알고리즘 보고서

201402437 조찬수

knapsack problem (주석 참고)

```
public class Knapsack {
    static int N;

    public static int[][] Knapsack(int val[], int wt[], int W) {
        N = wt.length; // 세로의 길이-1
        int[][] V = new int[N + 1][W + 1];// table 배열
        for (int col = 0; col <= W; col++) { // 첫 행을 전부 0으로 채운다.
            V[0][col] = 0;
        }
        for (int row = 0; row <= N; row++) { // 첫번째 열을 전부 0으로 채운다.
            V[row][0] = 0;
        }

        for (int i = 1; i <= N; i++) { // 행을 차례대로 이동

            for (int w = 1; w <= W; w++) { // 열을 차례대로 이동
                if (wt[i - 1] > w) { // i번째 물건을 못 넣는 경우, 변함 없으므로 V[i-1][w]
                    V[i][w] = V[i - 1][w];
                } else { // i번째 물건을 넣을 수 있는 경우
                    V[i][w] = max(V[i - 1][w], val[i - 1] + V[i - 1][w - wt[i - 1]]);
                }
            }
        }

        return V;
    }
```

```
// 프린트 해주는 부분
public static void print(int[][] V, int[] wt, int[] val, int N, int W) {
    int[] selected = new int[N + 1]; // max 무게를 형성하는 item을 알아내기 위한 배열

    int n = N;
    int w = W;

    while(n>0 && w>0) { //n,w 둘다 0보다 클 동안
        if(V[n][w] != V[n-1][w]) {//바로 위의 행의 값과 같지 않다면
            selected[n]=1; //해당 item을 포함
            w = w-wt[n-1]; //열의 인덱스를 줄인다.
        }
        n--;
    }
    for (int[] rows : V) {
        for (int col : rows) {
            System.out.format("%3d", col); // 한 줄씩 출력
        }
        System.out.println();
    }
    System.out.println("max : " + V[N][W]); //max값 출력
    System.out.print("item : ");
    for (int i = 1; i < N + 1; i++)
        if (selected[i] == 1) //포함된 item만 출력
            System.out.print(i + " ");
    System.out.println();
}
```

```java
public static int max(int a, int b) { //두 정수를 비교해서 더 큰 수를 리턴
    if (a > b)
        return a;
    else
        return b;
}
```

결과화면

```
<terminated> Knapsack [Java Application] C:₩Progr
배낭의 사이즈를 입력하세요(0~50) : 11
  0  0  0  0  0  0  0  0  0  0  0  0
  0  1  1  1  1  1  1  1  1  1  1  1
  0  1  6  7  7  7  7  7  7  7  7  7
  0  1  6  7  7 18 19 24 25 25 25 25
  0  1  6  7  7 18 22 24 28 29 29 40
  0  1  6  7  7 18 22 28 29 34 35 40
max : 40
item : 3 4
```