Nama : Masdar

Nim : D0424509

Prodi : Sistem Informasi

Tugas Jenis-jenis algoritma

Algoritma adalah serangkaian langkah atau prosedur sistematis yang digunakan untuk menyelesaikan suatu masalah atau mencapai tujuan tertentu. Dalam konteks pemrograman, algoritma memberikan instruksi jelas yang dapat diikuti oleh komputer untuk menghasilkan output dari input yang diberikan. Algoritma dapat diterapkan dalam berbagai bidang, termasuk matematika, ilmu komputer, dan rekayasa.

1. Algoritma Pencarian

   Digunakan untuk menemukan elemen tertentu dalam kumpulan data.

   - Pencarian Linier

     Mencari elemen dengan memeriksa setiap elemen satu per satu.

     ```
     def linear_search(arr, target):
       for index, value in enumerate(arr):
           if value == target:
               return index
     return -1
     ```

   - Pencarian Biner

     Mencari elemen dalam array yang terurut dengan membagi data menjadi dua bagian.

     ```
     def binary_search(arr, target):
       left, right = 0, len(arr) – 1
       while left <= right:
         mid = (left + right) // 2
         if arr[mid] == target:
             return mid
         elif arr[mid] < target:
             left = mid + 1 else:
     ```

```
            right = mid – 1
        return -1
```

2. Algoritma Pengurutan

Memungkinkan pengurutan data dalam urutan tertentu

- Bubble Sort

  Mengurutkan dengan membandingkan pasangan elemen dan menukarnya.

```
def bubble_sort(arr):
  n = len(arr)
  for i in range(n):
      for j in range(0, n-i-1):
          if arr[j] > arr[j+1]:
              arr[j], arr[j+1] = arr[j+1], arr[j]
  return arr
```

- Quick Sort

  Membagi array berdasarkan pivot dan mengurutkan bagian yang lebih kecil secara rekursif.

```
def quick_sort(arr):
  if len(arr) <= 1:
      return arr
  pivot = arr[len(arr) // 2]
  left = [x for x in arr if x < pivot]
  middle = [x for x in arr if x == pivot]
  right = [x for x in arr if x > pivot]
  return quick_sort(left) + middle + quick_sort(right)
```

3. Algoritma Graf

Algoritma graf adalah metode yang digunakan untuk menyelesaikan masalah yang dapat direpresentasikan dalam bentuk graf, yang terdiri dari simpul (vertex) dan sisi (edge).

- Dijkstra

```
Import heapq
Def dijkstra(graph, start):
    Queue = []
```

```
    Heapq.heappush(queue, (0, start))
    Distances = {node: float('infinity') for node in graph}
    Distances[start] = 0

    While queue:
        Current_distance, current_node = heapq.heappop(queue)

        If current_distance > distances[current_node]:
            Continue

        For neighbor, weight in graph[current_node].items():
            Distance = current_distance + weight
            If distance < distances[neighbor]:
                Distances[neighbor] = distance
                Heapq.heappush(queue, (distance, neighbor))
    Return distances
```

4. Algoritma Pemrograman Dinamis

   Mengatasi masalah dengan membagi menjadi sub-masalah dan menyimpan hasilnya untuk menghindari perhitungan ulang.

   - Fibonacci

```
Def fibonacci(n, memo={}):
    If n in memo:
        Return memo[n]
    If n <= 1:
        Return n
    Memo[n] = fibonacci(n-1, memo) + fibonacci(n-2, memo)
    Return memo[n]
```

5. Algoritma Greedy

   Mengambil keputusan terbaik pada setiap langkah tanpa mempertimbangkan konsekuensi jangka panjang, sering digunakan dalam optimasi.

- Masalah Koin

```
Def coin_change(coins, amount):
    Coins.sort(reverse=True)
    Result = []
    For coin in coins:
        While amount >= coin:
            Amount -= coin
            Result.append(coin)
Return result
```

6. Algoritma Backtracking

Mencari solusi dengan menjelajahi semua kemungkinan dan mundur jika solusi tidak valid, sering digunakan dalam teka-teki dan masalah kombinatorial.

- N-Queens Problem

```
Def solve_n_queens(n):
    Def backtrack(row, cols, diag1, diag2):
        If row == n:
            Result.append(board[:])
            Return
        For col in range(n):
            If col in cols or (row – col) in diag1 or (row + col) in diag2:
                Continue
            Board[row] = col
            Cols.add(col)
            Diag1.add(row – col)
            Diag2.add(row + col)
            Backtrack(row + 1, cols, diag1, diag2)
            Cols.remove(col)
            Diag1.remove(row – col)
            Diag2.remove(row + col)

    Result = []
    Board = [-1] *
```

```
    Backtrack(0, set(), set(), set())
  Return result
```

7. Algoritma Genetika

Menggunakan prinsip evolusi dan seleksi alam untuk menyelesaikan masalah optimasi, sering digunakan dalam masalah kompleks.

- Import random

```
  Def genetic_algorithm(population_size, generations):
    Population = [‘’.join(random.choices(‘01’, k=10)) for _ in range(population_size)]
    For _ in range(generations):
    Population = sorted(population, key=lambda x: fitness(x), reverse=True)[:population_size // 2]
    Population = population + crossover(population)
  Return population

  Def fitness(individual):
    Return sum(int(bit) for bit in individual)
  Def crossover(population):
    Offspring = []
    While len(offspring) < len(population):
    Parent1, parent2 = random.sample(population, 2)
    Point = random.randint(1, len(parent1) – 1)
    Offspring.append(parent1[:point] + parent2[point:])
 Return offspring
```

8. Algoritma Kriptografi

Algoritma kriptografi adalah metode matematis yang digunakan untuk mengamankan data dengan cara mengubah informasi menjadi format yang tidak dapat dibaca oleh pihak yang tidak berwenang.

- RSA (Sederhana)

```
  From Crypto.PublicKey import RS
  Def generate_rsa_keypair():
```

```
    Key = RSA.generate(2048)
    Private_key = key.export_key()
    Public_key = key.publickey().export_key()
Return private_key, public_key
```

9. Algoritma Machine Learning

Algoritma machine learning adalah metode yang memungkinkan komputer untuk belajar dari data dan membuat prediksi atau keputusan tanpa pemrograman eksplisit.

- Regresi Linier (Menggunakan scikit-learn)

```
From sklearn.linear_model import LinearRegression
Import numpy as np
X = np.array([[1], [2], [3], [4]])
Y = np.array([2, 3, 5, 7])
Model = LinearRegression()
Model.fit(X, y)
Predictions = model.predict(np.array([[5]]))
```