Python

Actividad Grupal: Resolución de ejercicios
Tema: seudocódigo
Subir la guía como ACT02-GRUPO#.pdf

Guía de Ejercicios Nº 02

Consideraciones: Leer el orientador de la clase

Objetivo Aplicar todo lo aprendido hasta el momento para resolver las diferentes problemáticas que presenta en Trabajo práctico. Al terminar habrás comprendido que es un algoritmo

Temas: Seudocódigo, algoritmos, entradas, salidas, instrucciones

Introducción

Los algoritmos de repetición, también conocidos como algoritmos de fuerza bruta, son métodos de resolución de problemas que implican probar todas las posibles soluciones hasta encontrar la correcta. En general, estos algoritmos no utilizan ninguna heurística ni optimización para reducir el espacio de búsqueda, lo que los hace simples, pero a menudo ineficientes.

Características de los Algoritmos de Fuerza Bruta:

1. Exhaustividad: Consideran todas las posibles soluciones.
2. Simplicidad: Son fáciles de implementar y entender.
3. No optimización: No usan técnicas para reducir el espacio de búsqueda.
4. Alta complejidad computacional: Pueden ser muy lentos y consumir muchos recursos, especialmente para problemas de gran tamaño.

1. Problema del Viajante (TSP):
    - Descripción: Encontrar el camino más corto que visita todas las ciudades exactamente una vez y regresa a la ciudad de origen.
    - Fuerza Bruta: Probar todas las permutaciones posibles de las ciudades y seleccionar la que tenga la distancia más corta.

2. Problema de la Mochila (Knapsack Problem):
    - Descripción: Dado un conjunto de ítems, cada uno con un peso y un valor, determinar el número de cada ítem que debe incluirse en una colección para que el peso total sea menor o igual a una capacidad dada y el valor total sea tan grande como sea posible.
    - Fuerza Bruta: Evaluar todas las combinaciones posibles de ítems para encontrar la combinación óptima.

Ventajas y Desventajas:

- Ventajas:
    - Fácil de implementar y entender.
    - Útil para problemas con un pequeño espacio de soluciones.
- Desventajas:
    - Ineficiente para problemas grandes debido a su alta complejidad temporal y espacial.
    - No es adecuado para aplicaciones en tiempo real debido a su lentitud.

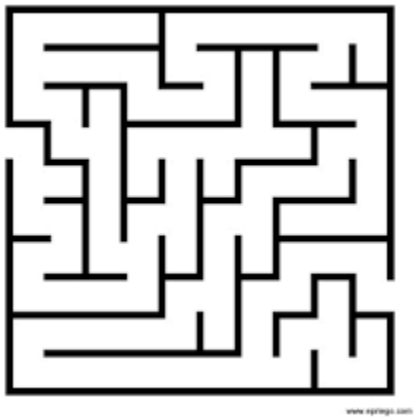
Aplicaciones:

- En situaciones donde el espacio de soluciones es pequeño.
- Como punto de referencia para comparar la eficacia de algoritmos más sofisticados.
- En problemas donde se necesita una solución exacta y no se puede utilizar heurísticas o aproximaciones.

Problemas Para Resolver

Ejercicio 1

Crear un algoritmo para que un robot, que sabe, cuando hay una puerta abierta, puede caminar y que sabe distinguir los movimientos adelante, atrás, izquierda y derecha. Pueda entrar a un laberinto y salir con la menor cantidad de instrucciones posibles. Es importante destacar que el robot, solo puede recibir una orden a la vez.



Ejercicio 2

Dado un algoritmo, probar si siguiendo paso a paso podemos resolver el problema sin encontrar errores