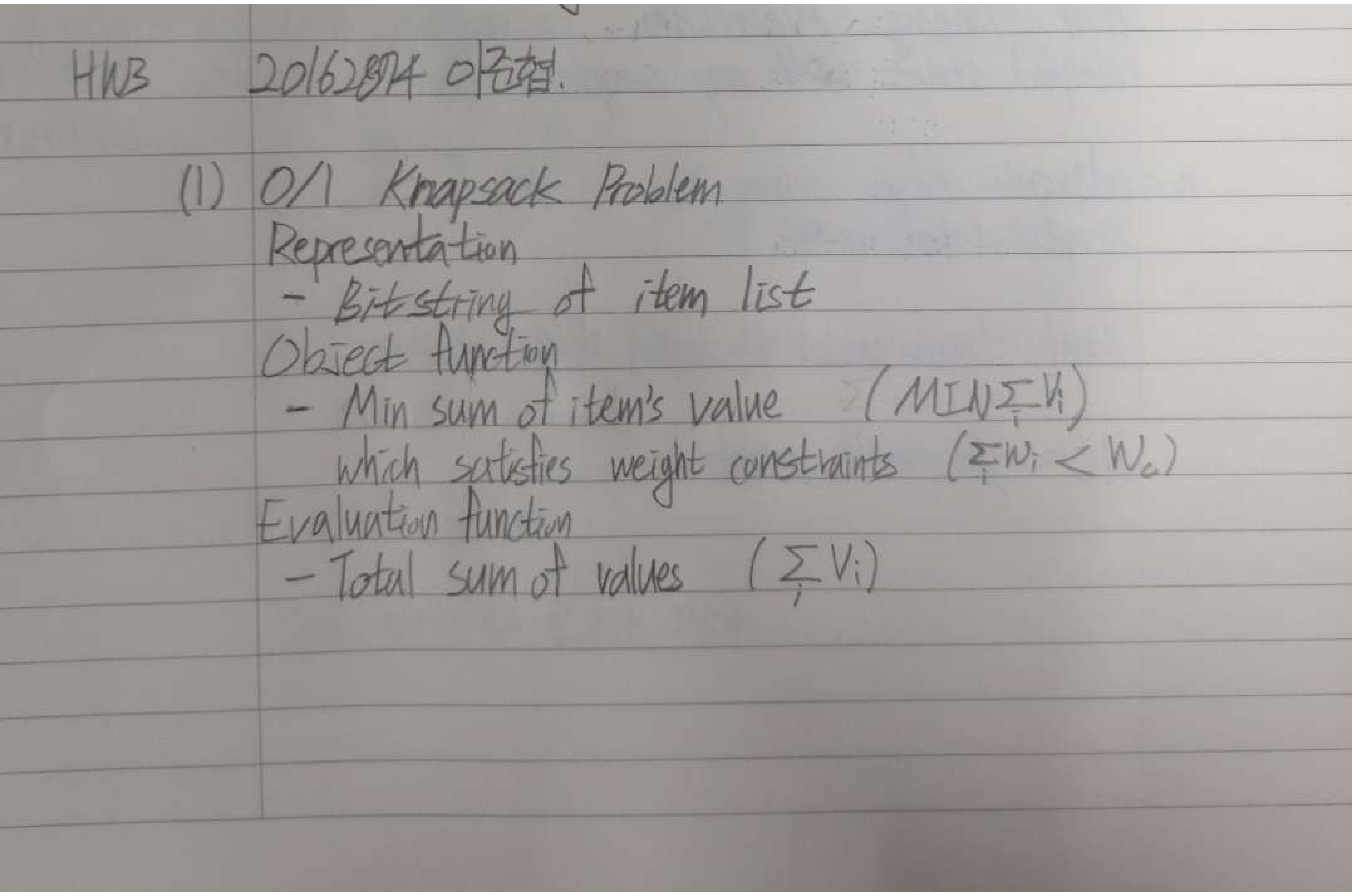

HW3 20162874 이준혁.

(1) 0/1 Knapsack Problem

Representation

- Bitstring of item list

Object function

- Min sum of item's value $(MIN \sum_i V_i)$ which satisfies weight constraints $(\sum_i W_i < W_a)$

Evaluation function

- Total sum of values $(\sum_i V_i)$

(2) Hill-Climbing Algorithm of Knapsack problem

(a)
```
let item # : N
Initial bit string "$\overbrace{000 \cdots 0}^{N}$" = Current
loop do
    Temp ← Current
    loop do
        New ← flip 1 bit of Temp from 0 → 1
        Check if New satisfy Weight Constraint
        Current = New value > Temp value ? New : Temp
    end
    if Current == Temp
        Break!
end
```

(b)
```
vector<string> transformation (string s) {
    int size = s.size(); vector<string> v;
    for (int i=0; i<size; i++)
        if (s[i]=='0') {
            s[i]=='1';
            v.push_back(s);
            s[i]=='0';
        }
    return v;
}
```

input : bit string of current item state
output : bit strings of neighbor item state.

(c) Example

4개의 아이템. 15kg

WN 3/3 ,4/4, 5/6 , 7/8, 9/10

S : 00000

(Neighbor)	temp	10000	01000	00100	00010	00001
	W/V	3/3	4/4	5/6	7/8	9/10

S : 00001

(Neighbor)	temp	10001	01001	00101	~~00011~~
	W/V	12/13	13/14	14/16	~~16/18~~
					overweight

S : 00101

(Neighbor)	temp	~~10101~~	~~01101~~	~~00111~~
	W/V	~~17/19~~	~~18/20~~	~~21/24~~
				overweight

⇒ choose item 3, 5 optimal solution