Knapsack Problem

- 0-1 knapsack problem -

과 목 명	알고리즘
제출일자	2017. 11. 16
분　　반	02
조　　원	전성배
학　　번	201302476

과제 설명

1. 과제 설명

가. Knapsack problem

1) W 만큼의 무게까지만 담을 수 있는 배낭(knapsack)이 있다.

2) n개의 물건(item)들이 있고, 물건마다 번호(i)를 붙여놓았다.

3) 각각의 물건들은 wi 만큼의 무게와, vi만큼의 가치를 가진다.

2. 과제 해결 방법

가. 어떤 item을 포함시켰을 때와 그렇지 않을 때의 두 경우를 비교해봐야 하므로 결국 모든, 또는 대부분의 경우에 대해 확인을 해보아야 한다.

$$OPT(i,w)=\begin{cases}0 & \text{if } i=0\\ OPT(i-1,w) & \text{if } w_i > w\\ \max\{OPT(i-1,w),\ \ v_i + OPT(i-1,w-w_i)\} & \text{otherwise}\end{cases}$$

W+1 →

n+1 ↓

	0	1	2	3	4	5	6	7	8	9	10	11
Ø	0	0	0	0	0	0	0	0	0	0	0	0
{1}	0	1	1	1	1	1	1	1	1	1	1	1
{1,2}	0	1	6	7	7	7	7	7	7	7	7	7
{1,2,3}	0	1	6	7	7	18	19	24	25	25	25	25
{1,2,3,4}	0	1	6	7	7	18	22	24	28	29	29	40
{1,2,3,4,5}	0	1	6	7	7	18	22	28	29	34	35	40

Item	Value	Weight
1	1	1
2	6	2
3	18	5
4	22	6
5	28	7

나. 배열(Table)에 OPT 값을 채우는 함수와, 완성된 배열을 분석하여 가치(value) 총합이 가장 높은 item 구성 및 value 합을 출력하는 함수를 함수를 각각 구현한다.

다. 입력 데이터 형식, 입력은 표준 입력을 사용한다.

라. 첫 째 줄에는 아이템 개수 n과 가방의 무게 제한 w가 주어진다.

마. 두번째 줄부터 n개의 줄에 걸쳐 아이템의 가치와 무게가 주어진다.

KnapsackProblem

3. KnapsackProblem

가. 필드값

```
public class knapsackProblem {

    static Scanner scanner = new Scanner(System.in);
    static int[][] items;
    static int[][] opt;
```

- 입력으로 사용할 Scanner와 item들의 정보를 담아 둘 배열, opt정보를 담아둘 배열을 이중배열을 통해 만들었습니다.

나. main()

```
public static void main(String[] args) {

    // Number of items
    int n = scanner.nextInt();
    int limit_weight = scanner.nextInt();

    opt = new int[n + 1][limit_weight + 1];
    // Array of items
    items = new int[n + 1][2];

    // Insert value and weight
    for (int i = 1; i < n + 1; i++) {
        // Value of item
        int value = scanner.nextInt();
        // Weight of itme
        int weight = scanner.nextInt();
        // Save value and weight
        items[i][0] = value;
        items[i][1] = weight;
    }
    scanner.close();

    fillOpt();
    printItem();

}
```

- item의 개수를 n으로 knapsack의 무게 한도를 limit_weight로 입력받아 opt사이즈와 items의 크기로 저장하여 사용하였습니다.
- 다음으로 입력되는 값들을 items의 value와 weight로 저장합니다.
- opt를 채우는 함수와 완성된 opt를 분석하여 가치 총합이 가장 높은 값과 해당 item들을 출력해주는 함수를 호출합니다.

다. fillOpt() / opt()

```
// Fill out 0-1 knapsack
private static void fillOpt() {
    for (int i = 0; i < opt.length; i++) {
        for (int j = 0; j < opt[0].length; j++) {
            opt[i][j] = opt(i, j);
        }
    }
}

private static int opt(int i, int w) {
    if (i == 0) {
        return 0;
    } else if (items[i][1] > w) {
        return opt(i - 1, w);
    } else {
        return Math.max(opt(i - 1, w), items[i][0] + opt(i - 1, w - items[i][1]));
    }
}
```

- fillOpt() 메소드를 사용하여 opt[][]에 opt()를 통해 return된 값을 저장합니다.
- opt() 메소드는 i가 0일 경우에는 어느 아이템도 아니기 때문에 0을 return합니다.
- items[i][]의 값이 w보다 크다면 i-1을 해주어 다른 아이템이 w보다 작은지 확인할 수 있도록 재귀적으로 함수를 다시 호출 해줍니다.
- (핵심) items[i][]의 값이 w보다 크지 않다면 opt(i-1)의 값보다 items[i][]와 opt(i-1)의 값에서 w - items[i][]의 값을 뺀 만큼과 더하여 두 수중에 더 큰 값을 return하여 줍니다.

라. printItem

```
// Print opt
private static void printItem() {
    int max = 0;
    for (int i = 0; i < opt.length; i++) {
        for (int j = 0; j < opt[0].length; j++) {
            System.out.printf("\t%d", opt[i][j]);
            if (max < opt[i][j]) {
                max = opt[i][j];
            }
        }
        System.out.println();
    }
    System.out.println("max : " + max);
    System.out.print("item : ");
    printMaxItems(max);
}

private static void printMaxItems(int max) {
    for (int i = 0; i < opt.length && max != 0; i++) {
        for (int j = 0; j < opt[0].length; j++) {
            if (max == opt[i][j]) {
                max -= items[i][0];
                System.out.print(i + " ");
                printMaxItems(max);
                return;
            }
        }
    }
}
```

- print는 opt에 있는 모든 값들을 아이템 항목에 맞게 출력하도록 합니다.

- print를 하는 중에 각각의 항목에 대해 max 변수를 사용하여 최대값을 update하면서 최종적인 max를 출력합니다.
- printMaxItems()에 max를 parameter를 넣어 max와 opt에 같은 값이 있다면 해당 값을 가지고 있는 index를 출력하여 주고 max는 해당 index의 value만큼 빼줍니다.
- 빼고 남은 max값에 대해 재귀적으로 함수를 다시 호출하여 max의 값을 갖는 index, 즉 item의 번호를 찾아서 출력해줍니다.
- 남은 value가 0이 될 때까지 반복하여 줍니다.

실행 결과

4. Knapsack Problem