

Aayush Shah

19BCE245

11 November 2021

Design and Analysis of Algorithms

Practical 9

• Code :

```
*
Aayush Shah
19BCE245
DAA Practical 9    | Knapsack using Dynamic Programming
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include <stdbool.h>

/* used by printm routine */
#define INDEX(a,i,j,lda)  a[ (i)*lda + (j) ]
#define INDENT(n)          for(int _ = 0; _ < n ; _++)
printf(" ")

/* used by knapsack01 routine */
#define M(i, j)            M[ (i)*(w+1) + (j) ]
#define max(a, b)          (((a) > (b)) ? (a) : (b))

void printm(const char *name,
            const int *a,
            const int lda,
            const int m,
            const int n)
{
    const int len = strlen(name) + 1;
    const int mid = m / 2;
```

```

    INDENT(len);
    printf("/ ");

    for( int i=0 ; i<n ; i++ ) printf(" ");
    printf(" \\n");
    for (int i=0 ; i<m ; i++) {
        if ( i == mid ) printf("%s ", name);
        else INDENT(len);
        printf("| ");
        for (int j=0 ; j<n ; j++) {
            printf("%5d ", INDEX(a,i,j,lda));
        }
        printf(" |\n");
    }
    INDENT(len);
    printf("\\ ");
    for( int i=0 ; i<n ; i++ ) printf(" ");
    printf("/\n");
}

int knapsack01(const int w,
               const int *weights,
               const int *value,
               const int n)
{
    int *M = ( int * ) malloc( (n+1)*(w+1) *
sizeof(int) );

    for(int i = 0 ; i < w+1 ; i++) M(0, i) = 0;
    for(int j = 0 ; j < n+1 ; j++) M(j, 0) = 0;

    // DP
    for ( int n_items = 1 ; n_items < n+1 ; n_items++ ) {
        for (int weight = 1 ; weight < w+1 ; weight++ ) {
            if ( weights[ n_items-1 ] <= weight ) {
                M( n_items , weight ) = max(
                    value[ n_items-1 ] + M( n_items-1 ,
weight - weights[ n_items-1 ] ) ,
                    M( n_items-1 , weight )
                );
            }
            else {

```

```

        M( n_items , weight ) = M( n_items-1 ,
weight );
    }
}

int max_loot = M( n, w );

printf("Memoization Table Entries ==> ", M, w+1, n+1,
w+1);

int res = max_loot;
int w_remaining = w;
bool flag = false;

printf("\n\t> Selected Weights : ");
for ( int n_items = n; n_items > 0 && res > 0;
n_items--) {
    if (res == M(n_items-1,w_remaining)){
        continue;
    }
    else {
        // This item is included.
        if(flag){
            printf(", %d", weights[n_items - 1]);
        }
        else{
            printf("%d", weights[n_items - 1]);
            flag = true;
        }

        // Since this weight is included its value is
deducted
        res = res - value[n_items - 1];
        w_remaining = w_remaining - weights[n_items -
1];
    }
}

free(M);

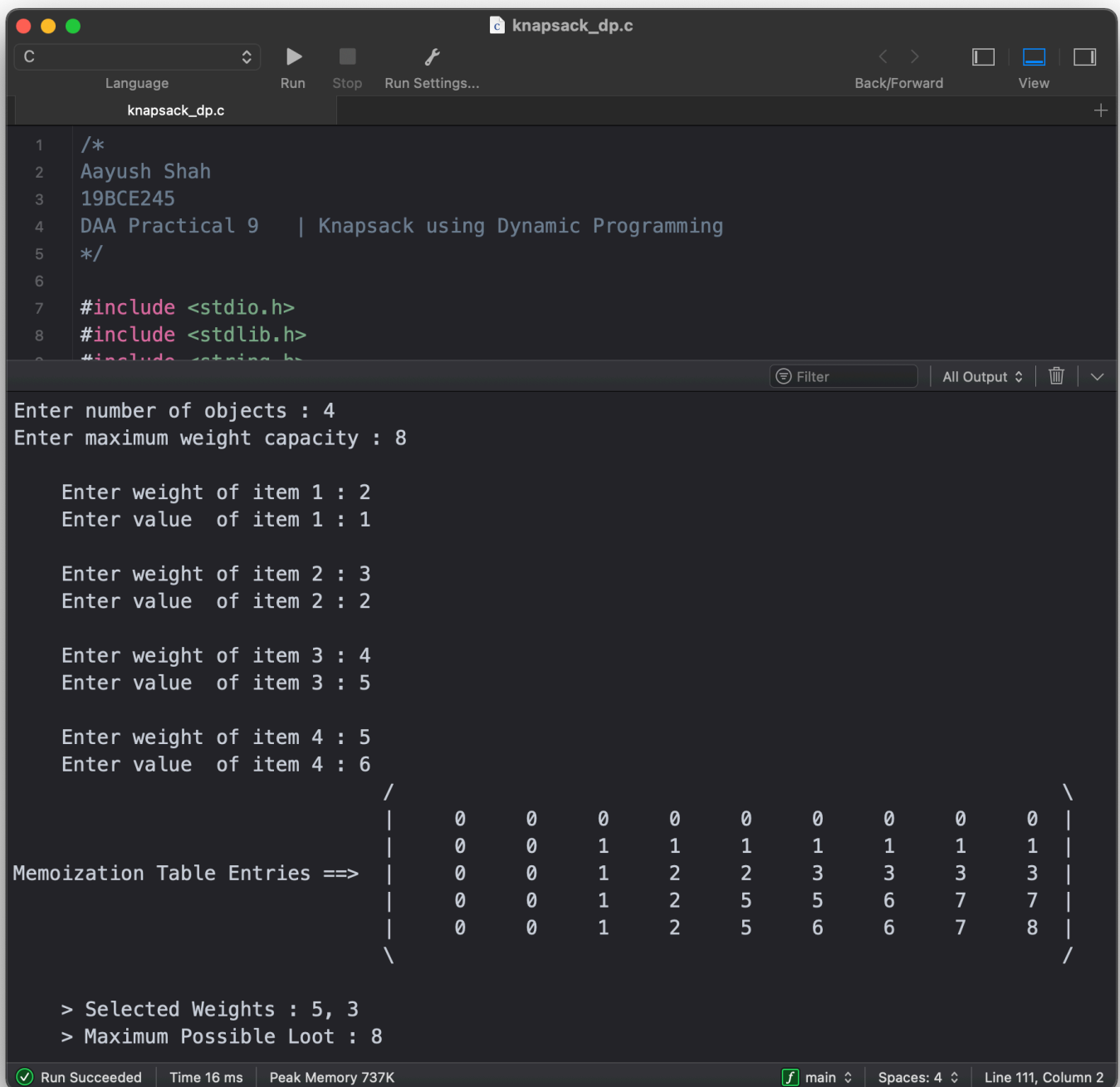
return max_loot;
}

```

```
int main()
{
    // int n = 4;          // n : number of objects
    // int w = 8;          // w : maximum weight capacity
    // int weights[] = {2, 3, 4, 5};
    // int value[] = {1, 2, 5, 6};

    int n,w;
    printf("Enter number of objects : ");
    scanf("%d",&n);
    int weights[n],value[n];
    printf("Enter maximum weight capacity : ");
    scanf("%d",&w);
    for(int i=0;i<n;i++){
        printf("\n\tEnter weight of item %d : ", i+1);
        scanf("%d",&weights[i]);
        printf("\tEnter value of item %d : ", i+1);
        scanf("%d",&value[i]);
    }
    printf("\n\t> Maximum Possible Loot : %d\n\n",
knapsack01(w, weights, value, n));
    return 0;
}
```

• Output :



```

1  /*
2  Aayush Shah
3  19BCE245
4  DAA Practical 9 | Knapsack using Dynamic Programming
5  */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9  #include <string.h>
10
11 int main() {
12     int n, W;
13     printf("Enter number of objects : ");
14     scanf("%d", &n);
15     printf("Enter maximum weight capacity : ");
16     scanf("%d", &W);
17
18     int wt[n], val[n];
19     for (int i = 1; i <= n; i++) {
20         printf("Enter weight of item %d : ", i);
21         scanf("%d", &wt[i]);
22         printf("Enter value of item %d : ", i);
23         scanf("%d", &val[i]);
24     }
25
26     // Memoization Table Entries ==>
27     int dp[n+1][W+1];
28     for (int i = 0; i <= n; i++) {
29         for (int j = 0; j <= W; j++) {
30             dp[i][j] = 0;
31         }
32     }
33
34     for (int i = 1; i <= n; i++) {
35         for (int j = 1; j <= W; j++) {
36             if (wt[i] > j) {
37                 dp[i][j] = dp[i-1][j];
38             } else {
39                 dp[i][j] = max(dp[i-1][j], dp[i-1][j-wt[i]] + val[i]);
40             }
41         }
42     }
43
44     printf("\n> Selected Weights : ");
45     int i = n, j = W;
46     while (i > 0 & j > 0) {
47         if (dp[i][j] != dp[i-1][j]) {
48             printf("%d, ", wt[i]);
49             j -= wt[i];
50         }
51         i--;
52     }
53     printf("\n> Maximum Possible Loot : %d\n", dp[n][W]);
54     return 0;
55 }

```

Enter number of objects : 4
Enter maximum weight capacity : 8

Enter weight of item 1 : 2
Enter value of item 1 : 1

Enter weight of item 2 : 3
Enter value of item 2 : 2

Enter weight of item 3 : 4
Enter value of item 3 : 5

Enter weight of item 4 : 5
Enter value of item 4 : 6

Memoization Table Entries ==>

	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1
1	0	0	1	2	2	3	3	3	3	3
2	0	0	1	2	5	5	6	7	7	7
3	0	0	1	2	5	6	6	7	8	8

> Selected Weights : 5, 3
> Maximum Possible Loot : 8

Run Succeeded | Time 16 ms | Peak Memory 737K | main | Spaces: 4 | Line 111, Column 2