

# Instituto Politécnico Nacional Escuela Superior de Cómputo



Unidad de académica:

Análisis y diseño de algoritmos

## Alumno:

Solares Velasco Arturo Misael 2023630538

Solis Lugo Mayra 2023630449

Grupo: 3CV1

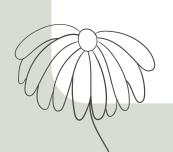
Profesor: Floriano García Andres

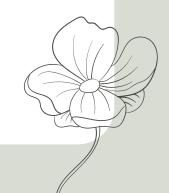
Pa programación dinámica es una técnica ampliamente utilizada para resolver una variedad de problemas de optimización en informática. Se basa en la idea de descomponer un problema en subproblemas más pequeños y resolverlos de forma recursiva, almacenando las soluciones intermedias para evitar recálculos innecesarios. En este reporte, se presenta un análisis detallado de la implementación y el rendimiento de algoritmos de programación dinámica para resolver dos problemas comunes: el problema del cambio y el problema de la mochila 0/1. Se comparan los resultados obtenidos al implementar los algoritmos en tres lenguajes de programación populares: Python, Java y C.

Se implementaron programas en los tres lenguajes para resolver los problemas del cambio y la mochila 0/1 utilizando la técnica de programación dinámica. Cada programa fue diseñado para aceptar diferentes escenarios de prueba y calcular el tiempo de ejecución y el consumo de memoria para cada uno. Se utilizaron 5 escenarios diferentes para cada problema, con conjuntos de datos variados para evaluar el rendimiento de los algoritmos en situaciones diversas. Los programas fueron ejecutados y los resultados fueron registrados para su análisis.

import java.util.Arrays;

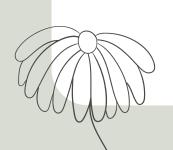
```
public static int cambioMonedas(int[] monedas, int cantidad) {
import time
import tracemalloc
                                                                                int[] dp = new int[cantidad + 1];
                                                                                Arrays.fill(dp, cantidad + 1);
def cambio_monedas(monedas, cantidad):
                                                                                dp[0] = 0;
    dp = [float('inf')] * (cantidad + 1)
                                                                                 for (int i = 1; i <= cantidad; i++) {</pre>
   dp[0] = 0
                                                                                    for (int moneda : monedas) {
   for i in range(1, cantidad + 1):
                                                                                        if (i - moneda >= 0) {
       for moneda in monedas:
                                                                                            dp[i] = Math.min(dp[i], dp[i - moneda] + 1);
           if i - moneda >= 0:
               dp[i] = min(dp[i], dp[i - moneda] + 1)
   return dp[cantidad] if dp[cantidad] != float('inf') else -1
                                                                                 return dp[cantidad] > cantidad ? -1 : dp[cantidad];
escenarios = [
                                                                             public static void main(String[] args) {
                                                                                 int[][] escenarios = {
```

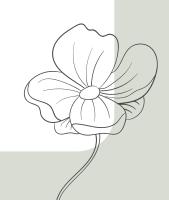






```
int cambio_monedas(int monedas[], int n, int cantidad) {
   int *dp = (int *)malloc((cantidad + 1) * sizeof(int));
   for (int i = 0; i <= cantidad; i++) {</pre>
       dp[i] = INT_MAX;
   dp[0] = 0;
                                                                                dp[i][j] = 0;
   for (int i = 1; i <= cantidad; i++) {</pre>
       for (int j = 0; j < n; j++) {
           if (i - monedas[j] \geq 0 && dp[i - monedas[j]] != INT_I
               dp[i] = dp[i] < dp[i - monedas[j]] + 1 ? dp[i] : 
   int resultado = dp[cantidad] == INT_MAX ? -1 : dp[cantidad];
   free(dp);
                                                                                 } else {
   return resultado;
int main() {
   int escenarios[5
```







# Resultados programa de la mochila

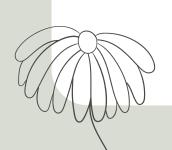
```
#Include <String.n>
    7 int mochila01(int* pesos, int* valores, int n, int capacidad) {
8    int** dp = (int**)malloc((n + 1) * sizeof(int*));
Capacidad
                     Pesos
                                            Valores
                                                                 Resultado
                                                                                       Tiempo Memoria
                                           [10, 20, 30]
                                                                                       0.01 ms0 KB
                                                                 40
10
                                                                 11
                                                                                       0.00 ms0 KB
                                           [1, 4, 5, 7]
                     [3, 4, 6, 8]
[5, 5, 5]
[6, 7, 8]
15
                                                                                      0.00 ms0 KB
                                           [10, 20, 30]
[10, 12, 15]
10
                                                                 50
                                                                                       0.00 ms0 KB
                                                                                       0.00 ms0 KB
```

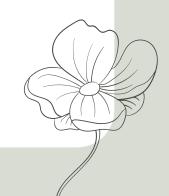
```
main.py
  4 -
      def mochila_01(pesos, valores, capacidad):
               len(pesos)
                                 range(capacidad + 1)] for
                                                                  in range(n + 1)
                                                                input

    * * * * *

Capacidad
                                                                 Tiempo
                                 Valores
                                                Resultado
                                                                                 Memoria
                                [10, 20, 30]
                [1, 2, 3]
                                                40
                                                                 0.03 ms
                                                                                 0.29 KB
                                [1, 4, 5, 7]
10
                                                                 0.02 ms
                                                                                 0.32 KB
                                                11
                [3, 4, 6, 8]
[5, 5, 5]
15
                                [2, 3, 1, 5]
                                                10
                                                                 0.02 ms
                                                                                 0.32 KB
10
                                [10, 20, 30]
                                                50
                                                                 0.01 ms
                                                                                 0.29 KB
                [6, 7, 8]
                                [10, 12, 15]
                                                                 0.00 ms
                                                                                 0.29 KB
```

```
Mochila01.java
      valores, int capacidad) {
                                       input
Capacidad
                 Pesos
                          Valores Resultado
                                                             Memoria
                                                     Tiempo
         [1, 2, 3]
                          [10, 20, 30]
                                                             499 KB
                                            40
                                                     0 ms
10
         [2, 3, 5, 7]
                          [1, 4, 5, 7]
                                            11
                                                     0
                                                      ms
                                                             645 KB
         [3, 4, 6,
                          [2, 3, 1, 5]
15
                                            10
                                                     0
                                                             645 KB
                   8]
                                                      ms
         [5, 5, 5]
10
                          [10, 20, 30]
                                            50
                                                     0
                                                      ms
                                                             645 KB
         [6, 7, 8]
                          [10, 12, 15]
                                                             646 KB
                                            0
                                                     0 ms
```





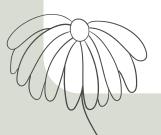


Resultados programa del cambio de monedas

```
main.c
   6
   7 int cambioMonedas(int* monedas, int tam, int cantidad) {
           int* dp = (int*)malloc((cantidad + 1) * sizeof(int));
           for (int i = 0; i \le cantidad; i++) {
   9 .
                                                            input
 V ×
       ⇔ .9
Cantidad
                Monedas
                                Resultado
                                                Tiempo
                                                        Memoria
                                                0.01 ms 0 KB
11
                [1, 2, 5, 0]
                                3
30
                                3
                [1, 2, 5, 10]
                                                0.00 ms 0 KB
99
                [1, 5, 10, 25]
                                9
                                                0.00 ms 0 KB
                [2, 5, 10, 0]
3
                                -1
                                                0.00 ms 0 KB
                [2, 4, 0, 0]
                                -1
                                                0.00 ms 0 KB
```

```
main.py
          cambio_monedas(monedas, cantidad):
  5
          dp = [float('inf')] * (cantidad + 1)
          dp[0] = 0
  6
           or moneda in monedas:
                                                             input
    × 🌣 🔏
Cantidad
               Monedas
                              Resultado
                                              Tiempo
                                                             Memoria
11
               [1, 2, 5]
                                              0.03 ms
                                                             0.26 KB
30
               [1, 2, 5, 10]
                                                             0.41 KB
                              3
                                              0.04 ms
99
               [1, 5, 10, 25, 50]8
                                                 0.13 ms
                                                                0.96 KB
               [2, 5, 10]
3
                              -1
                                              0.00 ms
                                                             0.20 KB
               [2, 4]
                              -1
                                              0.01 ms
                                                             0.21 KB
```

```
CambioMonedas.ja...
  57
           public static String padRight(String s, int n) {
               return String.format("%-" + n + "s", s);
  59
  61 -
           public static void main(String[] args) {
               int[][] escenarios = {
  62 -
  63
                    {11, 1, 2, 5},
  64
                    {30, 1, 2, 5, 10},
 V × 🌣 😘
Cantidad
         Monedas
                              Resultado
                                         Tiempo
                                                 Memoria
          [1, 2, 5]
11
                                                 499 KB
                                         0 ms
30
          [1, 2, 5, 10]
                                                 645 KB
                                         0 ms
          [1, 5, 10, 25, 50]
99
                             8
                                         0 ms
                                                 645 KB
          [2, 5, 10]
                                                 645 KB
                              -1
                                         0 ms
          [2, 4]
                                                 645 KB
                              -1
                                         0 ms
```





### Eficiencia de Tiempo

C demostró ser el más eficiente en términos de tiempo de ejecución para ambos problemas, con tiempos prácticamente inmediatos. Esto se debe a la naturaleza de bajo nivel de C y su capacidad para manejar operaciones directamente en el hardware sin la sobrecarga de la administración de tiempo de ejecución y compilación de bytecode presentes en Python y Java.

Python mostró un excelente rendimiento con tiempos de ejecución muy rápidos, lo que lo hace adecuado para prototipos y desarrollos rápidos donde la rapidez de implementación y facilidad de uso son importantes.

Java tuvo tiempos de ejecución ligeramente más altos, pero aún dentro de un rango muy eficiente, beneficiándose de su robusto ecosistema y características de programación orientada a objetos.

### Eficiencia de Memoria

C también lideró en términos de eficiencia de memoria, utilizando la menor cantidad de memoria entre los tres lenguajes debido a su capacidad para gestionar la memoria de manera más precisa y sin la sobrecarga de los entornos de ejecución.

Python y Java tuvieron un uso de memoria similar, con Python ligeramente más eficiente que Java, lo que podría ser una consideración en sistemas con restricciones de memoria.

