Code:

```
dict = {'00':'A', '10':'B', '20':'D', '21':'E', '11':'C', '22':'F', '23':'G'}

MAX, MIN = 1000, -1000

def minimax(depth, nodeIndex, maximizingPlayer,
          values, alpha, beta):

   if depth == 3:
      return values[nodeIndex]

   if maximizingPlayer:

      best = MIN

      for i in range(0, 2):
         print("Node : ",dict.get(str(depth)+str(nodeIndex))," Values -- alpha : ",alpha,"beta :
",beta)
         val = minimax(depth + 1, nodeIndex * 2 + i,
                     False, values, alpha, beta)
         best = max(best, val)
         alpha = max(alpha, best)
         print("Node : ",dict.get(str(depth)+str(nodeIndex))," Values -- alpha : ",alpha,"beta :
",beta)

         # Alpha Beta Pruning
         if beta <= alpha:
            break

      return best

   else:
      best = MAX

      for i in range(0, 2):
         print("Node : ",dict.get(str(depth)+str(nodeIndex))," Values -- alpha : ",alpha,"beta :
",beta)

         val = minimax(depth + 1, nodeIndex * 2 + i,
                     True, values, alpha, beta)
         best = min(best, val)
```

```
            beta = min(beta, best)
            print("Node : ",dict.get(str(depth)+str(nodeIndex))," Values -- alpha : ",alpha,"beta :
",beta)

            # Alpha Beta Pruning
            if beta <= alpha:
                break

        return best

if __name__  == "__main__":


    values = [2,3,5,9,0,1,7,5]
    graph = {
        'A' : ['B','C'], 'B' : ['D','E'], 'C' : ['F','G'], 'D' : [2,3], 'E' : [5,9], 'F' : [0,1], 'G' : [7,5]
    }
    print("The optimal value is :", minimax(0, 0, True, values, MIN, MAX))
```

OUTPUT: