CSC-654 Algorithms Analysis / Design Homework #5

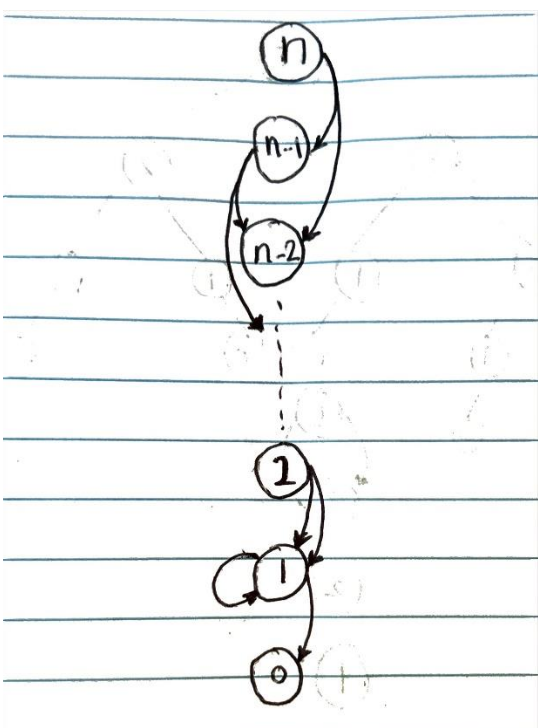
By Mostafa Abdelmegeed

Problem 15.1-3 from book

```
def cutrod_aux(pricelist, n, memos, sol, cost):

    #if n in memos:

    if memos[n] != -1:

        return memos[n]

    if n == 0:

        memos[n] = 0

        return 0

    bestprice = -math.inf

    for i in range(0, n):

        thisprice = pricelist[i] + cutrod_aux(pricelist, n - (i + 1), 
memos, sol, cost) - cost

        if thisprice > bestprice:

            sol[n] = i + 1

            bestprice = thisprice

    memos[n] = bestprice

    return bestprice



def cutrod_dp(pricelist, n, cost):

    memos = [-1] * (n+1)

    sol = [-1] * (n+1)

    rv = cutrod_aux(pricelist, n, memos, sol, cost)

    while (n > 0):

        n = n - sol[n]

    if rv == pricelist[n-1] - cost:

        return pricelist[n-1]


    return rv
```

Problem 15.1-5 from book

```
def fib_aux(n, memo):
    if n in memo:
        return memo[n]
    result = fib_aux(n-2,memo) + fib_aux(n-1, memo)
    memo[n] = result
    return result

def fib_db(n):
    memo = dict()
    memo[0] = 0
    memo[1] = 1
    return fib_aux(n, memo)
```



Assuming by edges the question means the number of times the recursion calls for the particular node is called, there will be **n+1** edges in the graph, and the number of nodes will be **n**

Problem 15.3-6 from book

Part I

Assuming commission 0, considering an optimal sequence S from currency 1 to n.

let P be the first part of the sequence (1 to k) and let Q be the second part of the sequence (k to n), where considering S to be a combination of {PQ}

Now considering another path for the first sequence, say P' to be optimal and P to not be.

Now there is a change in sequence, and the new sequence is S', where S' is a combo of {P'Q}

with this, we can deduce that S' is better than S, which is a contradiction of our original assumption.

Hence proving that an optimal substructure exists.

Part II

Let's let the conversion from i to j be 0.9. Let's let the conversion from j to k be 0.9. Finally, let's let the conversion from i to k be 0.89.

Let's also let the cost of making 1 trade .7 unit of currency i while making 2 trades costs 0.3 units of currency i, any number of trades more than that costs 100 currency units per trade.

Let's say I have 1 in currency i and I want currency k.

If we just look at the way the commissions are structured, by making 2 trades instead of 1 trade, we save .7 units of currency i. The best way to go from currency i to currency k would be go from i to j then k. Which would result in .81 - .3 = .51. Note this is better than going from i to k directly, which would be .89 − .7 = .19.

The best path from i to j would be not just i to j directly, however, since making that second trade saves you a fortune. That path would be i to k, then k to j, which would be 0.801 − .3 = .501 vs going directly would be .9 − .7 = .2.

Since the optimal path for a sub-problem if that problem was the whole problem is not a solution for when it is part of the overall problem, this problem no longer has optimal substructure property if ck can be arbitrary values.