```
import math

def minimax(curDepth, nodeIndex, maxTurn, scores, targetDepth):
  if(curDepth == targetDepth):
    return scores[nodeIndex]

  if(maxTurn):
    return max(minimax(curDepth + 1, nodeIndex * 2, False, scores, targetDepth),
               minimax(curDepth + 1, nodeIndex * 2 + 1, False, scores, targetDepth))
  else:
    return min(minimax(curDepth + 1, nodeIndex * 2, True, scores, targetDepth),
               minimax(curDepth + 1, nodeIndex * 2 + 1, True, scores, targetDepth))


scores = [3, 5, 6, 10, 1, 2, 0, -1]
treeDepth = math.log(len(scores), 2)
print("The optimal value is: ", end = "")
print(minimax(0,0,True,scores,treeDepth))
```

```
The optimal value is: 5
```

```
values = [3, 5, 6, 10, 1, 2, 0, -1,3,6,1,7,3,6,3,9]
treeDepth = math.log(len(values), 4)
print("The optimal value for b = 4 is: ", end = "")
print(minimax(0,0,True,values,treeDepth))

treeDepth = math.log(len(values), 2)
print("The optimal value for b = 2 is: ", end = "")
print(minimax(0,0,True,values,treeDepth))
```

```
The optimal value for b = 4 is: 6
The optimal value for b = 2 is: 3
```

Alpha Beta Pruning

```
MAX, MIN = 1000, -1000

def alphabeta(depth, nodeIndex, maximizingPlayer, values, alpha, beta,b,maxDepth):

  if(depth == maxDepth):
    return values[nodeIndex]

  if maximizingPlayer:
    best = MIN
    for i in range (0, b):
      val = alphabeta(depth + 1, nodeIndex *2 + i, False, values, alpha, beta,b,maxDepth
      best = max(best, val)
      alpha = max(alpha, best)
      if (beta <= alpha):
        break

    return best
```

```
  else:
    best = MAX
    for i in range(0, b):
      val = alphabeta(depth + 1, nodeIndex *2 + i, True, values, alpha, beta,b,maxDepth)
      best = min(best, val)
      beta = min(beta, best)
      if(beta <= alpha):
        break
    return best

values = [3,5,6,10,1,2,0,-1]
print("The optimal value is: ", alphabeta(0,0,True,values,MIN,MAX,2,3))
```

```
The optimal value is:  5
```

```
values = [3, 5, 6, 10, 1, 2, 0, -1,3,6,1,7,3,6,3,9]
print("The optimal value is: ", alphabeta(0,0,True,values,MIN,MAX,2,4))
```

```
The optimal value is:  3
```

```
values = [3, 5, 6, 10, 1, 2, 0, -1,3,6,1,7,3,6,3,9]
print("The optimal value is: ", alphabeta(0,0,True,values,MIN,MAX,4,2))
```

```
The optimal value is:  3
```