

Actividad 1.6 Algoritmos Greedy

Victor Misael Escalante Alvarado

A01741176

KnapSack

The screenshot displays a C++ IDE with the following components:

- Explorer:** Shows the project structure with files like `Knapsack.cpp`, `Knapsack.h`, and `main.cpp`.
- Code Editor:** Contains the implementation of the `Knapsack` class.


```

1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <fstream>
5
6 using namespace std;
7
8 // Variables globales
9 int W;
10 int n;
11
12 vector<int> values;
13 vector<int> weights;
14
15 // Función para leer los valores y pesos desde un archivo de texto
16 void LeerArf() {
17     ifstream archivo("Mochila.txt");
18
19     archivo >> n >> W; // Leer número de objetos y capacidad de la mochila
20
21     values.resize(n);
22     weights.resize(n);
23
24     for (int i = 0; i < n; i++) {
25         archivo >> values[i] >> weights[i];
26     }
27     archivo.close();
28
29     // Función comparadora para ordenar los objetos por su ratio valor/peso
30 bool Comparar(int a, int b) {
31     double ratioA = (double)values[a] / weights[a];
32     double ratioB = (double)values[b] / weights[b];
33     return ratioA > ratioB;
34 }
35
36 // Función greedy para el problema de la mochila 0/1
37 int knapsackGreedy(int W, int n, vector<int> values, vector<int> weights) {
38     vector<int> indices(n);
39     for (int i = 0; i < n; i++) {
40         indices[i] = i;
41     }

```
- Output Console:** Shows the execution results:


```

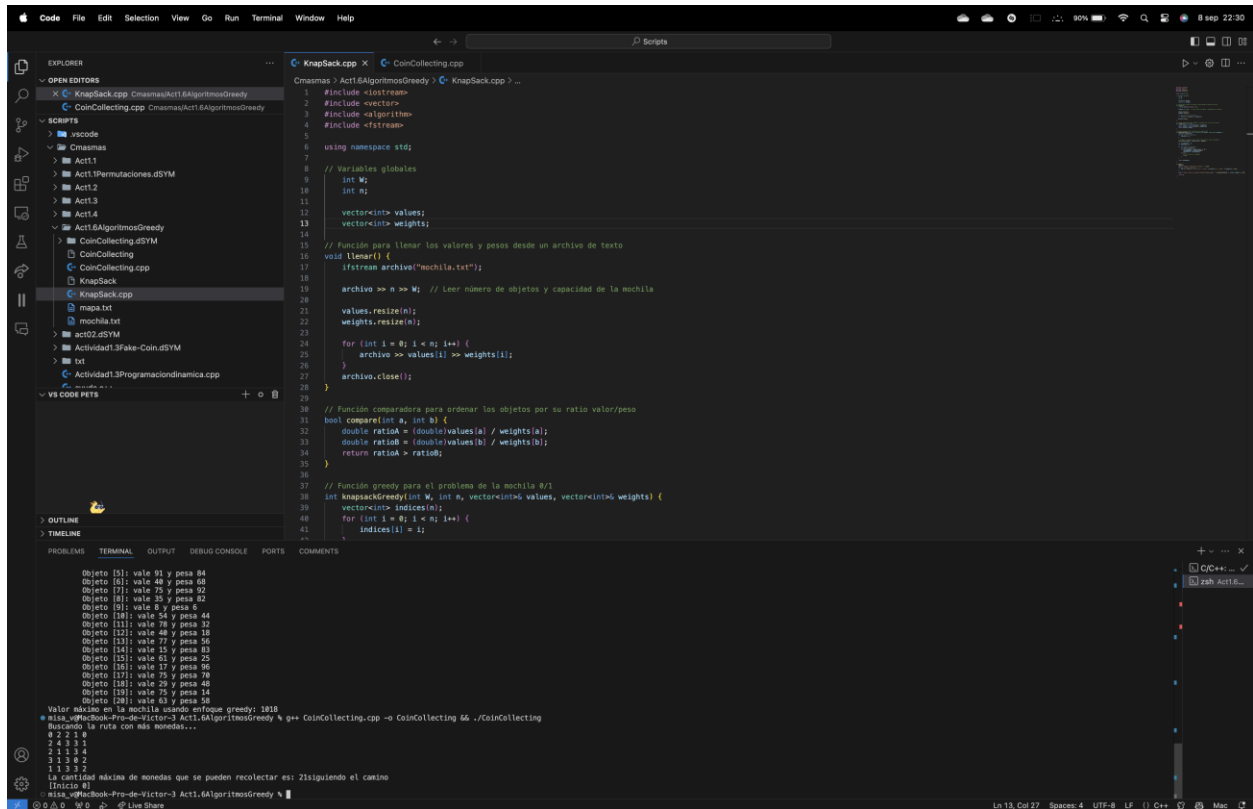
Objeto [0]: vale 75 y pesa 14
Objeto [1]: vale 43 y pesa 58
Valor máximo en la mochila usando enfoque greedy: 1018

```

¿Por qué es greedy? ¿En casos casos donde no se logra el óptimo? (recuerda que estos algoritmos son métodos aproximados).

Se considera Greedy puesto que tomamos en cuenta siempre el mejor aspecto de ratio entre valor y peso para tomar decisiones

Coin Collecting



```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <fstream>
5
6 using namespace std;
7
8 // Variables globales
9 int W;
10 int n;
11
12 vector<int> values;
13 vector<int> weights;
14
15 // Función para llenar los valores y pesos desde un archivo de texto
16 void llenar() {
17     ifstream archivo("mochila.txt");
18     archivo >> n >> W; // Leer número de objetos y capacidad de la mochila
19     values.resize(n);
20     weights.resize(n);
21
22     for (int i = 0; i < n; i++) {
23         archivo >> values[i] >> weights[i];
24     }
25     archivo.close();
26 }
27
28 // Función comparadora para ordenar los objetos por su ratio valor/peso
29 bool compare(int a, int b) {
30     double ratioA = (double)values[a] / weights[a];
31     double ratioB = (double)values[b] / weights[b];
32     return ratioA > ratioB;
33 }
34
35 // Función greedy para el problema de la mochila 0/1
36 int knapsackGreedy(int W, int n, vector<int> values, vector<int> weights) {
37     vector<int> indices(n);
38     for (int i = 0; i < n; i++) {
39         indices[i] = i;
40     }
41     sort(indices.begin(), indices.end(), compare);
42
43     int totalValue = 0;
44     for (int i = 0; i < n; i++) {
45         int index = indices[i];
46         if (weights[index] <= W - totalValue) {
47             totalValue += values[index];
48         }
49     }
50     return totalValue;
51 }
```

Objeto [1]: vale 91 y pesa 84
Objeto [2]: vale 48 y pesa 48
Objeto [3]: vale 75 y pesa 32
Objeto [4]: vale 35 y pesa 32
Objeto [5]: vale 8 y pesa 6
Objeto [6]: vale 54 y pesa 44
Objeto [7]: vale 78 y pesa 32
Objeto [8]: vale 77 y pesa 56
Objeto [9]: vale 35 y pesa 32
Objeto [10]: vale 61 y pesa 25
Objeto [11]: vale 17 y pesa 36
Objeto [12]: vale 75 y pesa 78
Objeto [13]: vale 29 y pesa 48
Objeto [14]: vale 75 y pesa 14
Objeto [15]: vale 63 y pesa 36

Valor máximo en la mochila usando enfoque greedy: 1818
Retorno la ruta con más monedas...

2 4 3 1
2 1 3 4
3 1 3 2
1 1 3 2

La cantidad máxima de monedas que se pueden recolectar es: 218igiendo el camino

[Inicio 0]

1818

¿Por qué es greedy? ¿En casos donde no se logra el óptimo? (recuerda que estos algoritmos son métodos aproximados).

Se considera greedy puesto que siempre se toma el camino en que se tengan más posibles monedas por esa columna o fila

Enlace a codespace :

<https://prod.liveshare.vsengsaas.visualstudio.com/join?66CE608AD8258CB35EB535D41C40CA014567>