NAMA : WAFFIQ ADKHILNIY MR

NIM : D0424310

PRODI : SISTEM INFORMASI

TUGAS MEMBUAT RANGKUMAN MATERI ARTIKEL JENIS-JENIS ALGORITMA

1. Algoritma Recursive

   Digunakan untuk menyelesaikan masalah dengan memanggil fungsi itu sendiri secara berulang hingga mencapai kondisi dasar. Recursive digunakan dalam banyak aplikasi seperti kalkulasi matematis yang berulang. Salah satu contohnya yaitu Faktorial dari sebuah bilangan. Misalnya, n! = n*(n-1)! hingga 1! = 1.

   Contoh Kasus: Faktorial 5! dihitung sebagai 5 * 4 * 3 * 2 * 1 = 120, menggunakan recursive, setiap fungsi menghitung ulang hingga mencapai angka dasar (1!).

   Gambar Recursive (Pseudocode):

```arduino
function faktorial(n):
    if n == 1:
        return 1
    else:
        return n * faktorial(n-1)
```

   - **Kelebihan**: Mempermudah masalah kompleks dengan memecah menjadi sub-masalah. Cocok untuk masalah rekursif seperti faktorial.
   - **Kekurangan**: Boros memori, bisa menyebabkan *stack overflow* jika tidak dioptimalkan.

2. Algoritma Sorting

   Sorting digunakan untuk mengurutkan data. Salah satu contoh adalah *Bubble Sort*, yang mengurutkan daftar dengan membandingkan dan menukar elemen yang berdekatan jika dalam urutan yang salah.

   Contoh Kasus: Mengurutkan daftar angka [4, 2, 7, 1] menjadi [1, 2, 4, 7] menggunakan Bubble Sort, di mana elemen-elemen ditukar hingga semuanya dalam urutan yang benar.

   Gambar Sorting (Bubble Sort):

```php
function bubbleSort(arr):
    for i from 0 to length(arr)-1:
        for j from 0 to length(arr)-i-1:
            if arr[j] > arr[j+1]:
                swap(arr[j], arr[j+1])
```

- **Kelebihan**: Sederhana dan mudah dipahami, cocok untuk data kecil.
- **Kekurangan**: Tidak efisien untuk data besar, kompleksitas $O(n^2)$.

3. Algoritma Searching

Algoritma ini digunakan untuk mencari elemen tertentu dalam struktur data. Algoritma yang populer adalah *Linear Search* dan *Binary Search*. Contoh *Binary Search*, yang bekerja dengan membagi daftar terurut menjadi dua bagian untuk menemukan target lebih cepat.

*Contoh Kasus:*Dalam daftar terurut [1, 3, 5, 7, 9, 15, 18], Binary Search mencari angka 7 dengan membandingkan nilai tengah (5), dan kemudian hanya mencari pada separuh yang benar.

Gambar Searching (Binary Search):

```sql
function binarySearch(arr, target):
    left = 0
    right = length(arr)-1
    while left <= right:
        mid = (left + right) / 2
        if arr[mid] == target:
            return mid
        else if arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
```

- **Kelebihan**: Sangat cepat pada data terurut, kompleksitas O(log n).
- **Kekurangan**: Hanya berlaku untuk data terurut.

4. Algoritma Greedy

Algoritma Greedy membuat keputusan lokal terbaik di setiap langkah untuk mencapai solusi global yang mungkin terbaik. Biasanya digunakan pada masalah optimasi. Contoh: Algoritma Dijkstra untuk menemukan jalur terpendek di antara node-node graf.

Contoh Kasus:

Dalam pencarian jalur terpendek dari satu titik ke titik lain di peta, algoritma Greedy memilih jalur terdekat di setiap langkah, meskipun hasil akhirnya belum tentu optimal untuk keseluruhan graf.

Gambar Greedy (Dijkstra):

```less
function dijkstra(graph, source):
    for each vertex v in graph:
        dist[v] = infinity
    dist[source] = 0
    while unsettled vertices exist:
        choose vertex u with smallest dist[u]
        for each neighbor v of u:
            dist[v] = min(dist[v], dist[u] + weight(u, v))
```

- **Kelebihan**: Cepat dan simpel, cocok untuk masalah optimasi tertentu.
- **Kekurangan**: Tidak selalu memberikan solusi global yang optimal.

5. Algoritma Backtracking

Algoritma ini mencoba semua solusi yang mungkin, dan jika solusi tersebut gagal, ia mundur dan mencoba jalur lain. Algoritma ini sering digunakan untuk masalah kombinatorik. Contoh: Penyelesaian Sudoku di mana angka yang dimasukkan salah bisa dibatalkan dan pencarian solusi dilanjutkan.

Contoh Kasus: Dalam permainan Sudoku, Backtracking memasukkan angka ke dalam kotak. Jika angka itu menyebabkan kesalahan pada aturan Sudoku, algoritma mundur dan mencoba angka lain hingga menemukan solusi yang benar.

Gambar Backtracking (Sudoku):

```typescript
function solveSudoku(board):
    if no empty cells:
        return true
    for each number from 1 to 9:
        if number is valid in empty cell:
            place number
            if solveSudoku(board):
                return true
            remove number
    return false
```

- **Kelebihan**: Mampu mengeksplorasi semua solusi, cocok untuk masalah kombinatorik seperti Sudoku.
- **Kekurangan**: Lambat pada ruang solusi besar, kompleksitas bisa eksponensial

6. Algoritma Randomized

Algoritma ini menggunakan elemen acak untuk menghasilkan solusi. Dalam beberapa kasus, algoritma ini memberikan solusi yang lebih cepat dibandingkan algoritma deterministik. Contoh: *Randomized Quick Sort* di mana elemen pivot dipilih secara acak untuk mengoptimalkan pembagian array.

Contoh Kasus: Mengacak urutan kartu dalam permainan kartu. Setiap kartu ditempatkan di posisi acak hingga seluruh deck teracak sempurna.

Gambar Randomized (Pengacakan):

```php
function shuffleDeck(deck):
    for i from length(deck)-1 down to 1:
        j = random(0, i)
        swap(deck[i], deck[j])
```

- **Kelebihan**: Menghindari kasus terburuk dengan elemen acak, lebih cepat dalam beberapa kasus.
- **Kekurangan**: Hasilnya tidak selalu konsisten, masih bisa lambat pada beberapa dataset.

Setiap algoritma memiliki keunggulan tersendiri, tergantung dari konteks dan kebutuhan masalah yang dihadapi.