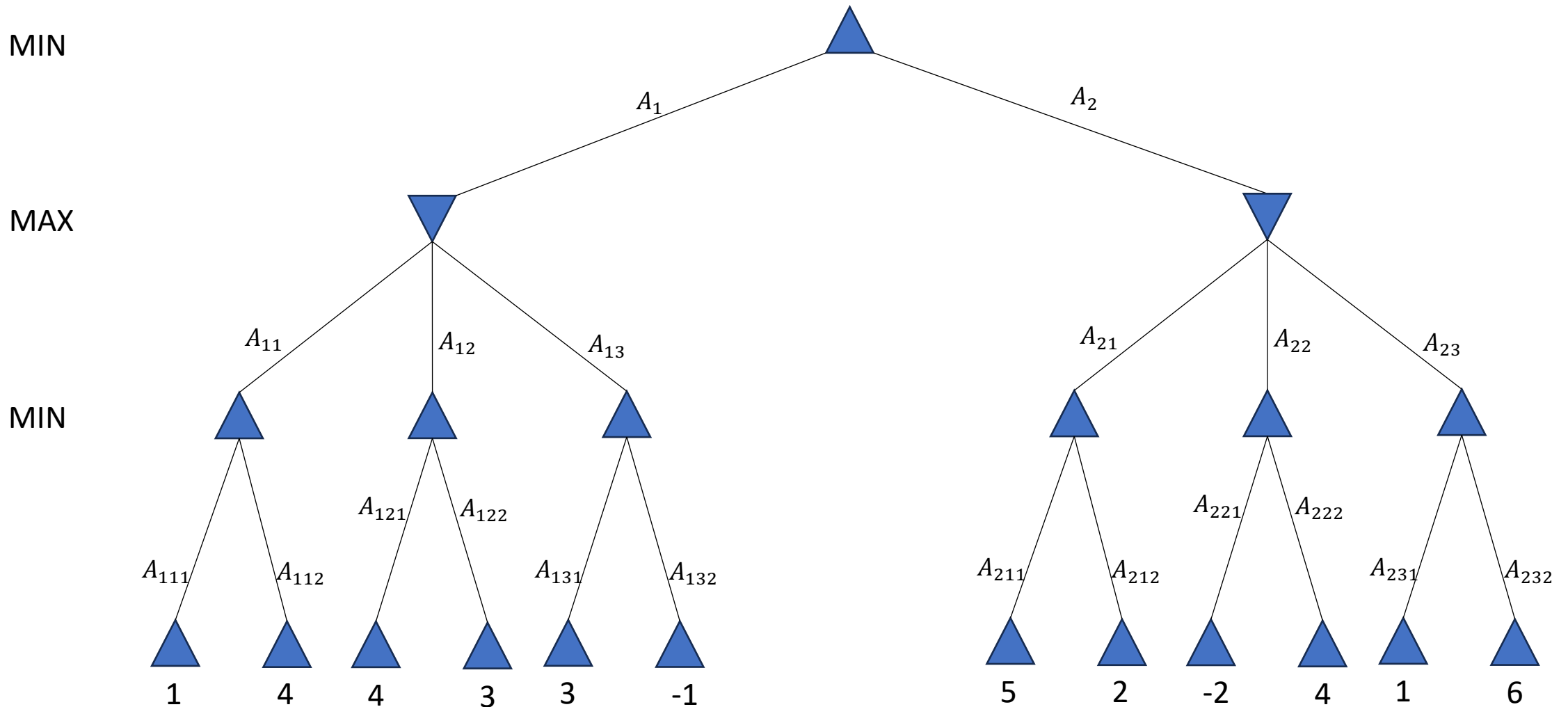
Aufgabe: Welchen Zug sollte MIN wählen und wie viele Punkte kann er erreichen?



Aufgabe: Welchen Zug sollte MAX wählen und wie viele Punkte kann er erreichen?



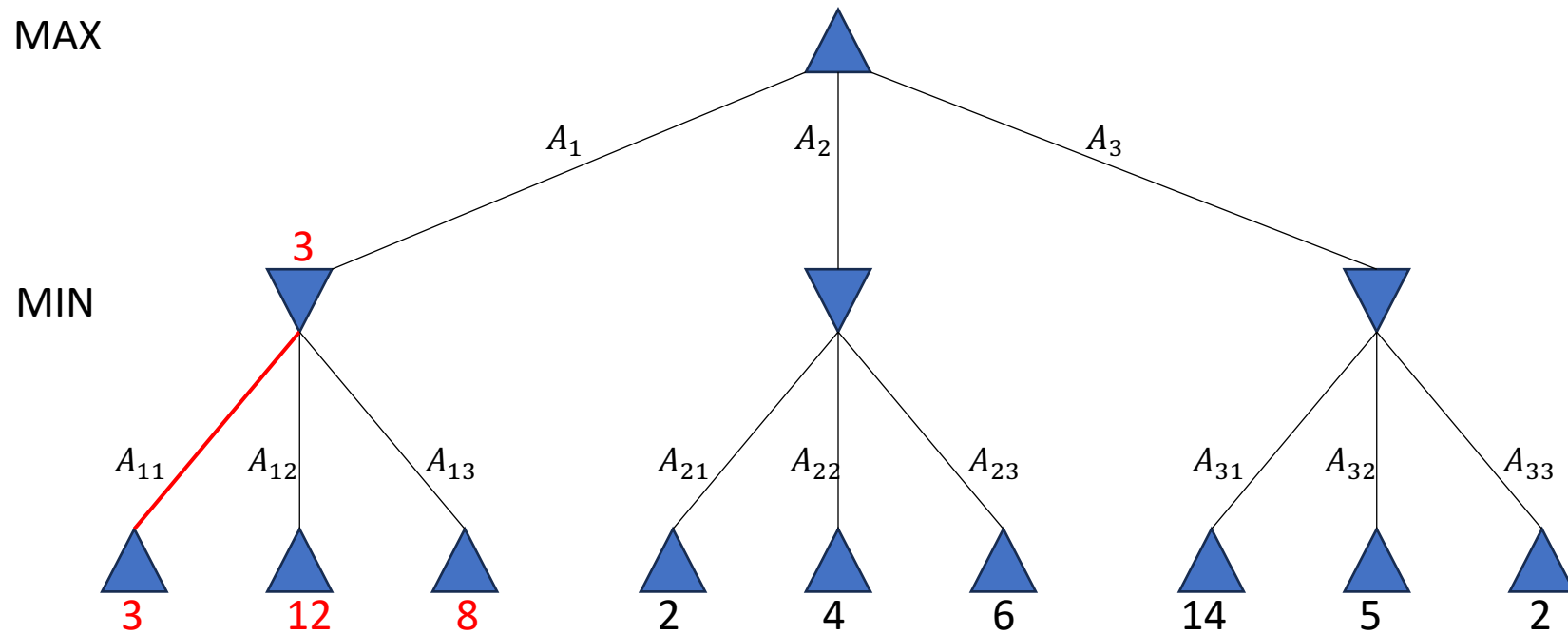
Aufgabe: Welchen Zug sollte MIN wählen und wie viele Punkte kann er erreichen?



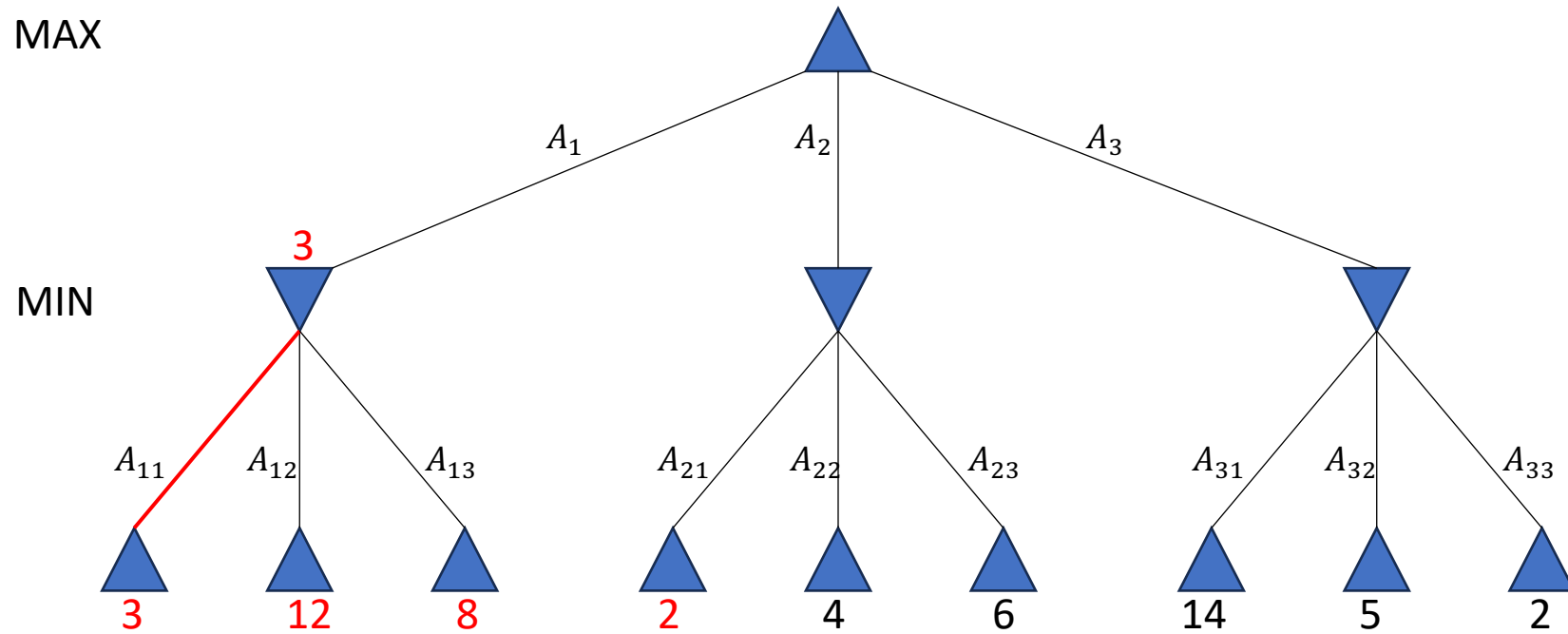
Verbesserung durch Alpha-Beta-Pruning

- Minimax besucht *alle* Blätter des Spielbaums
→ exponentiell viele Spielzustände
- Wenn wir eine Bewertungsfunktion nutzen wird das Ergebnis ungenauer → besser erst möglich tief im Suchbaum bewerten
- also will man trotzdem möglichst große Teile des exponentiell wachsenden Baums untersuchen
- Erkenntnis: Wenn ein (rationaler) Spieler bestimmte Züge nie machen wird, weil sie für ihn garantiert zu keiner besseren Bewertung führen
→ dann können wir diesen Teil des Spielbaums einfach ignorieren
- Alpha-Beta-Pruning

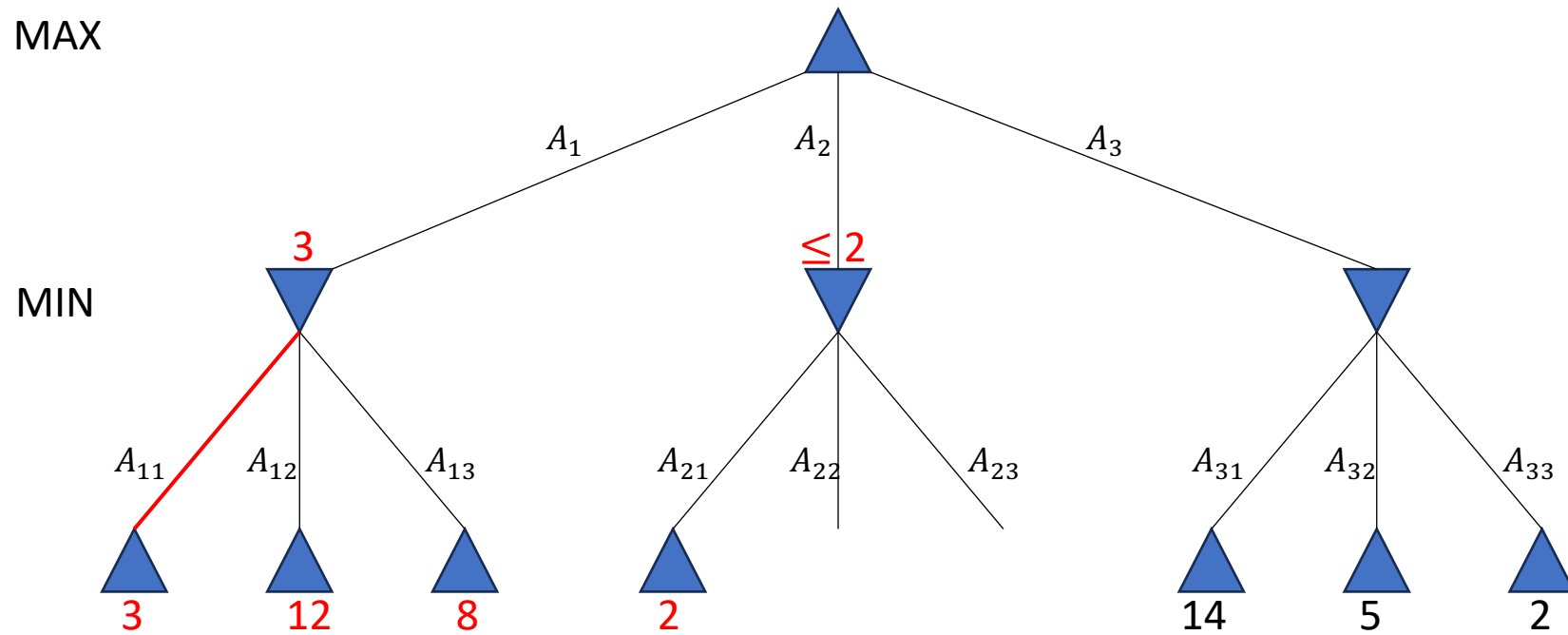
Alpha-Beta-Pruning



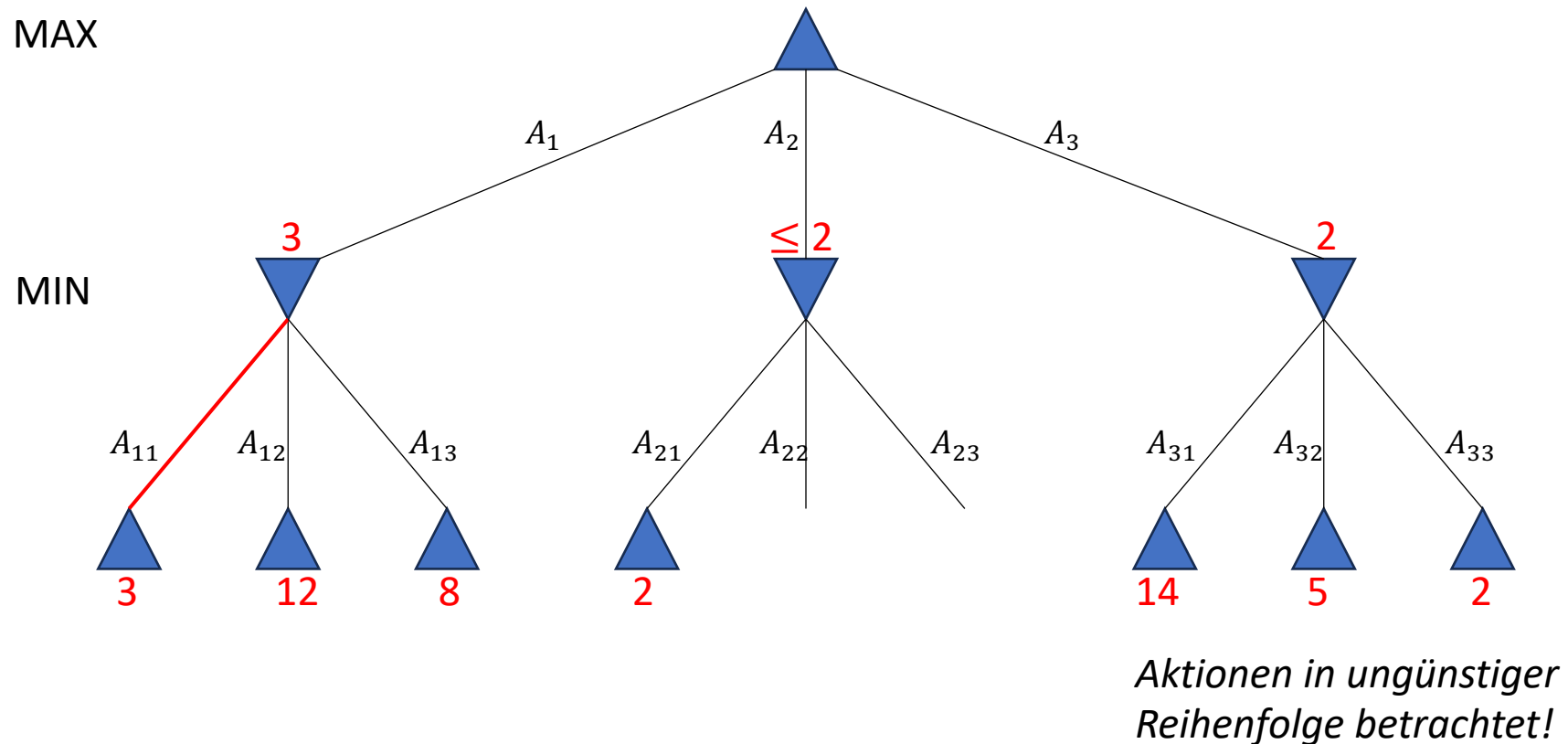
Alpha-Beta-Pruning



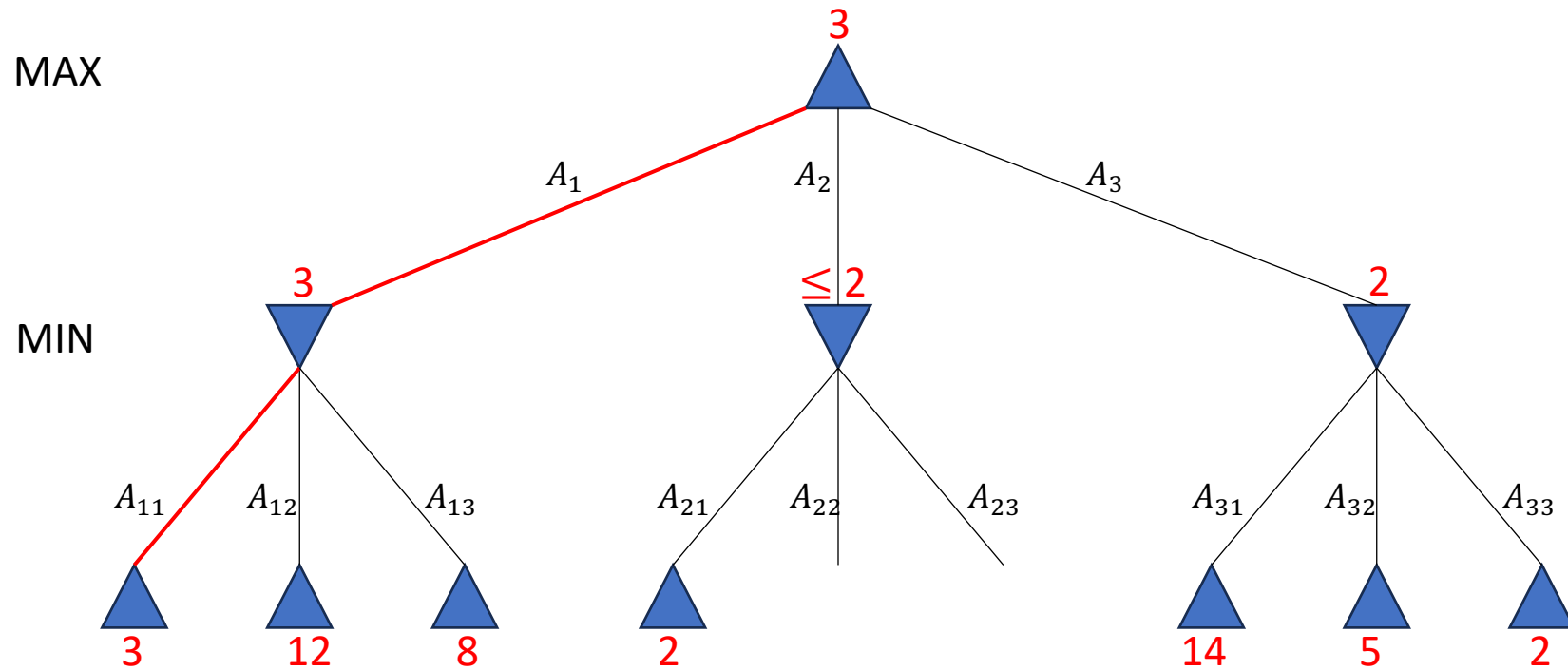
Alpha-Beta-Pruning



Alpha-Beta-Pruning



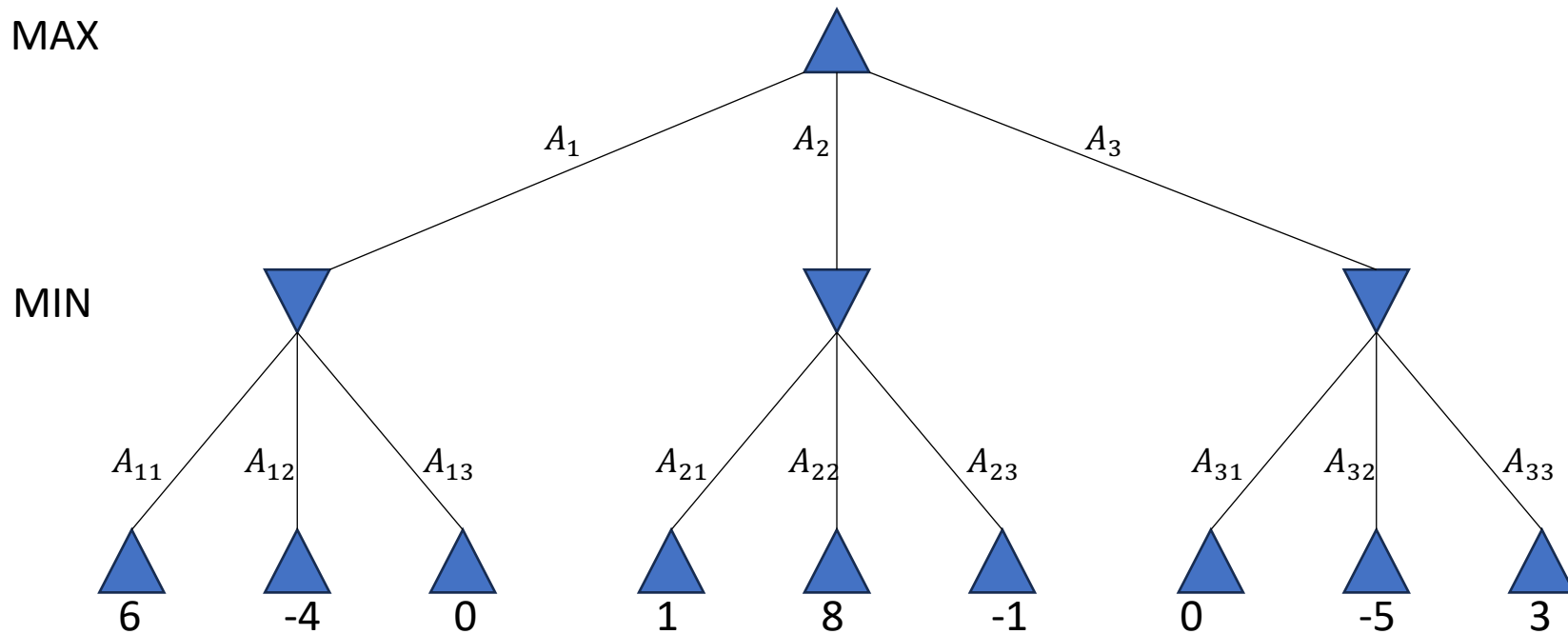
Alpha-Beta-Pruning



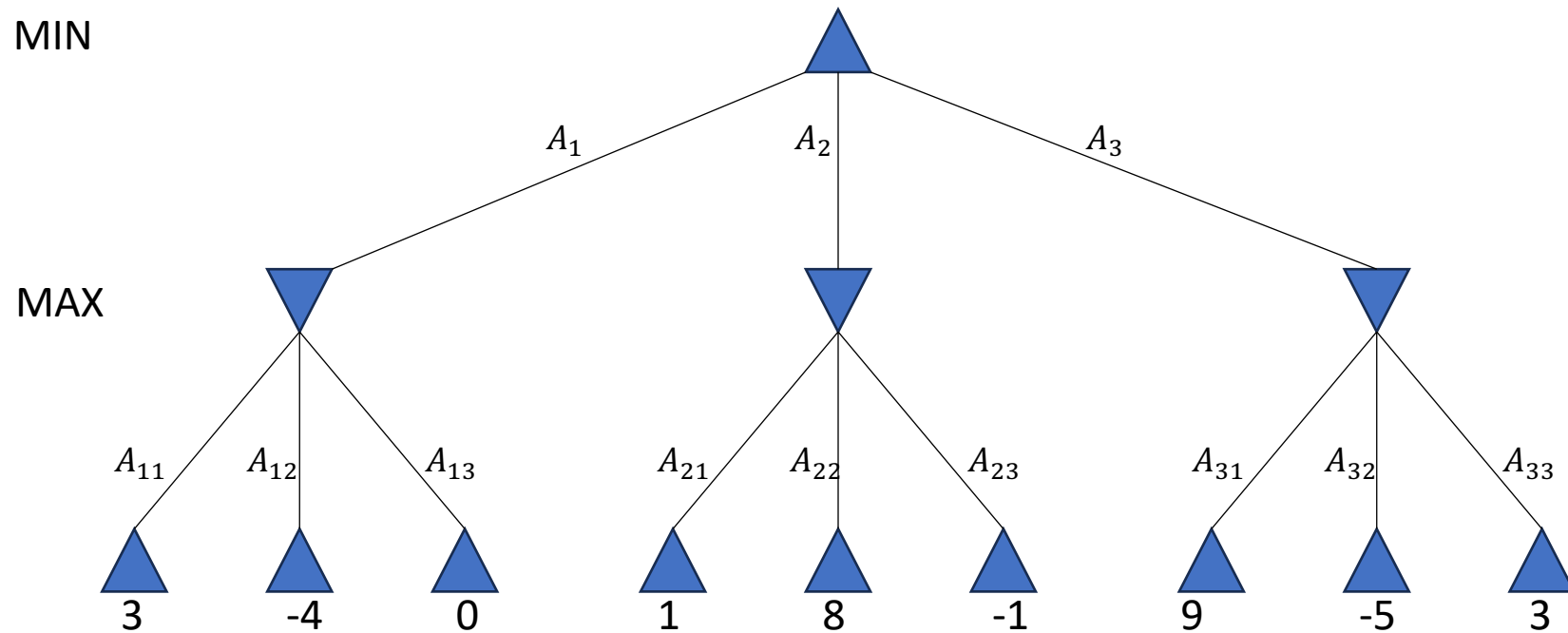
Wir betrachten jetzt die Aufgaben von vorhin noch einmal.

An welchen Stellen könnte die Suche durch Alpha-Beta-Pruning vorzeitig beendet werden?

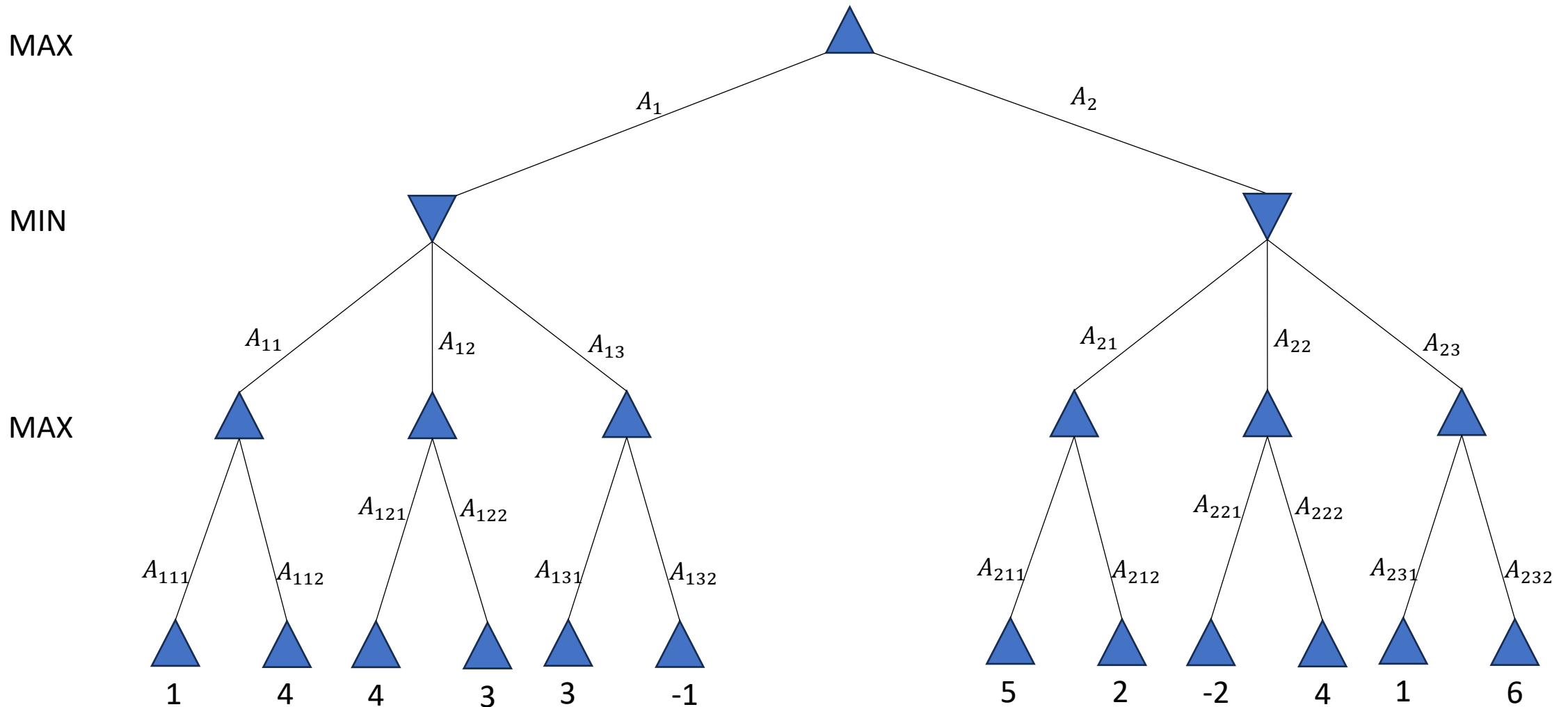
Aufgabe: Welchen Zug sollte MAX wählen und wie viele Punkte kann er erreichen?



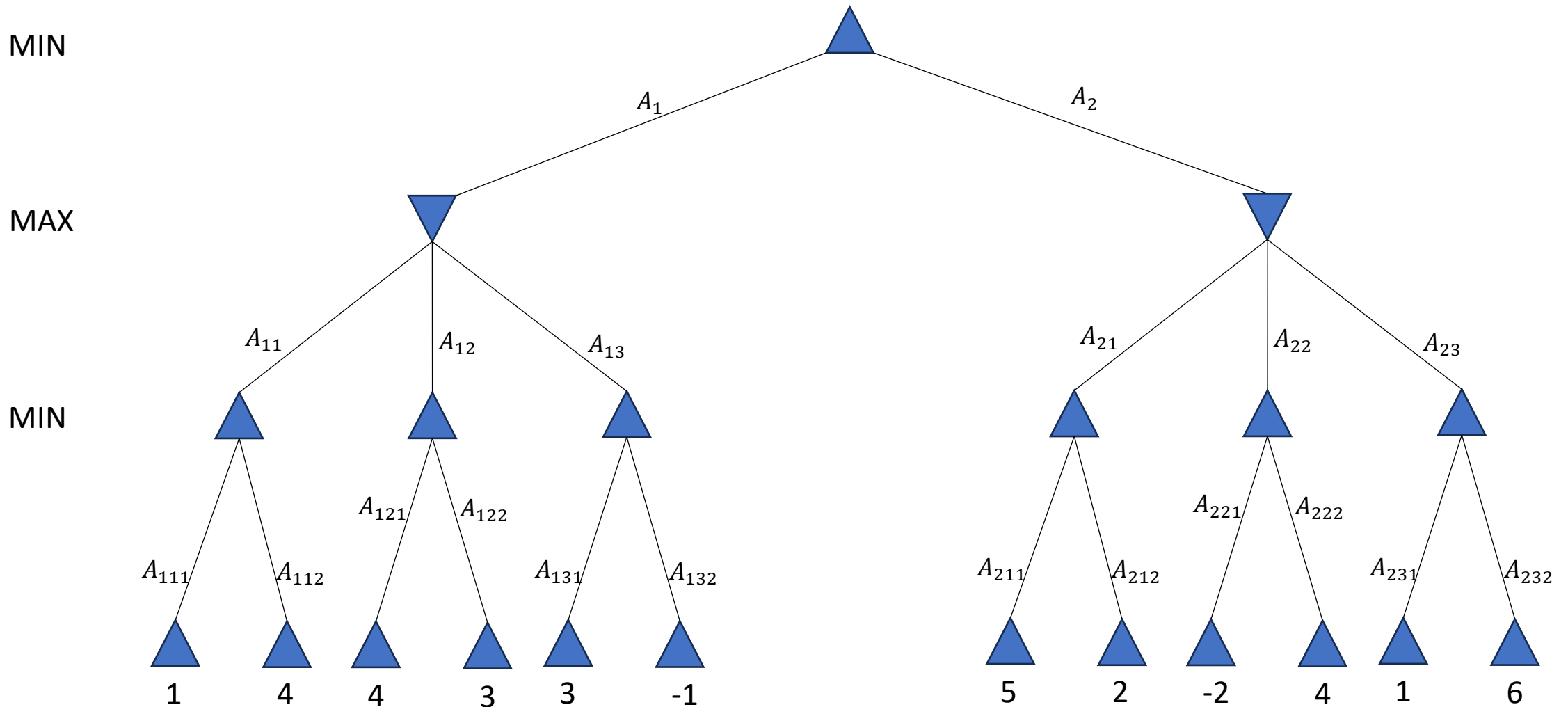
Aufgabe: Welchen Zug sollte MIN wählen und wie viele Punkte kann er erreichen?



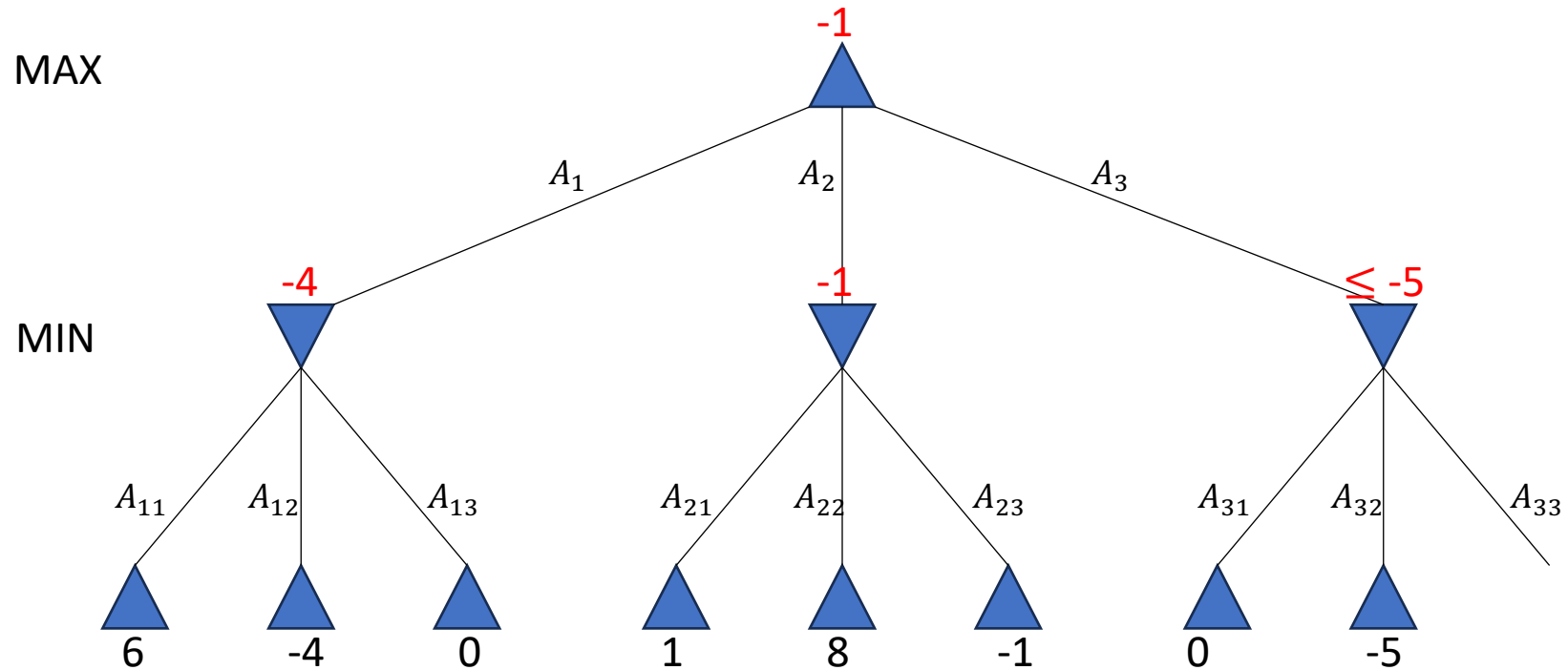
Aufgabe: Welchen Zug sollte MAX wählen und wie viele Punkte kann er erreichen?



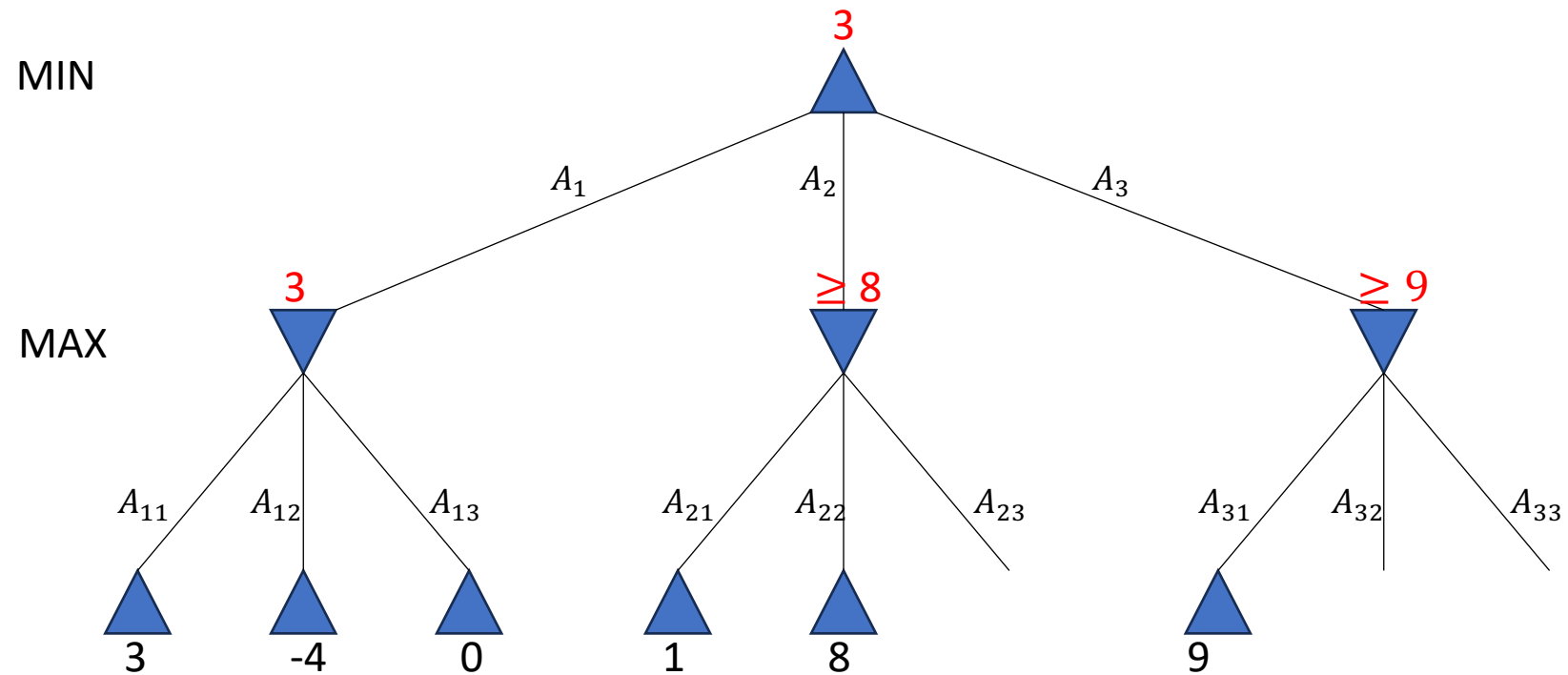
Aufgabe: Welchen Zug sollte MIN wählen und wie viele Punkte kann er erreichen?



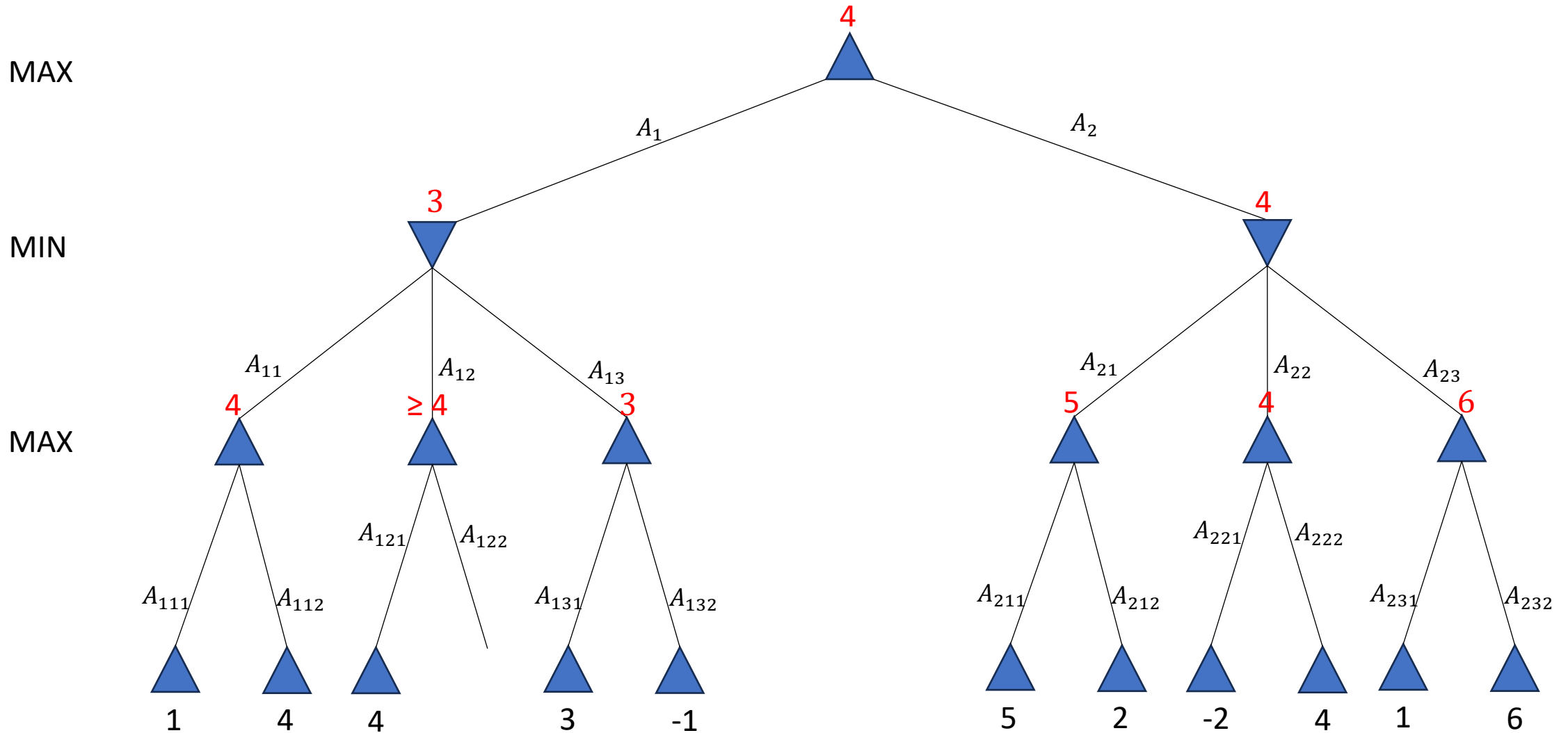
Lösung mit Alpha-Beta-Pruning



Lösung mit Alpha-Beta-Pruning



Lösung mit Alpha-Beta-Pruning



Lösung mit Alpha-Beta-Pruning

