LintCode 92

http://lintcode.com/en/problem/backpack/

Yifeng Zeng

Description

92. Backpack

Idea Report

[3,4,5,8] => 10

	0	1	2	3	4	5	6	7	8	9	10
0	T	F	F	F	F	F	F	F	F	F	F
3	T	F	F	T	F	F	F	F	F	F	F
4	T	F	F	T	T	F	F	T	F	F	F
5	T	F	F	T	T	T	F	T	T	T	F
8	T	F	F	T	T	T	F	T	T	T	F

Code

```
public class Solution {
    // DP AC
    public int backPack(int m, int[] A) {
        boolean[][] f = new boolean[A.length + 1][m + 1];
        f[0][0] = true;
        int rows = A.length;
        for (int i = 1; i <= rows; i++) {
            f[i][0] = true;
            for (int j = 1; j <= m; j++) {
                if (i == 3) {
                    System.out.println(i + "," + j);
```

```
                }
                f[i][j] |= f[i-1][j];
                if (j - A[i-1] >= 0) {
                    f[i][j] |= f[i - 1][j - A[i-1]];
                }
            }
        }

        for (int i = m; i >= 0; i--) {
            if (f[A.length][i]) {
                return i;
            }
        }
        return 0;
    }
}
```

```
// DP with space optimization AC
public int backPack(int m, int[] A) {
    // boolean[][] f = new boolean[A.length + 1][m + 1];
    boolean[] f = new boolean[m + 1];
    f[0] = true;
    for (int i = 1; i <= A.length; i++) {
        for (int j = m; j >= A[i-1]; j--) {
            f[j] |= f[j - A[i-1]];
        }
    }

    for (int i = m; i >= 0; i--) {
        if (f[i]) {
            return i;
        }
    }
    return 0;
}
```

DFS for loop without memoization

Code

```
// for loop TLE
public int backPack(int m, int[] A) {
    int[] min = new int[1];
    min[0] = m + 1;
    helper(m, A, 0, min);
    return m - min[0];
```

```
}

private void helper(int m, int[] A, int index, int[] min) {
    System.out.println(m + ", " + index + ", " + min[0]);
    if (m >= 0) {
        min[0] = Math.min(min[0], m);
    }

    if (index == A.length || m < 0) {
        return;
    }

    for (int i = index; i < A.length; i++) {
        helper(m - A[i], A, i + 1, min);
    }
}
```

DFS choose/not choose without memoization, TLE

```
// public int backPack(int m, int[] A) {
//     for (int i = m; i >= 0; i--) {
//         if (sumTo(A, i)) {
//             return i;
//         }
//     }
//     return 0;
// }

private boolean sumTo(int[] A, int m) {
    return helper(m, A, 0);
}
private boolean helper(int m, int[] A, int index) {
    if (m <= 0) {
        return m == 0;
    }
    if (index >= A.length) {
        return false;
    }

    return helper(m, A, index + 1) || helper(m - A[index], A, index + 1);
}
```

DFS choose/not choose with memoization, Memory Limit Exceeded

```
// public int backPack(int m, int[] A) {
```

```
//     for (int i = m; i >= 0; i--) {
//         if (sumTo(A, i)) {
//             return i;
//         }
//     }
//     return 0;
// }

private boolean sumTo(int[] A, int m) {
    int[][] memo = new int[A.length + 1][m + 1];
    // for (int[] r : memo) {
    //     System.out.println(Arrays.toString(r));
    // }
    return helper(m, A, 0, memo);
}
private boolean helper(int m, int[] A, int index, int[][] memo) {
    if (m <= 0) {
        return m == 0;
    }
    if (index >= A.length) {
        return false;
    }

    if (memo[index][m] != 0) {
        return memo[index][m] == 1;
    }

    boolean res = false;
    res = helper(m, A, index + 1, memo);
    if (res) {
        memo[index][m] = 1;
        return true;
    }
    res = helper(m - A[index], A, index + 1, memo);
    if (res) {
        memo[index][m] = 1;
        return true;
    }
    memo[index][m] = -1;
    return false;
}
```

Summary

-