The Design and Analysis of Algorithms
National Taiwan University, Fall 2020

Homework 4

Yu-Chieh Kuo B07611039

This homework answers the problem set sequentially.

1. Let K be the size of the knapsack, $S[i]$ be the size of the i-th item, and $P(n, K)$ be the solution with the number of items n and the size of the knapsack K. The algorithm can be rewrite as below.

Algorithm 1 ModifyKnapsack(K,S,P)

Let int k := K, Answer := []
for $i := n \dots 1$ **do**
  if $P[i, k].exist = false$ **then**
    return 0
  end if
  if $P[i, k].belong = true$ **then**
    $Answer.append(S[i])$
    $k := K - S[i]$
  end if
end for
if $k \neq 0$ **then**
  return 0
end if
return Answer

2. With the description of the statement, we can rewrite the knapsack algorithm as below.

Algorithm 2 Knapsack(K,S,P)

```
Let P[0,0].exist := true
Let P[0,0].belong := 0
for k := 1...K do
    P[0,k].exist := false
end for
for i := 1...n do
    for k := 0...K do
        P[i,k].exist := false
        if P[i-1,k].exist then
            P[i,k].exist := true
            P[i,k].belong := 0
        else if k - S[i] >= 0 then
            if P[i,k-S[i]].exist then
                P[i,k].exist := true
                P[i,k].belong := P[i,k-S[i]].belong + 1
            end if
        end if
    end for
end for
```

3. Let n be the size of a set of integers, $X[]$ be the set of integers, S be the sum of the set, the algorithm can be represented as below.

Algorithm 3 FindPartition(X[],n,S)

if S is odd **then**
    return false
end if
Let int $s := \frac{S}{2}$, $dp[n][s]$, $set1, set2 = []$
if $dp[n][s].exist$ **then**
    while $n > 0$ **do**
        if $dp[n][s].belong$ **then**
            $set1.append(X_n)$
            $s := s - x_n$
        else
            $set2.append(X_n)$
        end if
        $n := n - 1$
    end while
else
    return false

4. (a) The algorithm can be write as below.

Algorithm 4 HanoiTower(n,A,B,C)

if $n := 1$ **then**
    move from A to B
else if $n > 1$ **then**
    HanoiTower(n-1,A,C,B)
    move the largest disk from A to B
    HanoiTower(n-1,C,B,A)

To Explain how induction works here, we can use the inductive method. In the base case, we have 1 disk and we will move it from A to B. Assume that moving $n-1$ disks is available, when moving n disks, we will move $n-1$ disks from A to C, move the largest disk to B, then move $n-1$ disk from C to B.

(b) Let $S(n)$ be the total steps of moves for Hanoi Tower, we have $S(1) = 1$ and $S(n) = 2S(n-1) + 1$. With easy computation, we can find $S(n) = 2^n - 1$.

5. Without the recursive step, we can also write the algorithm to deal with the Hanoi Puzzle problem. The algorithm can be written as below. At first, we need to define a function *move(m,n)* to make the legal move between two legs m,n and print the move.

Algorithm 5 NonRecursiveHanoiPuzzle(n,A,B,C)

```
Let step := 0
while step < 2^n − 1 do
    if n is even then
        move(A,C)
        move(A,B)
        move(C,B)
    else
        move(A,B)
        move(A,C)
        move(C,B)
    end if
    step := step + 1
end while=0
```