Greedy Algorithms: Making Change

Algorithm for Counting Change

Note: The algorithms below assume that the last “coin” always has denomination of 1; We also assume that all array elements are automatically initialized to 0;

Recursive Algorithm

```
ALGORITHM MakeChangeRecursive(K, M[J..K], Amount)
begin
    if Amount = 0 then //base case: no change left
        Change[1..K];
        Change[J] ← 0;
        return Change[1..K]
    else
        if J = K then //base case: reached the end coins array
            Change[1..K];
            Change[K] ← Amount;
            return Change[1..K]
        else //recursion step
            tmp ← Amount / M[J];
            Amount ← Amount - tmp * M[i];
            Change ← MakeChangeRecursive(K, M[J+1..K], Amount);
            Change[J] ← tmp;
            return Change[1..K];
        endif
    endif
end
```
Iterative Algorithm

```
ALGORITHM MakeChange(K, M[1..K], Amount)
begin
    Change[1..K]
i ← 1;
    while Amount > 0 do
        Change[i] ← Amount/M[i];
        Amount ← Amount - Change[i]*M[i];
        i ← i + 1;
    endwhile
    return Change[1..K];
end
```