2. Knapsack

Time complexity: O(nlogn)

Algorithm:

1. Input:
    - Take input for the number of items (`n`).
    - Take input for the weights and values of each item.
    - Take input for the capacity of the knapsack (`W`).
    -
2. Initialize:
    - Create a 2D array `K` of size `(n+1) x (W+1)` to store the maximum value that can be obtained for different weights and number of items.
    -
3. Base Case Initialization:
    - Set `K[0][w] = 0` for all `w` (weight) from 0 to `W`, since with 0 items, the maximum value that can be obtained is 0.(// initialize w)
    - Set `K[i][0] = 0` for all `i` (number of items) from 0 to `n`, since with 0 capacity, the maximum value that can be obtained is 0.(// initialize n)
    -
4. Dynamic Programming Iteration:
    - Iterate `i` from 1 to `n` (number of items) and `w` from 1 to `W` (capacity of knapsack):
        - If `wt[i - 1] <= w` (weight of the current item is less than or equal to the current capacity `w`):
            - `K[i][w] = max(val[i - 1] + K[i - 1][w - wt[i - 1]], K[i - 1][w]).`
        - Else:
            - `K[i][w] = K[i - 1][w].`

5. Return Result:
    - Return `K[n][W]`, which represents the maximum value that can be obtained with `n` items and a knapsack capacity of `W`.

Code:

```
#include <stdio.h>
#define MAX_ITEMS 100

int max(int a, int b) {
    return (a > b) ? a : b;
}
int knapsack(int W, int wt[], int val[], int n) {
    int i, w;
    int K[n + 1][W + 1];

    for (i = 0; i <= n; i++) {
        for (w = 0; w <= W; w++) {
            if (i == 0 || w == 0)
                K[i][w] = 0;
            else if (wt[i - 1] <= w)
                K[i][w] = max(val[i - 1] + K[i - 1][w - wt[i - 1]], K[i - 1][w]);
            else
                K[i][w] = K[i - 1][w];
        }
    }
    return K[n][W];
}

int main() {
    int n, W;
    printf("Enter the number of items: ");
    scanf("%d", &n);

    int wt[MAX_ITEMS], val[MAX_ITEMS];
    printf("Enter the weights and profit of items:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &wt[i], &val[i]);
    }

    printf("Enter the capacity of the knapsack: ");
    scanf("%d", &W);
    printf("Maximum value that can be obtained: %d\n", knapsack(W, wt, val, n));
    return 0;
}
```