CSE 150B

Minimax and 2048

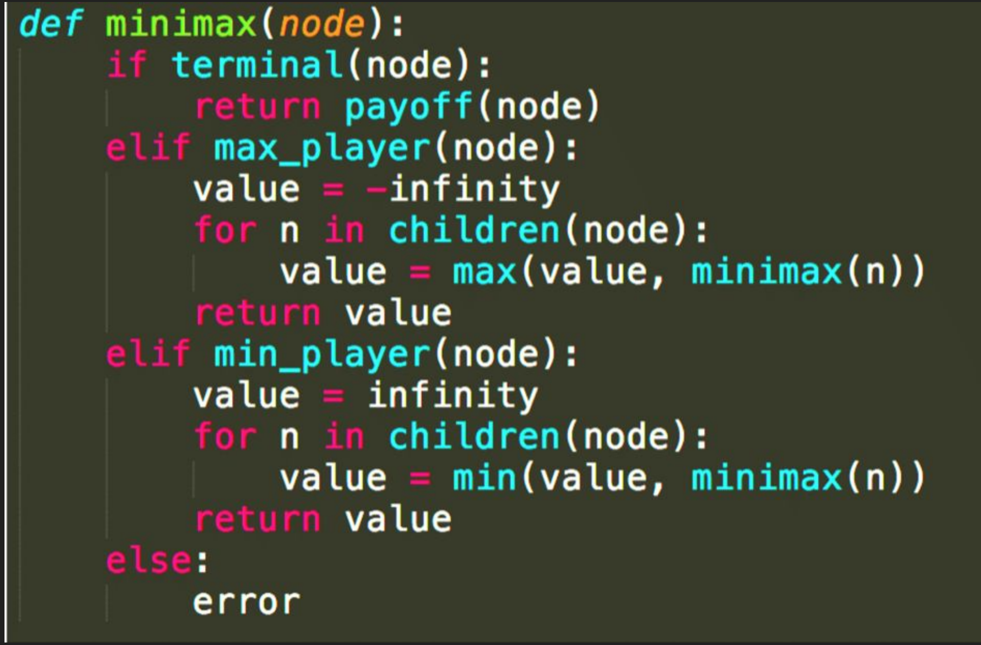
Aravind Mahadevan

Agenda

- Minimax tree example
    - Regular minimax example
    - Alpha-Beta Pruning example
- 2048 workflow

Minimax

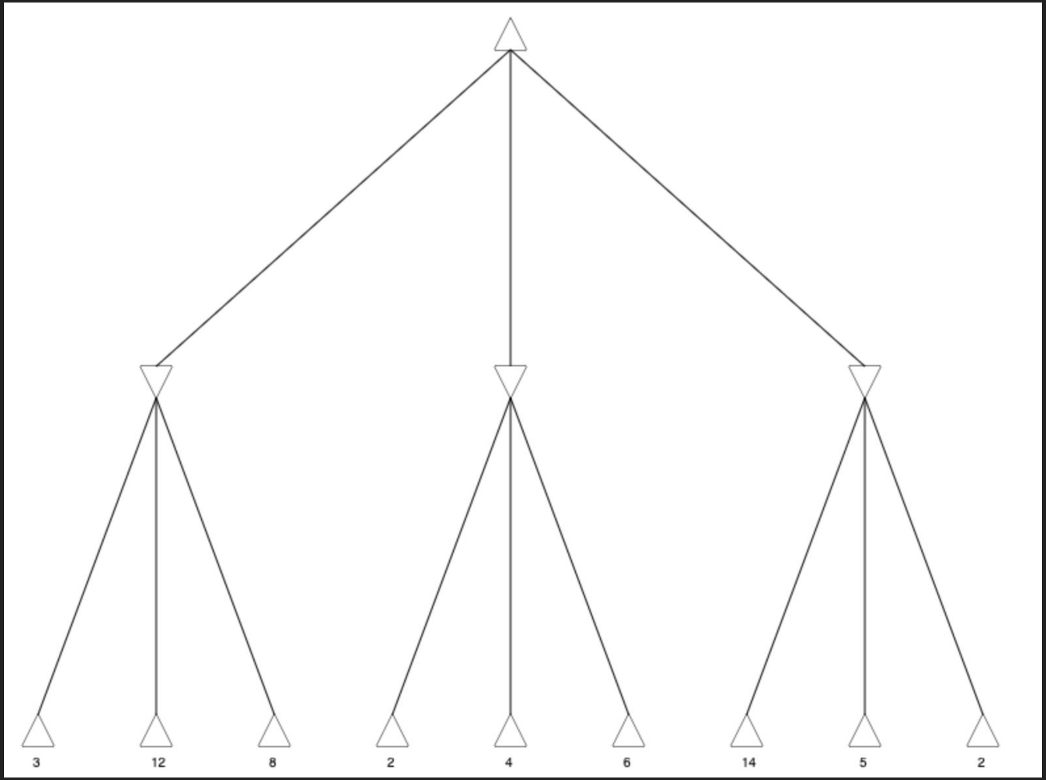
- Modeling a 2 player zero sum game with perfect information
- Create a tree of possibilities that terminates after a finite sequence of actions.
- Assume both player are rational



```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

Example of minimax

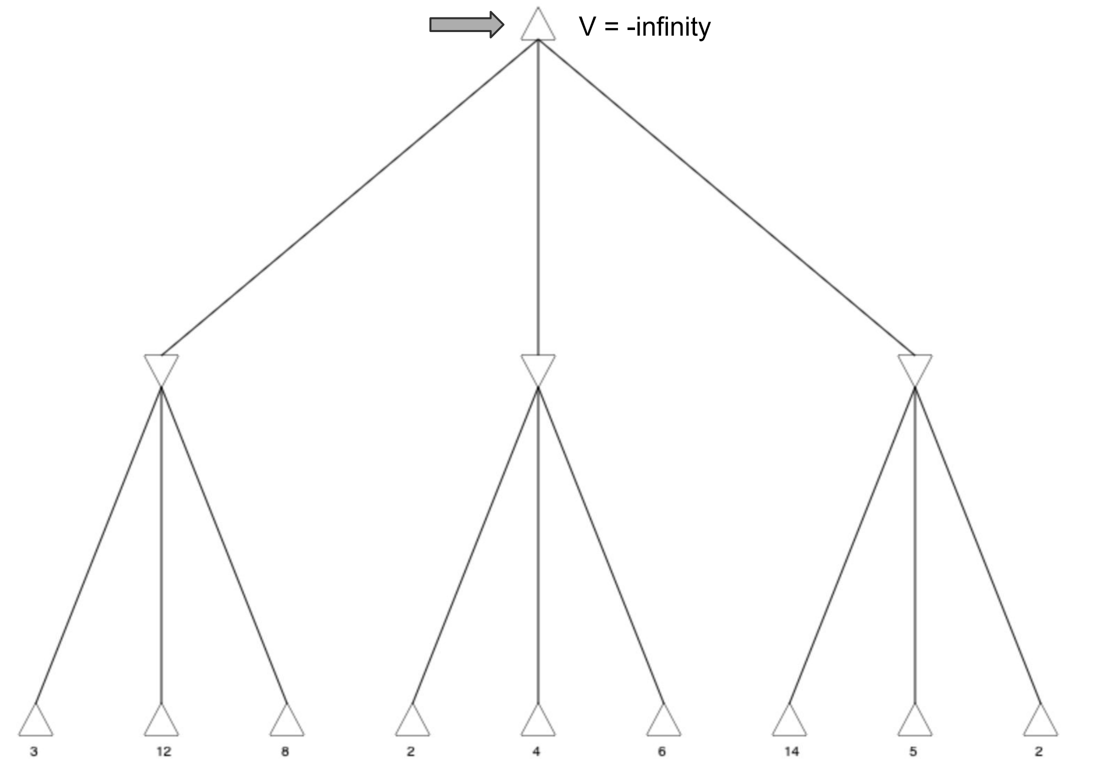
- Will run minimax algorithm on the tree shown below



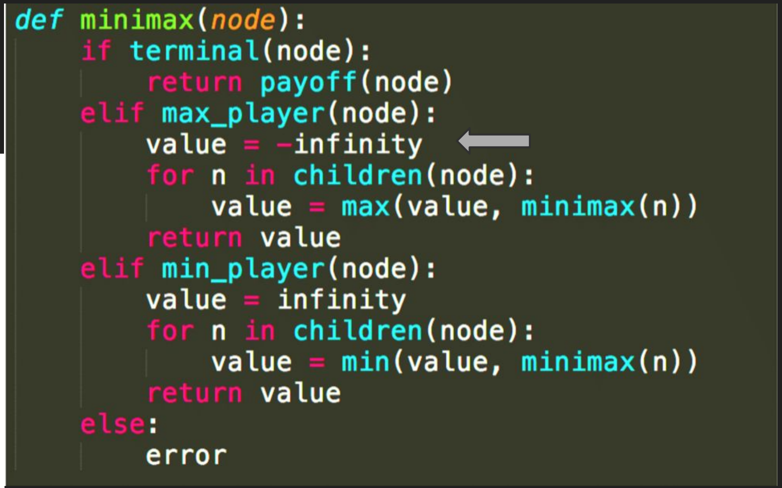
△ Max player

▽ Min player

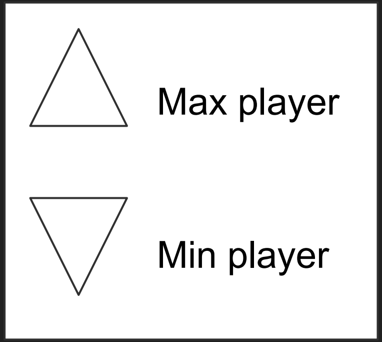
Example of minimax



V = -infinity



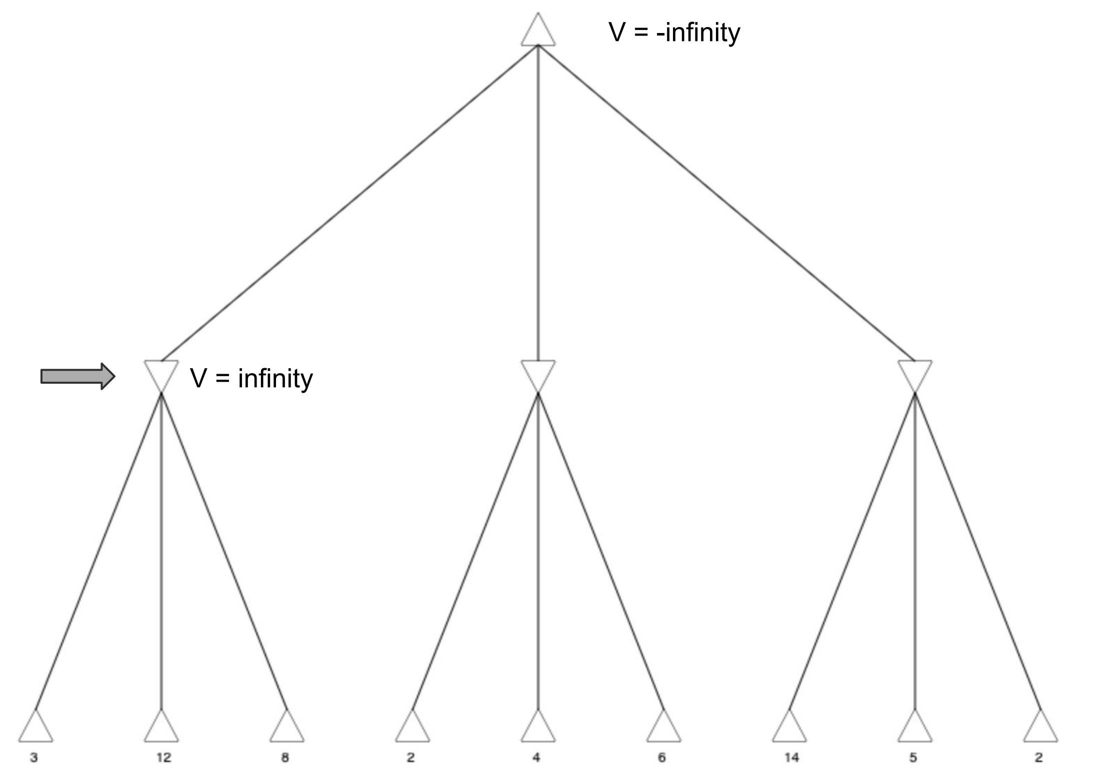
```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

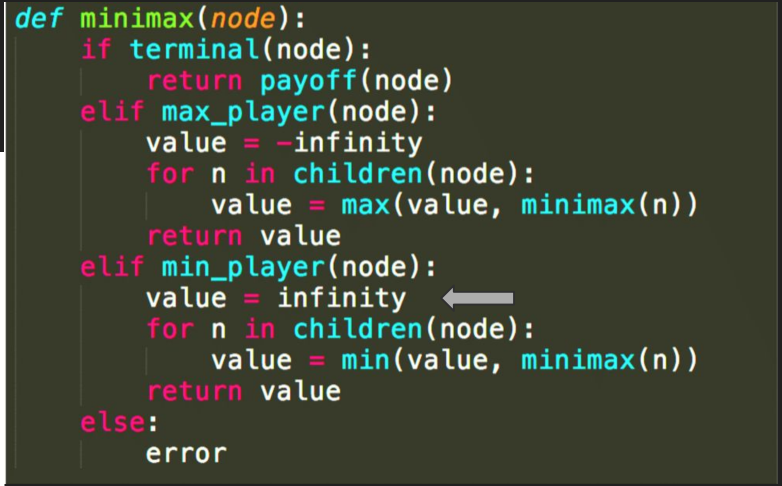


Max player

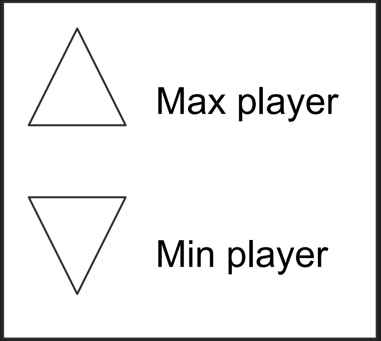
Min player

Example of minimax

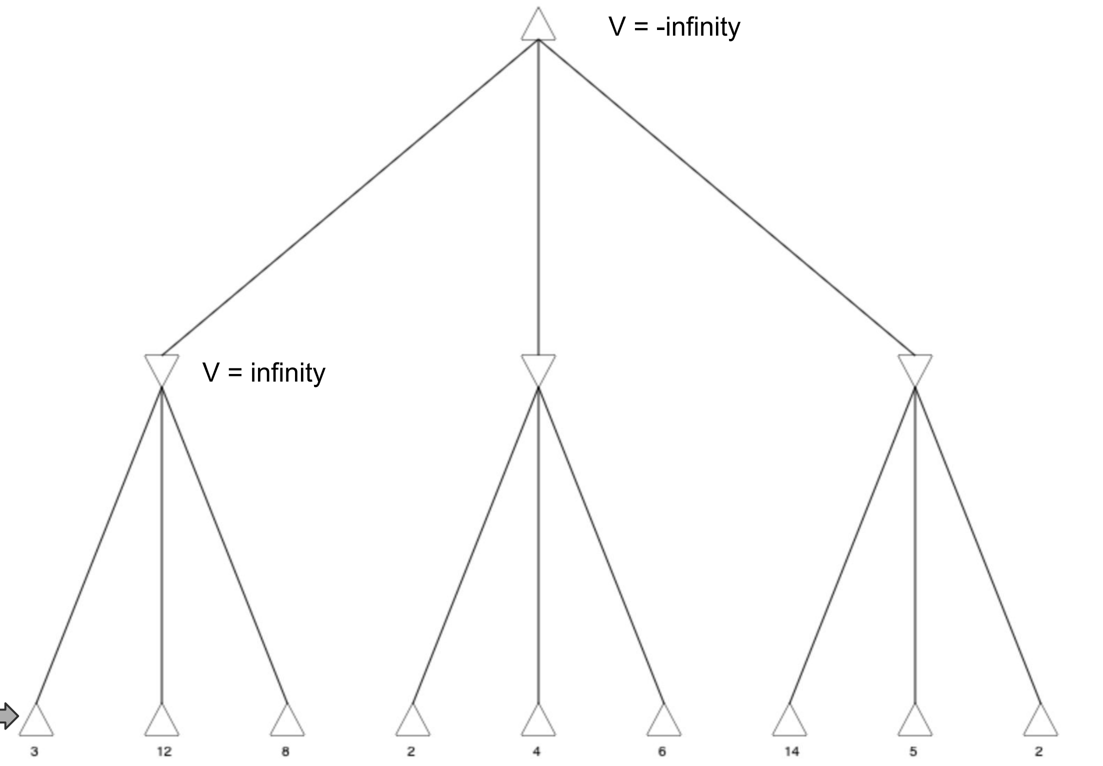


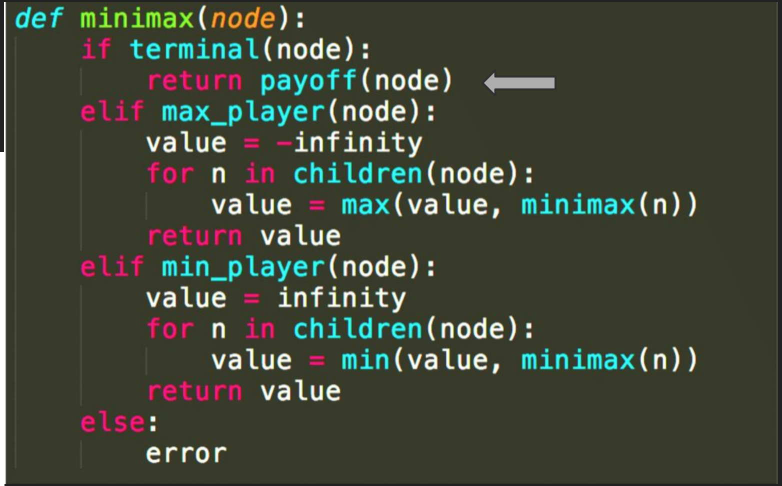


```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

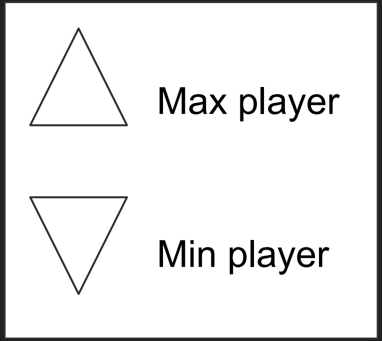


Example of minimax

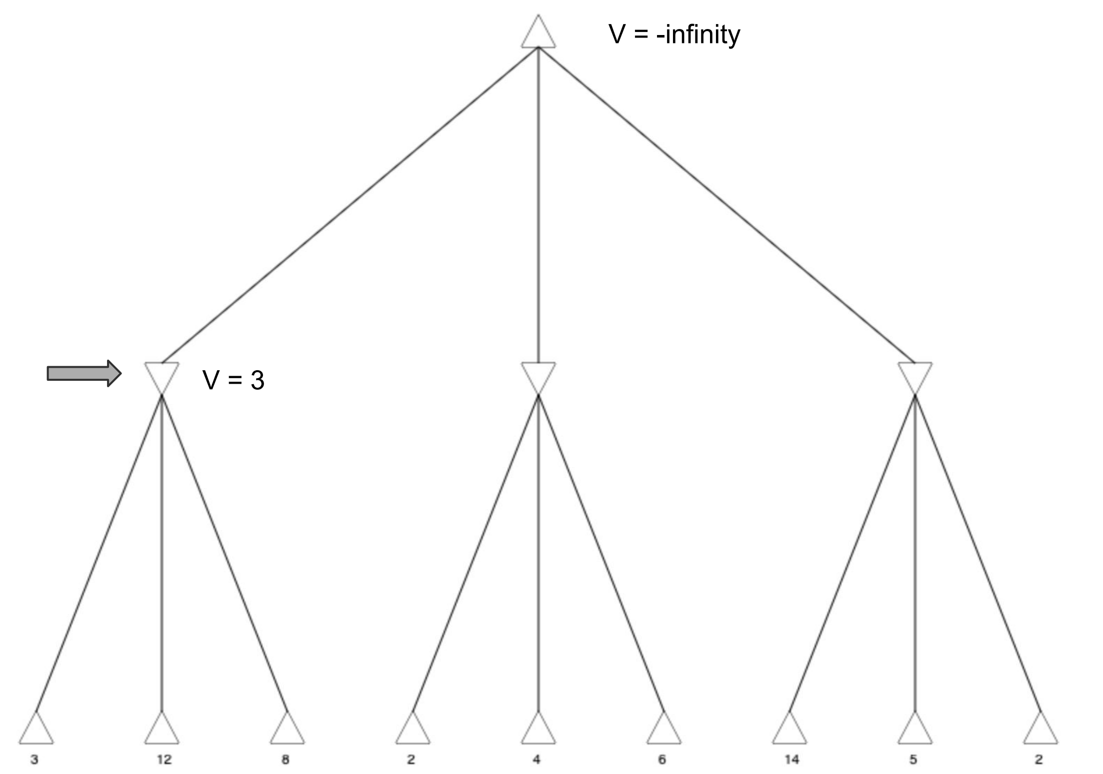


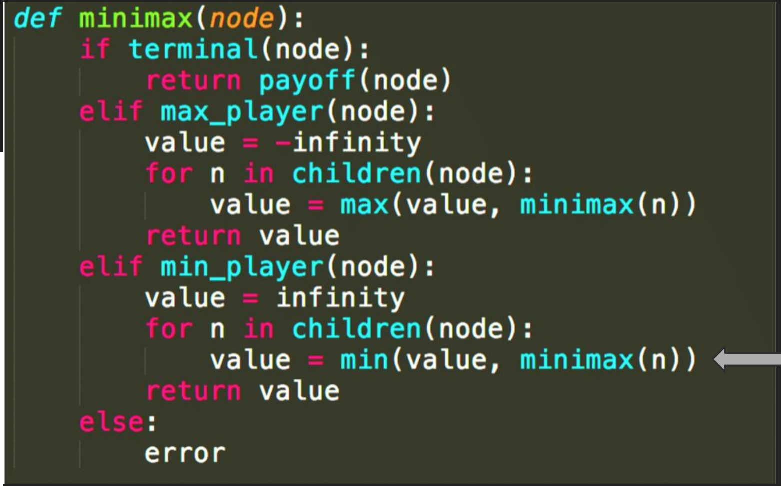


```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```



Example of minimax



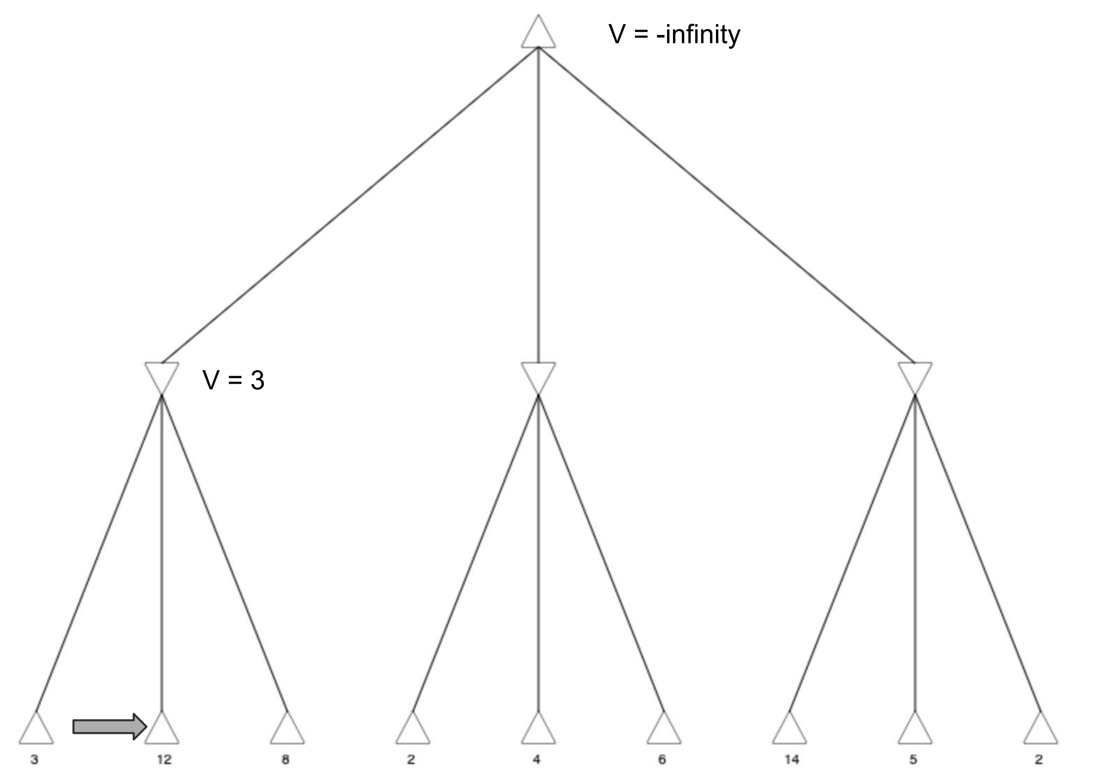


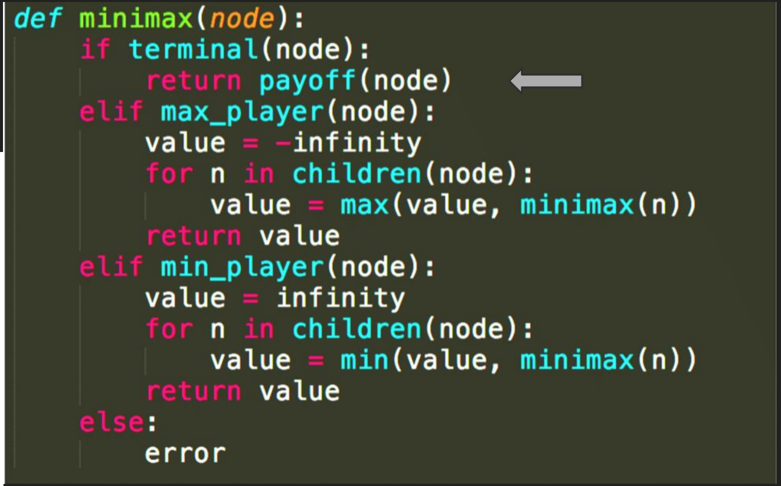
```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

△ Max player

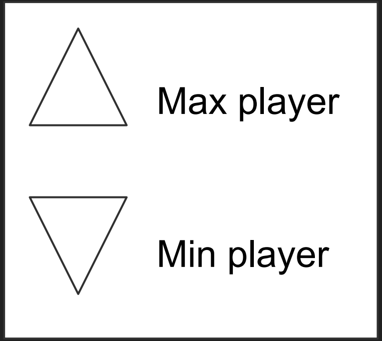
▽ Min player

Example of minimax

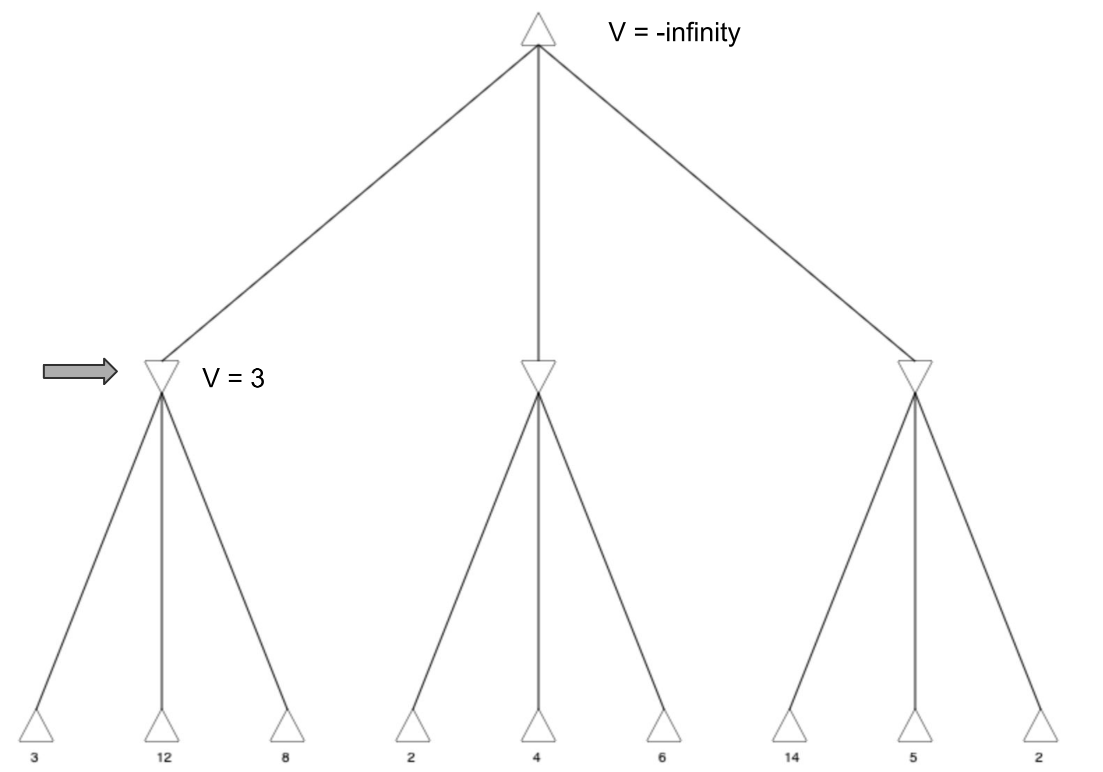


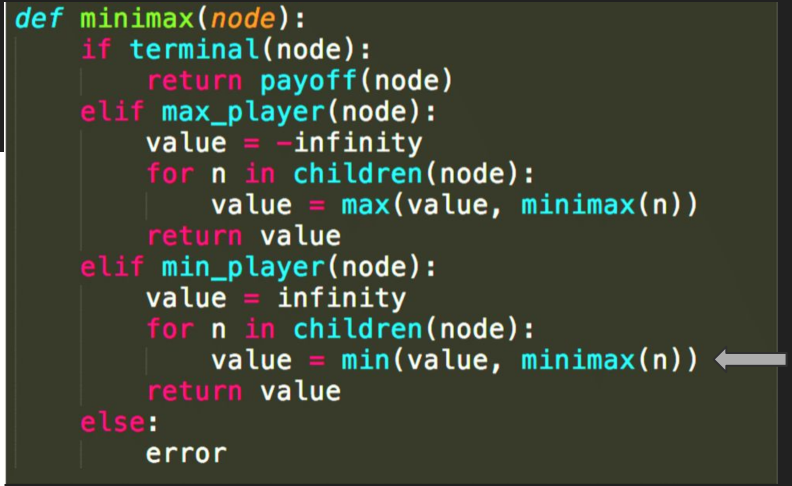


```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```



Example of minimax



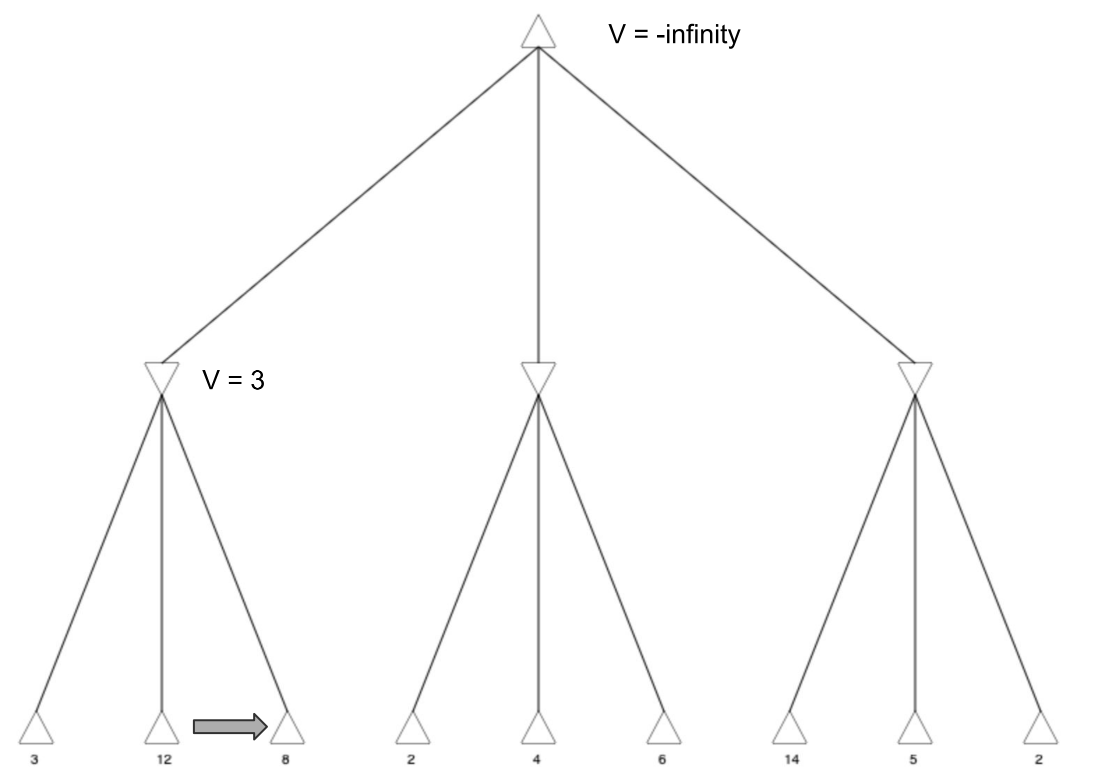


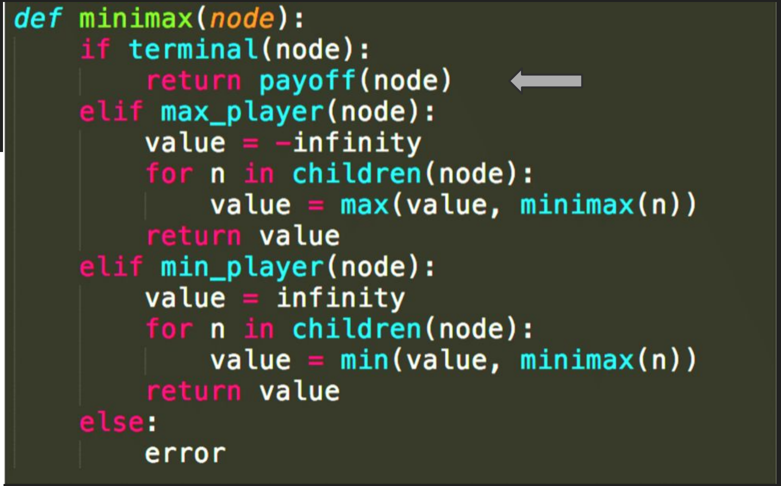
```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

△ Max player

▽ Min player

Example of minimax



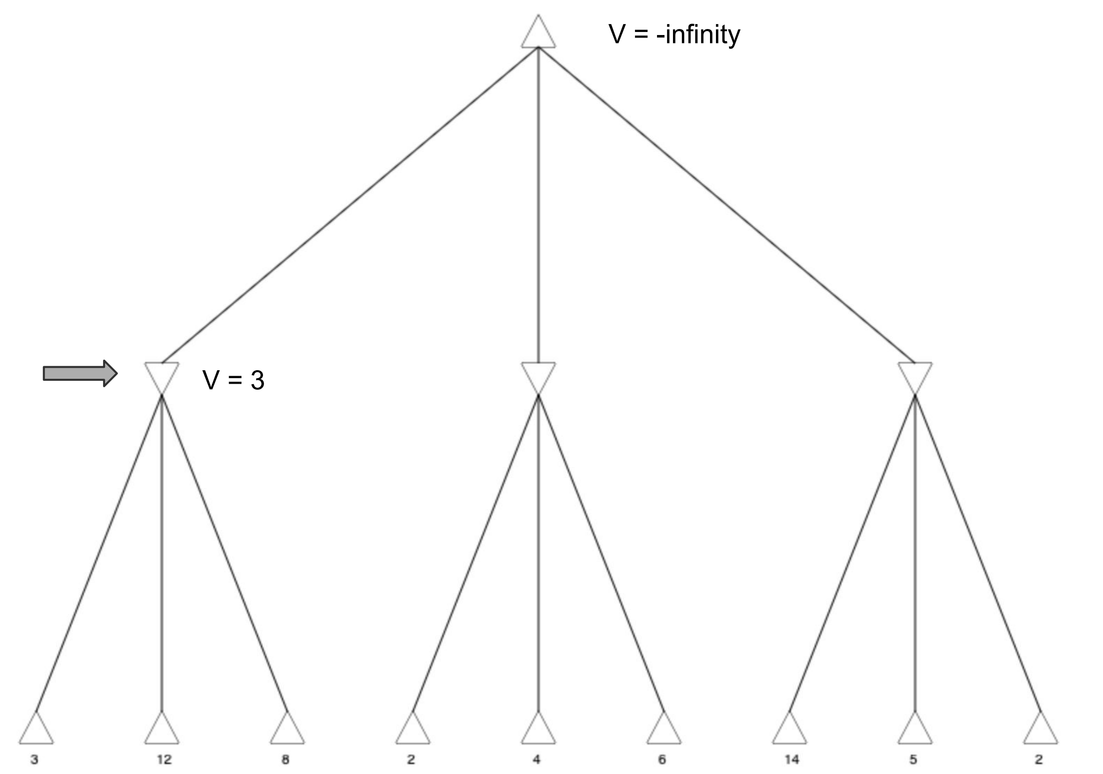


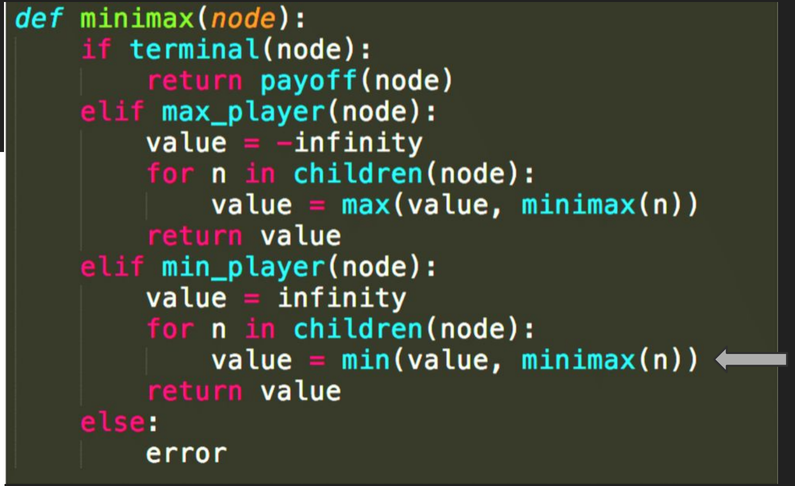
```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

△ Max player

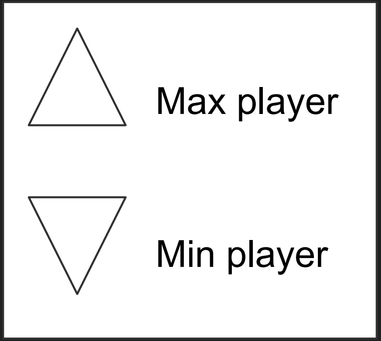
▽ Min player

Example of minimax





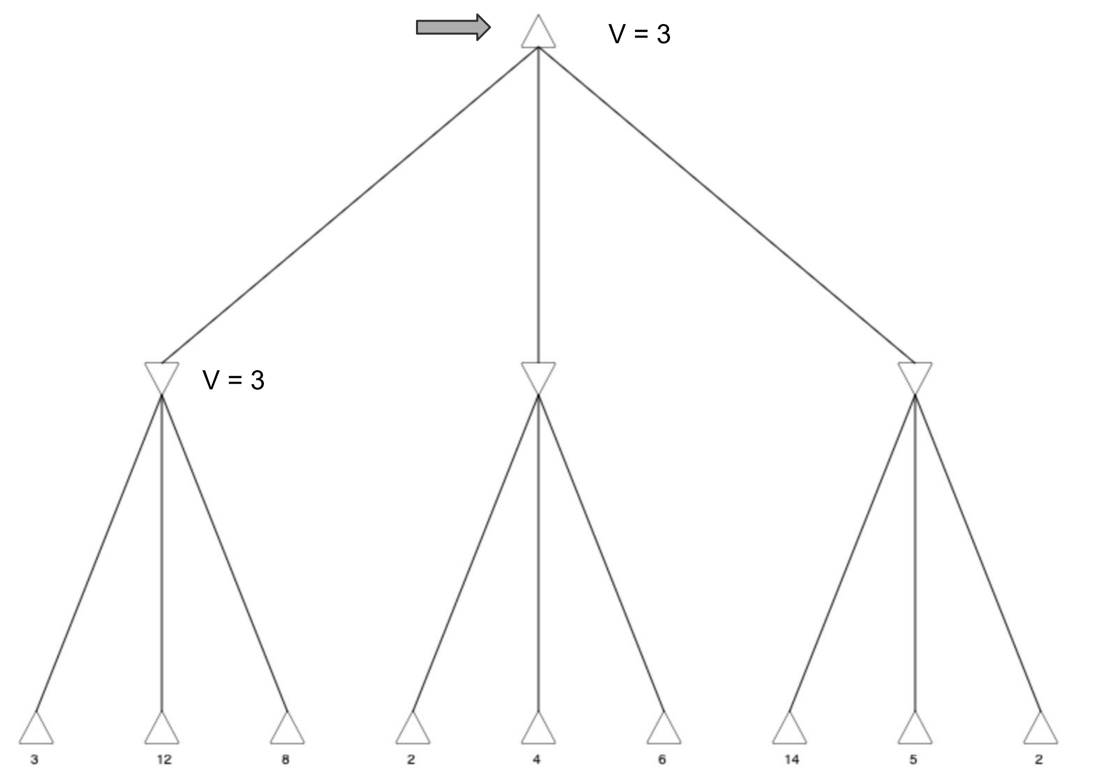
```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

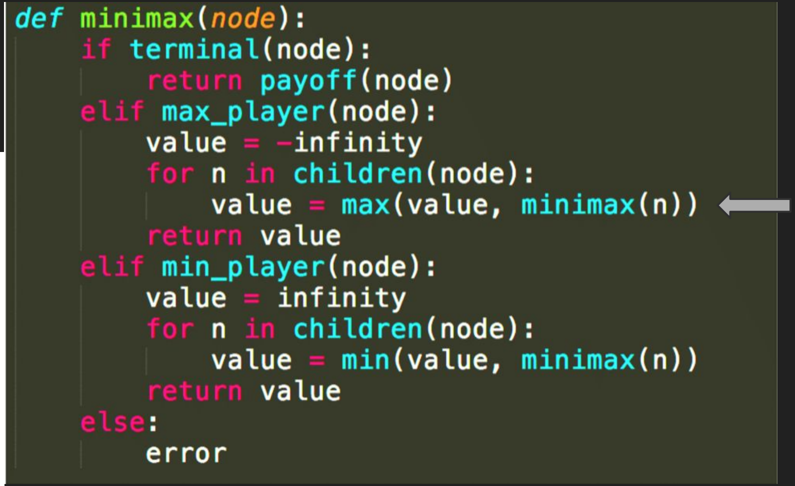


Max player

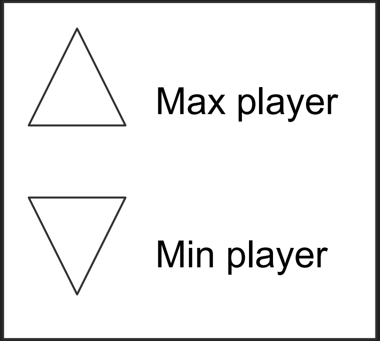
Min player

Example of minimax

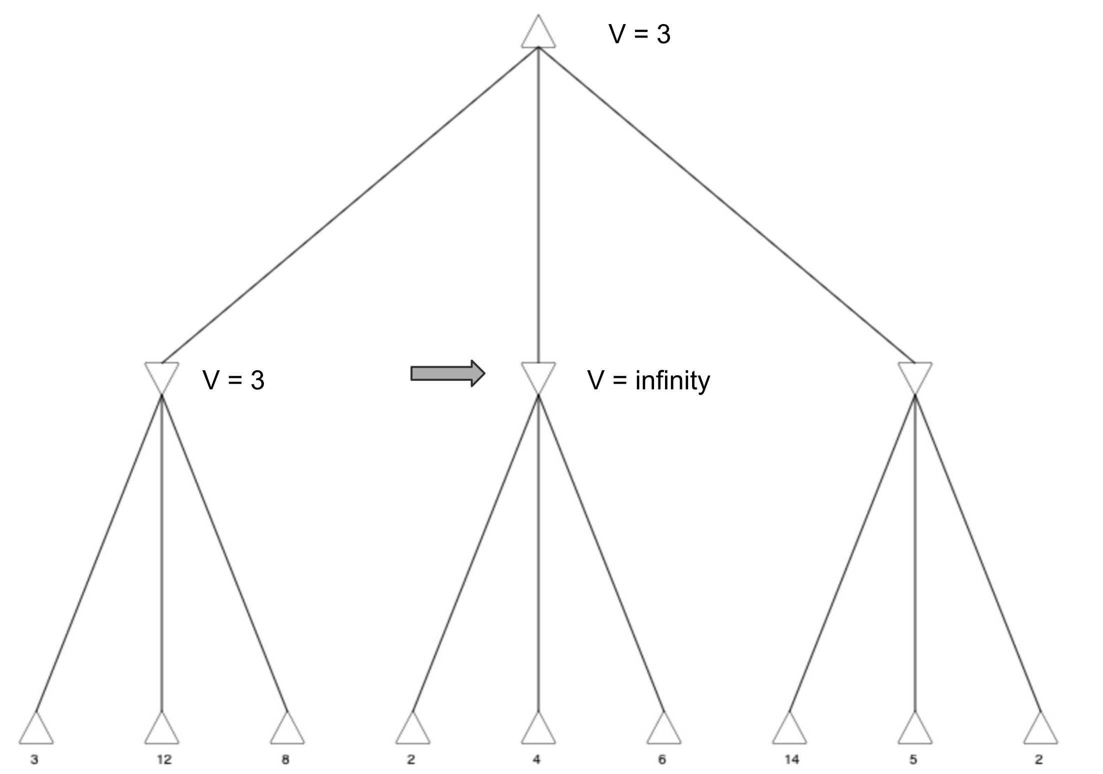


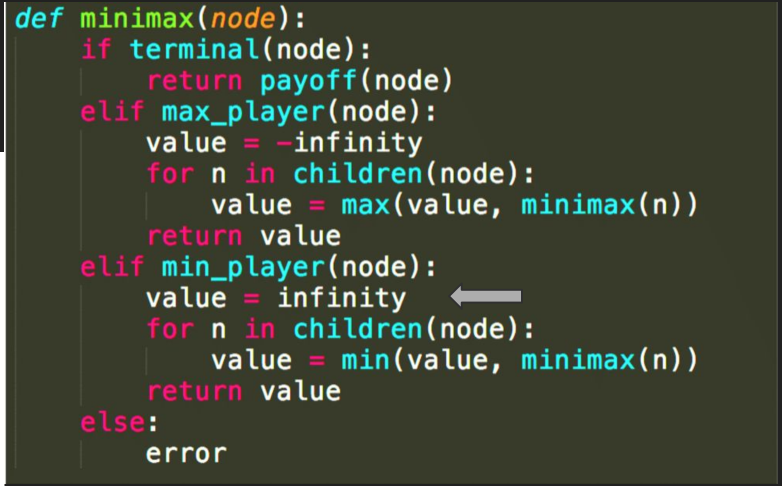


```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

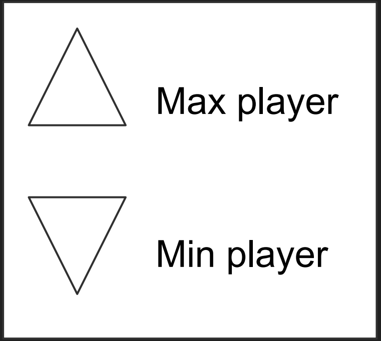


Example of minimax

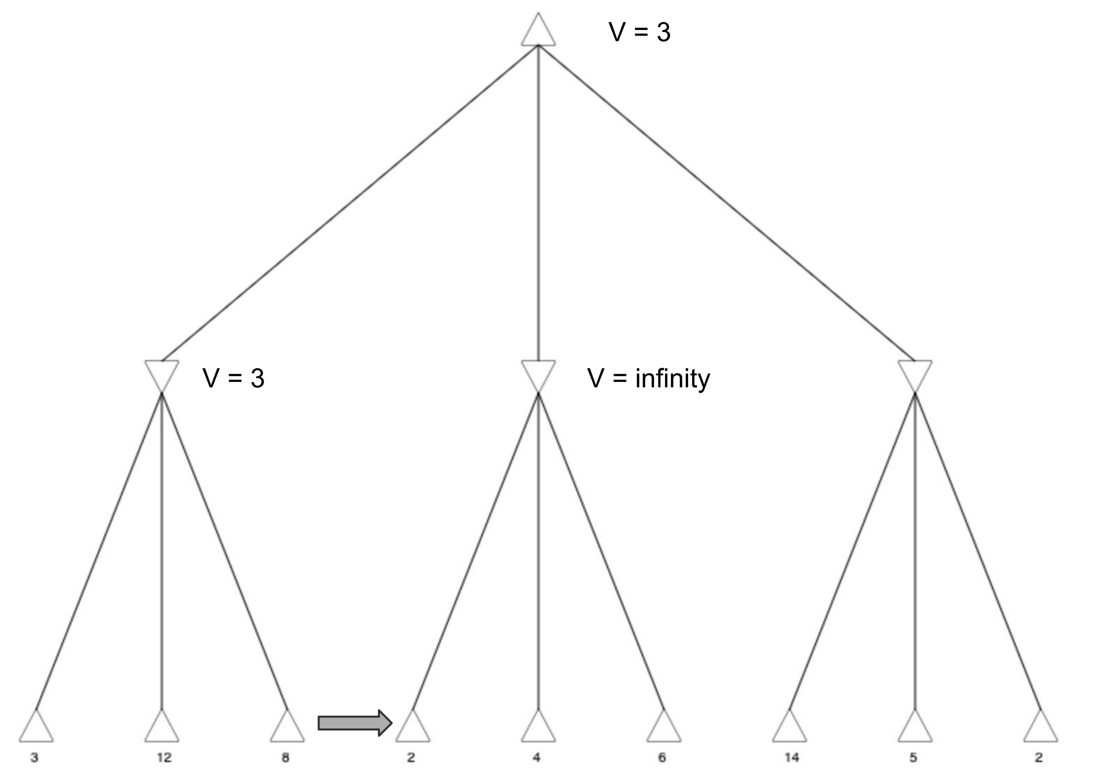


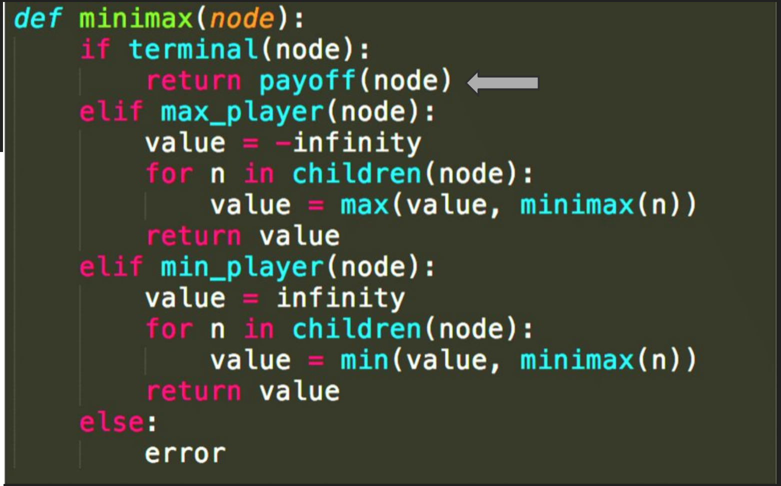


```python
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

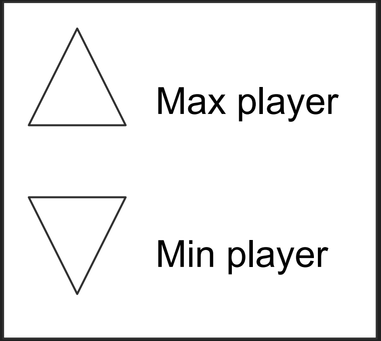


Example of minimax

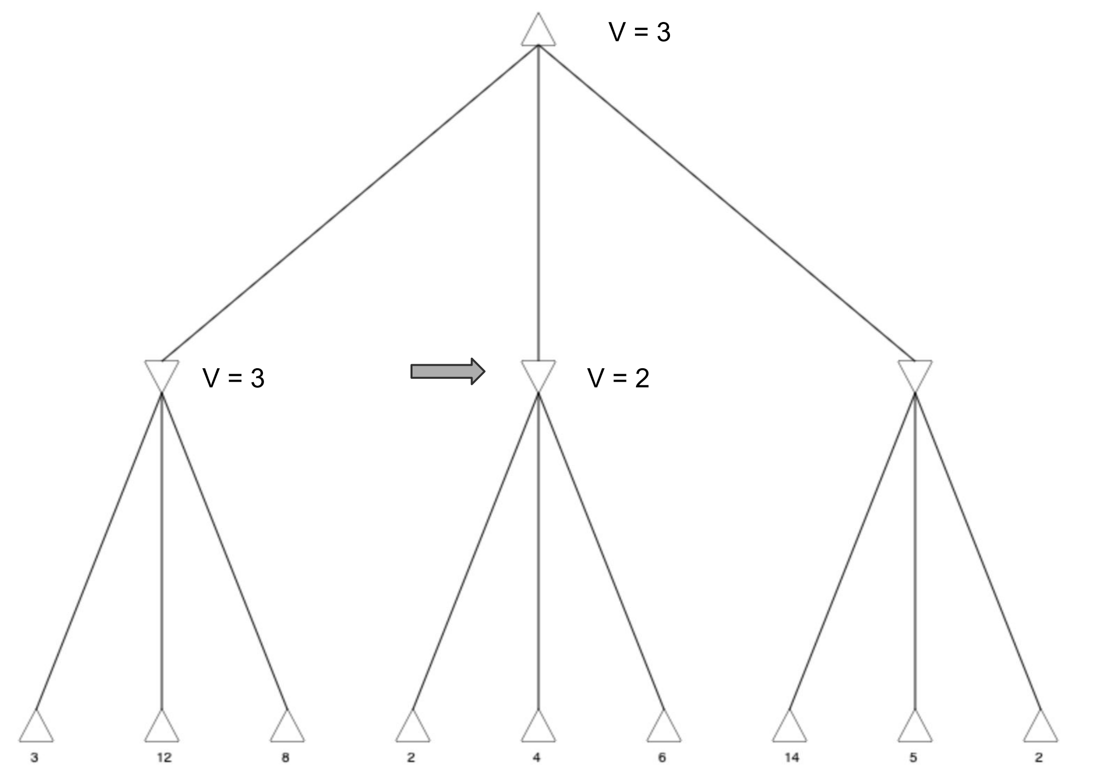


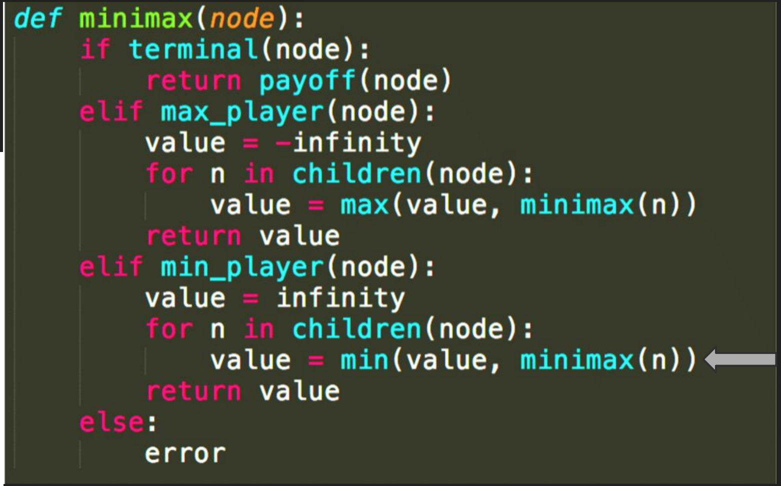


```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```



Example of minimax



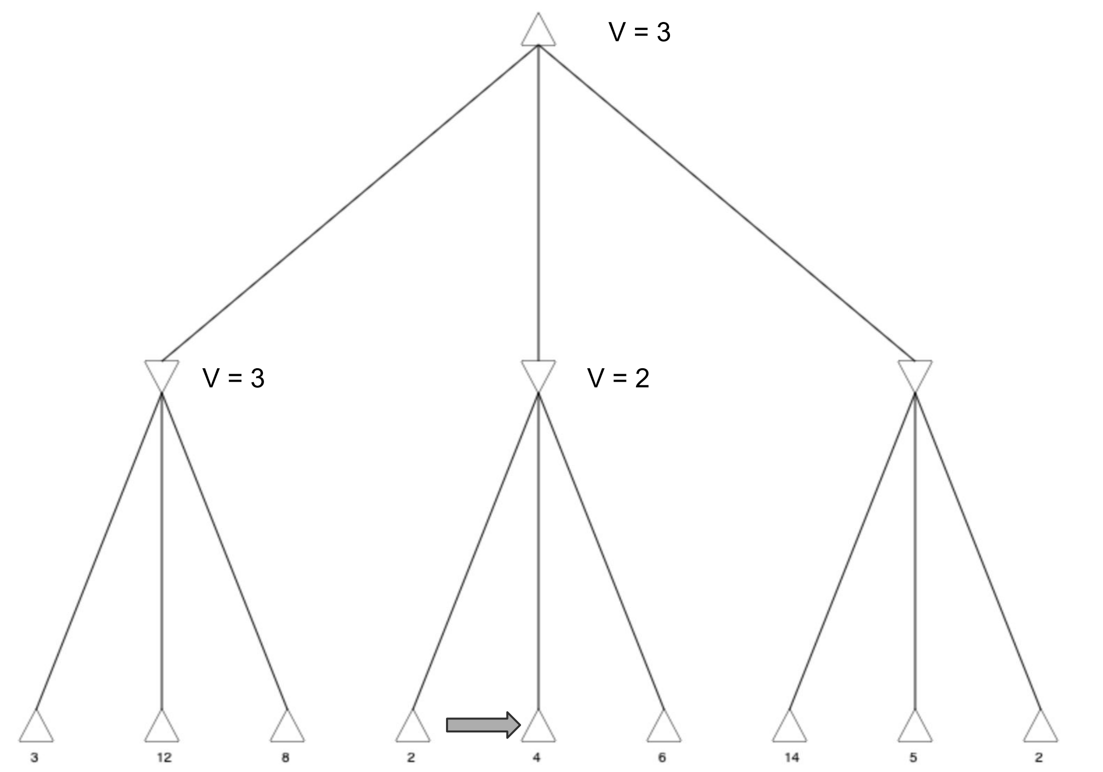


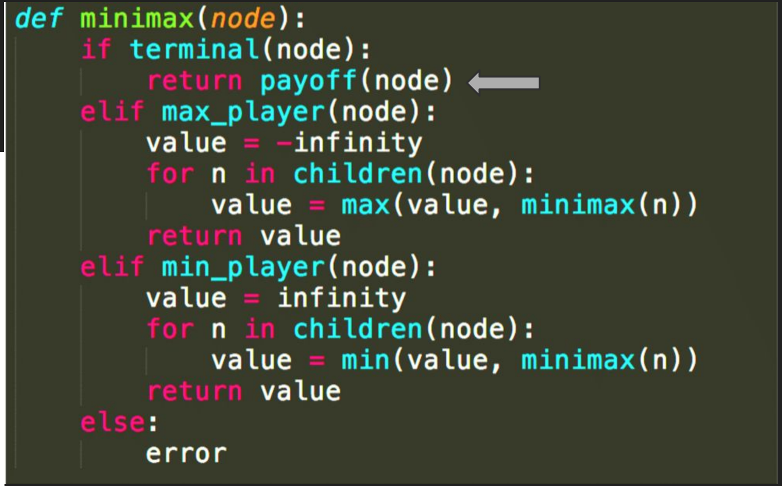
```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

△ Max player

▽ Min player

Example of minimax



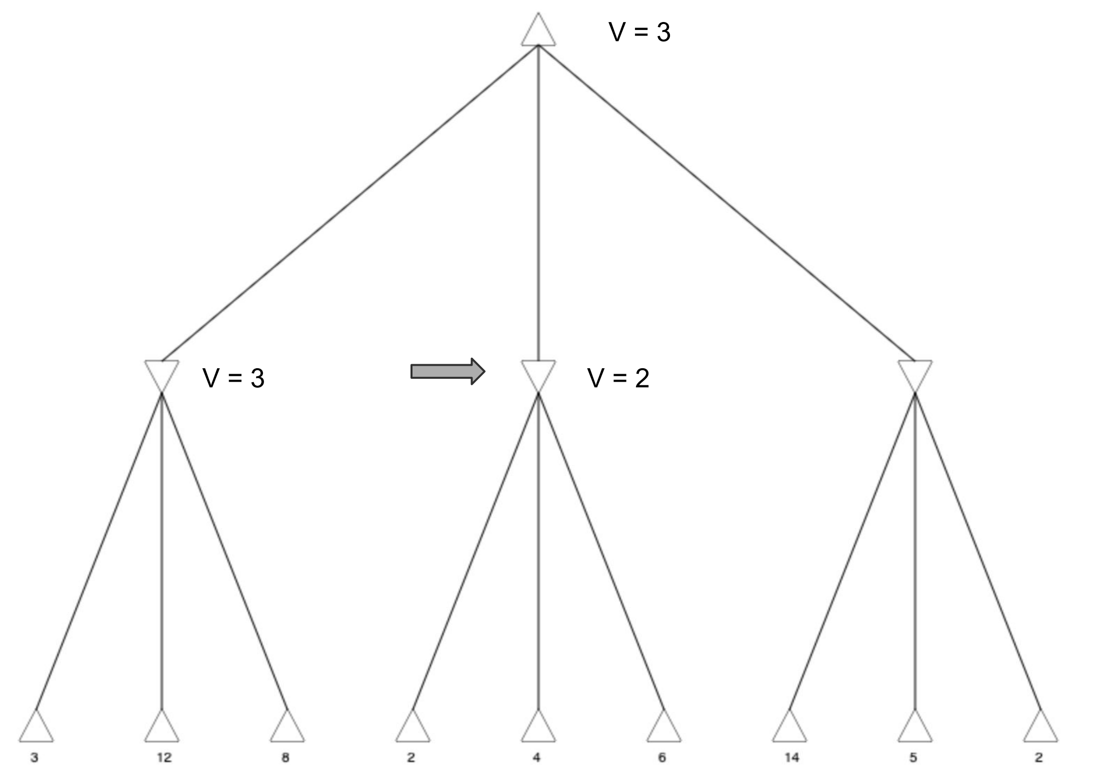


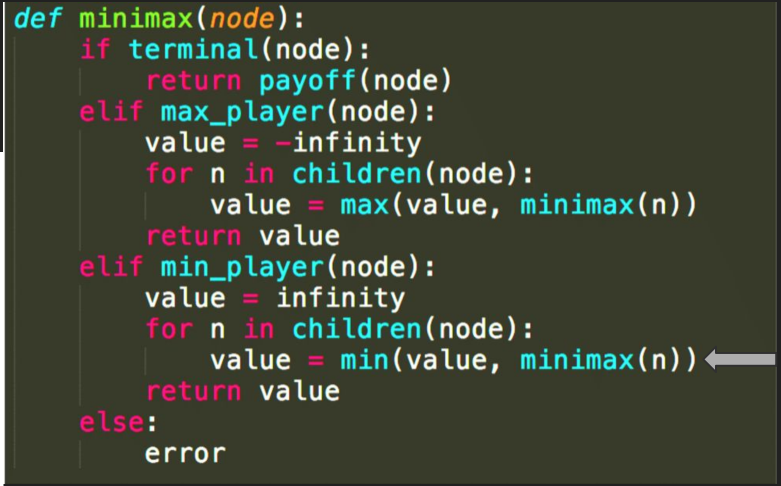
```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

△ Max player

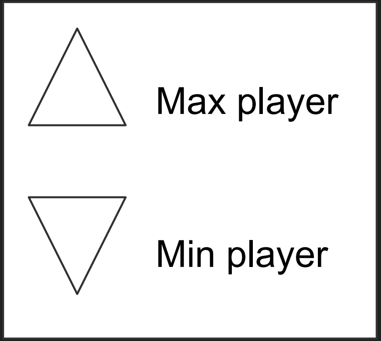
▽ Min player

Example of minimax

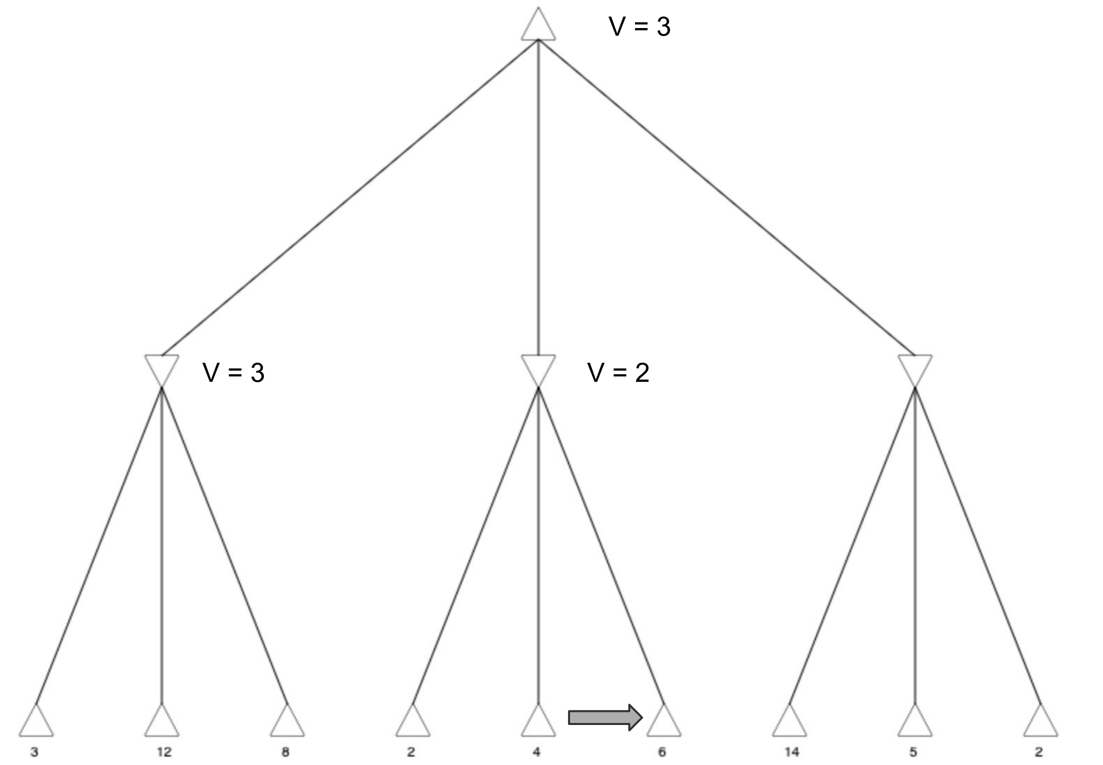


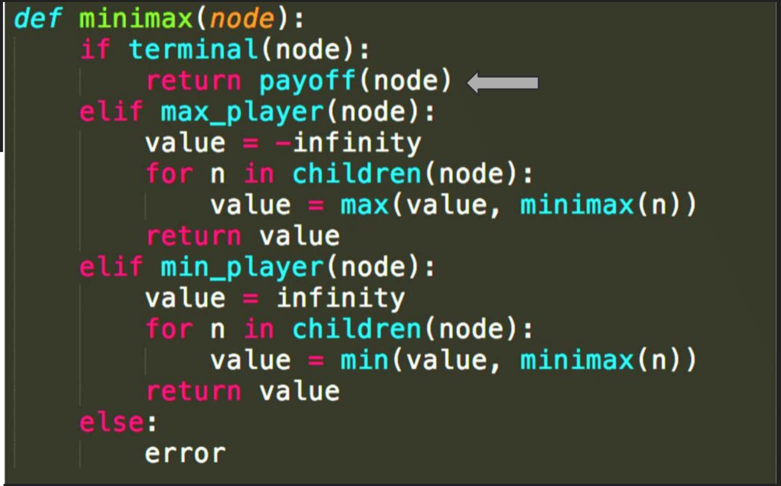


```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

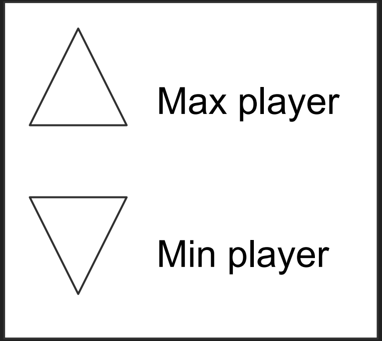


Example of minimax

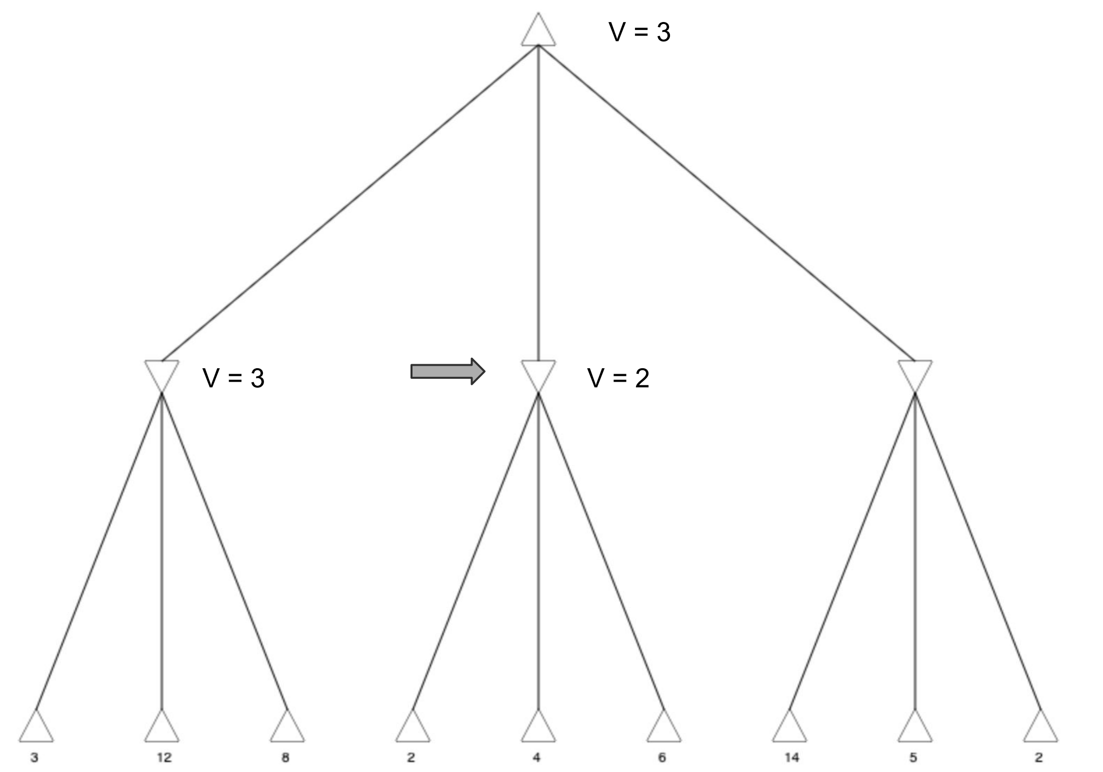


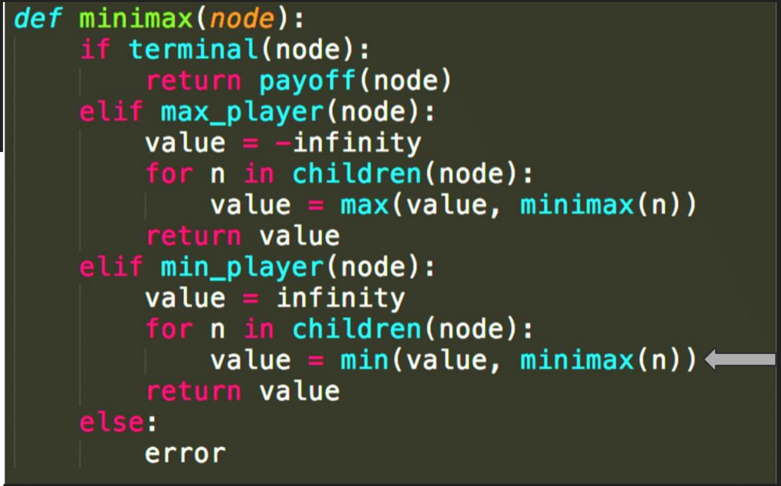


```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

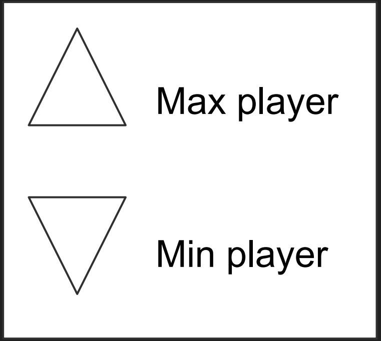


Example of minimax

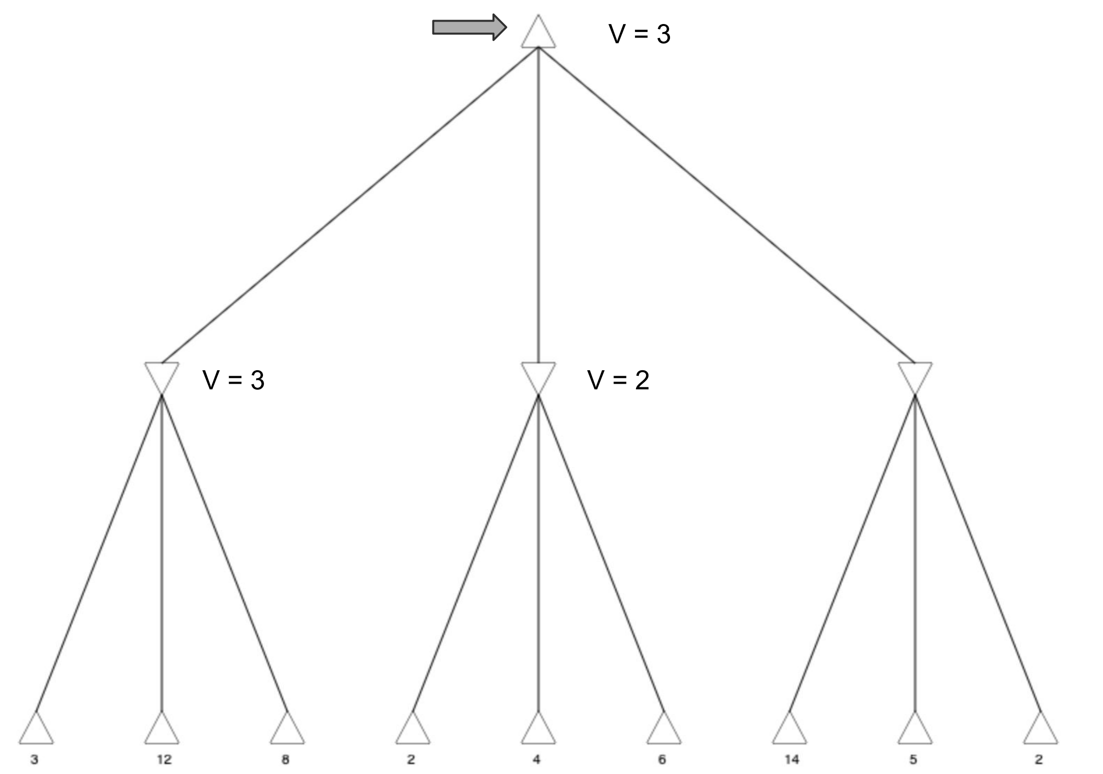


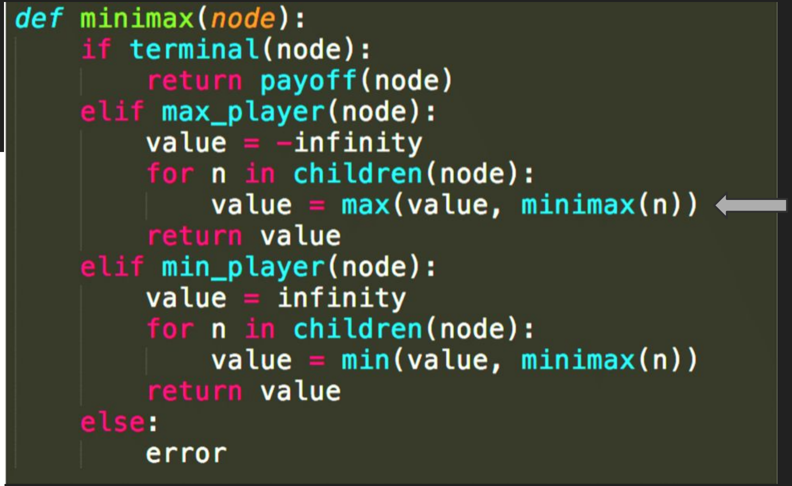


```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

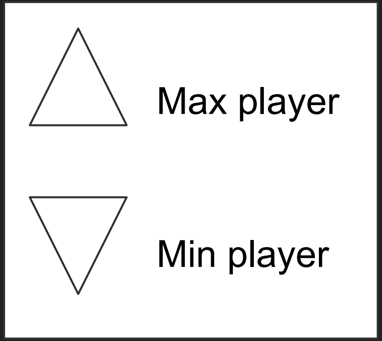


Example of minimax





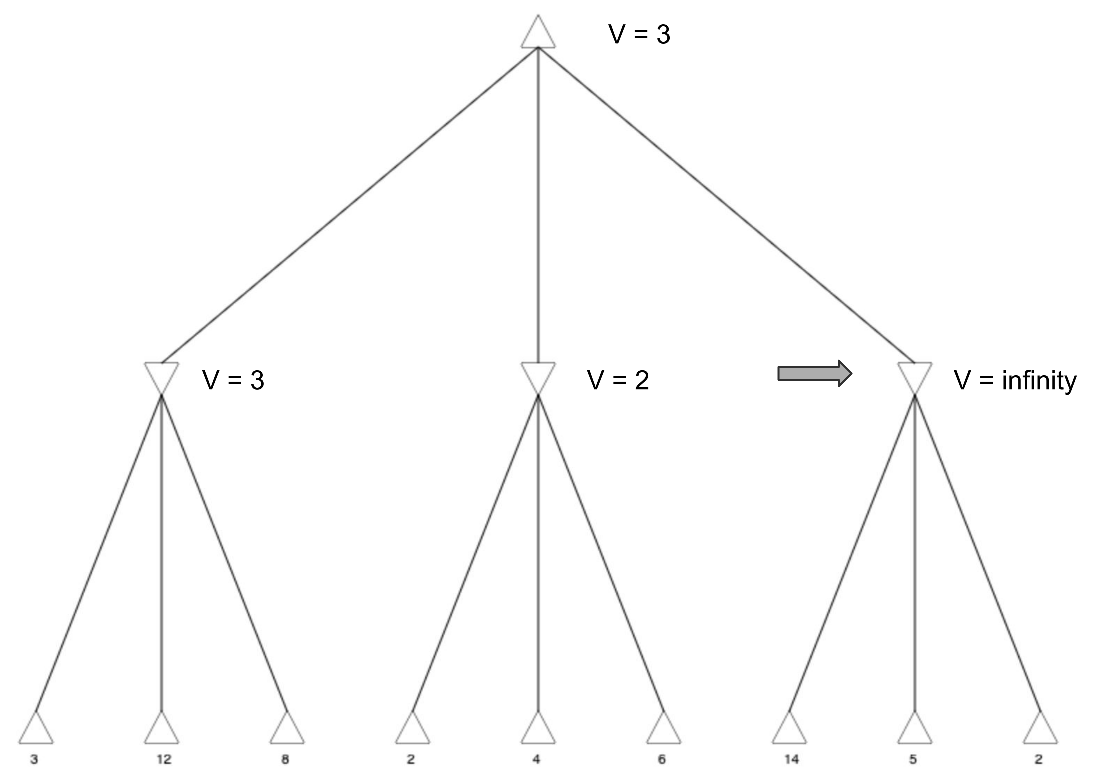
```python
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

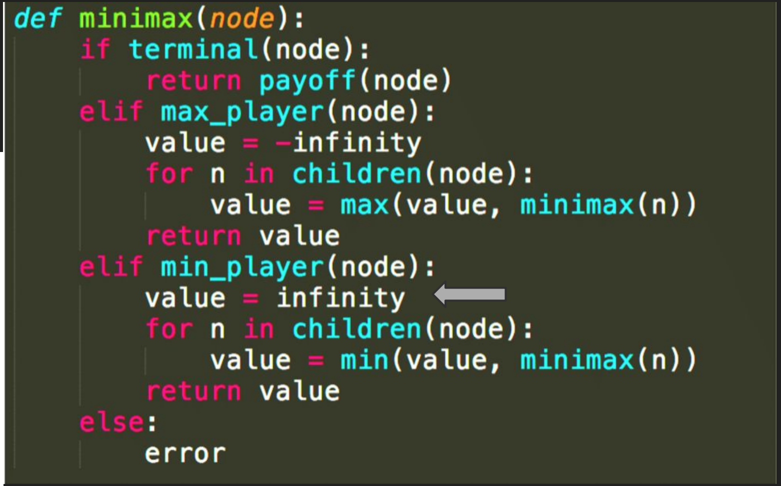


Max player

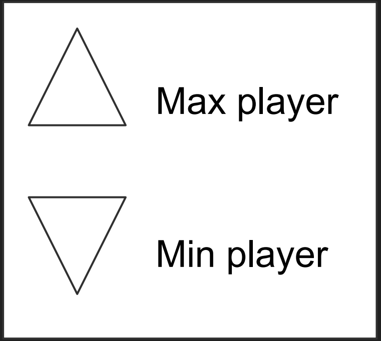
Min player

Example of minimax

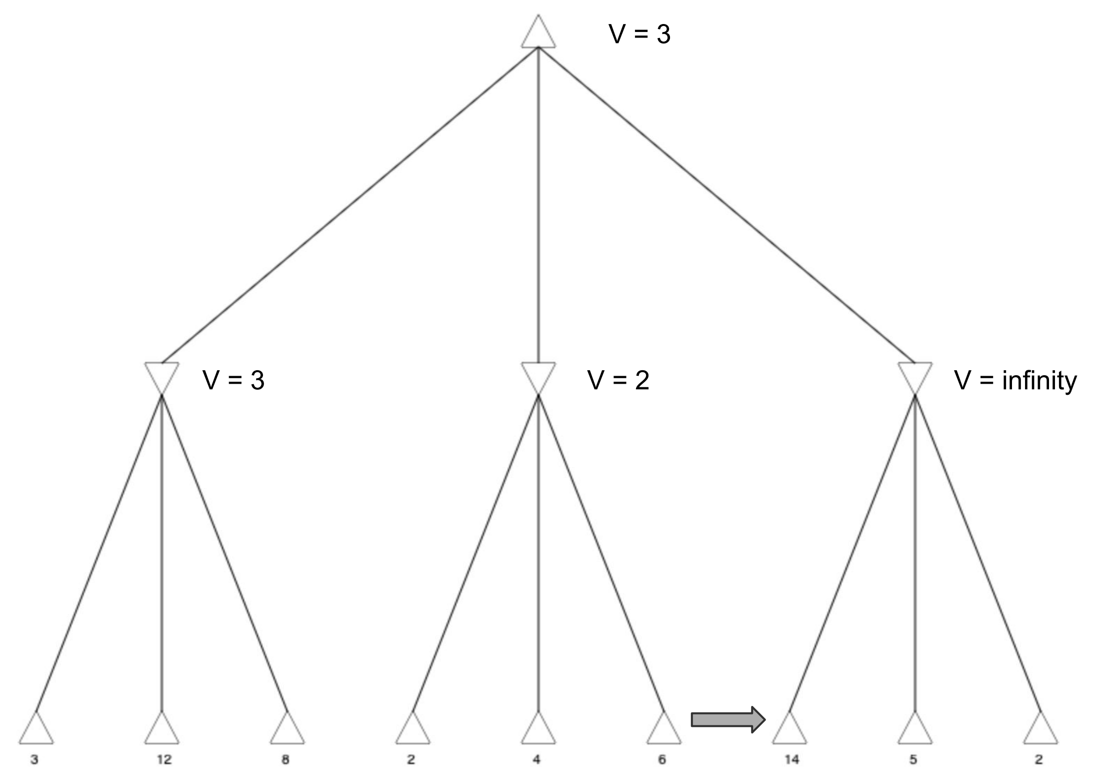


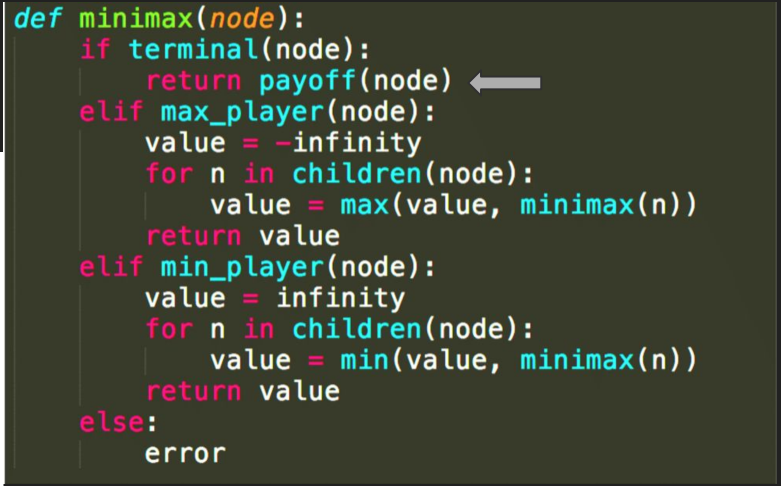


```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

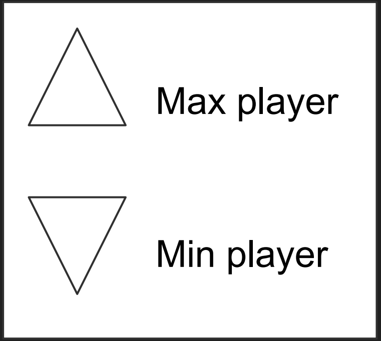


Example of minimax

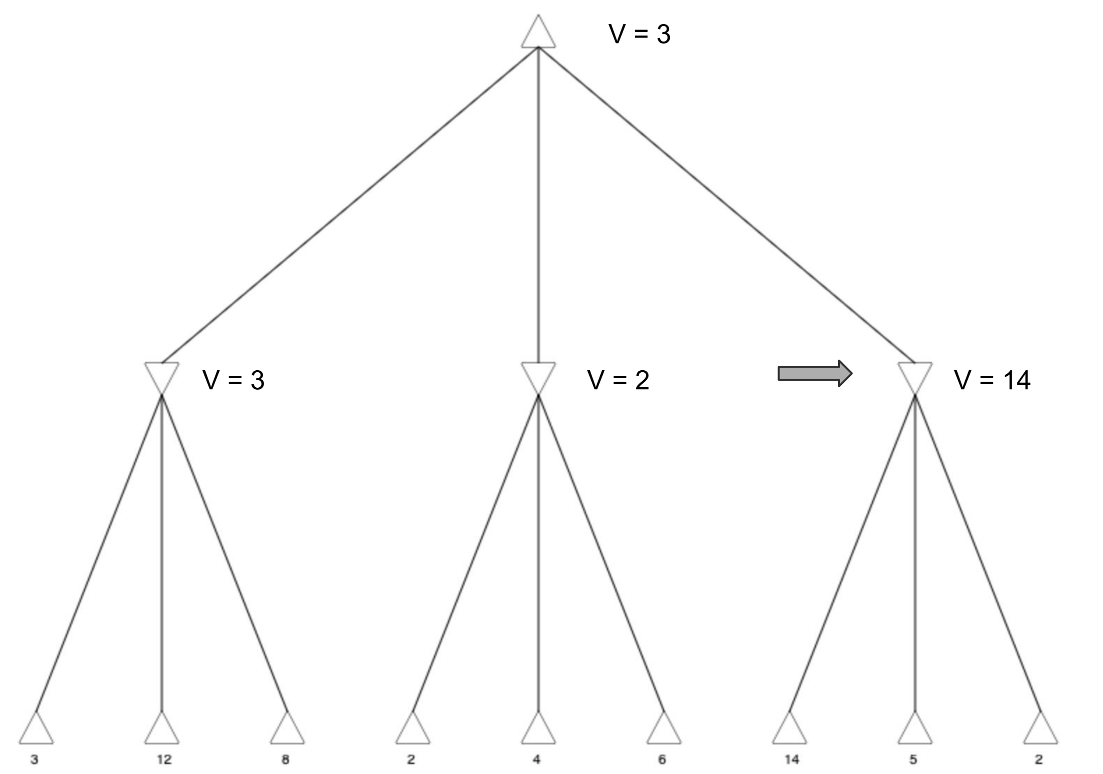


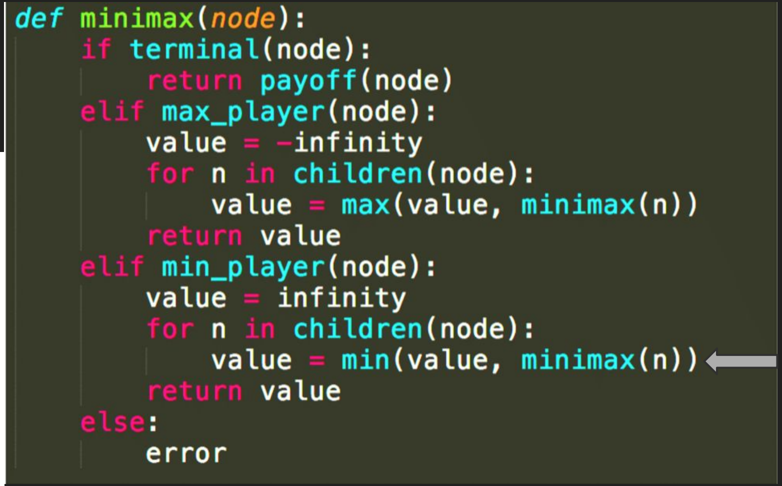


```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```



Example of minimax



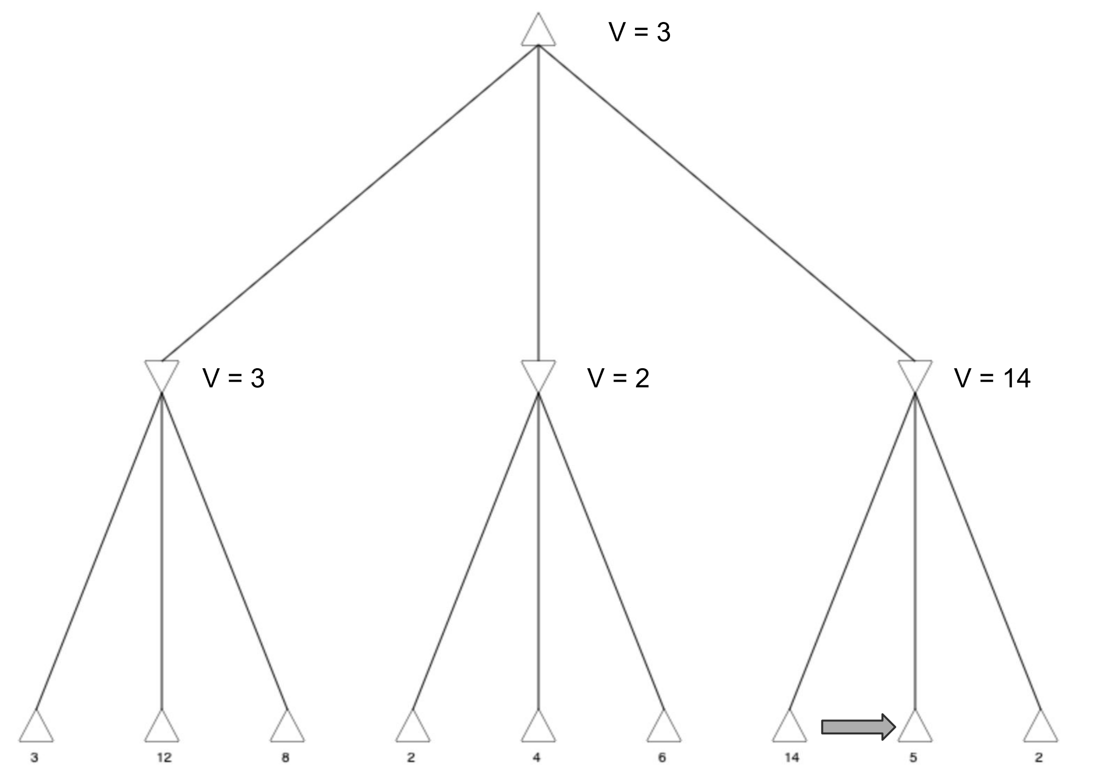


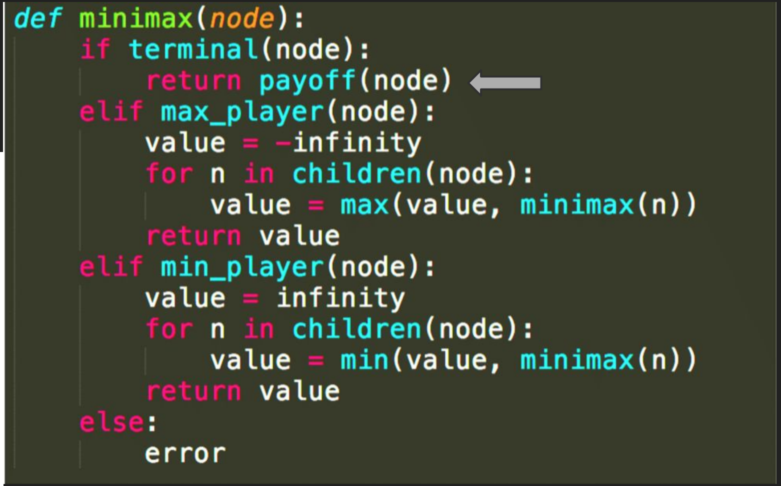
```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

△ Max player

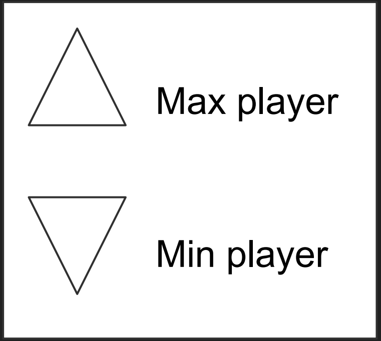
▽ Min player

Example of minimax

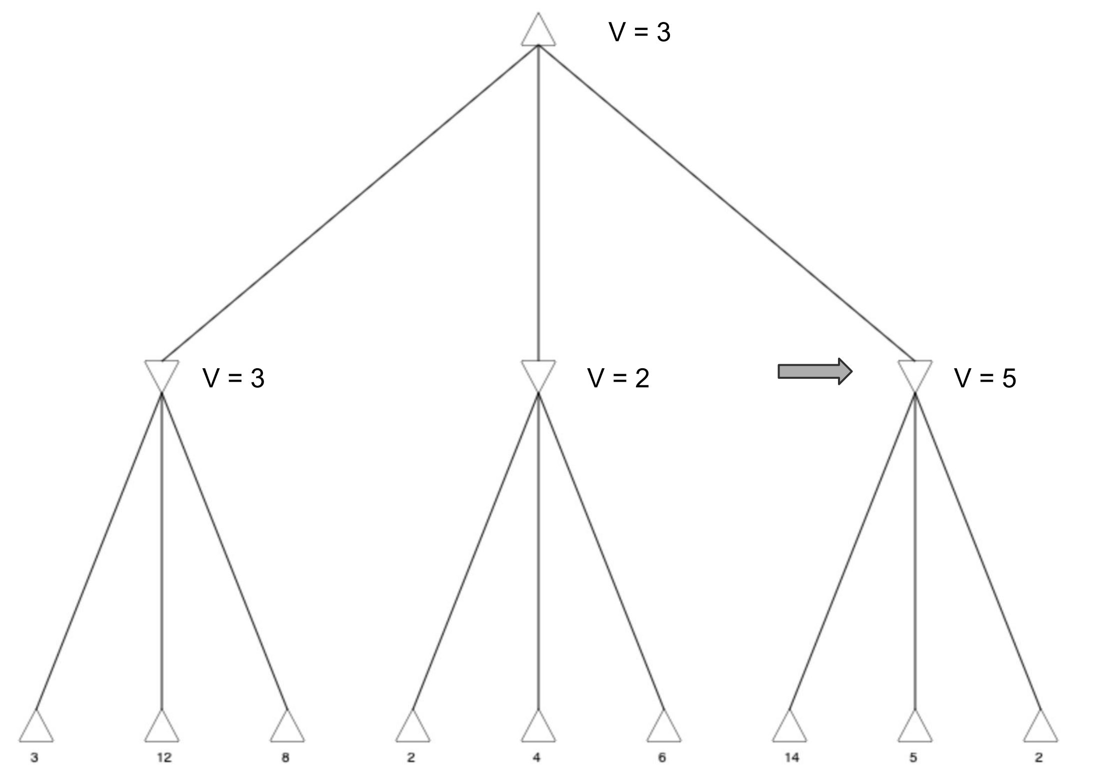


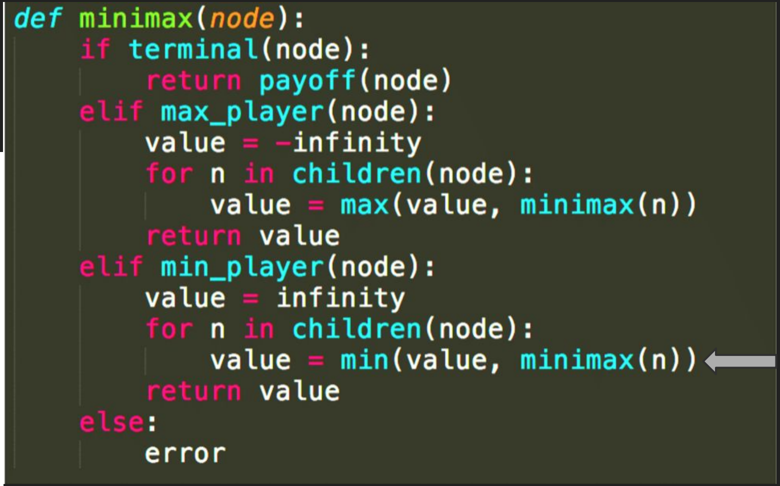


```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

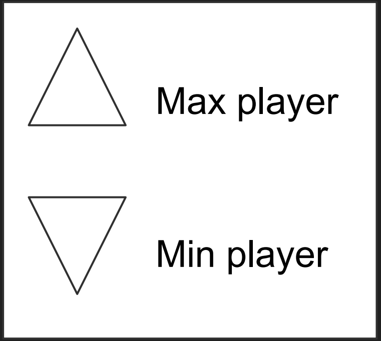


Example of minimax

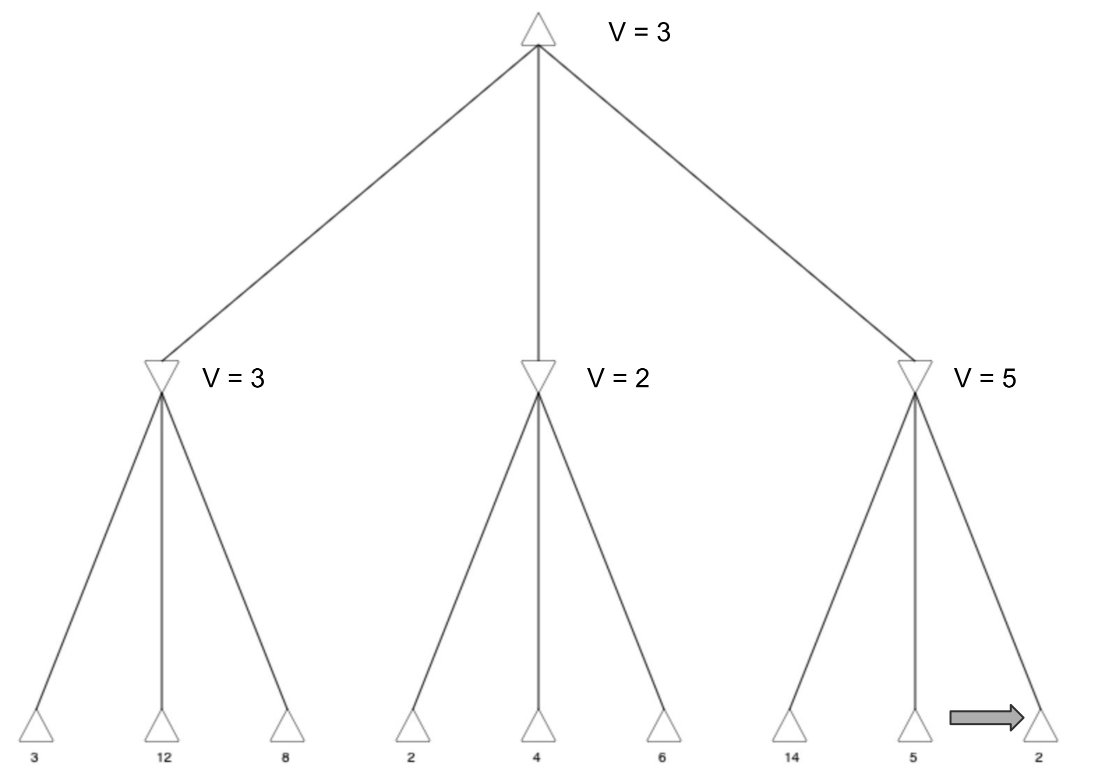


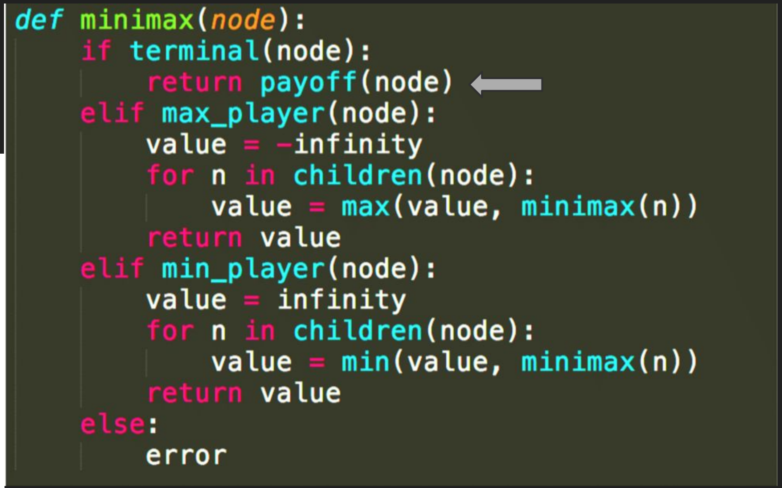


```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

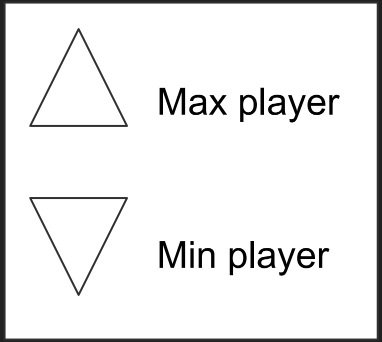


Example of minimax

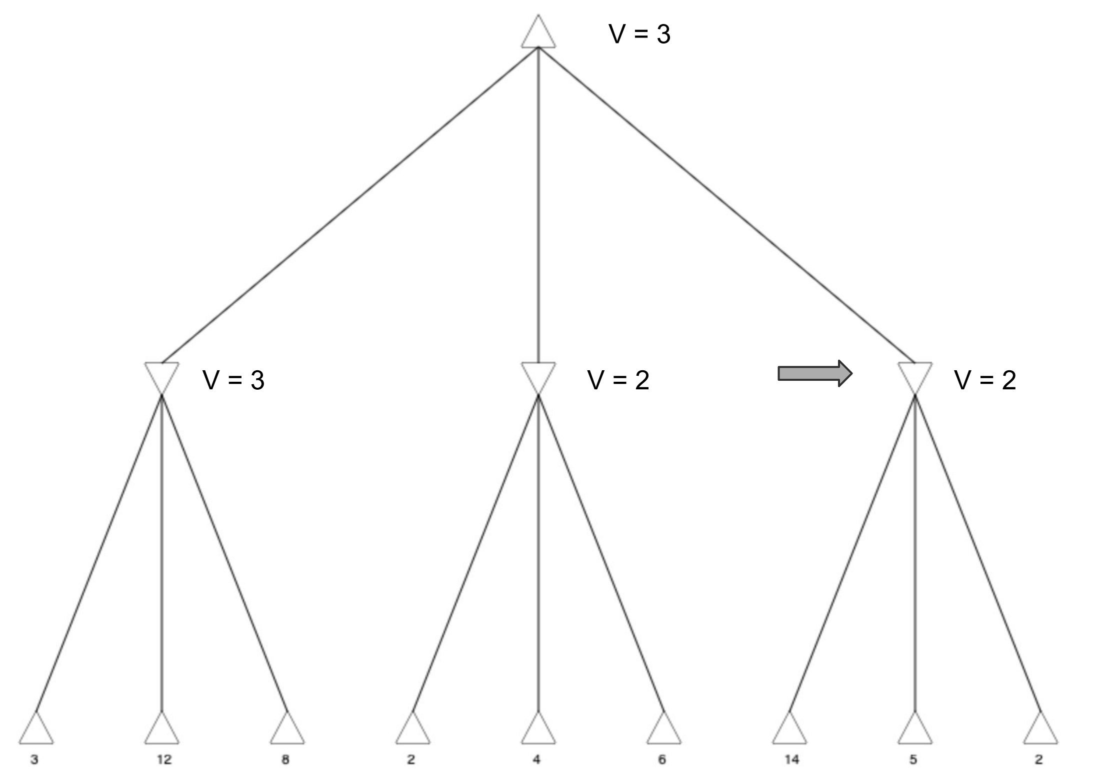


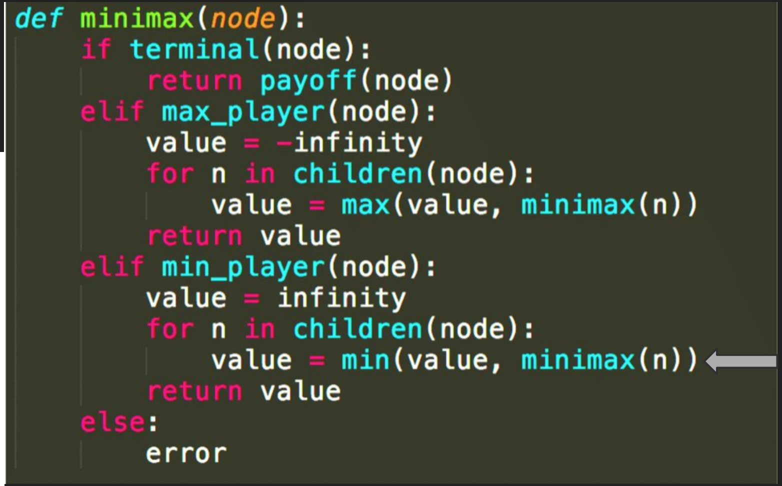


```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

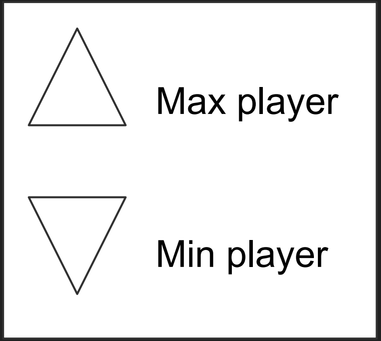


Example of minimax

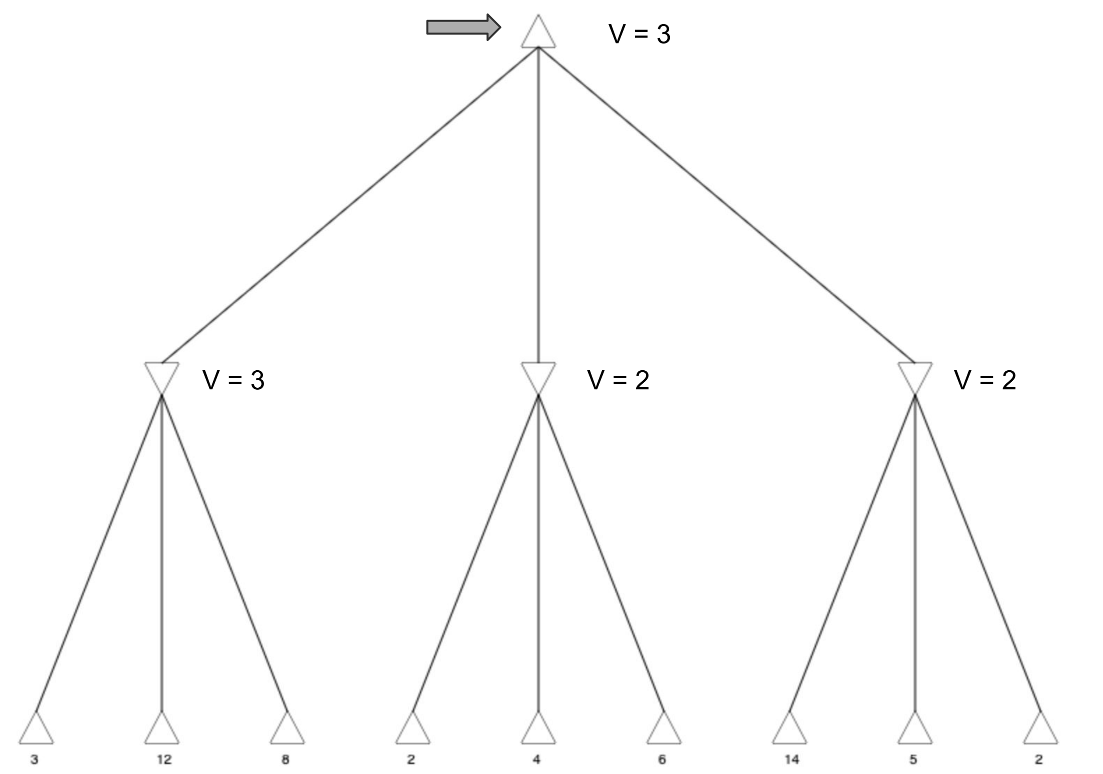


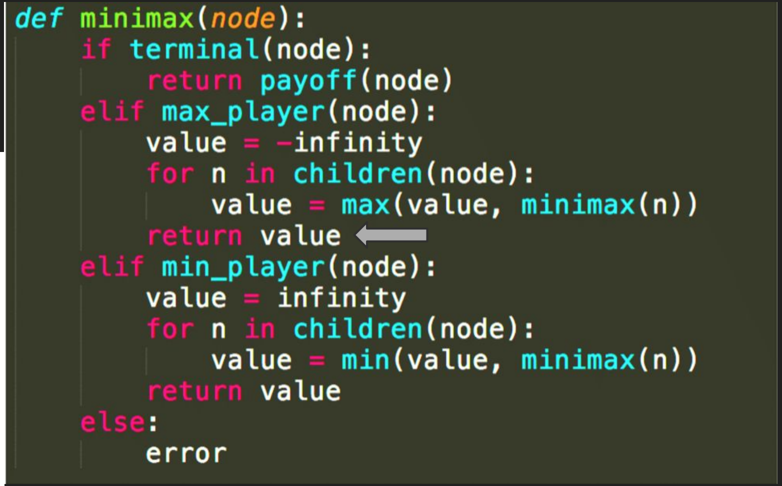


```
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```



Example of minimax





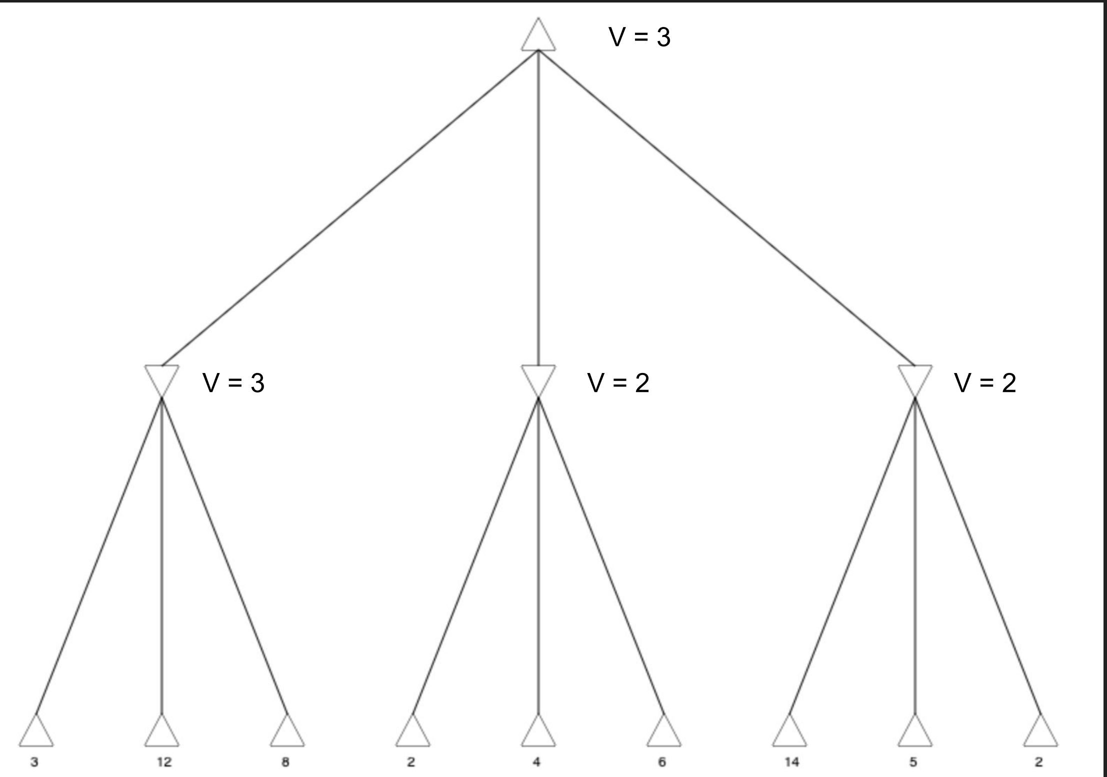
```python
def minimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, minimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, minimax(n))
        return value
    else:
        error
```

△ Max player

▽ Min player

FINAL RESULT: 3

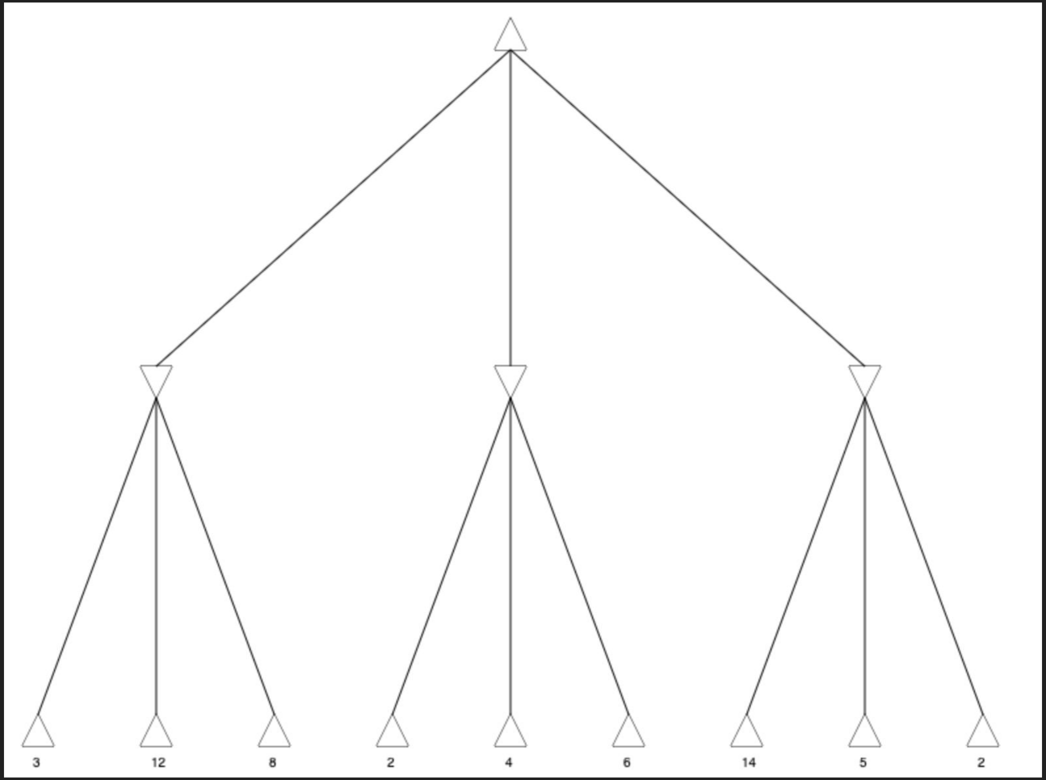
Example of minimax



- For minimax, we must visit each node of the tree in order to obtain the correct minimax value at the root of the tree.
- If tree is very large, this becomes computationally expensive.
- One optimization technique that is commonly used is alpha-beta pruning.

Example of Alpha-Beta Pruning

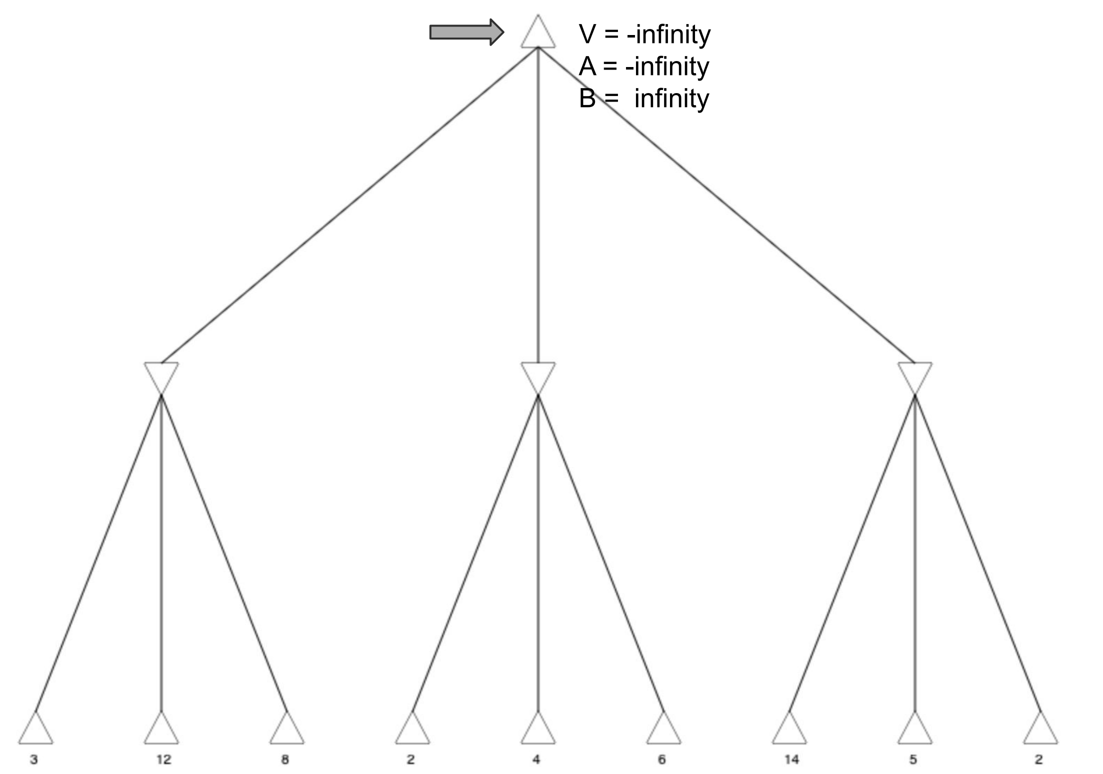
- Will run minimax algorithm with alpha beta pruning on the tree shown below

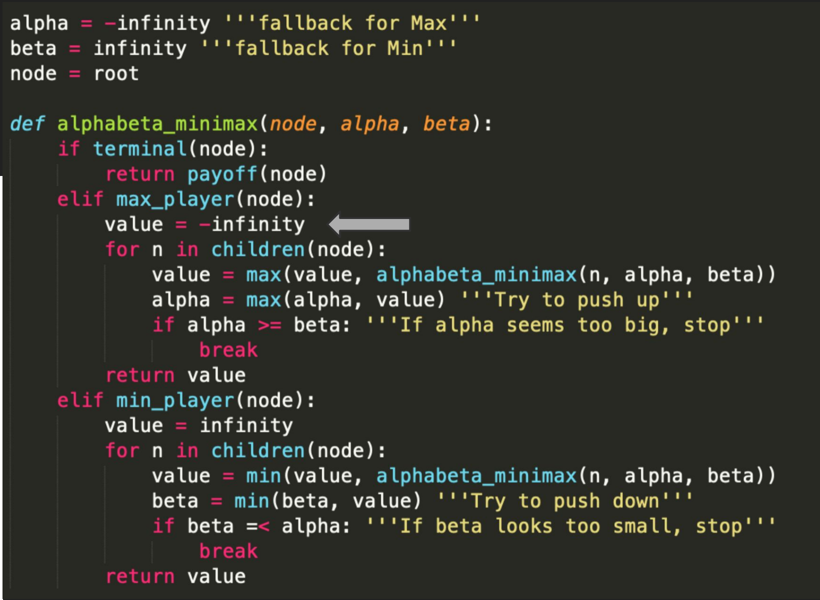


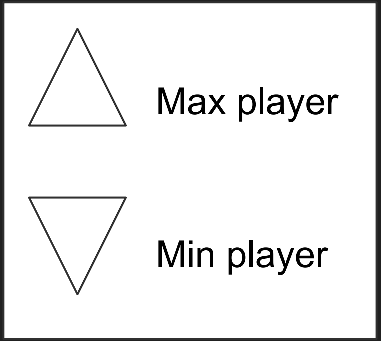
△ Max player

▽ Min player

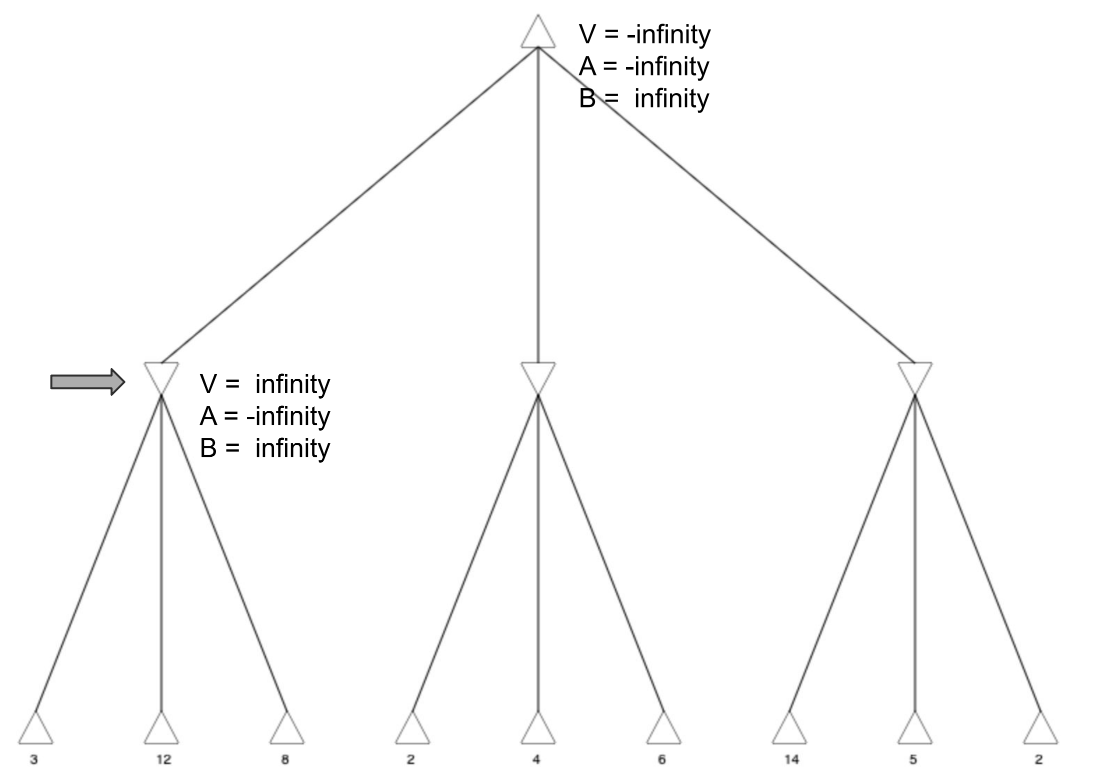
Example of alpha-beta minimax


V = -infinity
A = -infinity
B = infinity


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

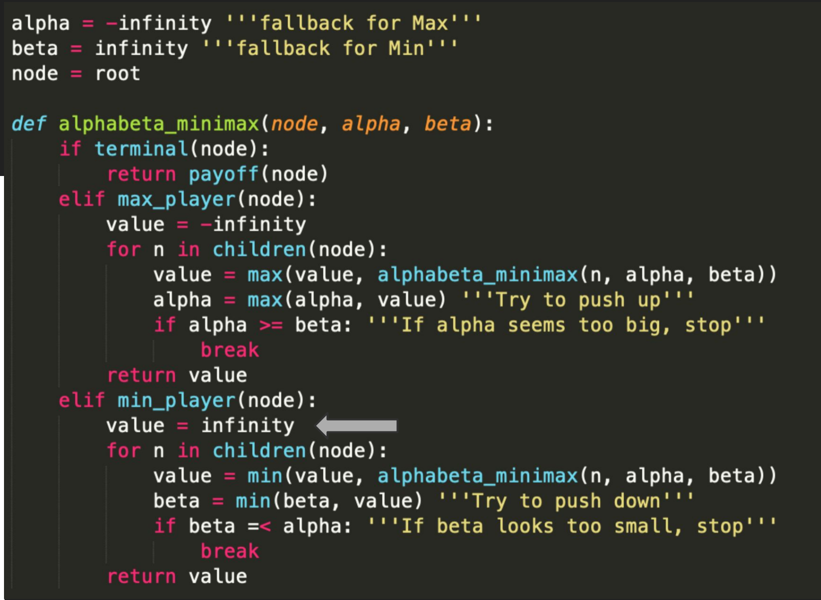


Example of alpha-beta minimax

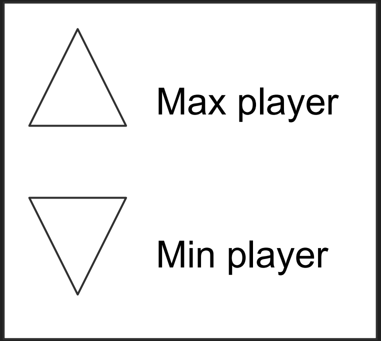


V = -infinity
A = -infinity
B = infinity

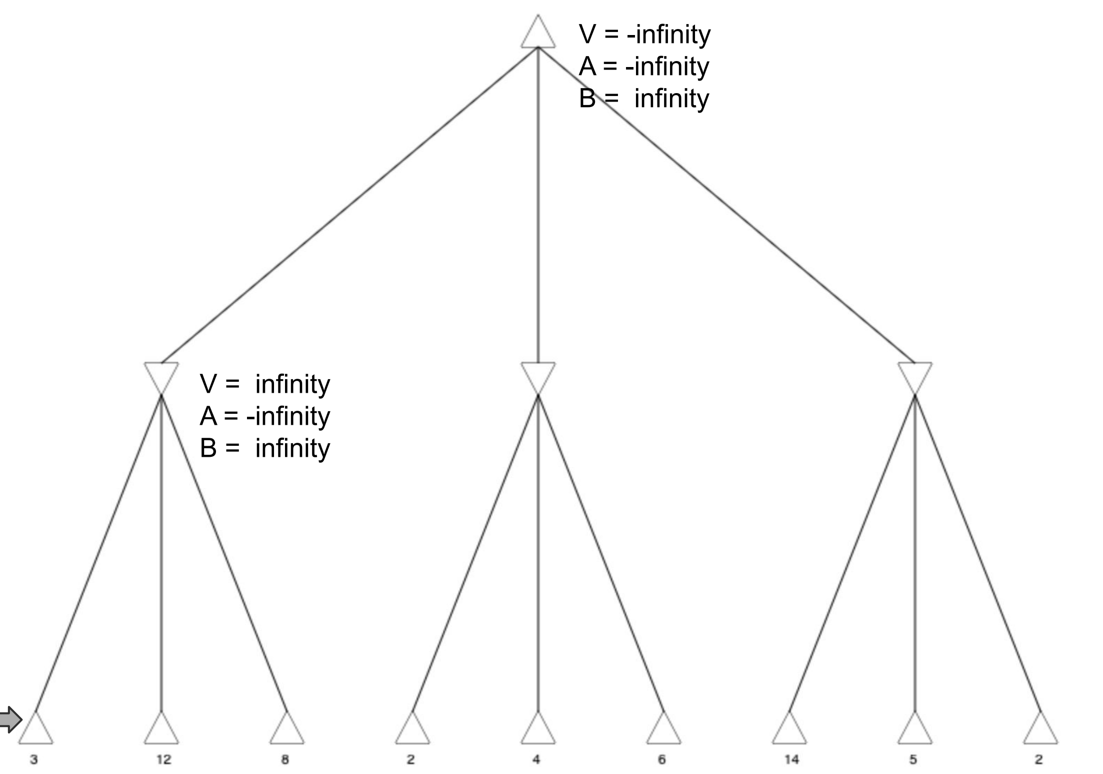
V = infinity
A = -infinity
B = infinity

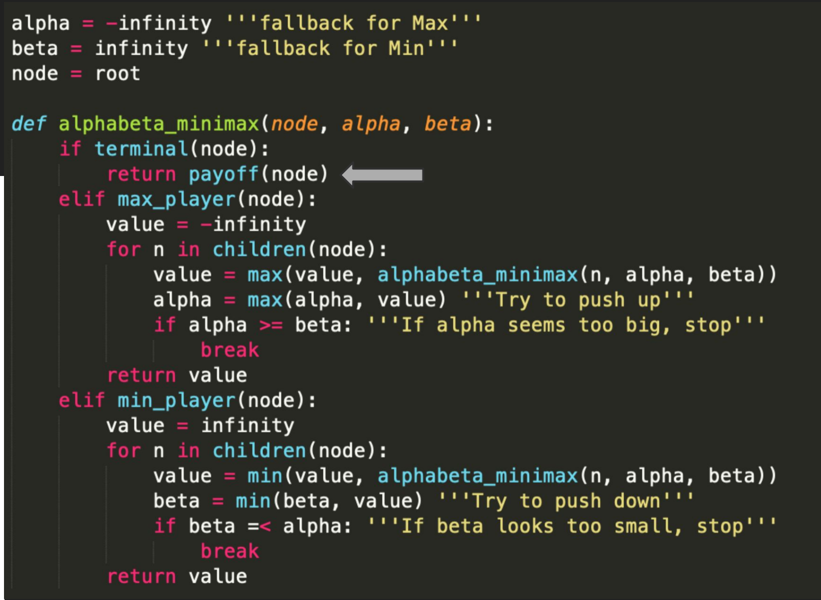


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

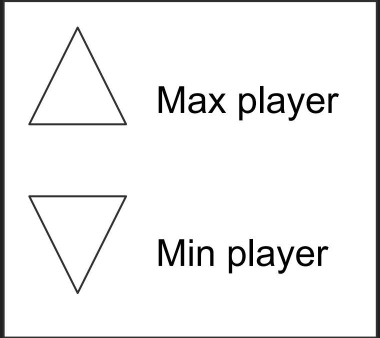


Example of alpha-beta minimax

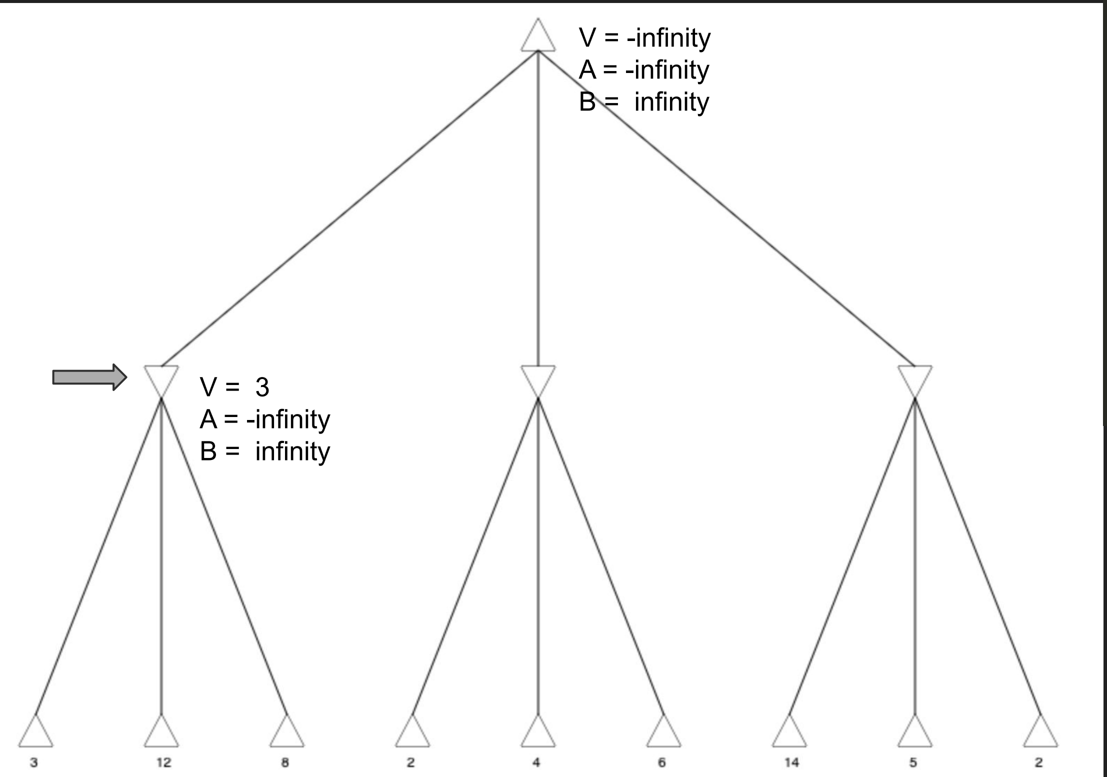




```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

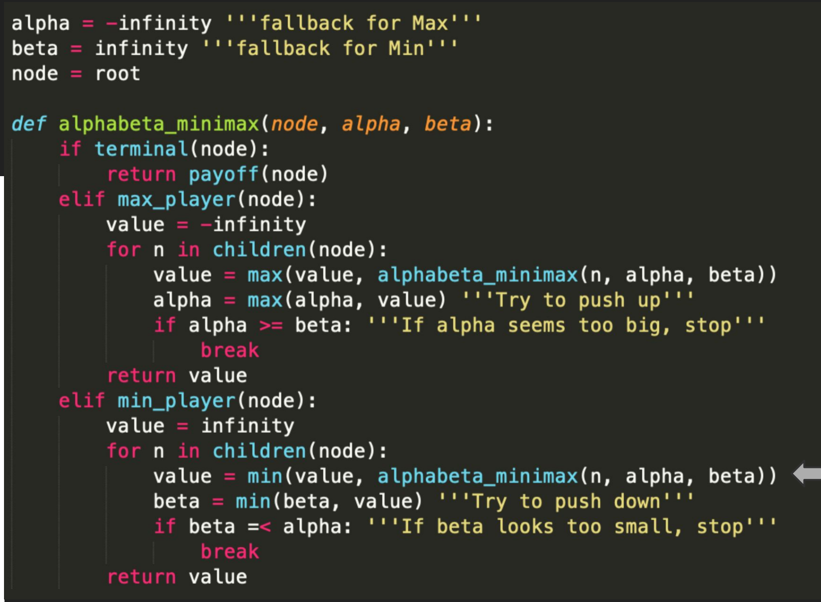


Example of alpha-beta minimax

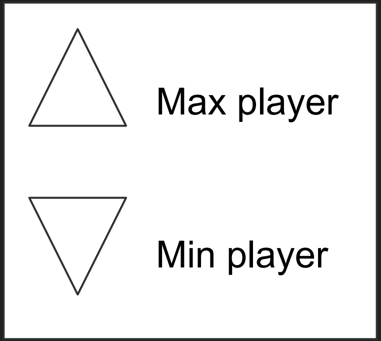


V = -infinity
A = -infinity
B = infinity

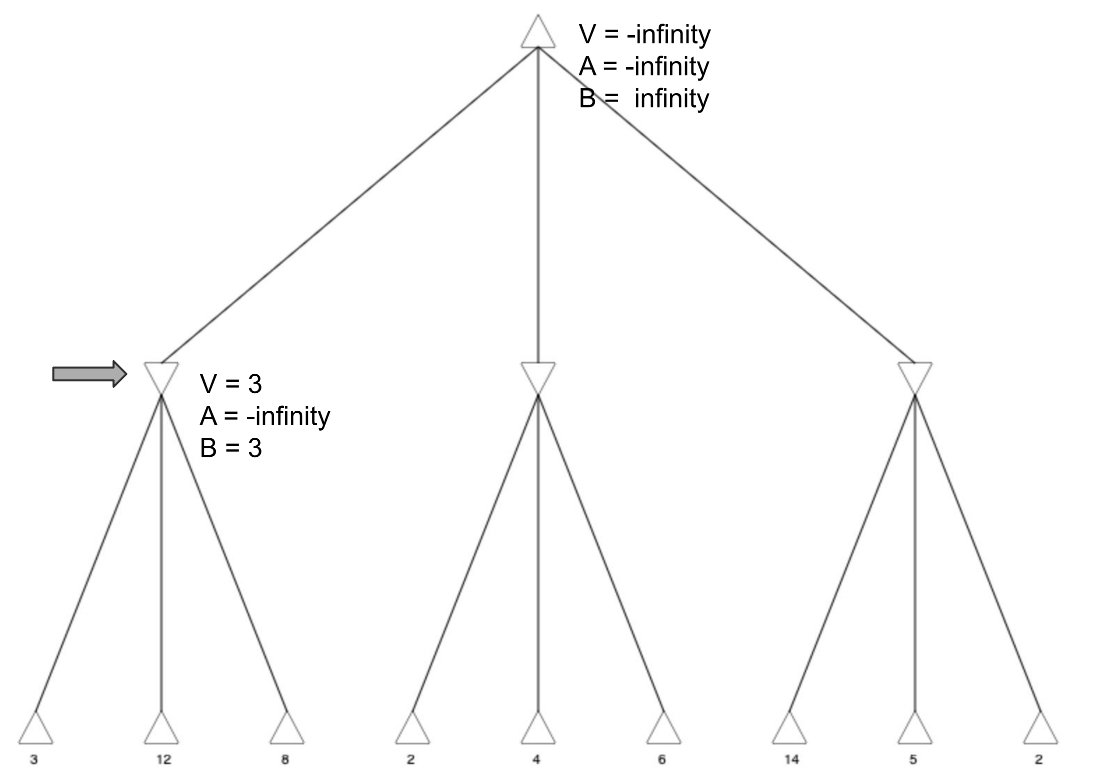
V = 3
A = -infinity
B = infinity

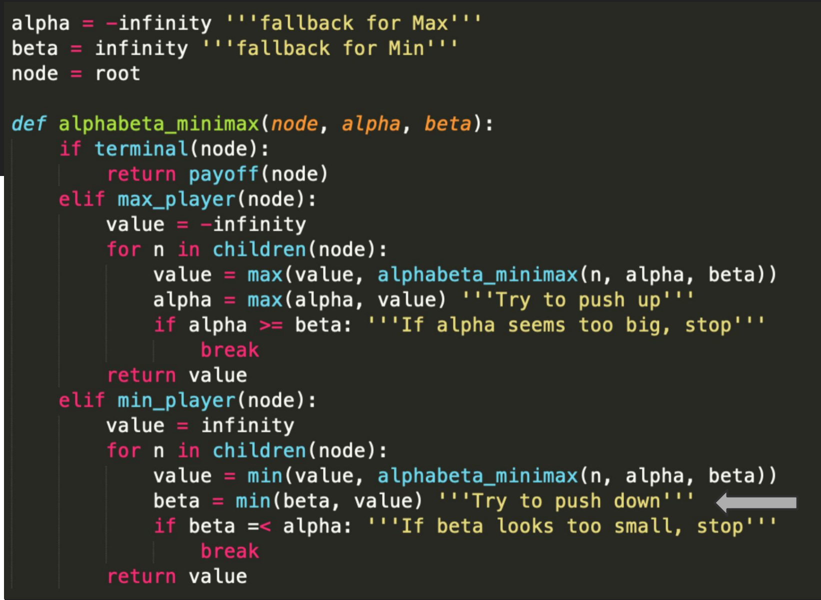


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

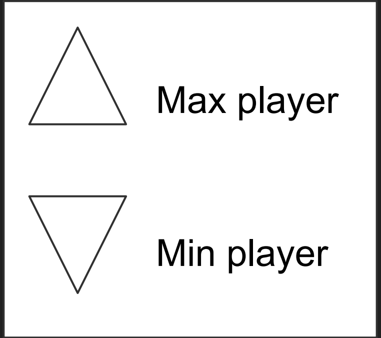


Example of alpha-beta minimax

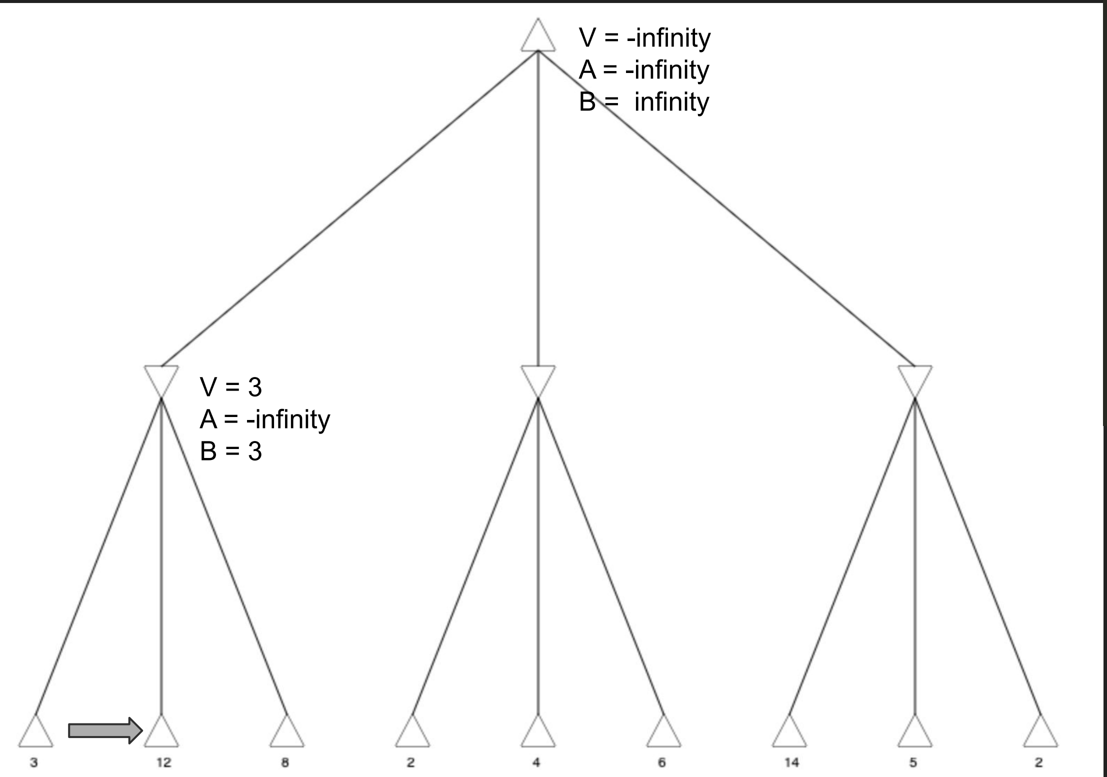


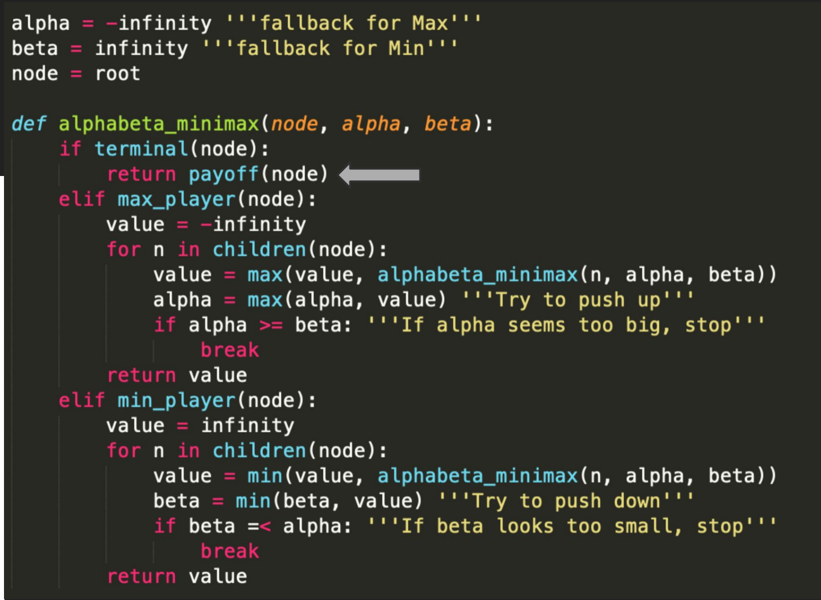


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

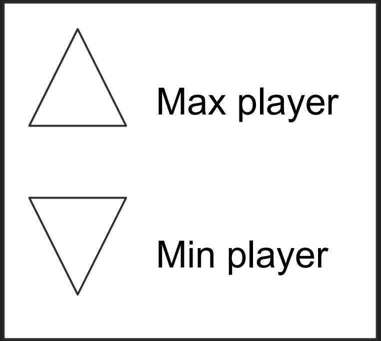


Example of alpha-beta minimax

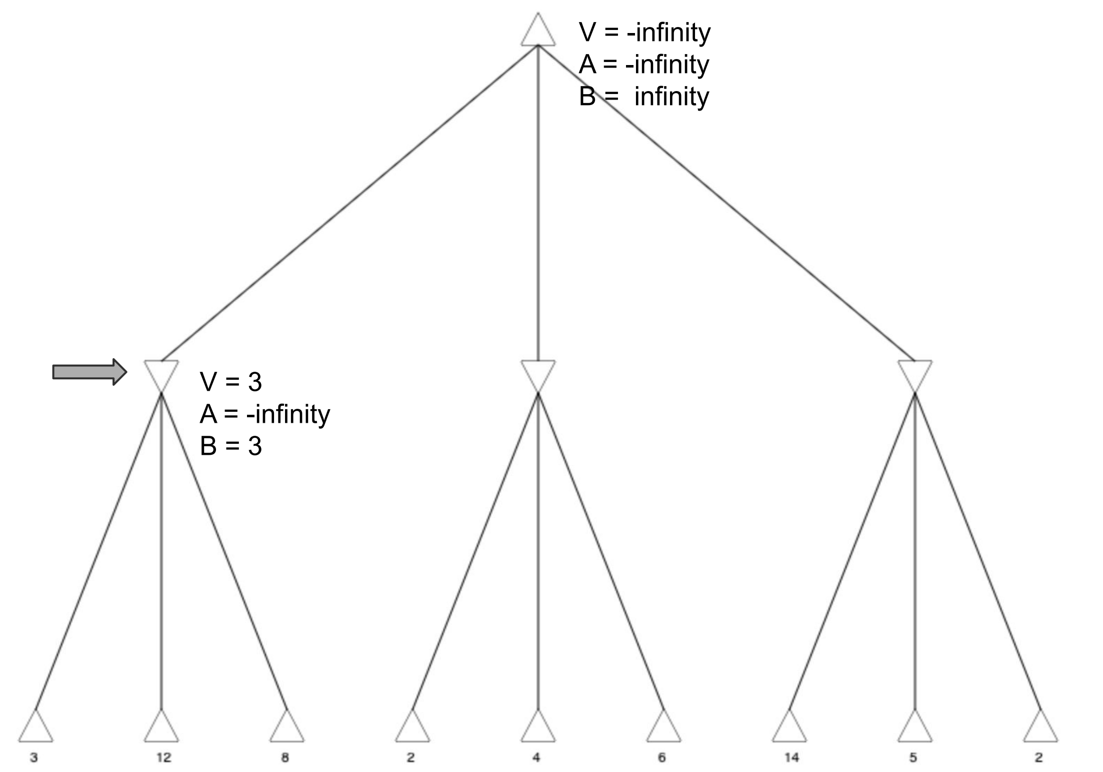


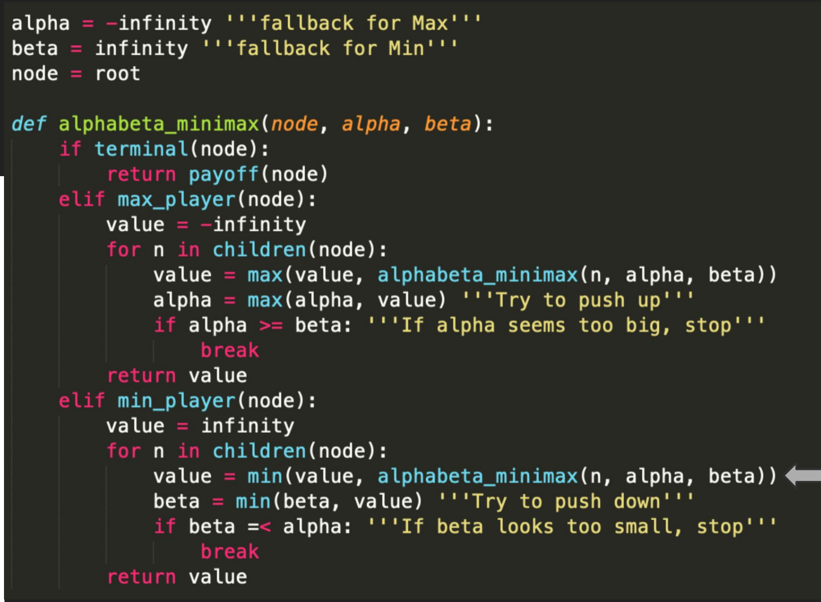


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

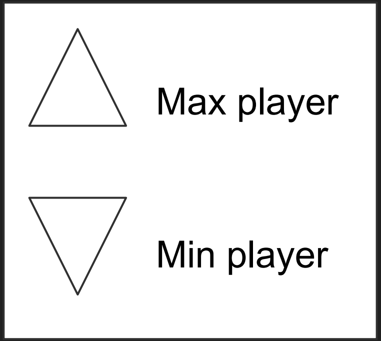


Example of alpha-beta minimax

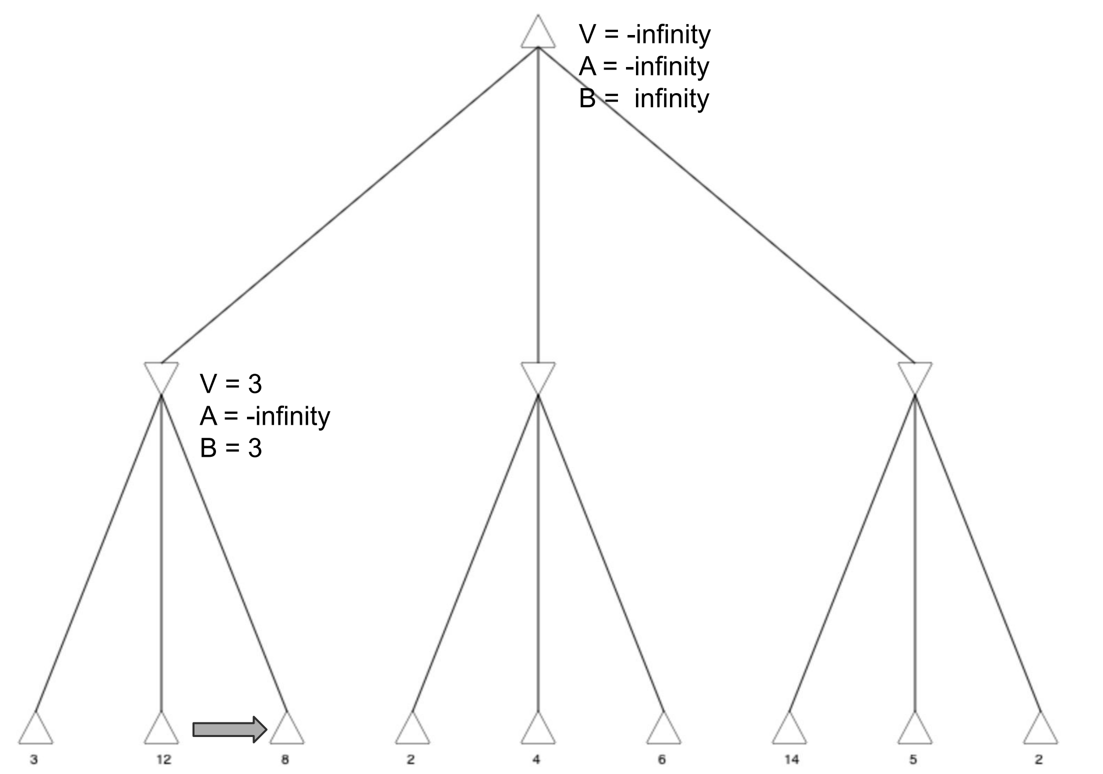




```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

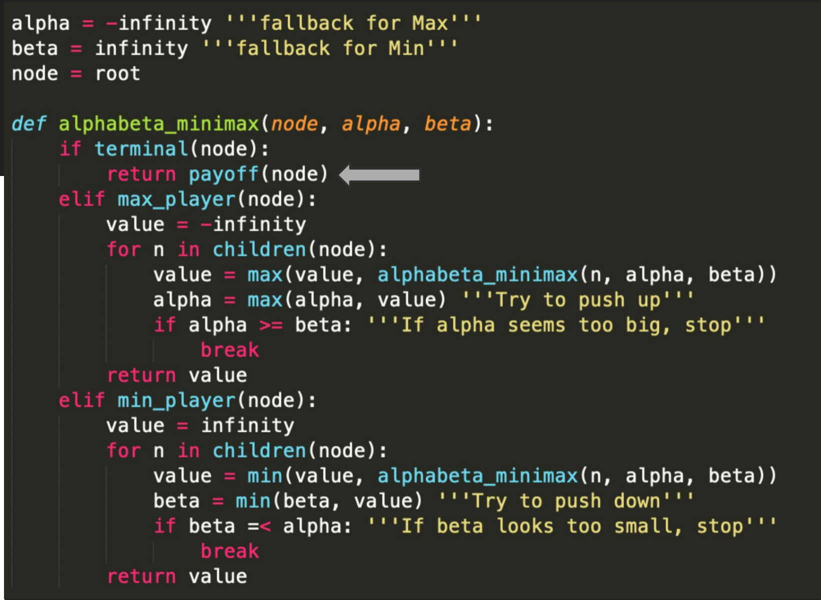


Example of alpha-beta minimax

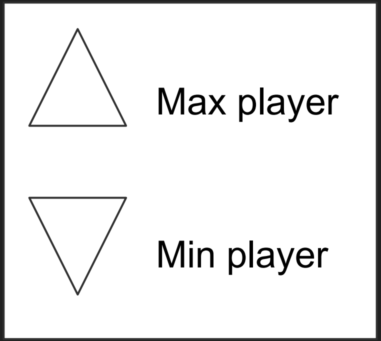


V = -infinity
A = -infinity
B = infinity

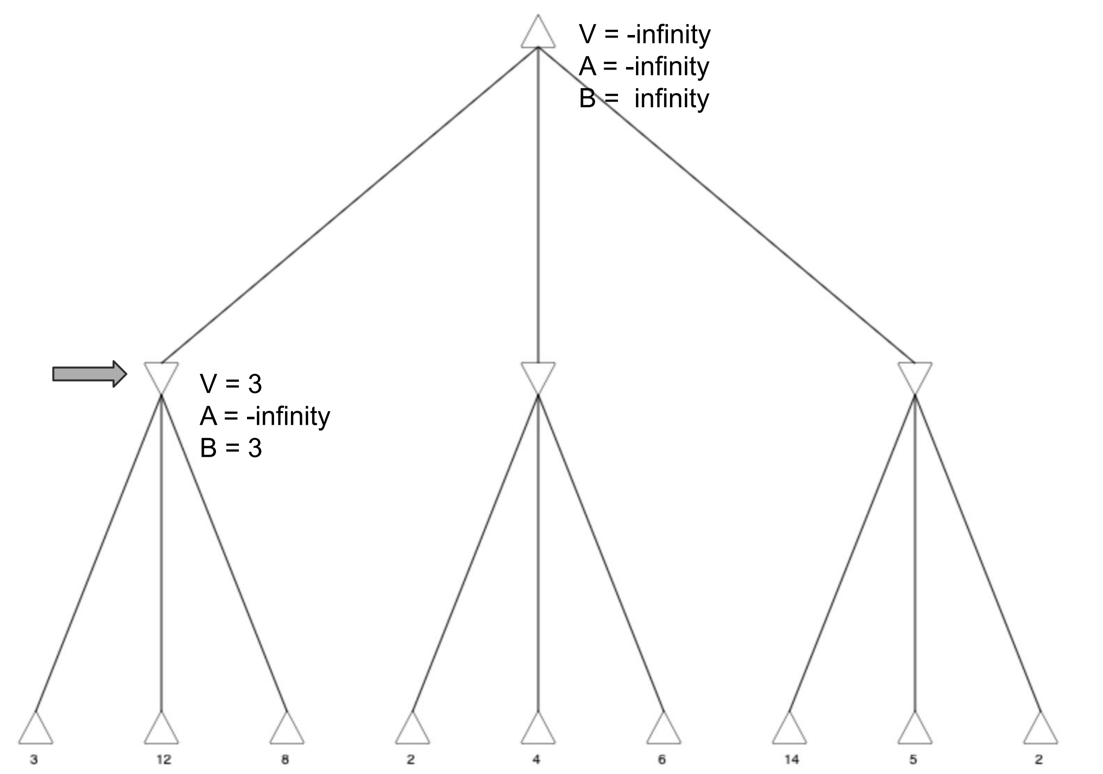
V = 3
A = -infinity
B = 3

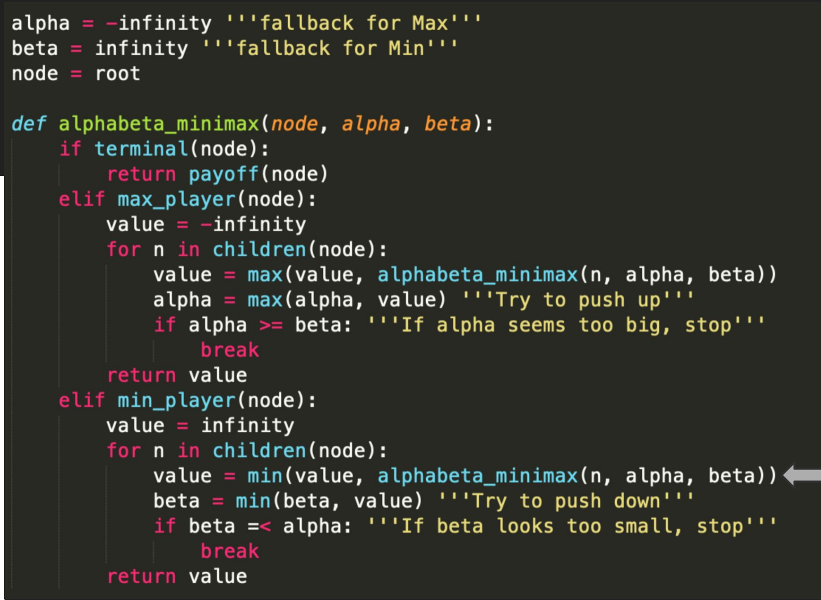


```python
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

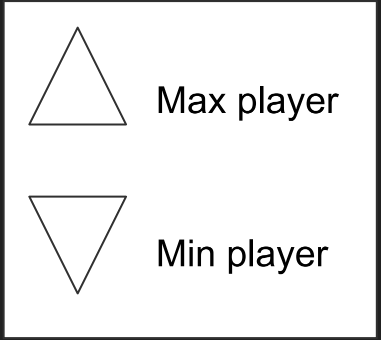


Example of alpha-beta minimax

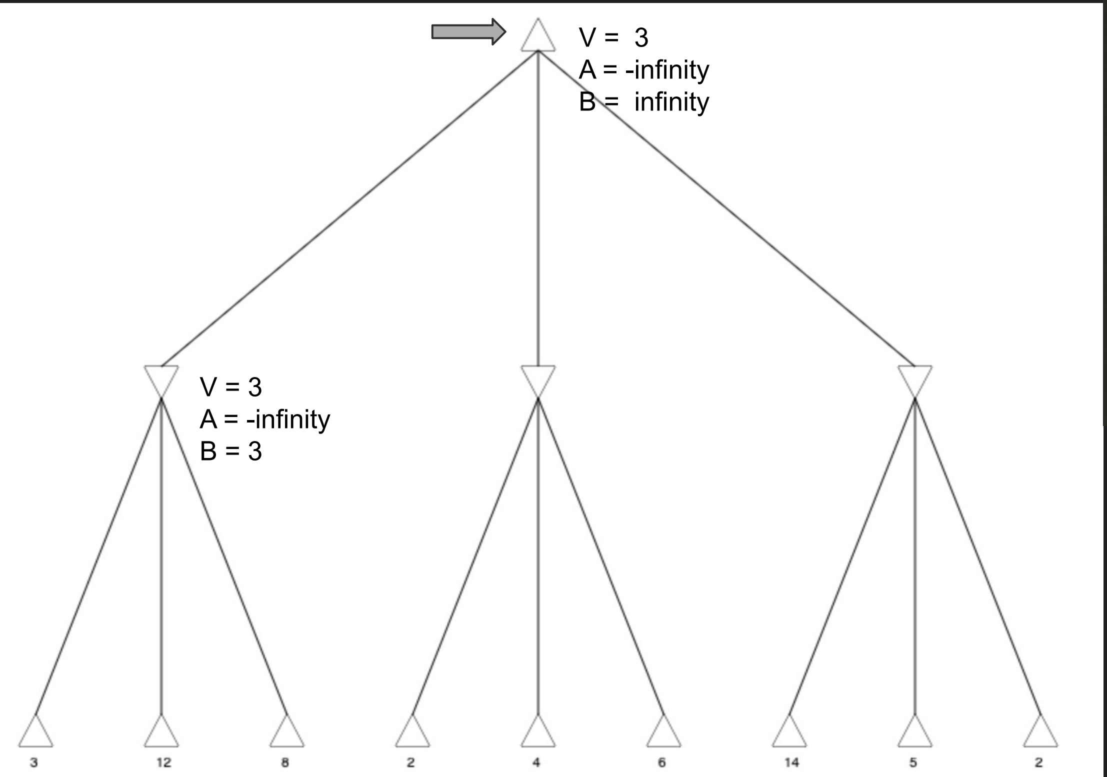


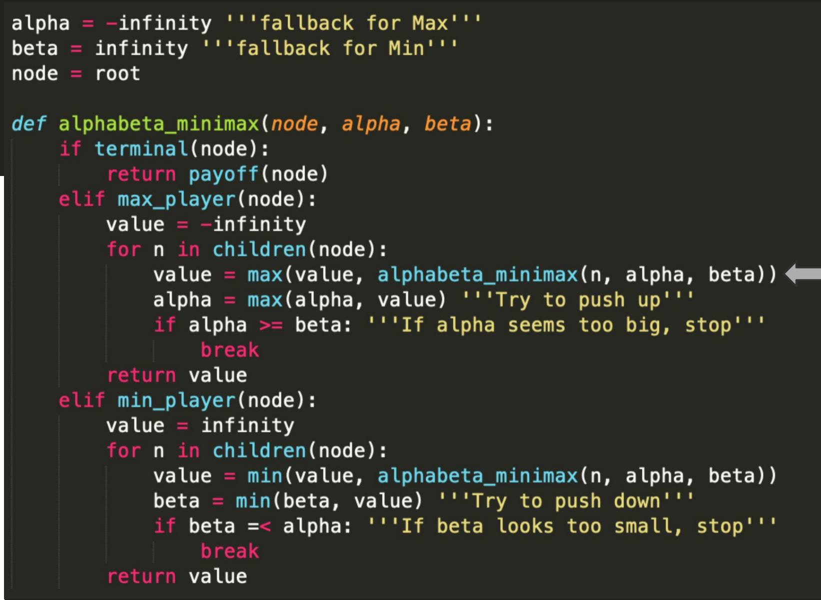


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

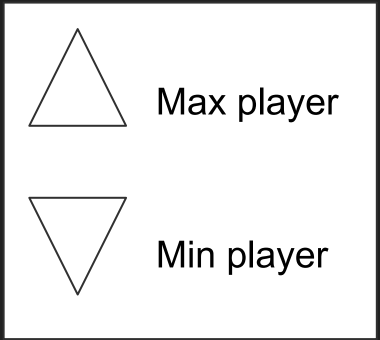


Example of alpha-beta minimax

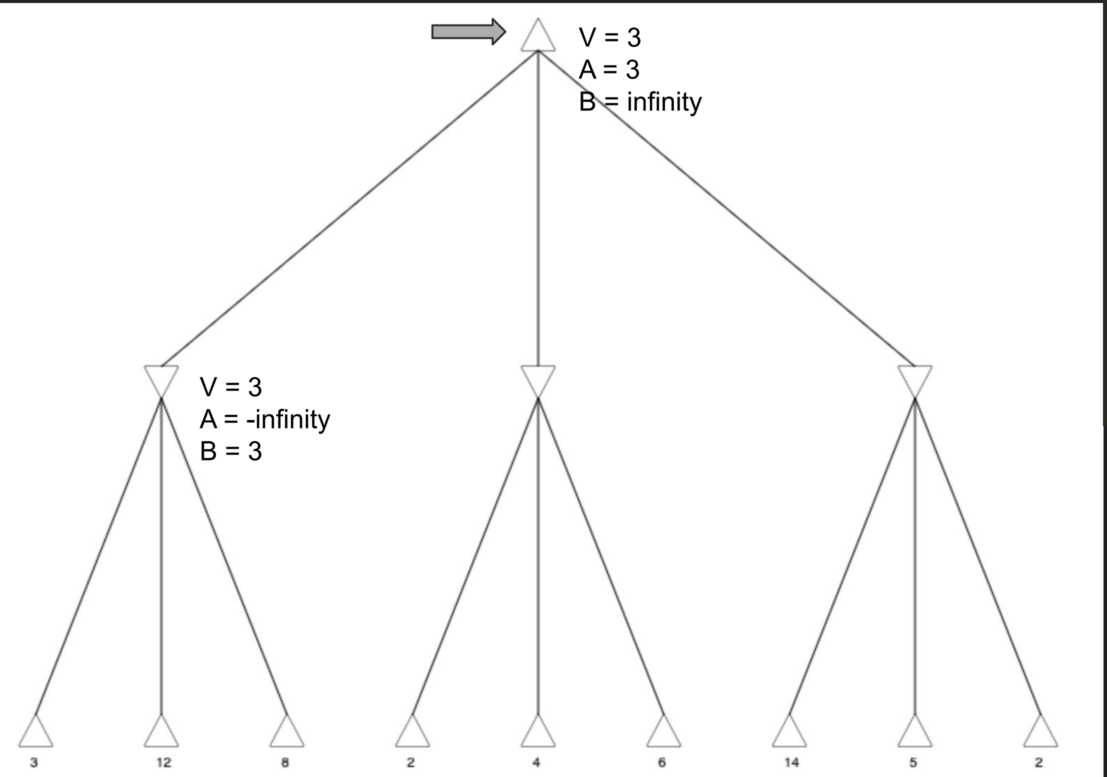


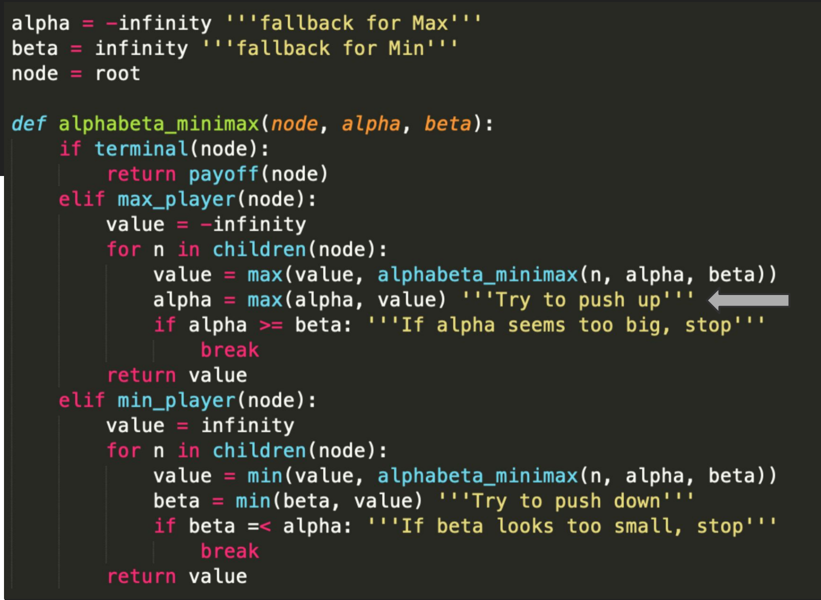


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

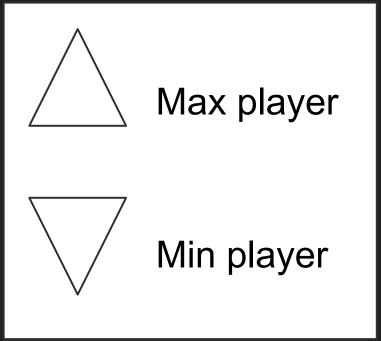


Example of alpha-beta minimax

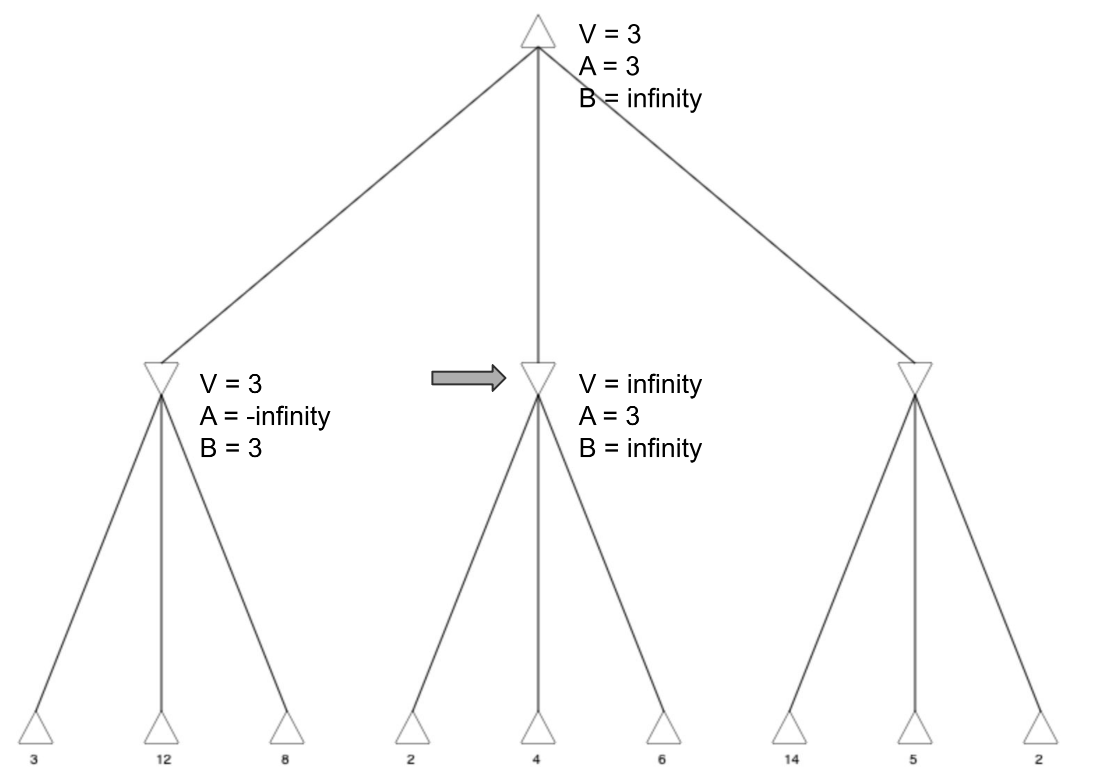


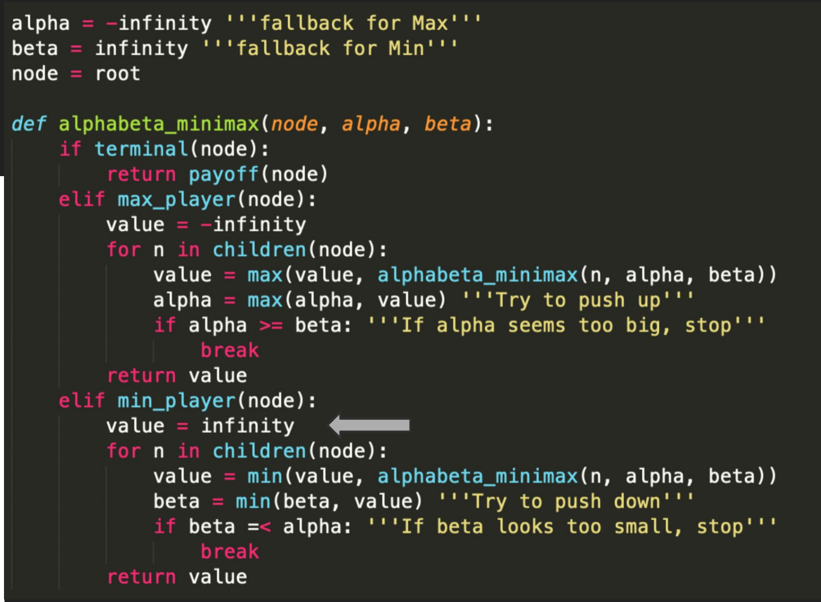


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

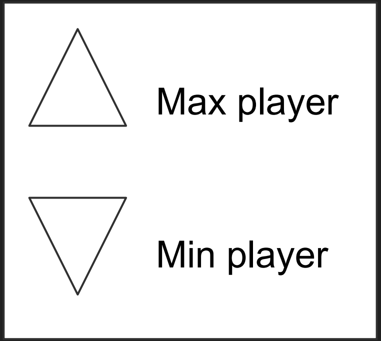


Example of alpha-beta minimax

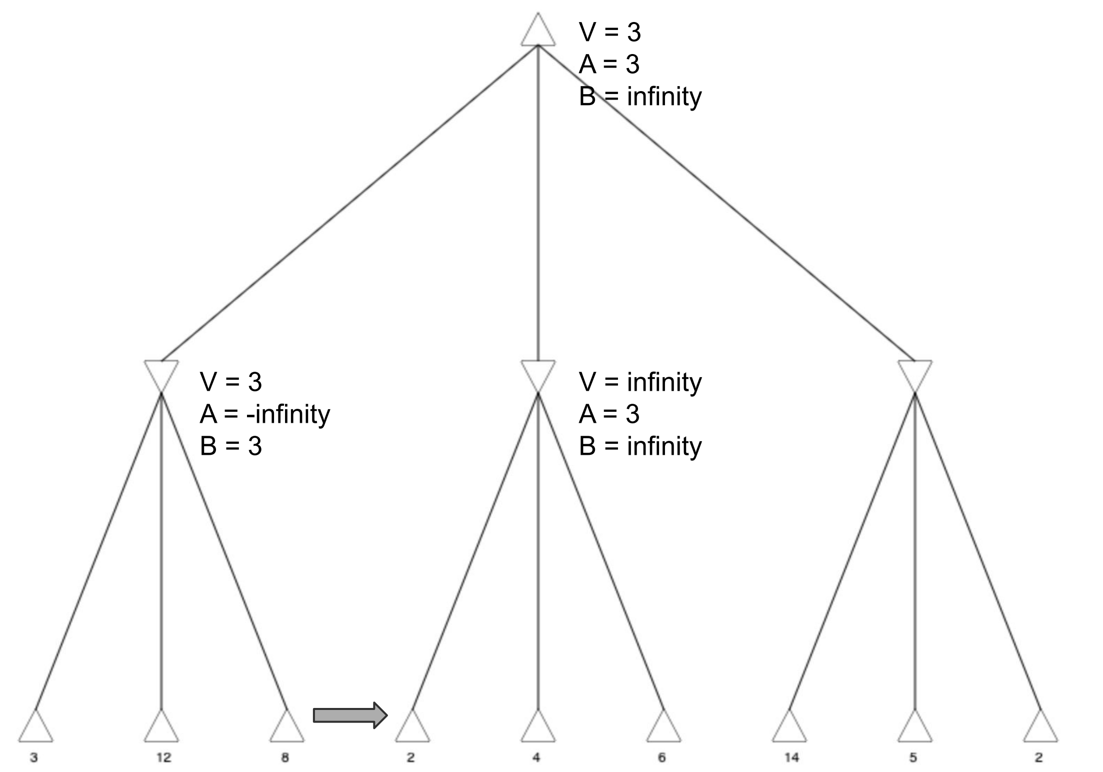


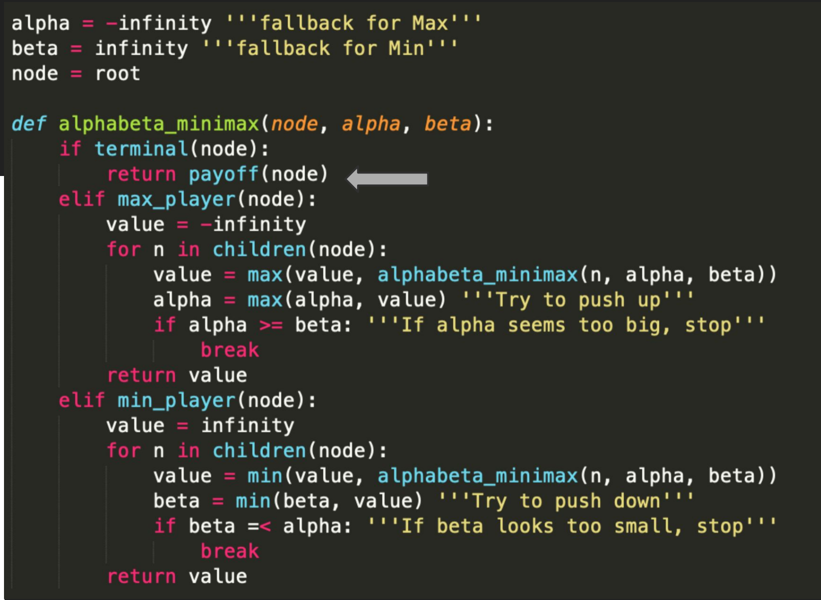


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

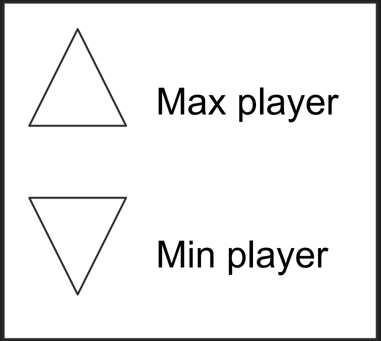


Example of alpha-beta minimax

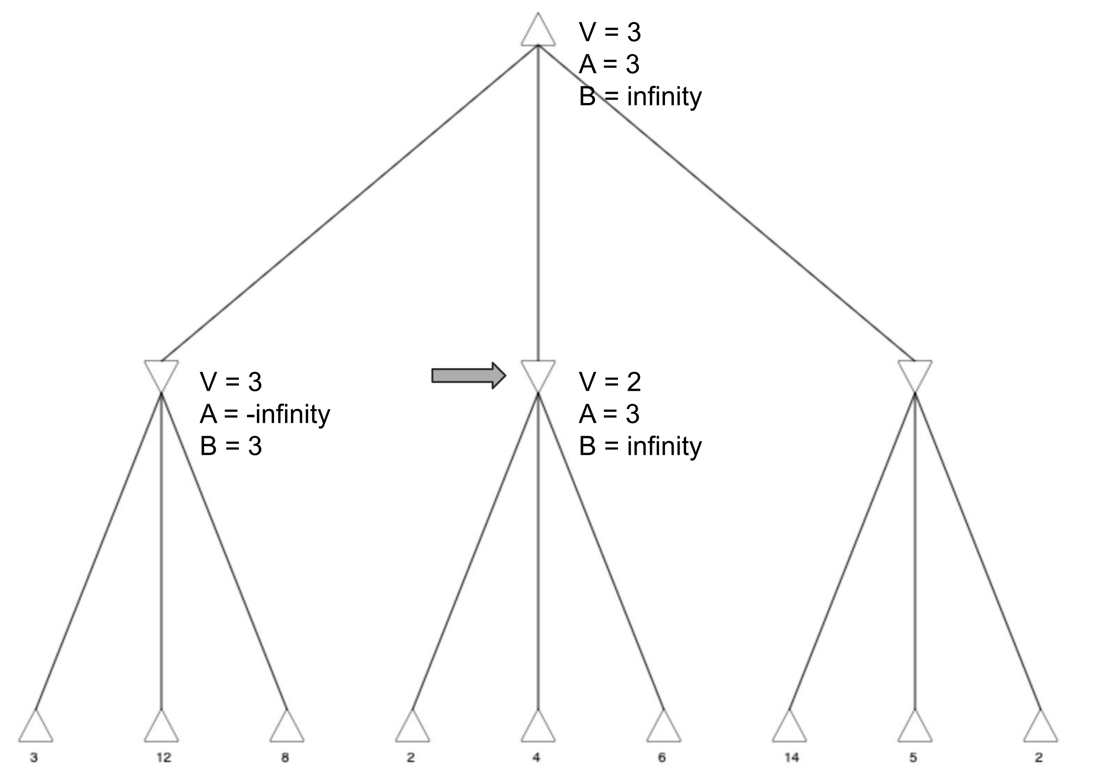




```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```



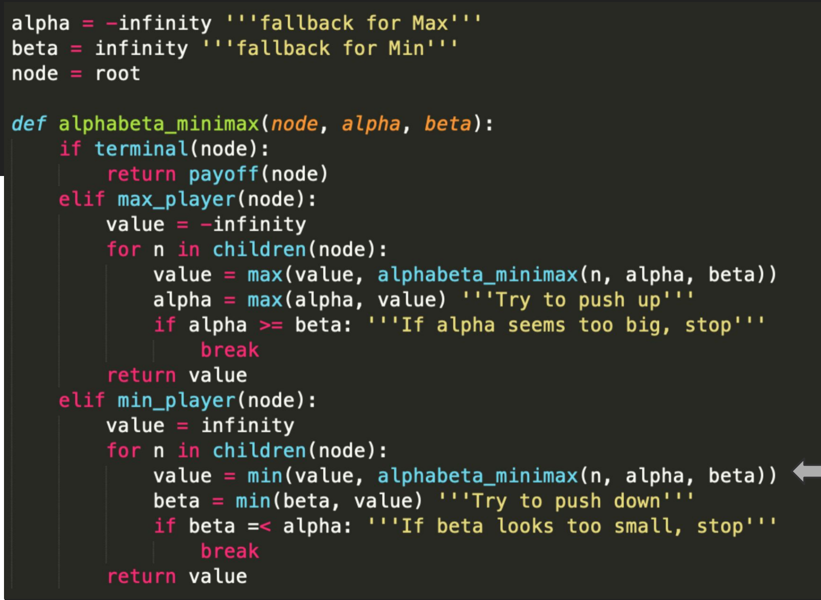
Example of alpha-beta minimax



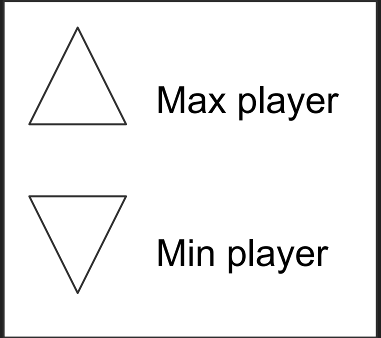
V = 3
A = 3
B = infinity

V = 3
A = -infinity
B = 3

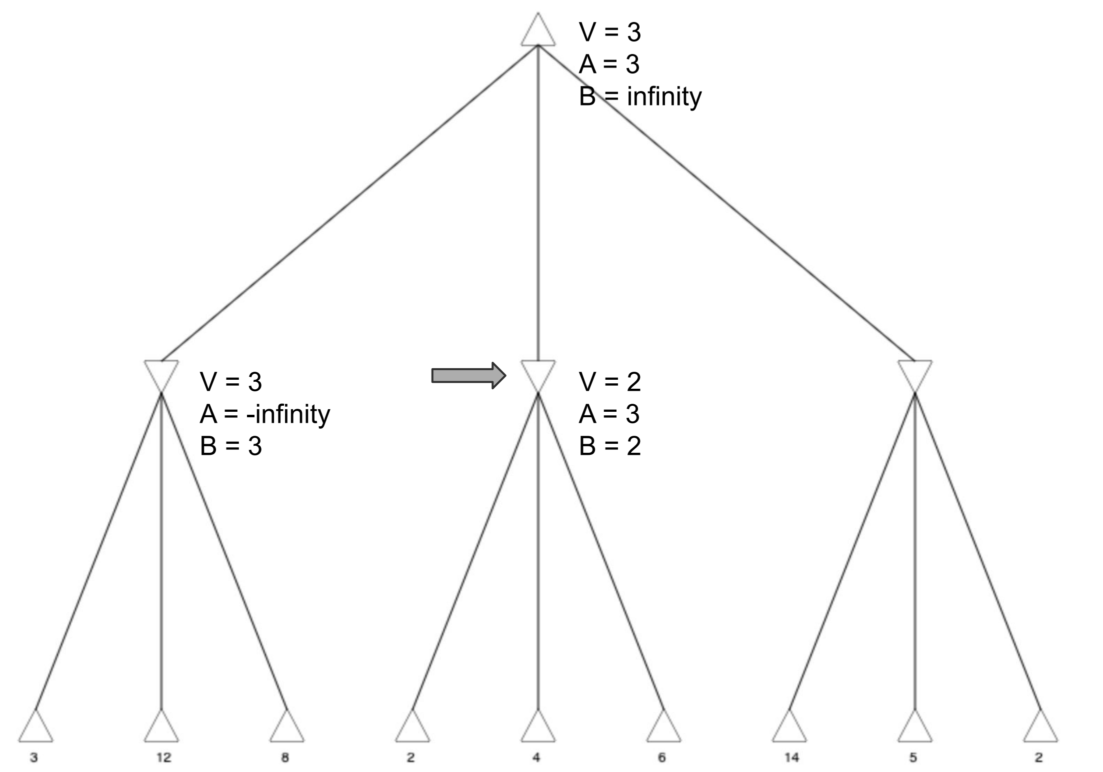
V = 2
A = 3
B = infinity

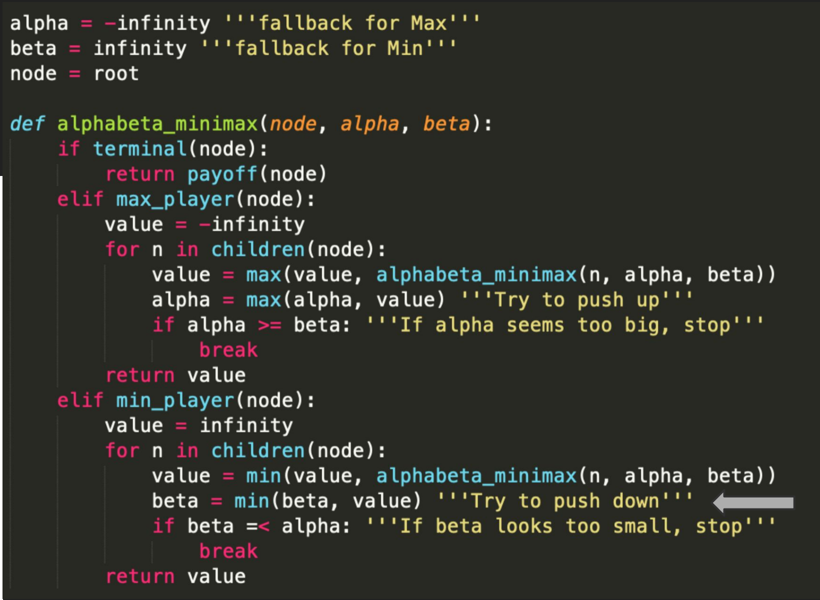


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

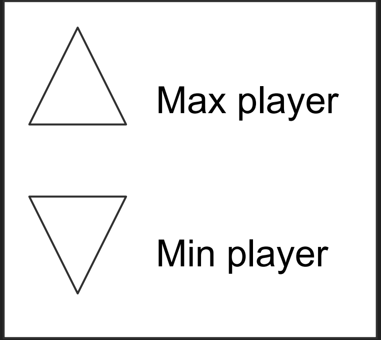


Example of alpha-beta minimax

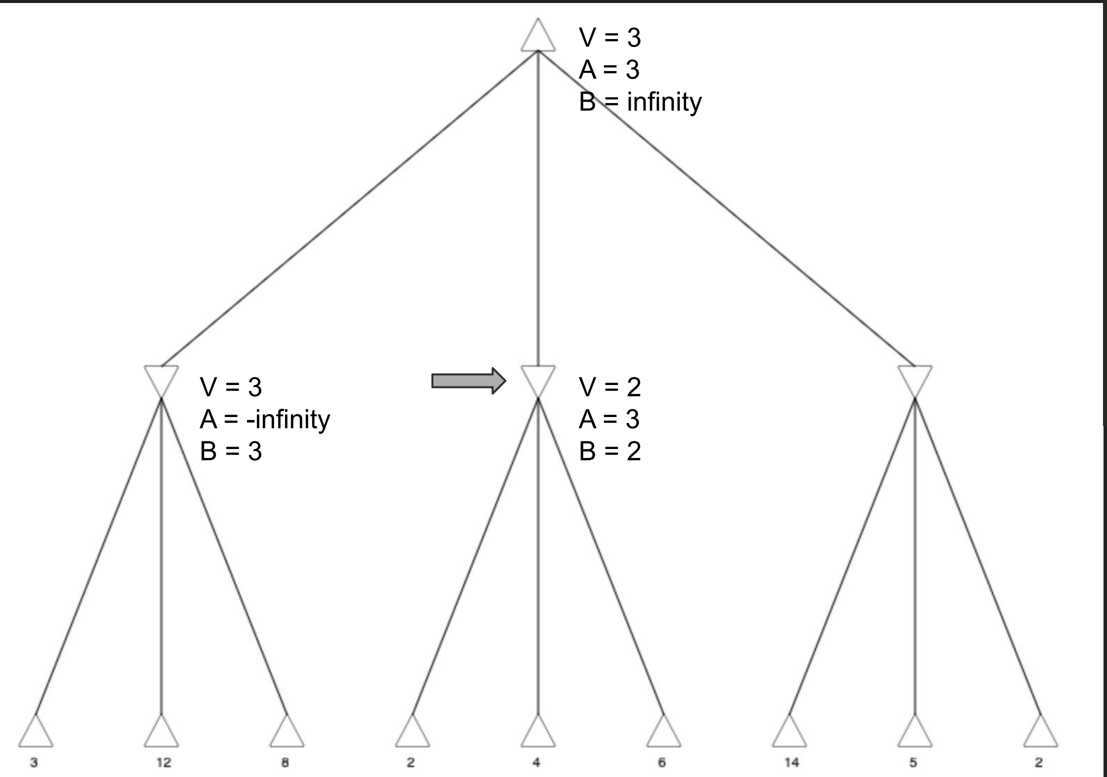


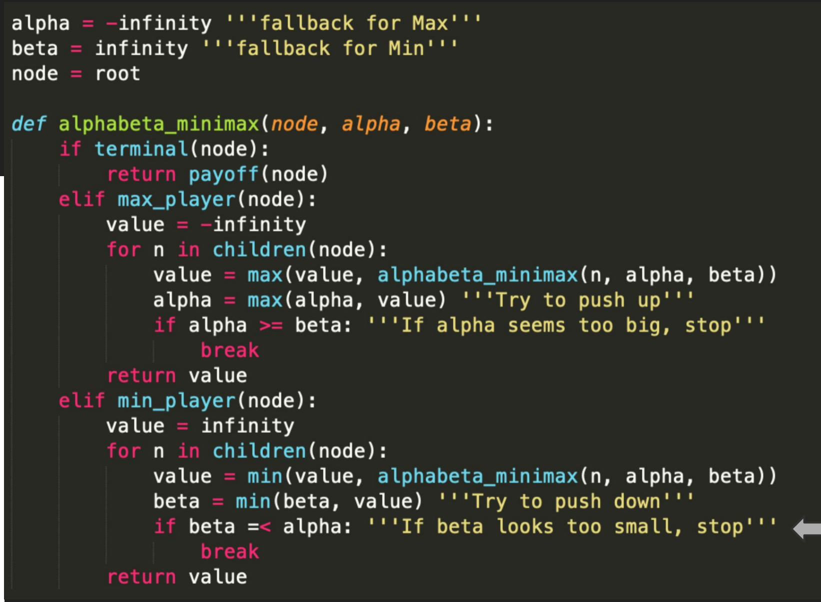


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

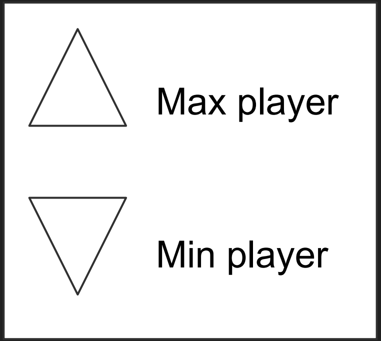


Example of alpha-beta minimax

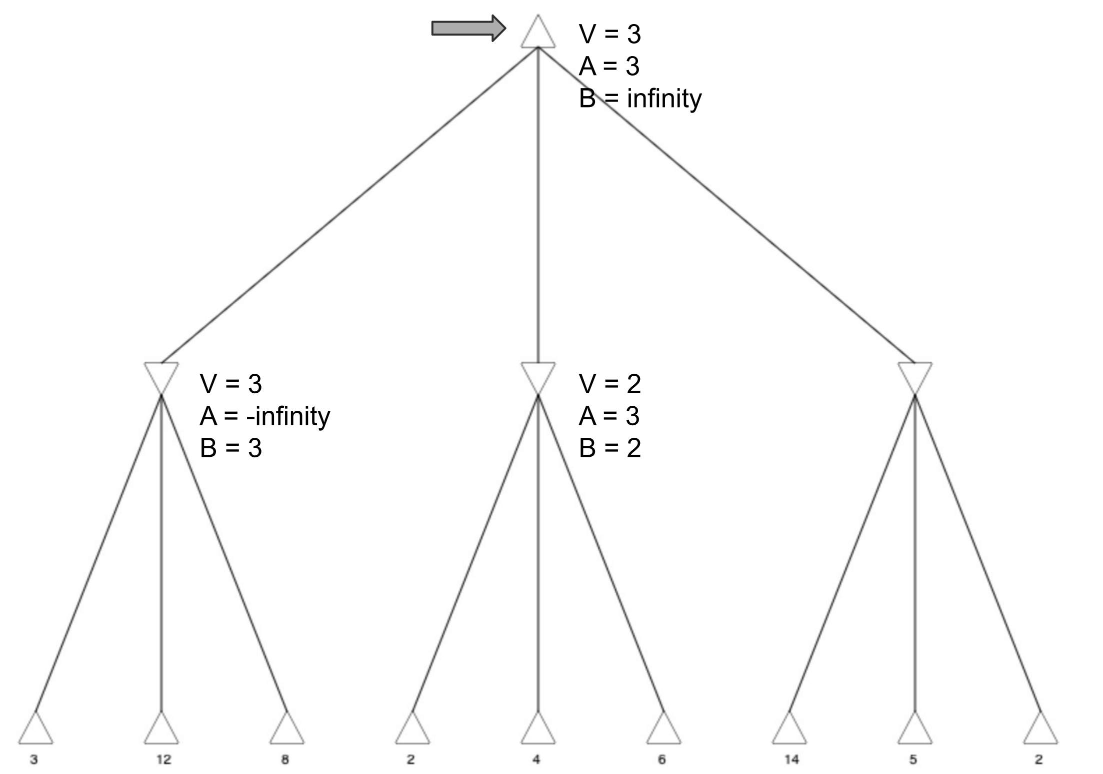


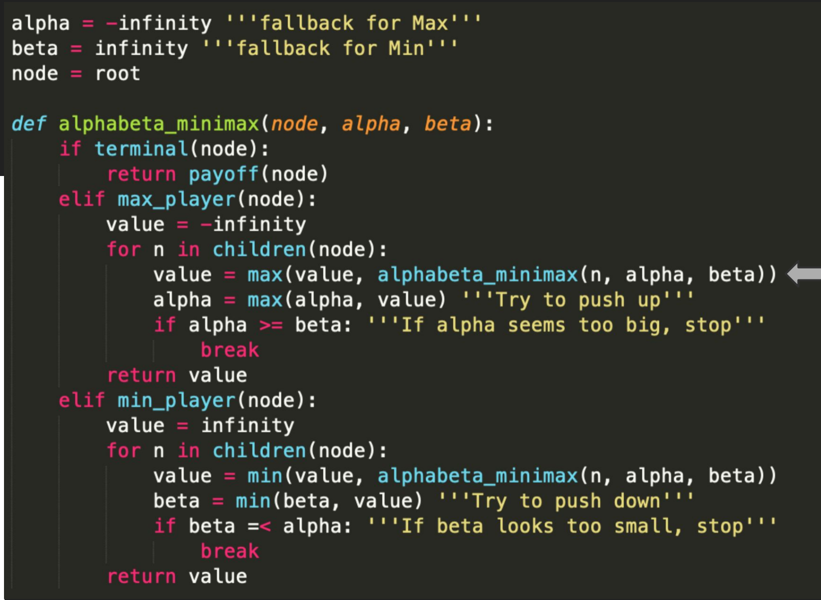


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

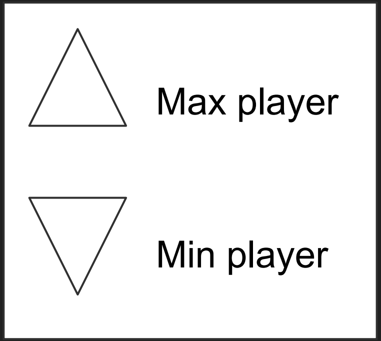


Example of alpha-beta minimax

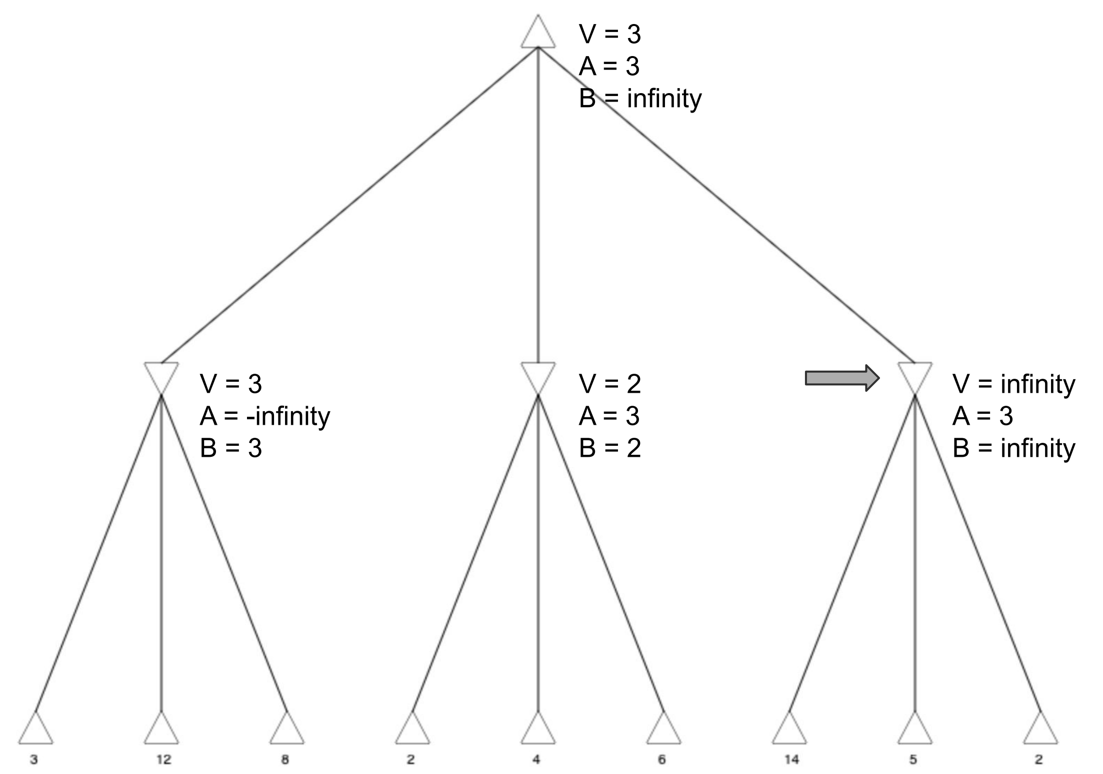


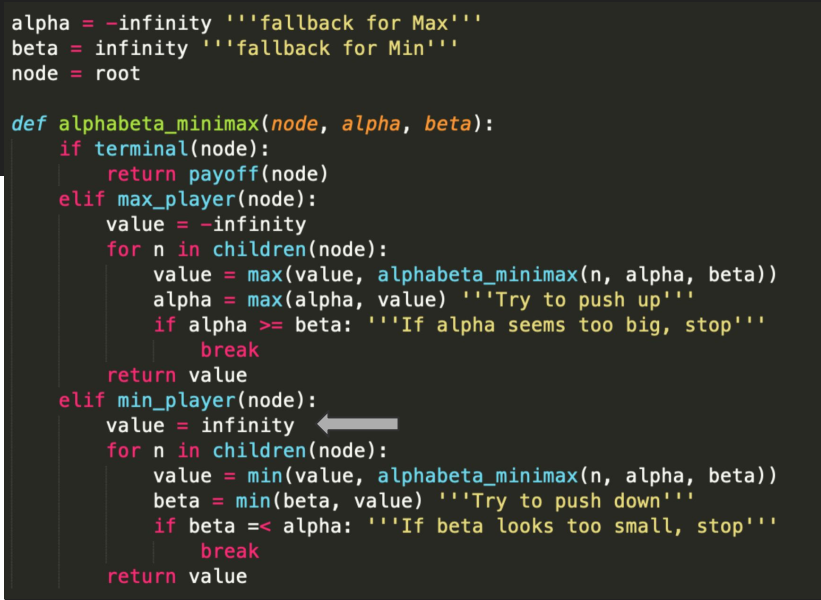


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

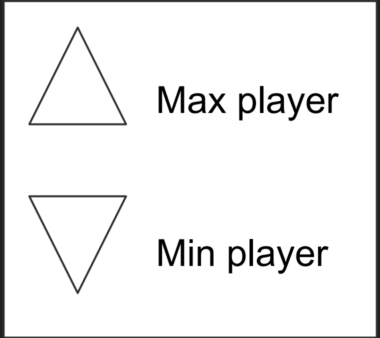


Example of alpha-beta minimax

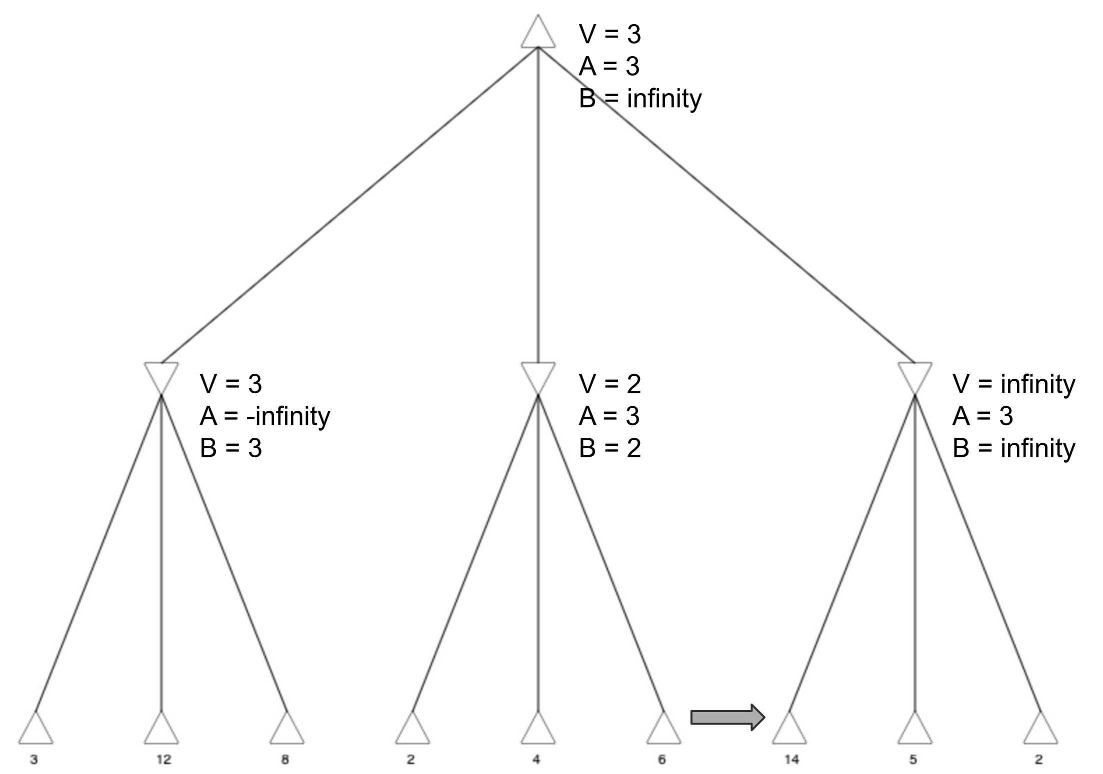


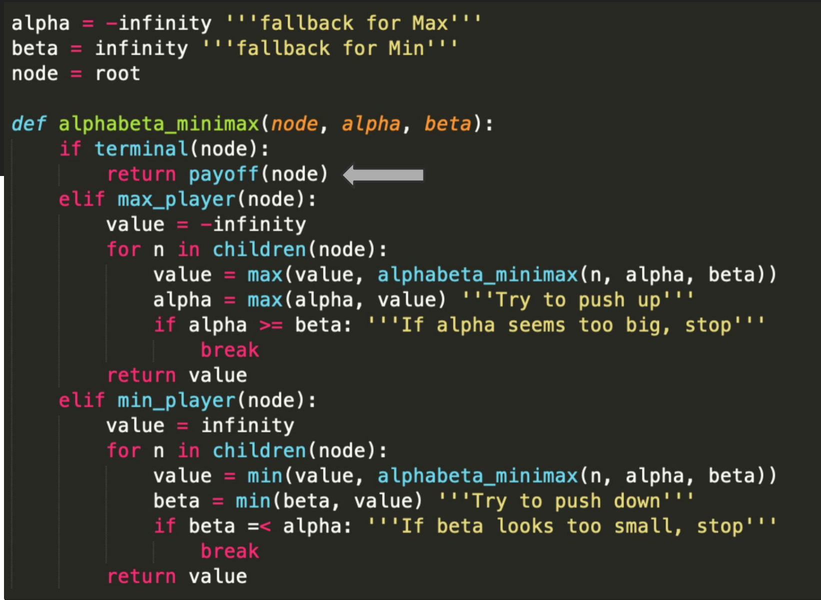


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

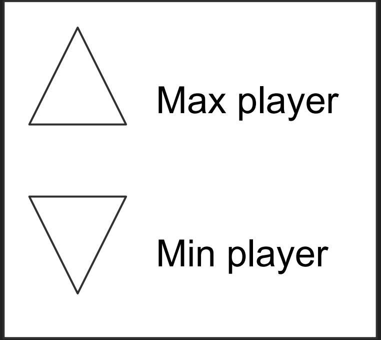


Example of alpha-beta minimax

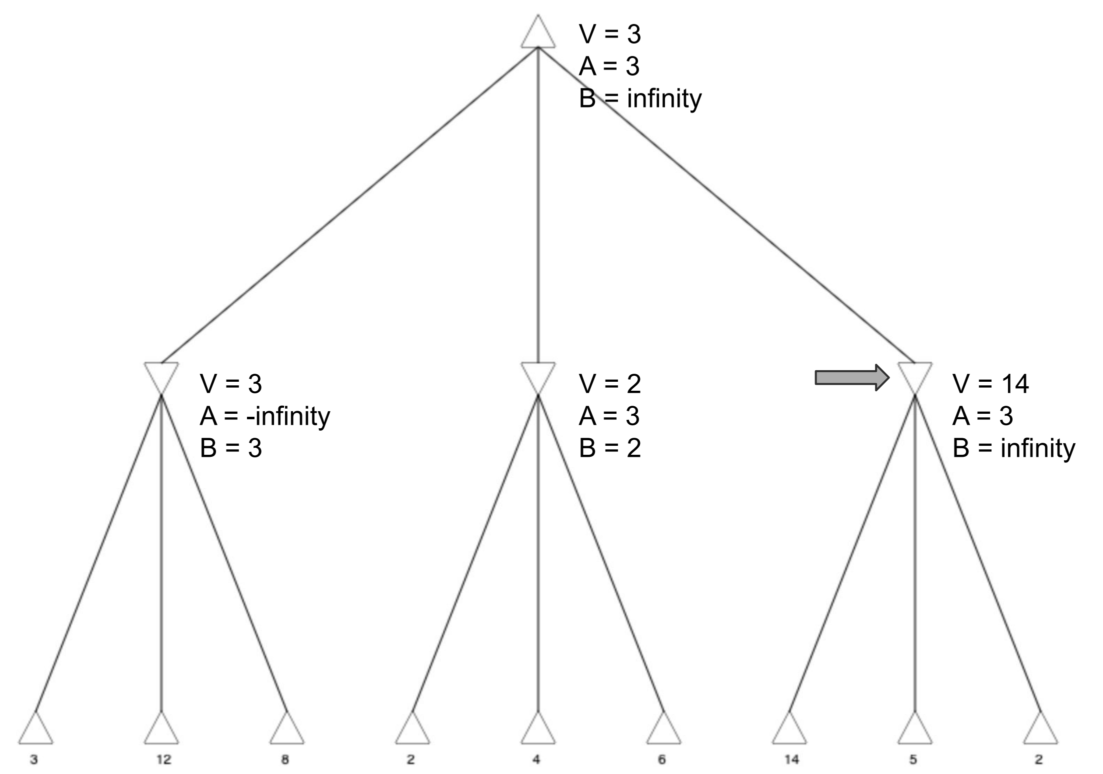


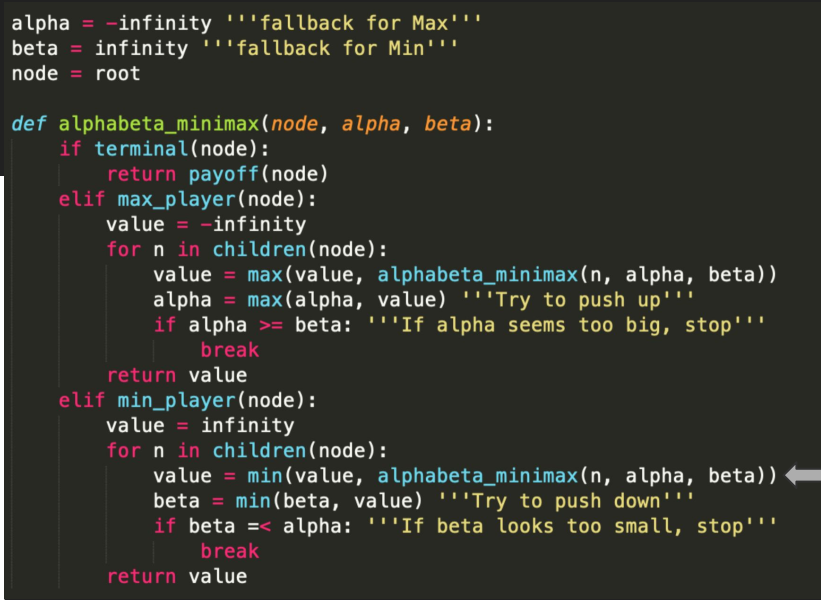


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

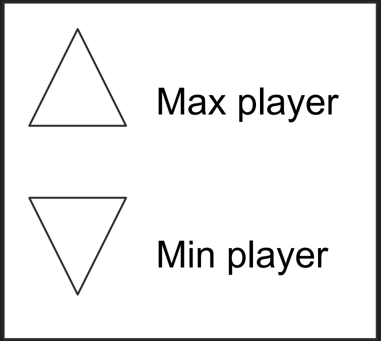


Example of alpha-beta minimax

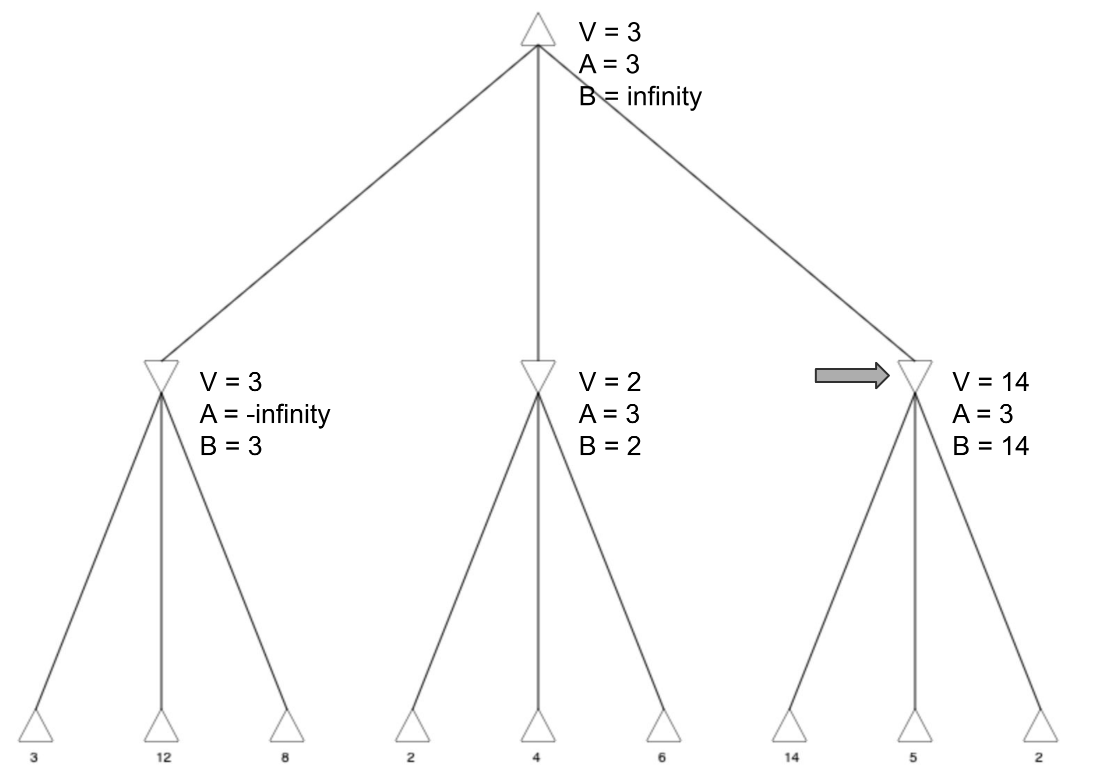


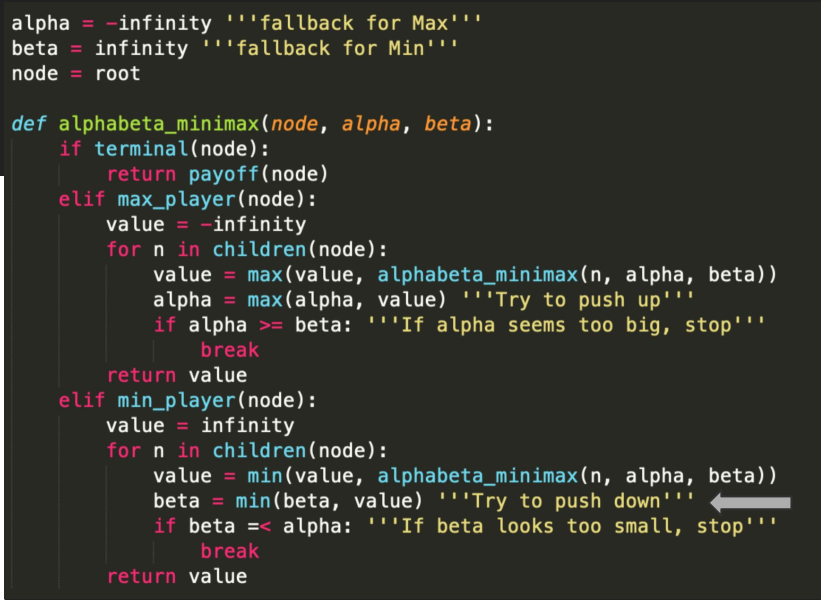


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

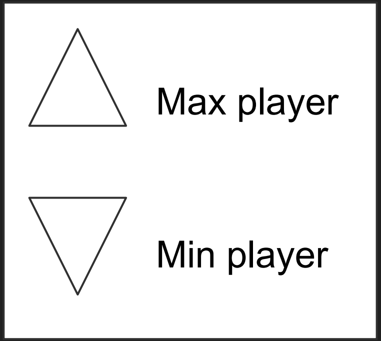


Example of alpha-beta minimax

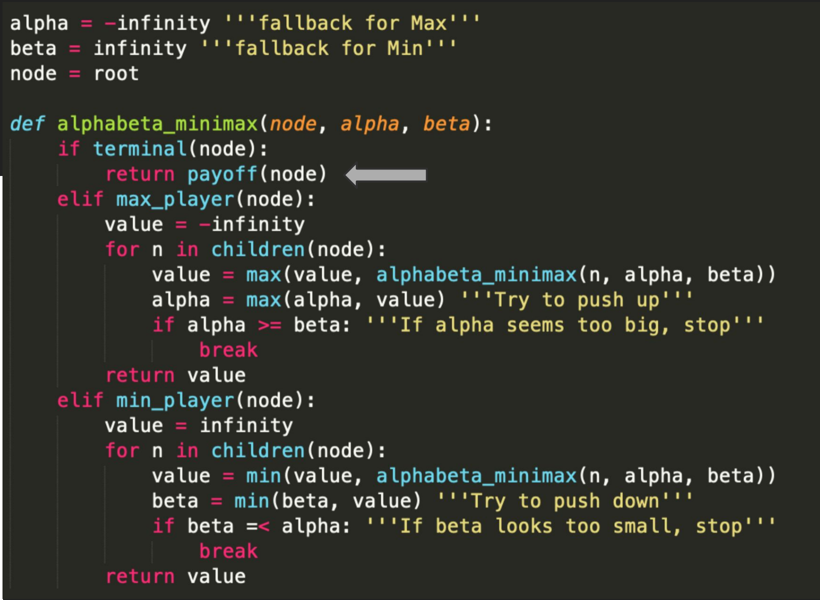




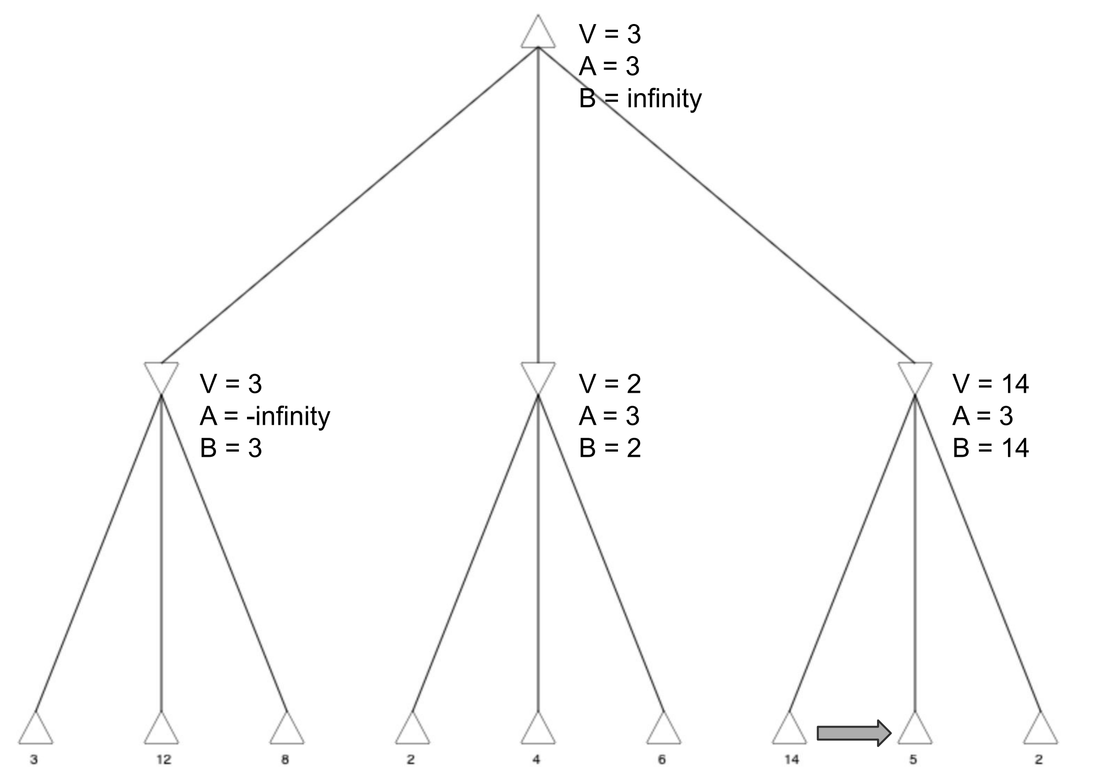
```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

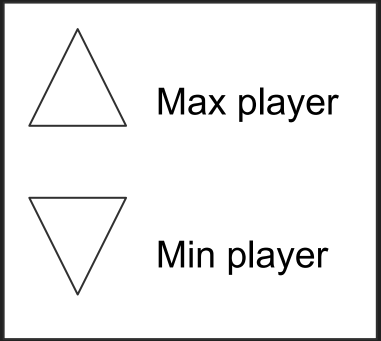


Example of alpha-beta minimax

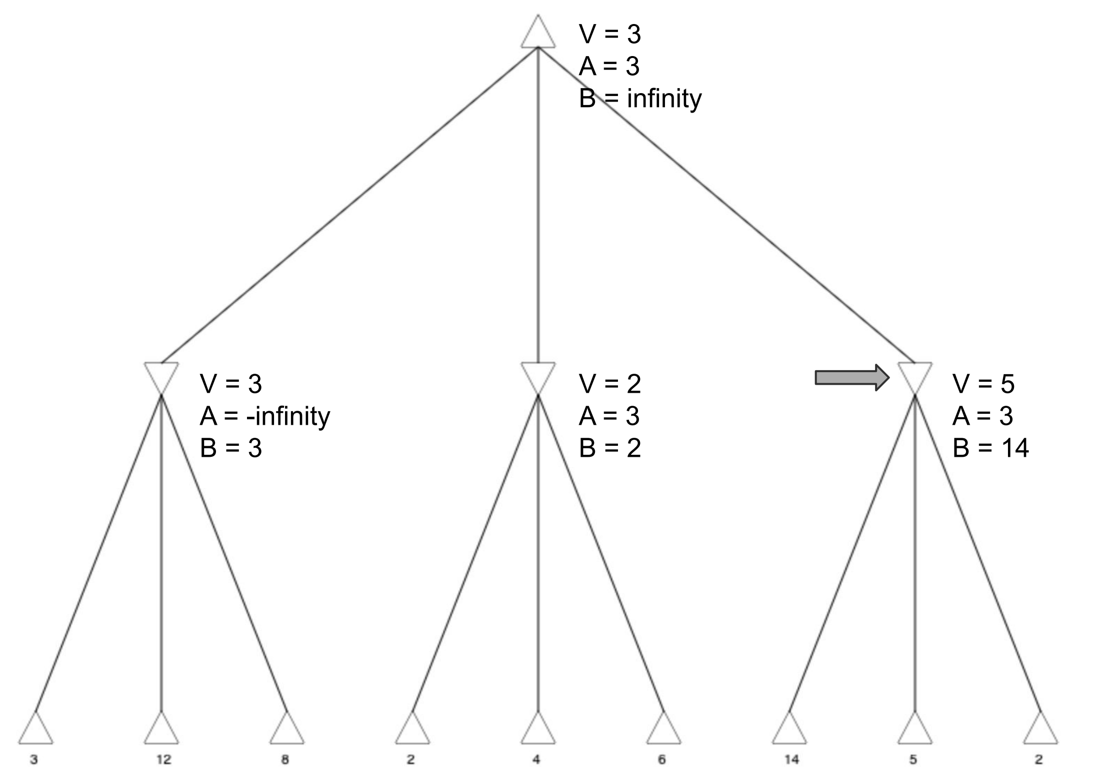


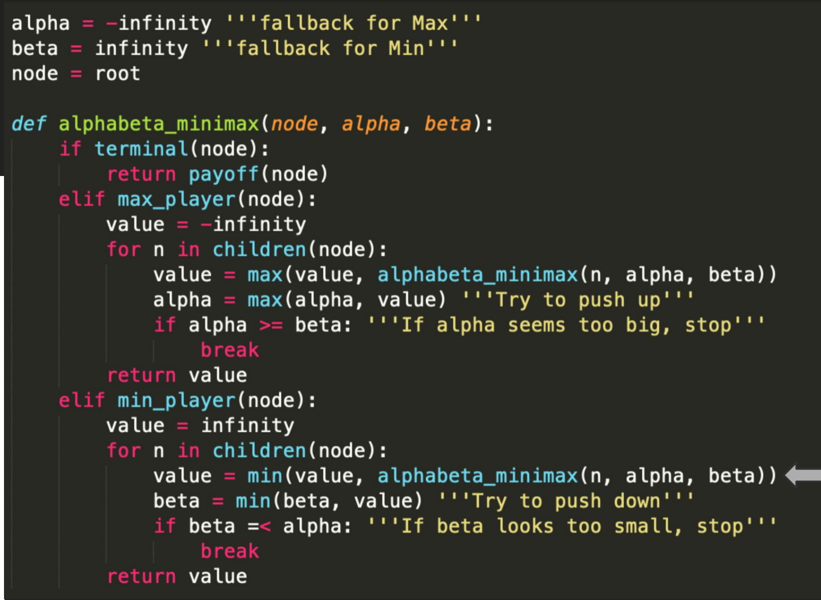
```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```



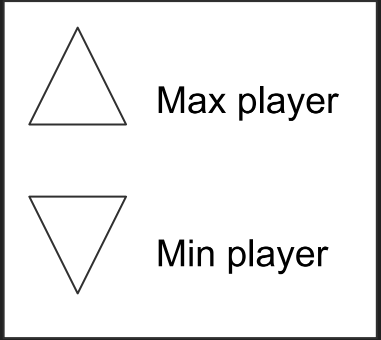


Example of alpha-beta minimax

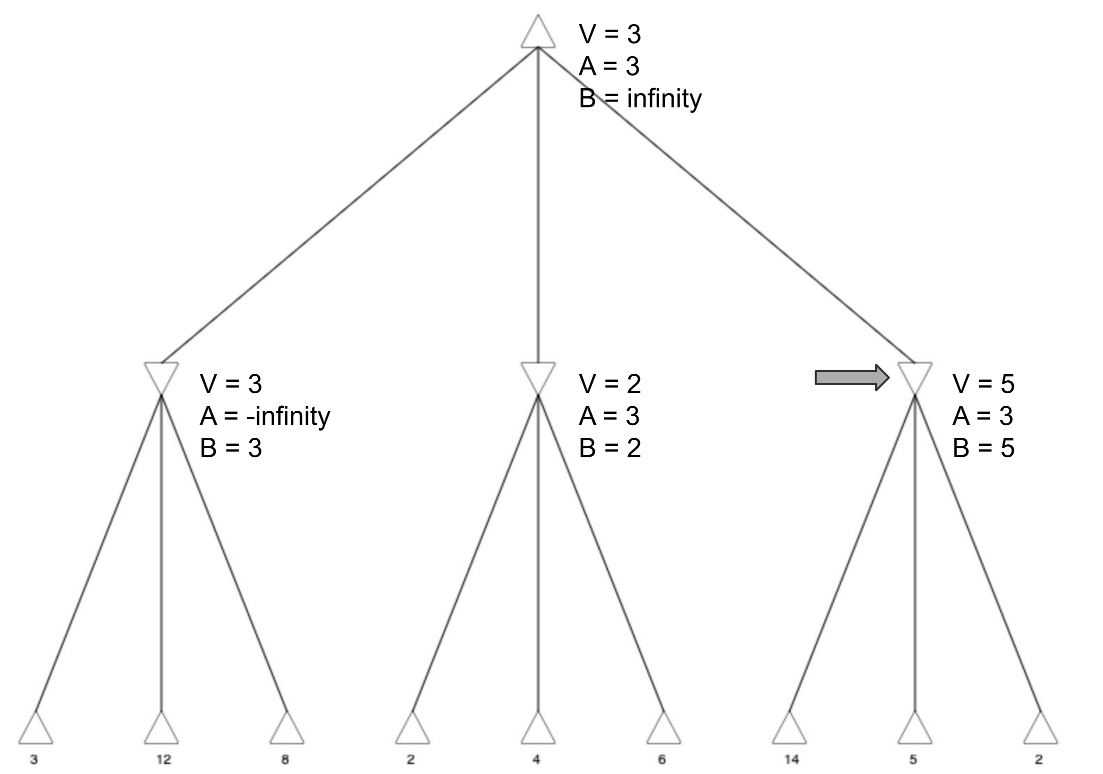


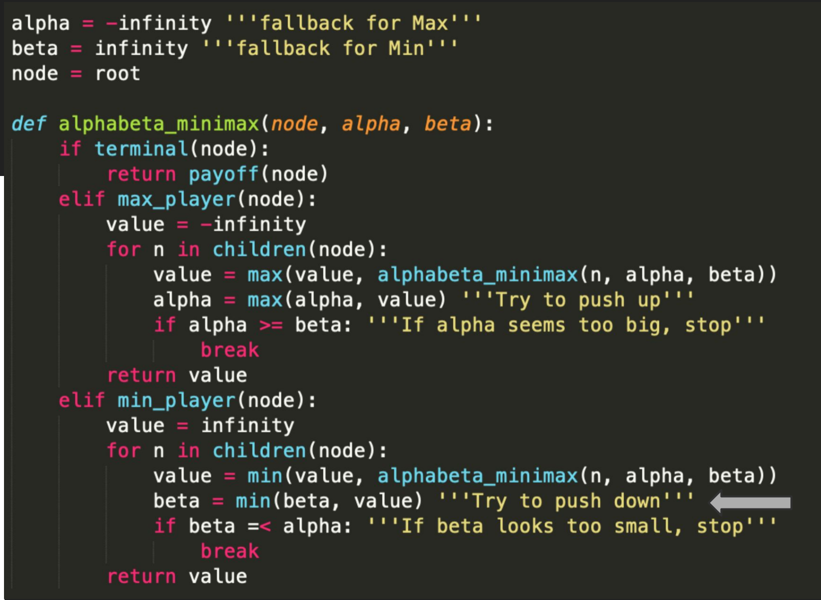


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

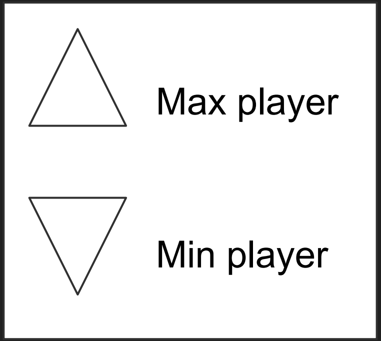


Example of alpha-beta minimax

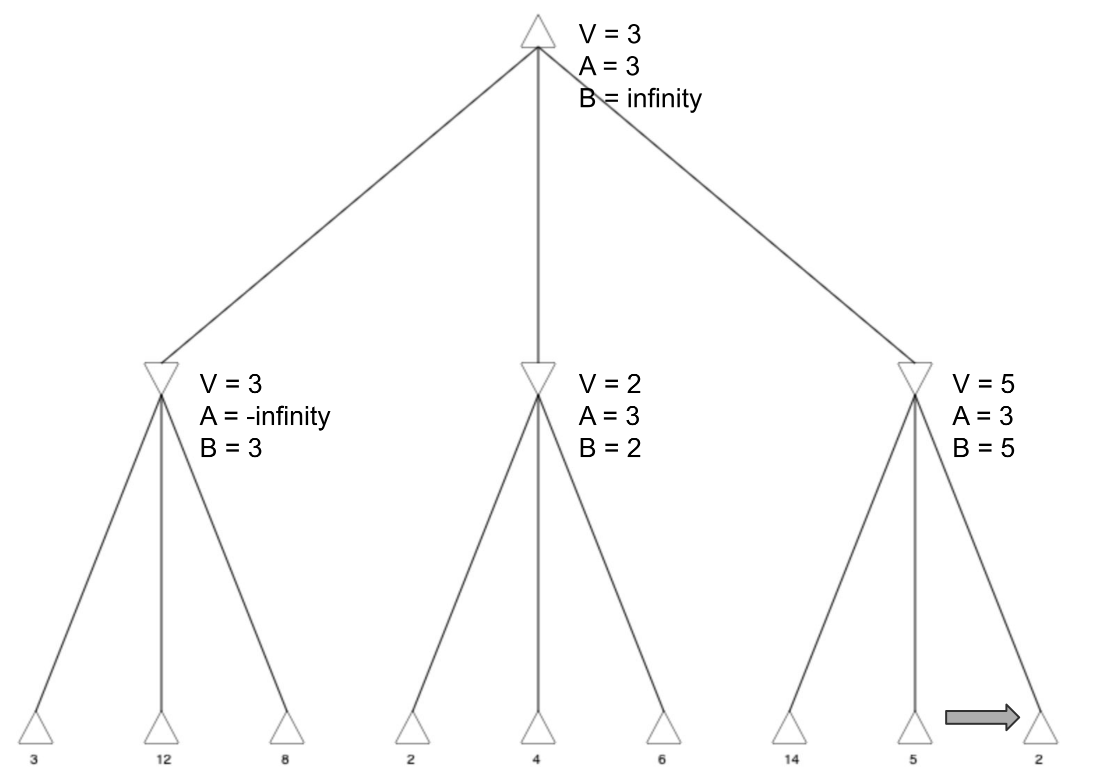


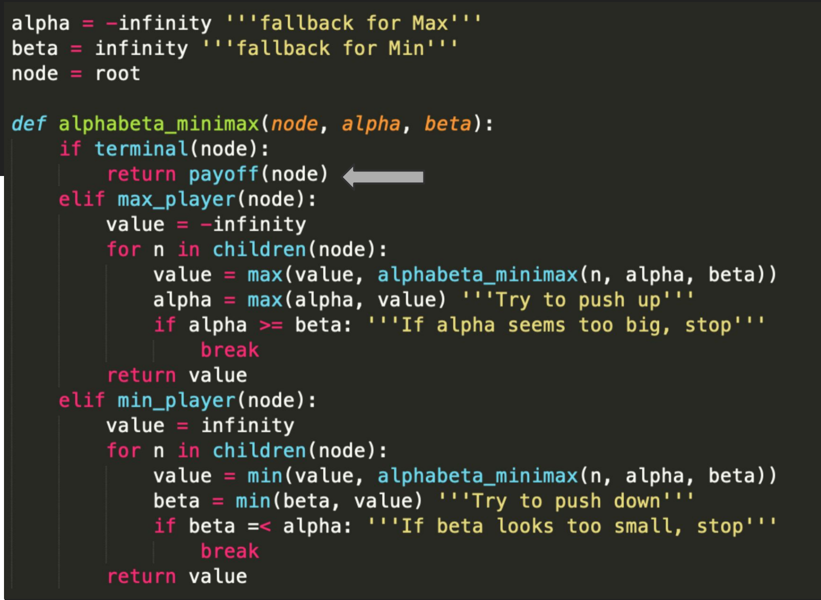


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

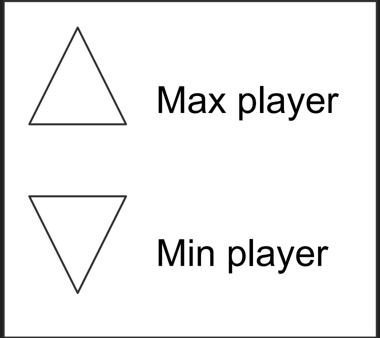


Example of alpha-beta minimax

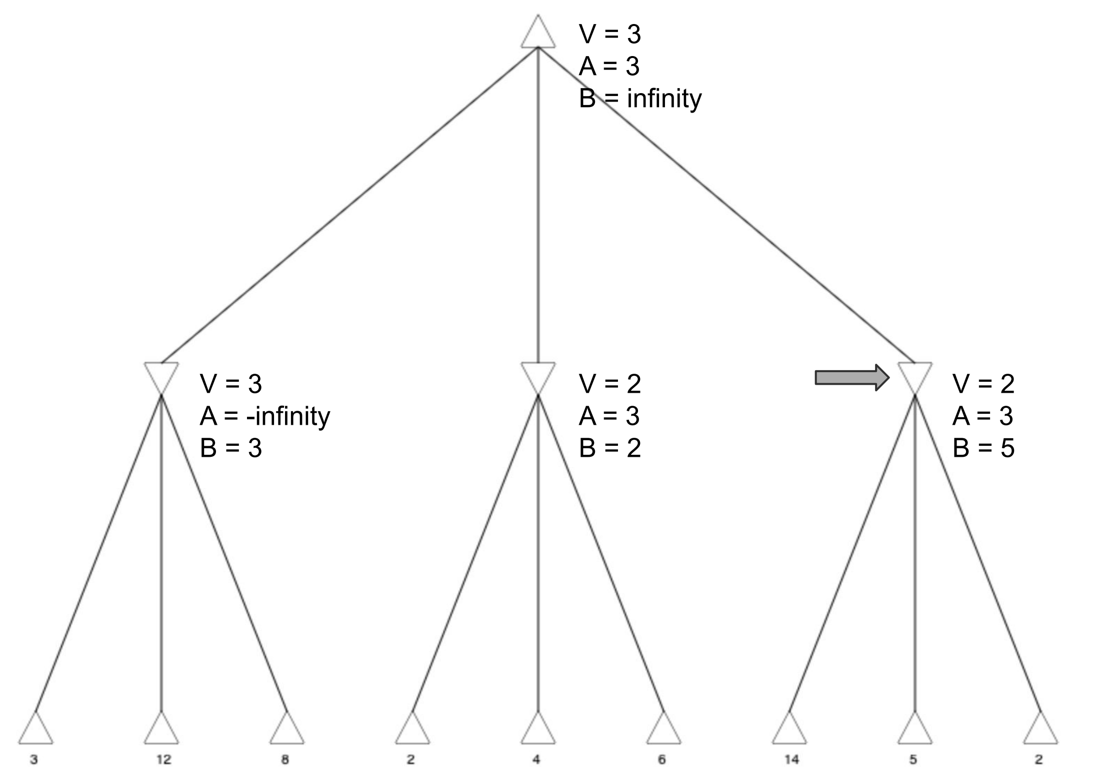


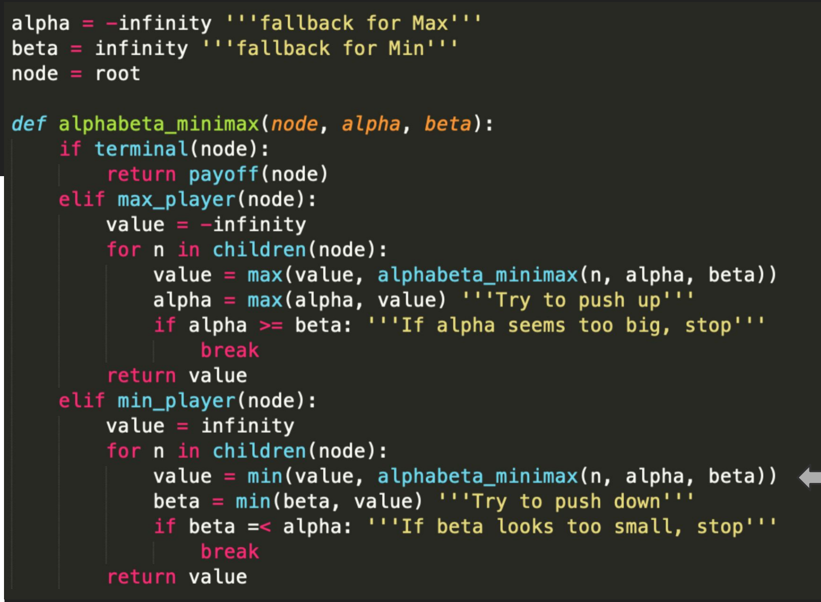


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

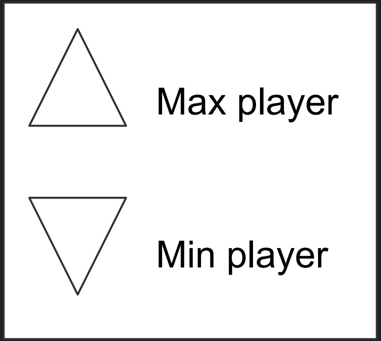


Example of alpha-beta minimax

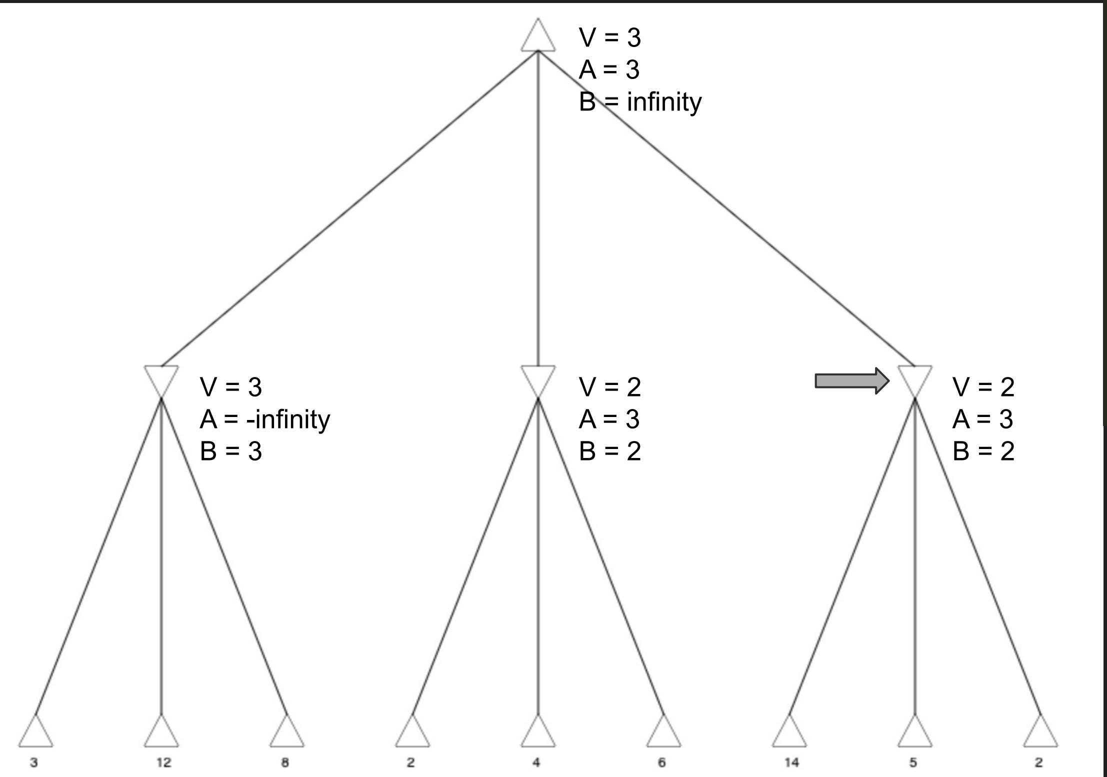


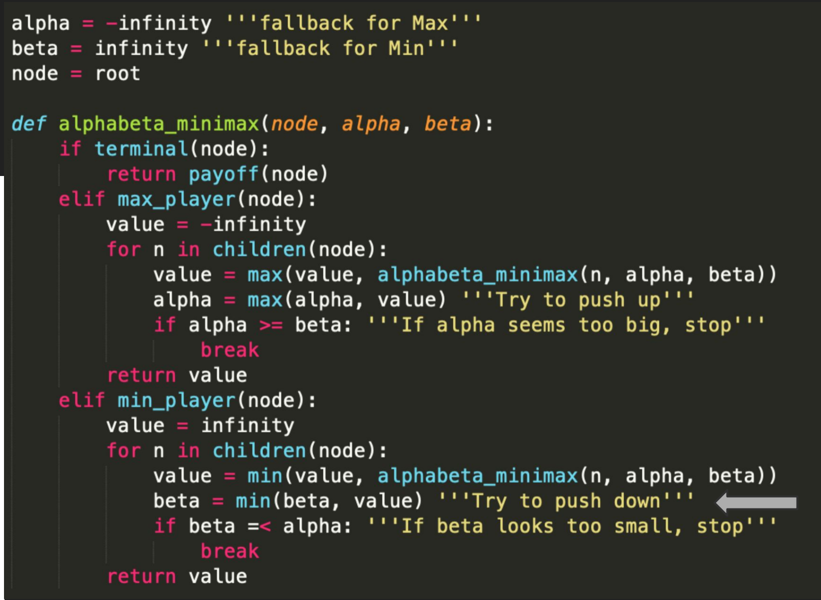


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

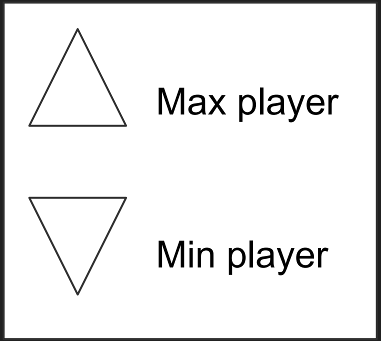


Example of alpha-beta minimax

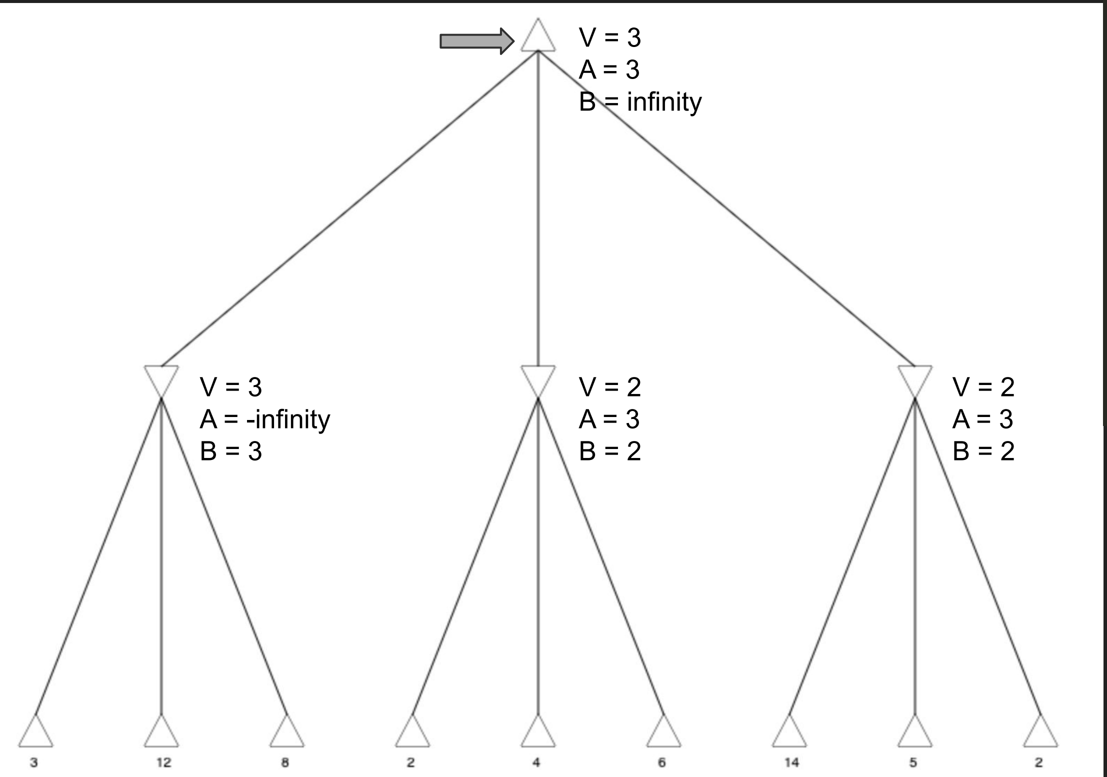


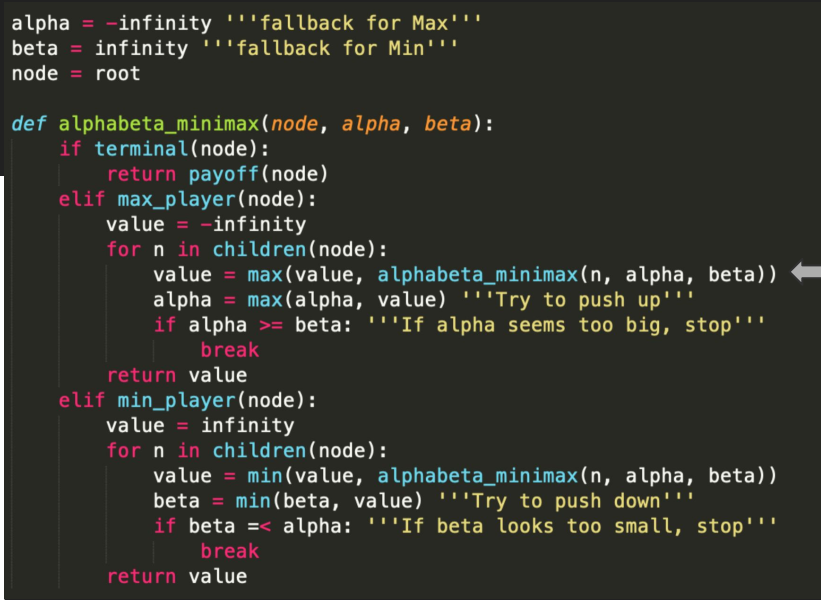


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```

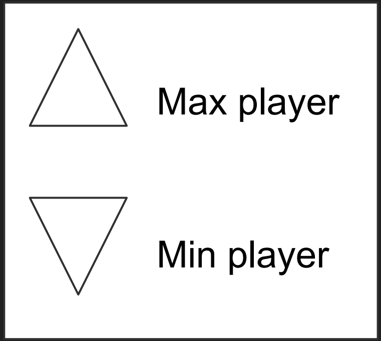


Example of alpha-beta minimax



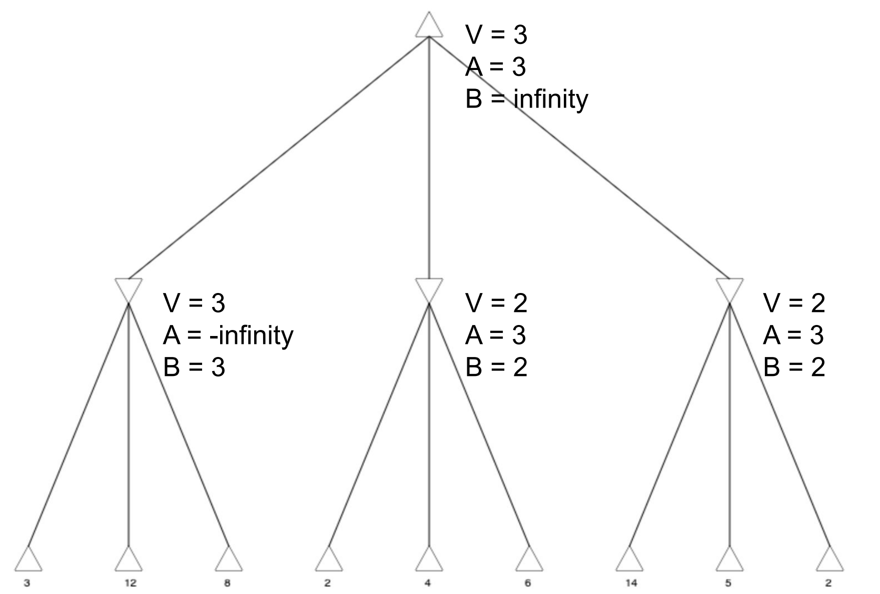


```
alpha = -infinity '''fallback for Max'''
beta = infinity '''fallback for Min'''
node = root

def alphabeta_minimax(node, alpha, beta):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, alphabeta_minimax(n, alpha, beta))
            alpha = max(alpha, value) '''Try to push up'''
            if alpha >= beta: '''If alpha seems too big, stop'''
                break
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, alphabeta_minimax(n, alpha, beta))
            beta = min(beta, value) '''Try to push down'''
            if beta =< alpha: '''If beta looks too small, stop'''
                break
        return value
```



FINAL RESULT: 3

Comparison of two algorithms



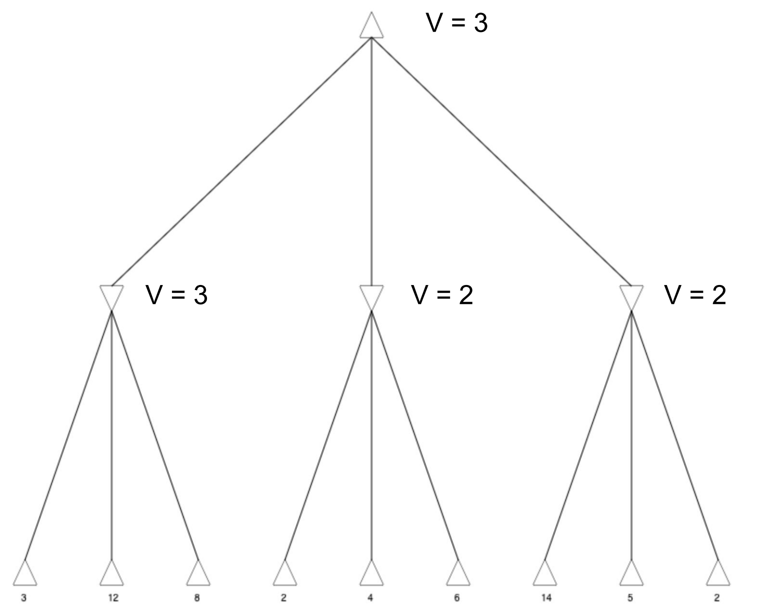
V = 3
A = 3
B = infinity

V = 3
A = -infinity
B = 3

V = 2
A = 3
B = 2

V = 2
A = 3
B = 2

FINAL RESULT: 3



V = 3

V = 3

V = 2

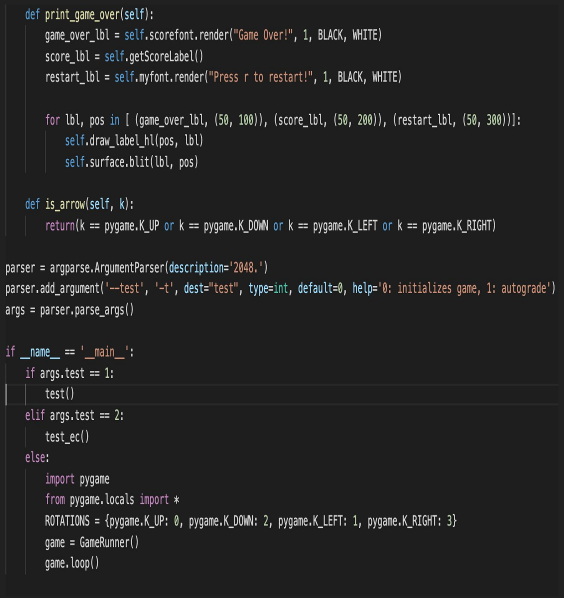
V = 2

FINAL RESULT: 3

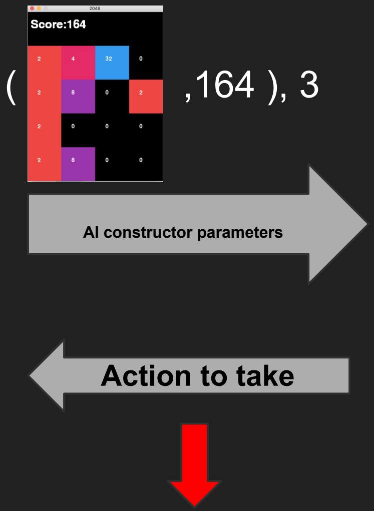
- Both methods return the same value for the root, but we observed that alpha-beta pruning does not visit all the nodes of the tree whereas regular minimax does.
- For more alpha-beta pruning practice: http://homepage.ufp.pt/jtorres/ensino/ia/alfabeta.html

Workflow of 2048

main.py



```
    def print_game_over(self):
        game_over_lbl = self.scorefont.render("Game Over!", 1, BLACK, WHITE)
        score_lbl = self.getScoreLabel()
        restart_lbl = self.myfont.render("Press r to restart!", 1, BLACK, WHITE)

        for lbl, pos in [ (game_over_lbl, (50, 100)), (score_lbl, (50, 200)), (restart_lbl, (50, 300))]:
            self.draw_label_hl(pos, lbl)
            self.surface.blit(lbl, pos)

    def is_arrow(self, k):
        return(k == pygame.K_UP or k == pygame.K_DOWN or k == pygame.K_LEFT or k == pygame.K_RIGHT)

parser = argparse.ArgumentParser(description='2048.')
parser.add_argument('--test', '-t', dest="test", type=int, default=0, help='0: initializes game, 1: autograde')
args = parser.parse_args()

if __name__ == '__main__':
    if args.test == 1:
        test()
    elif args.test == 2:
        test_ec()
    else:
        import pygame
        from pygame.locals import *
        ROTATIONS = {pygame.K_UP: 0, pygame.K_DOWN: 2, pygame.K_LEFT: 1, pygame.K_RIGHT: 3}
        game = GameRunner()
        game.loop()
```



Score:164

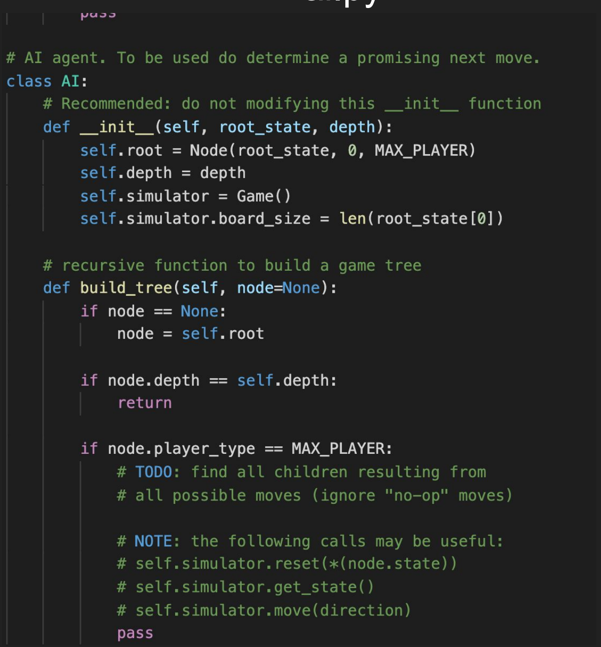
2	4	32	0
2	8	0	2
2	0	0	0
2	8	0	0

(,164), 3

AI constructor parameters

Action to take

ai.py



```
        pass

# AI agent. To be used do determine a promising next move.
class AI:
    # Recommended: do not modifying this __init__ function
    def __init__(self, root_state, depth):
        self.root = Node(root_state, 0, MAX_PLAYER)
        self.depth = depth
        self.simulator = Game()
        self.simulator.board_size = len(root_state[0])

    # recursive function to build a game tree
    def build_tree(self, node=None):
        if node == None:
            node = self.root

        if node.depth == self.depth:
            return

        if node.player_type == MAX_PLAYER:
            # TODO: find all children resulting from
            # all possible moves (ignore "no-op" moves)

            # NOTE: the following calls may be useful:
            # self.simulator.reset(*(node.state))
            # self.simulator.get_state()
            # self.simulator.move(direction)
            pass
```

- To let ai make decision on move to take, when pygame finished initializing and boar visible, hit Enter.
- Initialize AI with tuple containing board state and score along with depth of tree.

Main functions to implement in ai.py

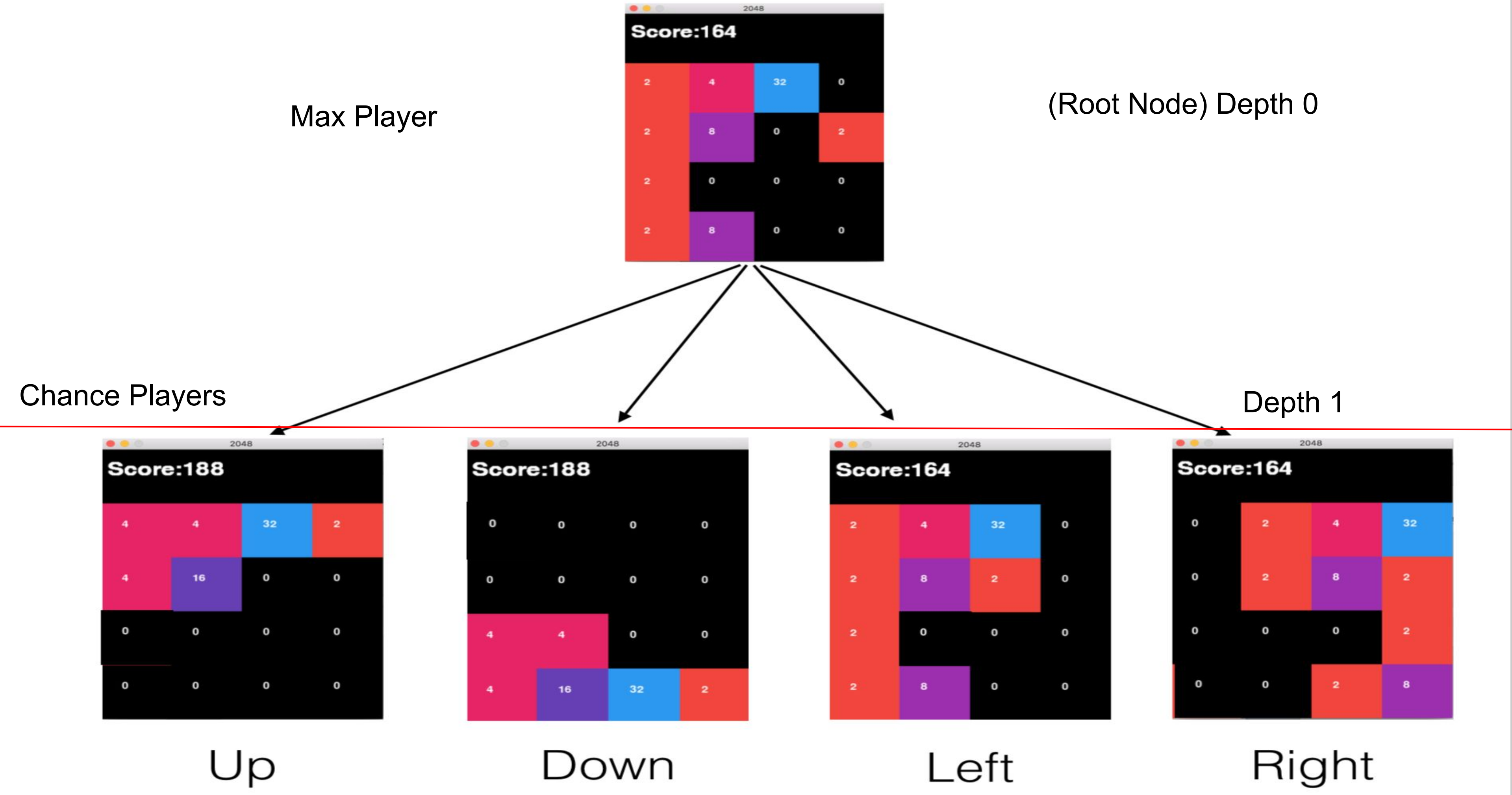
- build_tree
- expectimax

Growing the Tree

- In the build tree method, recursively build the tree to the depth specified in the AI class’ constructor.
- Start growing from the root node which is a max player.

Growing the Tree (MAX nodes)

- Max player nodes simulate how humans play 2048.
- This means that you can take 4 such actions at a max node: UP (0) , DOWN (2), LEFT (1), RIGHT (3).
- This also means a max player node can at maximum have 4 children nodes.
- Use the simulator to retrieve the updated board and score as a result of these actions.
- Prior to getting new children, ensure simulator has the same board and score as specified in the max node.



Max Player

(Root Node) Depth 0

2048

Score:164

2	4	32	0
2	8	0	2
2	0	0	0
2	8	0	0

Chance Players

Depth 1

2048

Score:188

4	4	32	2
4	16	0	0
0	0	0	0
0	0	0	0

Up

2048

Score:188

0	0	0	0
0	0	0	0
4	4	0	0
4	16	32	2

Down

2048

Score:164

2	4	32	0
2	8	2	0
2	0	0	0
2	8	0	0

Left

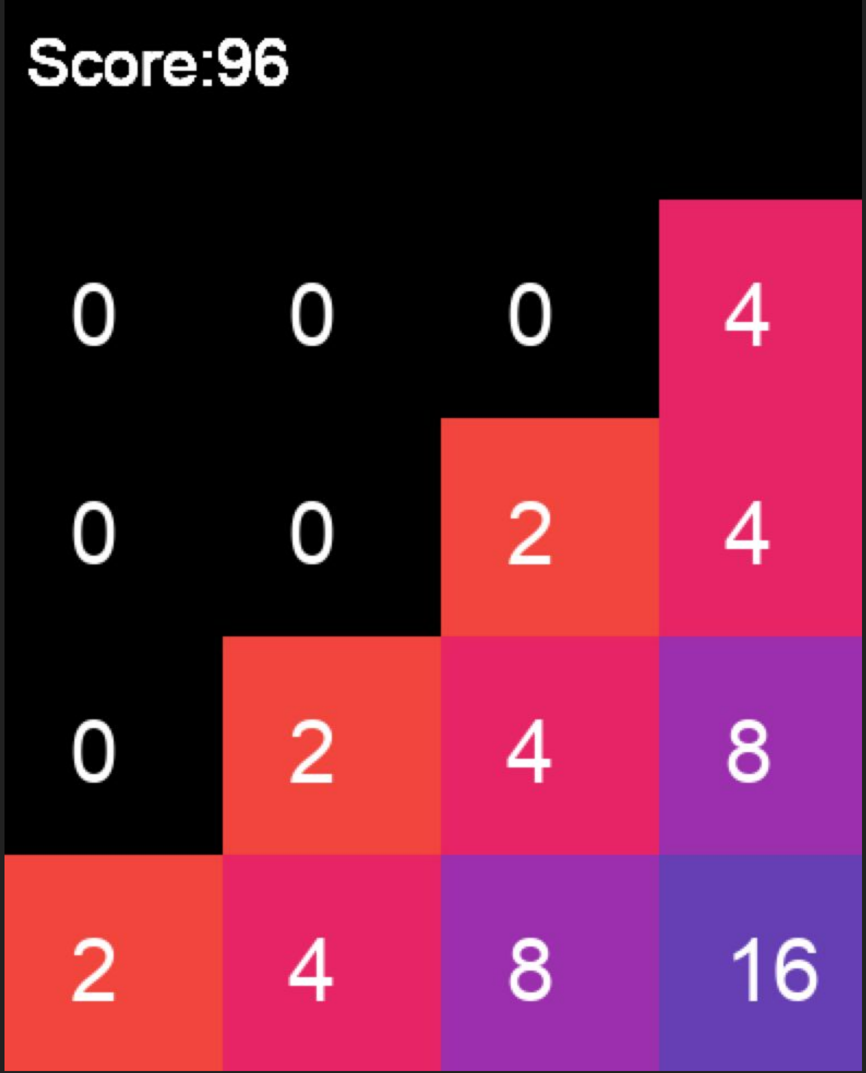
2048

Score:164

0	2	4	32
0	2	8	2
0	0	0	2
0	0	2	8

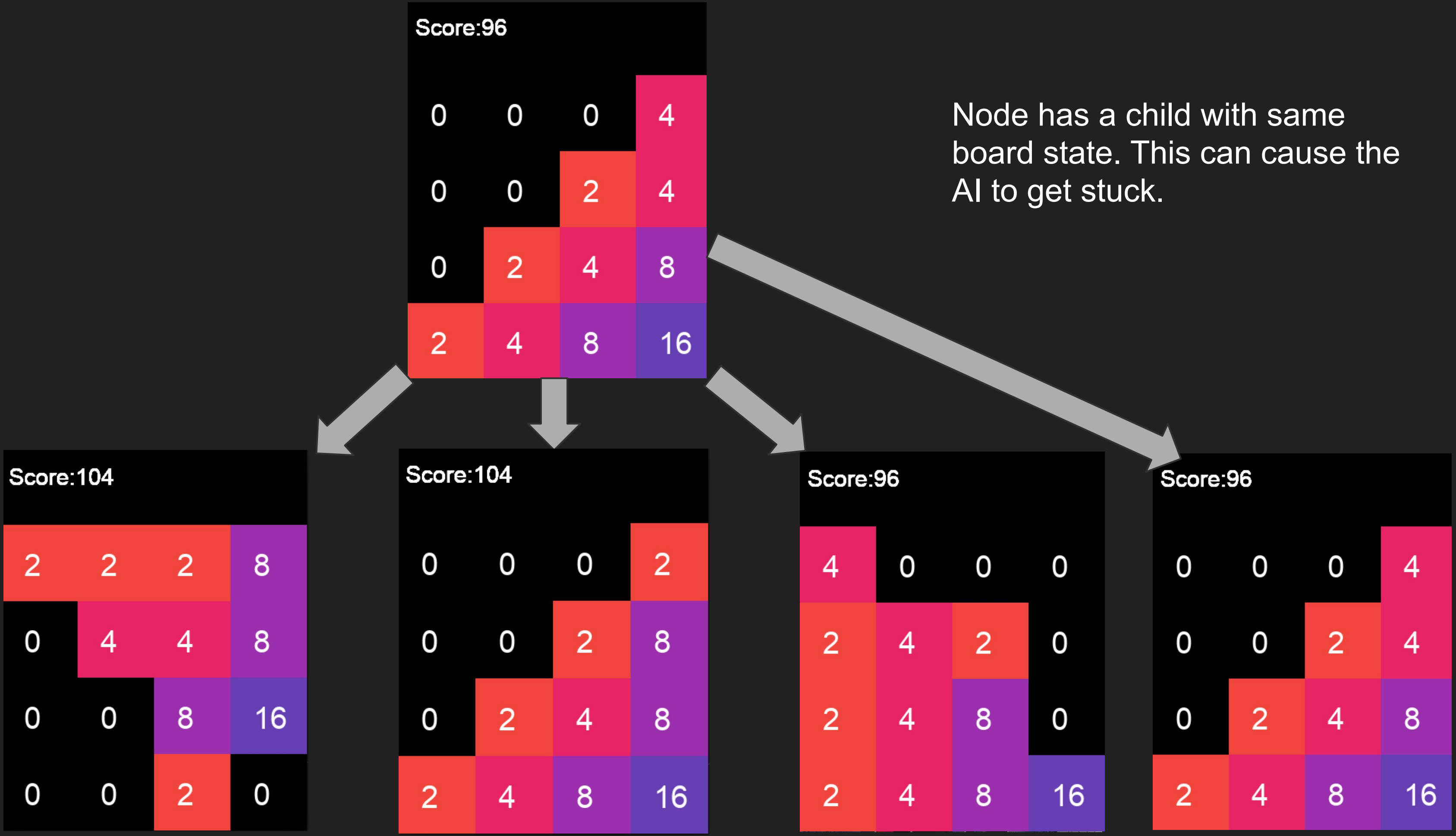
Right

Edge case to handle when growing max player node



Score:96

0	0	0	4
0	0	2	4
0	2	4	8
2	4	8	16



Score:96

0	0	0	4
0	0	2	4
0	2	4	8
2	4	8	16

Node has a child with same board state. This can cause the AI to get stuck.

Score:104

2	2	2	8
0	4	4	8
0	0	8	16
0	0	2	0

UP

Score:104

0	0	0	2
0	0	2	8
0	2	4	8
2	4	8	16

DOWN

Score:96

4	0	0	0
2	4	2	0
2	4	8	0
2	4	8	16

LEFT

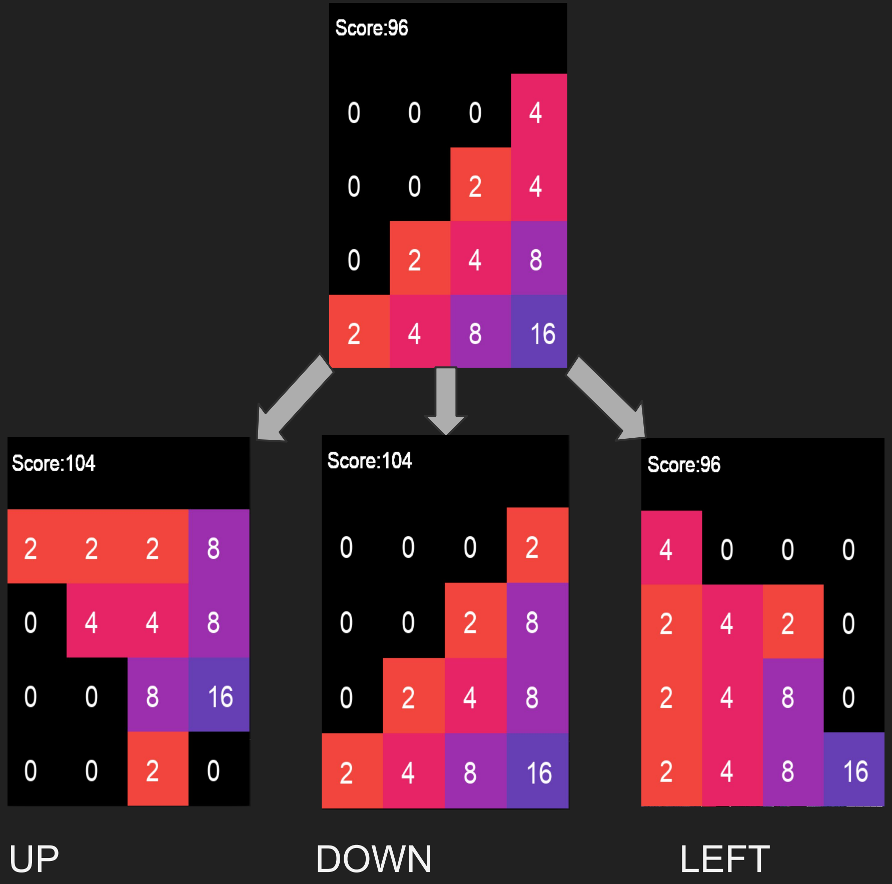
Score:96

0	0	0	4
0	0	2	4
0	2	4	8
2	4	8	16

RIGHT

Key takeaway for growing tree (Max players nodes)

- Make sure child node’s board is unique.
- Use the boolean returned by the move method to check if move resulted in unique board.
- Ensure simulator has same board state and score as the node, prior to doing move.
- Check simulator method reset() and get_state() which will help in child node creation.



Score:96

0	0	0	4
0	0	2	4
0	2	4	8
2	4	8	16

Score:104

2	2	2	8
0	4	4	8
0	0	8	16
0	0	2	0

UP

Score:104

0	0	0	2
0	0	2	8
0	2	4	8
2	4	8	16

DOWN

Score:96

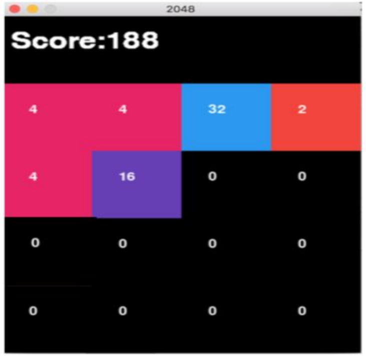
4	0	0	0
2	4	2	0
2	4	8	0
2	4	8	16

LEFT

Growing the GameTree (Chance nodes)

- Chance player nodes simulates how the computer plays 2048.
- In this game, the computer randomly places 2 on empty tiles.
- This means at a given chance player node, the number of children is equal to the number of empty tiles on the board of the chance node.
- Similar to max player nodes, ensure that the simulator has the same board state and score as the node.
- The simulator method get_open_tiles(), will be useful for finding all the empty tiles of the board at a chance player node.

Chance Player



2048

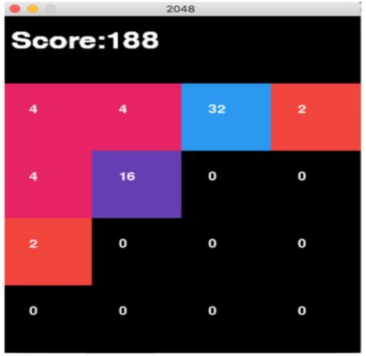
Score:188

4	4	32	2
4	16	0	0
0	0	0	0
0	0	0	0

Depth 1

Max Players

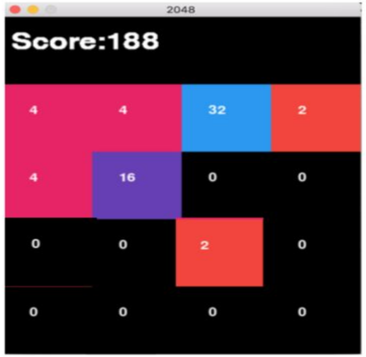
Depth 2



2048

Score:188

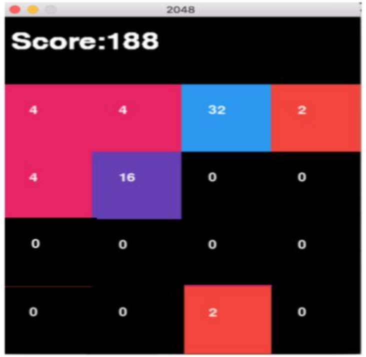
4	4	32	2
4	16	0	0
2	0	0	0
0	0	0	0



2048

Score:188

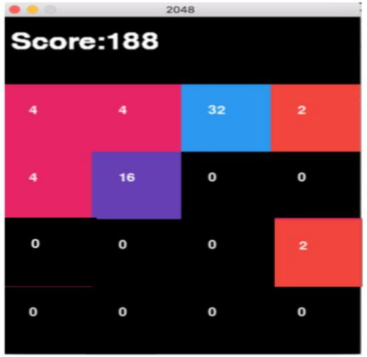
4	4	32	2
4	16	0	0
0	0	2	0
0	0	0	0



2048

Score:188

4	4	32	2
4	16	0	0
0	0	0	0
0	0	2	0



2048

Score:188

4	4	32	2
4	16	0	0
0	0	0	2
0	0	0	0

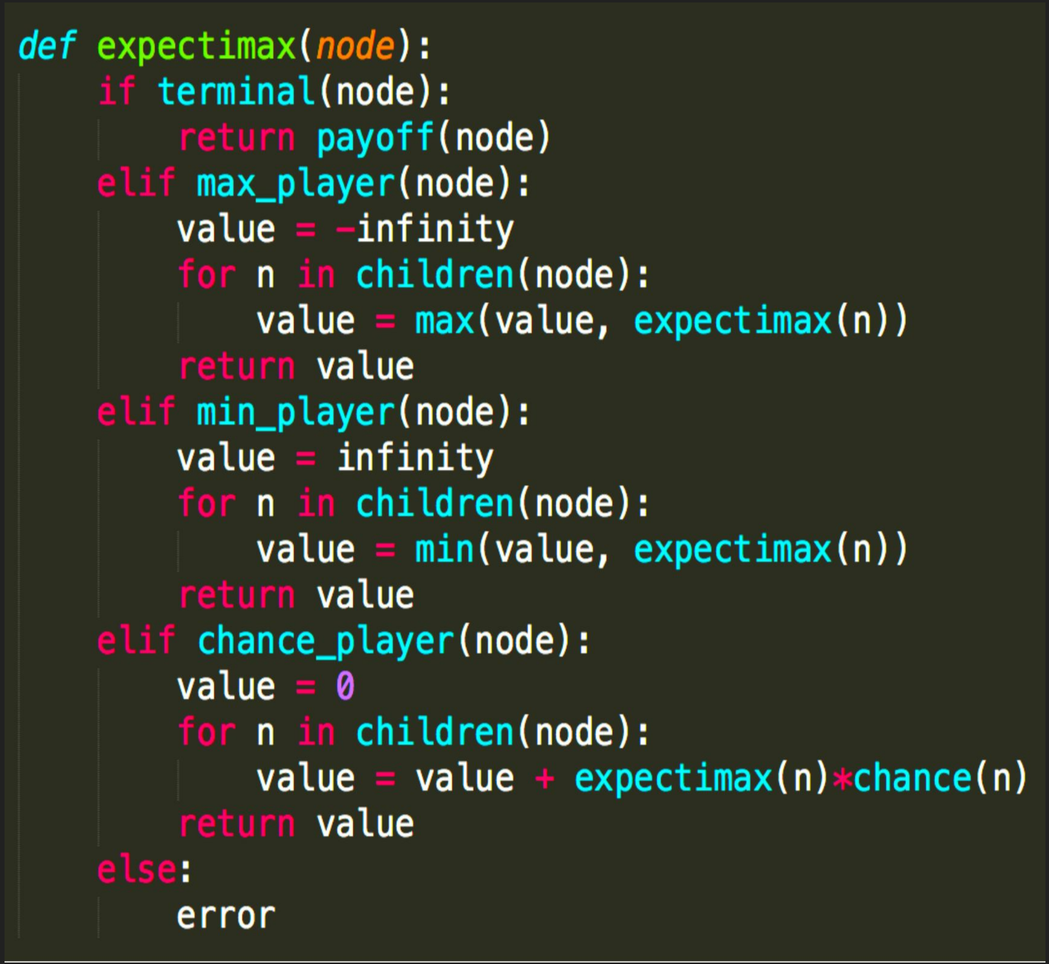
...

Growing the GameTree (TLDR)

- Grow the tree starting from the root node passed into your game tree constructor.
- The root node is a max player.
- Children of max player nodes are chance player nodes.
- Children of chance player nodes are max player nodes.
- Children of a max player is determined by taking board of max player and performing moves: up, down, left, and right.
- Children of a chance player is determined by taking the board of chance player and placing two in empty tile locations.
- Ensure simulator has same board state and score as seen in the node prior to growing out node.
- Use deep copies so you don’t overwrite node’s board state.

expectimax()

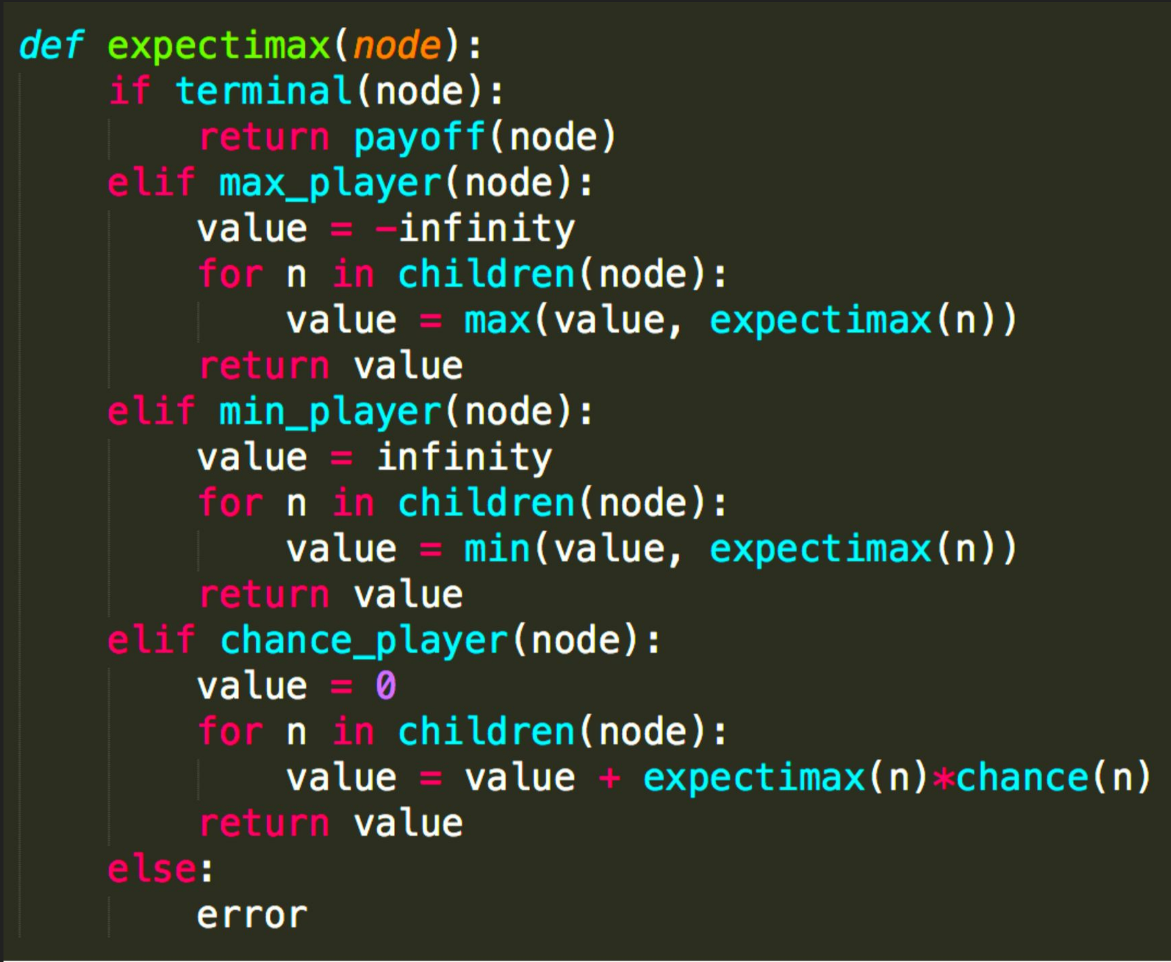
- Same as expectimax function seen here, minus the case for min_player.
- Function will be called on the root node of fully built game tree.
- Function returns a tuple containing (best direction, best value) for max player nodes and (None, best value) for chance player nodes.



```
def expectimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, expectimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, expectimax(n))
        return value
    elif chance_player(node):
        value = 0
        for n in children(node):
            value = value + expectimax(n)*chance(n)
        return value
    else:
        error
```

expectimax()

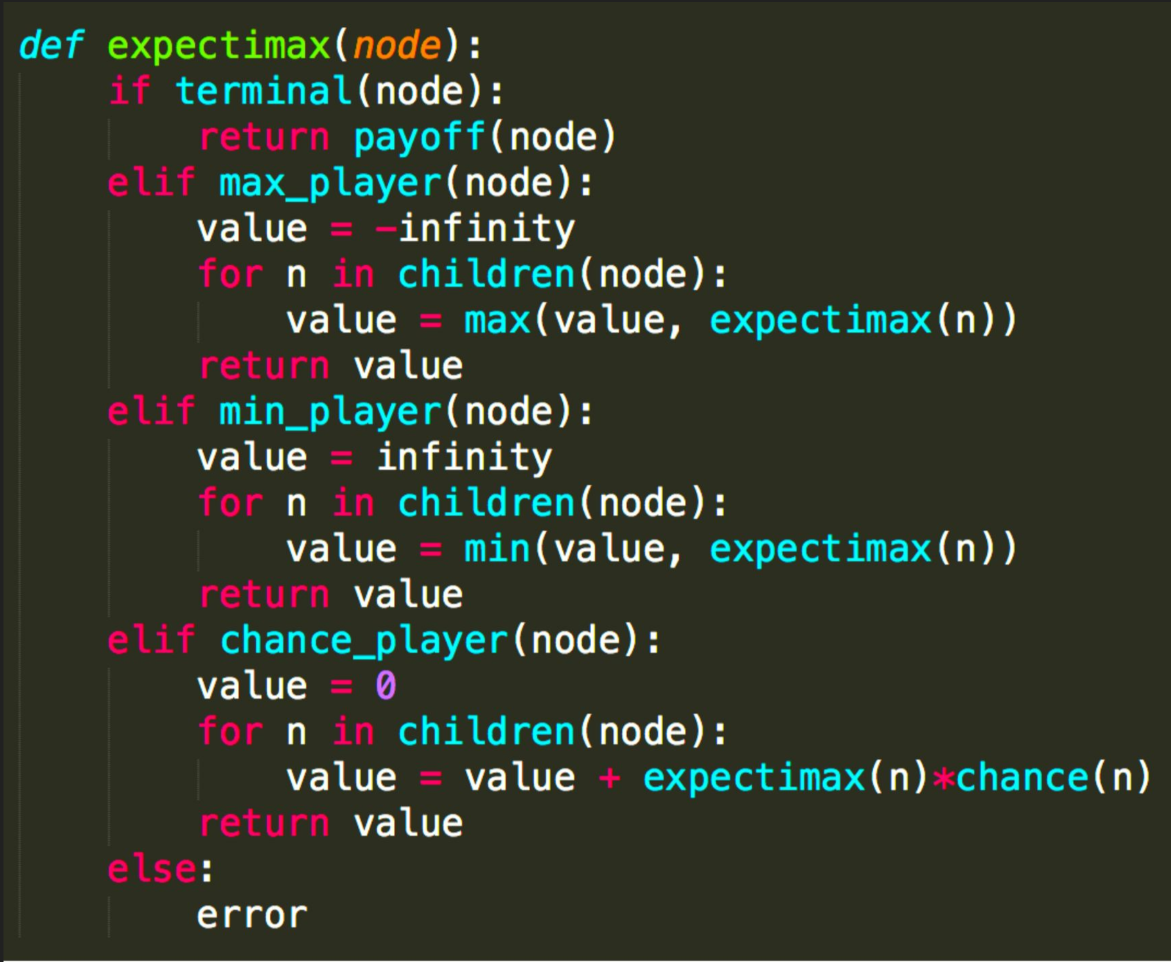
- Terminal node of tree corresponds to leaf node of tree/any node that doesn’t have children.
- For a max player, to find the best direction, you see which child node has the highest expectimax value returned, and choose the direction that resulted in that child node.



```
def expectimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, expectimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, expectimax(n))
        return value
    elif chance_player(node):
        value = 0
        for n in children(node):
            value = value + expectimax(n)*chance(n)
        return value
    else:
        error
```

expectimax()

- At a chance player node, chance(n) can be interpreted as the probability of the computer taking the action resulting in child board.
- At a chance player node, can assume computer can take any action with equal probability.



```
def expectimax(node):
    if terminal(node):
        return payoff(node)
    elif max_player(node):
        value = -infinity
        for n in children(node):
            value = max(value, expectimax(n))
        return value
    elif min_player(node):
        value = infinity
        for n in children(node):
            value = min(value, expectimax(n))
        return value
    elif chance_player(node):
        value = 0
        for n in children(node):
            value = value + expectimax(n)*chance(n)
        return value
    else:
        error
```

Submission Checklist

- Correct Implementation of depth 3 tree should be able to reach score of 5000 and reach 512 tile often.
- If you are doing the extra credit, write compute_decision_ec().
- Ensure that implementation works with provided main.py, game.py, and tests.py prior to turn in.
- Only submit ai.py to Gradescope.

GOOD LUCK