REPORT

수업명	알고리즘
과제번호	#9
제출일	2018.11.28
학과	컴퓨터학부
교수명	최재영
출석번호	232
이름	이승현

1. 프로그램 개요

① Sum of Subsets 문제를 교재의 Algorithm 5.4를 구현하여 해결하고, Monte Carlo 기법으로 노드의 개수를 추정하는 프로그램입니다.

② 주어진 문제의 조건은 다음과 같습니다

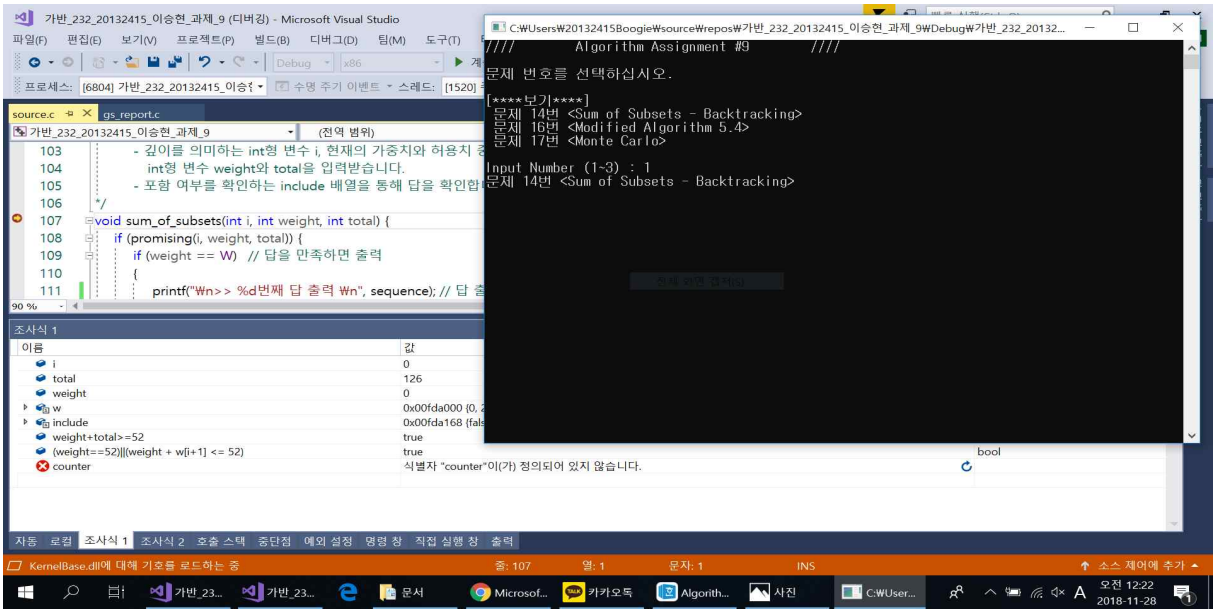
$$W = 52,\ w_1 = 2,\ w_2 = 10,\ w_3 = 13,\ w_4 = 17,\ w_5 = 22,\ w_6 = 42$$

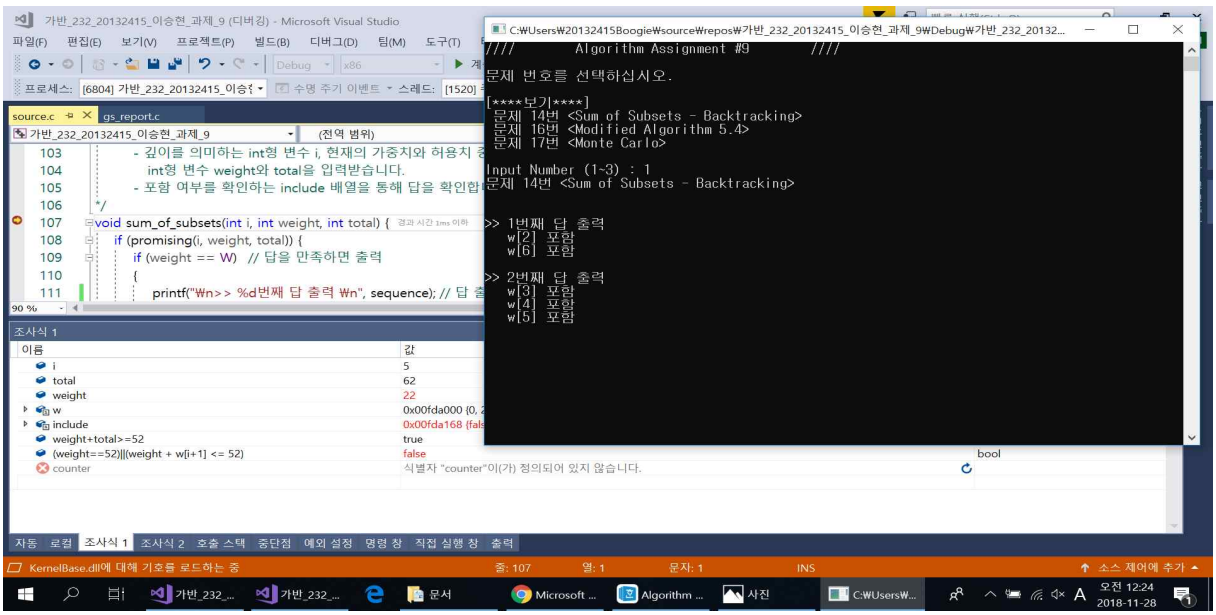
② 사용자에게 문제 번호를 입력받아 결과를 출력합니다.

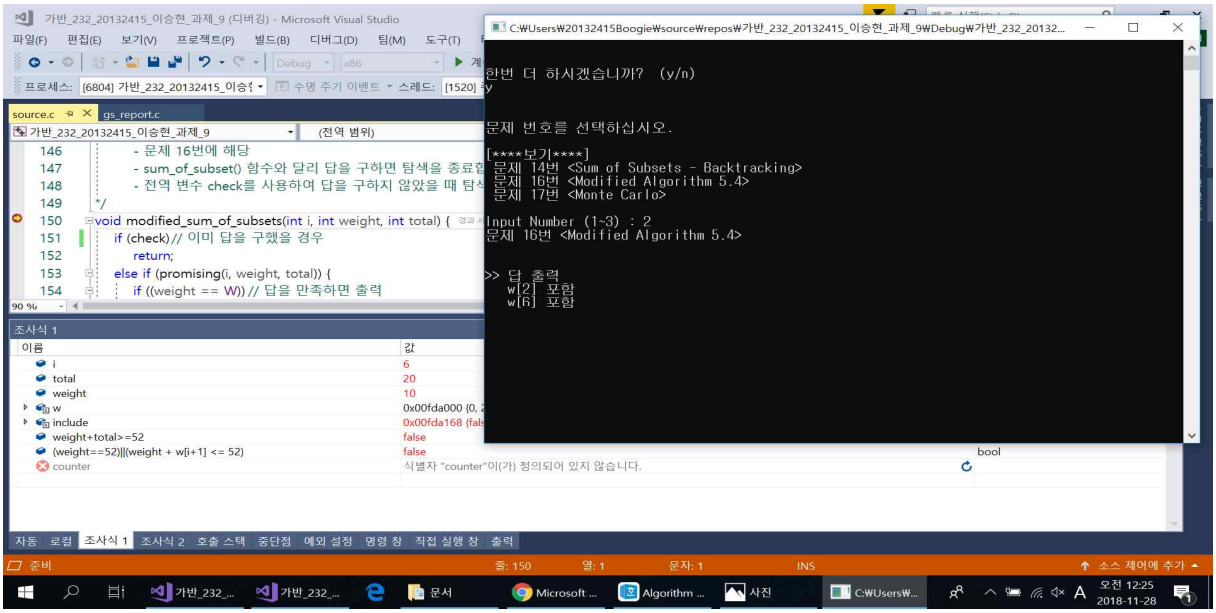
③ 출력이 끝난 후, 사용자에게 재실행 여부를 입력받습니다.

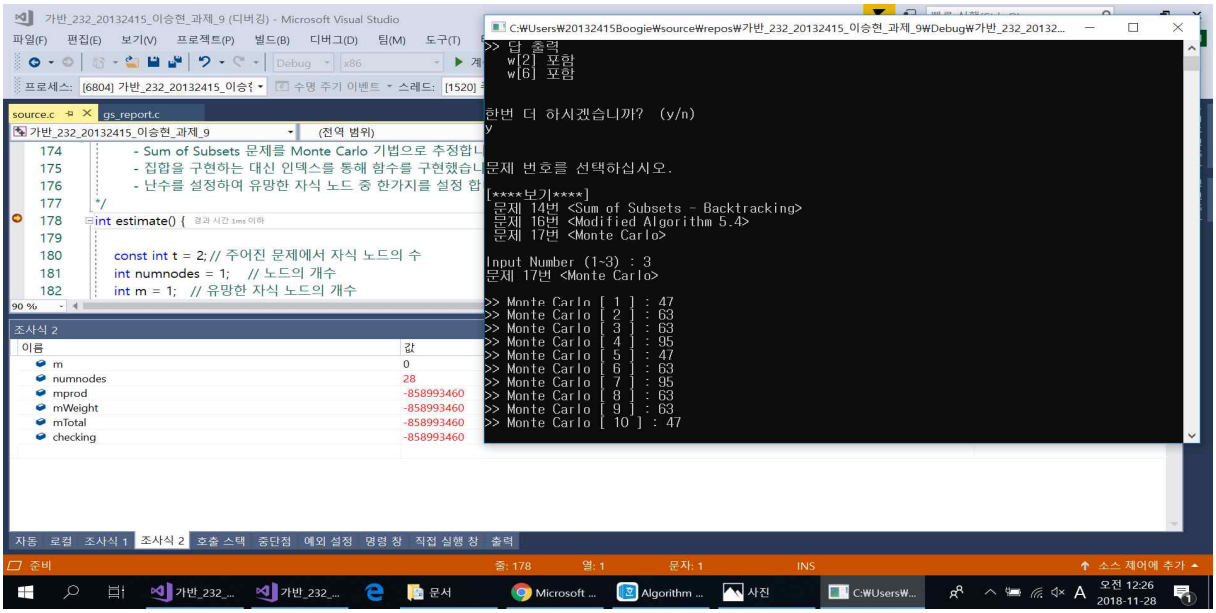
④ C로 구현한 프로그램입니다.

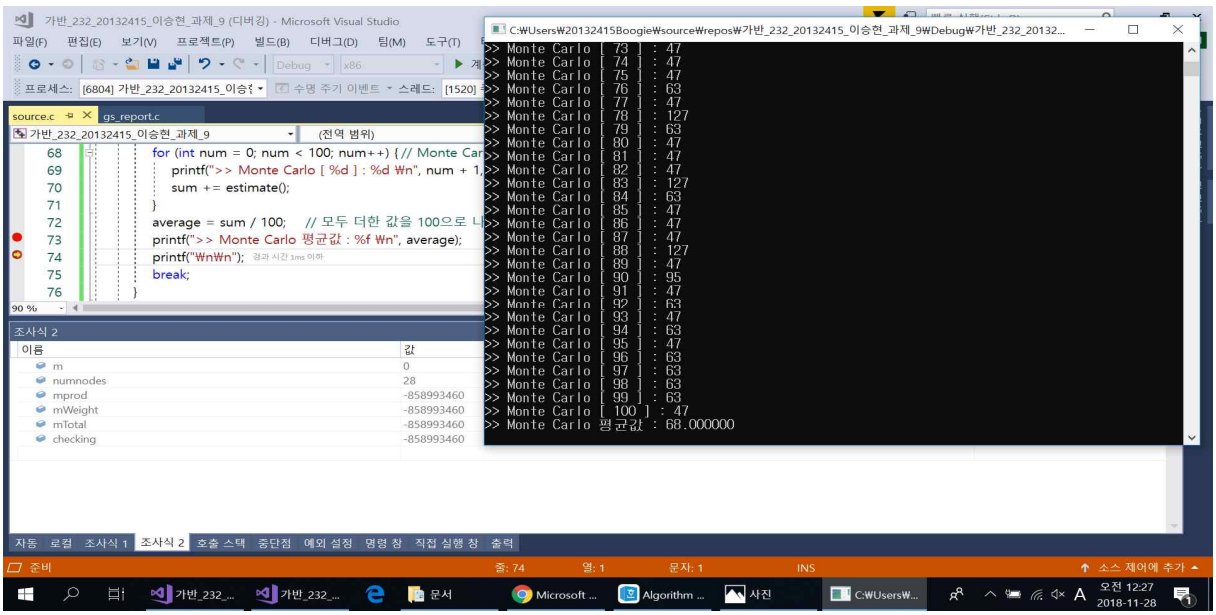
2. 프로그램 실행 화면






C:₩Users₩20132415Boogie₩source₩repos₩가반_232_20132415_이승현_과제_9₩Debug₩가반_232_20132...
한번 더 하시겠습니까? (y/n)
y
문제 번호를 선택하십시오.
[****보기****]
문제 14번 <Sum of Subsets - Backtracking>
문제 16번 <Modified Algorithm 5.4>
문제 17번 <Monte Carlo>
Input Number (1~3) : 2
문제 16번 <Modified Algorithm 5.4>
>> 답 출력
w[2] 포함
w[6] 포함
source.c
qs_report.c
- 문제 16번에 해당
- sum_of_subset() 함수와 달리 답을 구하면 탐색을 종료
- 전역 변수 check를 사용하여 답을 구하지 않았을 때 탐
void modified_sum_of_subsets(int i, int weight, int total) {
if (check)// 이미 답을 구했을 경우
return;
else if (promising(i, weight, total)) {
if ((weight == W)) // 답을 만족하면 출력
조사식 1
이름
값
i
6
total
20
weight
10
w
0x00fda000 {0,
include
0x00fda168 {fals
weight+total>=52
false
(weight==52)||(weight + w[i+1] <= 52)
false
bool
counter
식별자 "counter"이(가) 정의되어 있지 않습니다.
자동 로컬 조사식 1 조사식 2 호출 스택 중단점 예외 설정 명령 창 직접 실행 창 출력
준비
줄: 150 열: 1 문자: 1 INS
소스 제어에 추가
오전 12:25
2018-11-28


C:₩Users₩20132415Boogie₩source₩repos₩가반_232_20132415_이승현_과제_9₩Debug₩가반_232_20132...
>> 답 출력
w[2] 포함
w[6] 포함
한번 더 하시겠습니까? (y/n)
y
문제 번호를 선택하십시오.
[****보기****]
문제 14번 <Sum of Subsets - Backtracking>
문제 16번 <Modified Algorithm 5.4>
문제 17번 <Monte Carlo>
Input Number (1~3) : 3
문제 17번 <Monte Carlo>
>> Monte Carlo [1] : 47
>> Monte Carlo [2] : 63
>> Monte Carlo [3] : 63
>> Monte Carlo [4] : 95
>> Monte Carlo [5] : 47
>> Monte Carlo [6] : 63
>> Monte Carlo [7] : 95
>> Monte Carlo [8] : 63
>> Monte Carlo [9] : 63
>> Monte Carlo [10] : 47
source.c
qs_report.c
- Sum of Subsets 문제를 Monte Carlo 기법으로 추정합니
- 집합을 구현하는 대신 인덱스를 통해 함수를 구현했습니
- 난수를 설정하여 유망한 자식 노드 중 한가지를 설정 합
int estimate() {
const int t = 2;// 주어진 문제에서 자식 노드의 수
int numnodes = 1; // 노드의 개수
int m = 1; // 유망한 자식 노드의 개수
조사식 2
이름
값
m
0
numnodes
28
mprod
-858993460
mWeight
-858993460
mTotal
-858993460
checking
-858993460
자동 로컬 조사식 1 조사식 2 호출 스택 중단점 예외 설정 명령 창 직접 실행 창 출력
준비
줄: 178 열: 1 문자: 1 INS
소스 제어에 추가
오전 12:26
2018-11-28


C:₩Users₩20132415Boogie₩source₩repos₩가반_232_20132415_이승현_과제_9₩Debug₩가반_232_20132...
>> Monte Carlo [73] : 47
>> Monte Carlo [74] : 47
>> Monte Carlo [75] : 47
>> Monte Carlo [76] : 63
>> Monte Carlo [77] : 47
>> Monte Carlo [78] : 127
>> Monte Carlo [79] : 63
>> Monte Carlo [80] : 47
>> Monte Carlo [81] : 47
>> Monte Carlo [82] : 47
>> Monte Carlo [83] : 127
>> Monte Carlo [84] : 63
>> Monte Carlo [85] : 47
>> Monte Carlo [86] : 47
>> Monte Carlo [87] : 47
>> Monte Carlo [88] : 127
>> Monte Carlo [89] : 47
>> Monte Carlo [90] : 95
>> Monte Carlo [91] : 47
>> Monte Carlo [92] : 63
>> Monte Carlo [93] : 47
>> Monte Carlo [94] : 63
>> Monte Carlo [95] : 47
>> Monte Carlo [96] : 63
>> Monte Carlo [97] : 63
>> Monte Carlo [98] : 63
>> Monte Carlo [99] : 63
>> Monte Carlo [100] : 47
>> Monte Carlo 평균값 : 68.000000
for (int num = 0; num < 100; num++) {// Monte Car
printf(">> Monte Carlo [%d] : %d ₩n", num + 1,
sum += estimate();
}
average = sum / 100; // 모두 더한 값을 100으로 나
printf(">> Monte Carlo 평균값 : %f ₩n", average);
printf("₩n₩n");
break;
}
조사식 2
이름
값
m
0
numnodes
28
mprod
-858993460
mWeight
-858993460
mTotal
-858993460
checking
-858993460
자동 로컬 조사식 1 조사식 2 호출 스택 중단점 예외 설정 명령 창 직접 실행 창 출력
준비
줄: 74 열: 1 문자: 1 INS
소스 제어에 추가
오전 12:27
2018-11-28

3. 결론

3-1) 문제 13번은 손으로 구현하였습니다.

3-2) 문제 14번

- 두 개의 정답이 있습니다.
- ① $w_2 + w_6 = 10 + 42 = 52 = W$
- ② $w_3 + w_4 + w_5 = 13 + 17 + 22 = 52 = W$

3-3) 문제 16번

- 정답을 하나 찾으면 탐색을 중지합니다.
- $w_2 + w_6 = 10 + 42 = 52 = W$

3-4) 문제 17번

- 100회의 횟수를 반복했을 때, 결과값의 평균은 대체적으로 62 ~ 69의 값을 가지는 것을 확인 할 수 있습니다.

4. 소스코드

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <time.h>

#define W 52    // 문제에서 주어진 최대 허용량
#define nSize 8  // 배열의 크기
#define INF 1000000      // 무한수

int w[nSize] = { 0, 2, 10, 13, 17, 22, 42, INF };    // 가중치 배열
bool include[nSize] = {false, false, false, false, false , false , false, false };      // 포함 유무를 확인하는 배열
int weight = 0;  // 현재의 누적 가중치
int sequence = 1;          // 정답의 순서를 나타내기 위한 변수
bool check = false;        // 최초로 정답을 구했을 시 탐색 중단을 위한 check 변수

void sum_of_subsets(int i, int weight, int total);
void modified_sum_of_subsets(int i, int weight, int total);
bool promising(int i, int weight, int total);
int estimate();

int main(void) {
        int total;// 현재 상태에서 넣을 수 있는 가중치들의 합
        int num;
        int sum;          // Monte Carlo 기법을 통해 계산한 노드 개수들의 합

        float average;    // 평균치를 저장하기 위한 변수

        srand(time(NULL));

        char input = 'y';          // 프로그램 종료 여부를 확인하기 위한 char형 변수

        int choose = 0;            //  문제 번호 선택을 위한 int형 변수
        printf("////%8sAlgorithm Assignment #9%8s//// ₩n", " ", " ");
        while (input == 'y')       // while loop : 'y' 입력 시 프로그램 반복
        {
                printf("₩n문제 번호를 선택하십시오. ₩n₩n");
                printf("[****보기****]₩n%51. 문제 14번 <Sum of Subsets - Backtracking>₩n%52. 문제 16번
<Modified Algorithm 5.4>₩n%53. 문제 17번 <Monte Carlo>₩n₩n");
                printf("Input Number (1~3) : ");
                scanf("%d", &choose);    // 과제의 문제 번호를 선택
                switch (choose)          // 입력한 숫자에 따라 프로그램을 실행합니다.
                {
                case 1:                  // case 1 : 문제 14번
                {
                        printf("문제 14번 <Sum of Subsets - Backtracking>₩n₩n");
                        total = 126;
                        weight = 0;
                        for (num = 0; num < nSize; num++)       // include 배열 초기화
                                include[num] = false;
                        sum_of_subsets(0, weight, total);
                        printf("₩n₩n");
                        break;
                }
                case 2:
                {
                        printf("문제 16번 <Modified Algorithm 5.4>₩n₩n");
                        total = 126;
```

```
                        weight = 0;
                        for (num = 0; num < nSize; num++)       // include 배열 초기화
                                include[num] = false;
                        modified_sum_of_subsets(0, weight, total);
                        printf("₩n₩n");
                        break;
                }
                case 3:
                {
                        sum = 0;
                        printf("문제 17번 <Monte Carlo>₩n₩n");
                        for (int num = 0; num < 100; num++) {  // Monte Carlo 기법을 100회동안 반복
                                // n번째 계산 값 출력
                                printf(">> Monte Carlo [ %d ] : %d ₩n", num + 1, estimate());
                                sum += estimate();
                        }
                        average = sum / 100;              // 모두 더한 값을 100으로 나누어 평균치를 구한다
                        printf(">> Monte Carlo 평균값 : %f ₩n", average);
                        printf("₩n₩n");
                        break;
                }
                }
                while (1) {                 // while loop : 프로그램 종료 여부 선택
                        printf("한번 더 하시겠습니까?  (y/n)₩n");
                        scanf("%c", &input);
                        if (input == 32 || input == 10)
                                scanf("%c", &input);
                        if (input != 'y' && input != 'n')
                                printf("잘못 입력하셨습니다.₩n");
                        else
                        {
                                printf("₩n");
                                break;
                        }
                }
        }
        return 0;
}

/*
        sum_of_subsets () 함수
                - Algorithm 5.4를 구현한 함수
                - 문제 14번에 해당
                - BackTracking 기법을 재귀를 통해 구현
                - 깊이를 의미하는 int형 변수 i, 현재의 가중치와 허용치 중 남은 가중치를 의미하는
                  int형 변수 weight와 total을 입력받습니다.
                - 포함 여부를 확인하는 include 배열을 통해 답을 확인합니다.
*/
void sum_of_subsets(int i, int weight, int total) {
        if (promising(i, weight, total)) {
                if (weight == W)// 답을 만족하면 출력
                {
                        printf("₩n>> %d번째 답 출력 ₩n", sequence);       // 답 출력을 구분
                        for (int j = 0; j < nSize; j++)
                                if (include[j] == true)
                                        printf("    w[%d] 포함 ₩n", j);
                        sequence++;
                }
                else {
```

```
                        include[i + 1] = true;      // case1 : 넣을 때
                        sum_of_subsets(i + 1, weight + w[i + 1], total - w[i + 1]);
                        include[i + 1] = false;             // case2 : 넣지 않을 때
                        sum_of_subsets(i + 1, weight, total - w[i+1]);
                }
        }
}
/*
        promising()함수
                - Algorithm 5.4를 구현한 함수
                - 노드가 pomising한지 아닌지를 판단하는 함수
                - 깊이를 의미하는 int형 변수 i, 현재의 가중치와 허용치 중 남은 가중치를 의미하는
                  int형 변수 weight와 total을 입력받습니다.
                - 남은 가중치와 현재 가중치가 목표 보다 적은 경우와
                - 모든 깊이를 탐색했으나 목표치를 못채운 경우 또는 채워도 목표치보다 적을 경우
                - false를 리턴합니다.
                - 문제에서 주어진 깊이보다 더 깊게 내려가도 false를 리턴합니다.
*/
bool promising(int i, int weight, int total) {
        if (i == 7 )
                return false;
        else
                return (weight + total >= W) && (weight == W || weight + w[i + 1] <= W);
}

/*
        modified_sum_of_subsets () 함수
                - Algorithm 5.4를 수정하여 구현한 함수
                - 문제 16번에 해당
                - sum_of_subset() 함수와 달리 답을 구하면 탐색을 종료합니다.
                - 전역 변수 check를 사용하여 답을 구하지 않았을 때 탐색 진행
*/
void modified_sum_of_subsets(int i, int weight, int total) {
        if (check)          // 이미 답을 구했을 경우
                return;
        else if (promising(i, weight, total)) {
                if ((weight == W))          // 답을 만족하면 출력
                {
                        printf("₩n>> 답 출력 ₩n");
                        for (int j = 0; j < nSize; j++)
                                if (include[j] == true)
                                        printf("    w[%d] 포함 ₩n", j);
                        check = true;
                }
                else {
                        include[i + 1] = true;      // case1 : 넣을 때
                        modified_sum_of_subsets(i + 1, weight + w[i + 1], total - w[i + 1]);
                        include[i + 1] = false;             // case2 : 넣지 않을 때
                        modified_sum_of_subsets(i + 1, weight, total - w[i+1]);
                }
        }
}
/*
        estimate() 함수
                - Algorithm 5.2 를 구현한 함수
                - 문제 17번에 해당
                - Sum of Subsets 문제를 Monte Carlo 기법으로 추정합니다.
                - 집합을 구현하는 대신 인덱스를 통해 함수를 구현했습니다.
                - 난수를 설정하여 유망한 자식 노드 중 한가지를 설정 합니다.
```

```
*/
int estimate() {

	const int t = 2;  // 주어진 문제에서 자식 노드의 수
	int numnodes = 1;	// 노드의 개수
	int m = 1;	// 유망한 자식 노드의 개수
	int mprod = 1;  // m의 누적값
	int mWeight = 0;	// 유망한지 판단하는 데 쓰이는 가중치 변수
	int mTotal = 126; // 유망한지 판단하는 데 쓰이는 남아있는 가중치 총합 변수
	int checking;	// 난수를 저장하기 위한 변수
	/*
		for 반복문
			- 인덱스 i가 0부터 문제에서 주어진 깊이까지 반복
			- 유망한 자식이 더 이상 없으면 함수 종료
	*/
		for (int i = 0; i < nSize; i++) {
			mprod = mprod * m;	// 유망한 자식 노드의 수인 m을 곱한 값의 누적값
			numnodes = numnodes + mprod * t;	// node의 수 += mprod+t
			// 노드를 포함하는 경우가 유망할 때
			if (promising(i + 1, mWeight + w[i + 1], mTotal - w[i + 1]))
			{
				// 노드를 포함하지 않아도 유망한 경우
				if (promising(i + 1, mWeight, mTotal - w[i + 1]))
				{
					m = 2;  // 유망한 자식 노드의 수
					checking = rand() % m;  // 난수 설정
					switch (checking) {
					case 0:   // 0 : w[i+1]을 포함하지 않는 경우
						mTotal = mTotal - w[i + 1];
						break;
					case 1:   // 0 : w[i+1]을 포함하는 경우
						mWeight = mWeight + w[i + 1];
						mTotal = mTotal - w[i + 1];
						break;
					}
				}
				else {	// w[i+1 ]노드를 포함하는 경우만 유망할 때
					m = 1;
					mWeight = mWeight + w[i + 1];
					mTotal = mTotal - w[i + 1];
				}
			}
			// w[ i+1 ]노드를 포함하지 않는 경우만 유망할 때
			else if (promising(i + 1, mWeight, mTotal - w[i + 1]))
			{
				m = 1;
				mWeight = mWeight + w[i + 1];
				mTotal = mTotal - w[i + 1];
			}
			else {	// 유망한 노드가 없을 때
				m = 0;
				return numnodes;
			}
		}
	return numnodes;
}
```