Mata Kuliah: Struktur Data
Pertemuan ke 12
Topik: Algoritma Searching

Pendahuluan

Algoritma searching merupakan teknik pencarian data yang sangat penting dalam berbagai bidang seperti Artificial Intelligence (AI), Data Mining, Big Data, dan Internet of Things (IoT). Pada materi ini, kita akan membahas beberapa algoritma searching lanjutan dan aplikasinya dalam berbagai bidang.

1. Linear Search :

Pencarian data secara linier.

- Kelebihan: Sederhana dan mudah diimplementasikan.
- Kekurangan: Kurang efisien untuk data besar.

Aplikasi: Pengurutan data sederhana, pencarian data dalam array.

Contoh Kode PHP:

```
function linearSearch($arr, $target) {
 for ($i = 0; $i < count($arr); $i++) {
 if ($arr[$i] == $target) {
 return $i;
 }
 }
 return -1;
}
```

2. Binary Search:

Pencarian data secara biner.

- Kelebihan: Efisien dan cepat.
- Kekurangan: Memerlukan data yang terurut.

Aplikasi: Pengurutan data besar, pencarian data dalam database.

Contoh Kode PHP:

```
function binarySearch($arr, $target) {
 $low = 0;
 $high = count($arr) - 1;
 while ($low <= $high) {
 $mid = floor(($low + $high) / 2);
 if ($arr[$mid] == $target) {
 return $mid;
```

```
} elseif ($arr[$mid] < $target) {
$low = $mid + 1;
} else {
$high = $mid - 1;
}
}
return -1;
}
```

3. Hashing :

Pencarian data menggunakan fungsi hash.

- Kelebihan: Efisien dan cepat.
- Kekurangan: Memerlukan fungsi hash yang baik.

Aplikasi: Pengurutan data besar, pencarian data dalam database.

Contoh Kode PHP:

```
function hashing($arr, $target) {
$hashTable = array();
foreach ($arr as $value) {
$hashTable[hash('sha256', $value)] = $value;
}
return isset($hashTable[hash('sha256', $target)]) ? $hashTable[hash('sha256',
$target)] : null;
}
```

4. Depth-First Search (DFS):

Pencarian data secara mendalam.

- Kelebihan: Efisien untuk data yang memiliki struktur graf.
- Kekurangan: Kurang efisien untuk data yang tidak memiliki struktur graf.

Aplikasi: Pengurutan data graf, pencarian data dalam jaringan sosial.

Contoh Kode PHP:

```
function dfs($graph, $target) {
$visited = array();
$stack = array($graph[0]);
while (!empty($stack)) {
$node = array_pop($stack);
if ($node == $target) {
return true;
```

```
}
if (!in_array($node, $visited)) {
$visited[] = $node;
foreach ($graph[$node] as $neighbor) {
$stack[] = $neighbor;
}
}
}
return false;
}
```

5. Breadth-First Search (BFS):

Pencarian data secara lebar.

- Kelebihan: Efisien untuk data yang memiliki struktur graf.
- Kekurangan: Kurang efisien untuk data yang tidak memiliki struktur graf.

Aplikasi: Pengurutan data graf, pencarian data dalam jaringan sosial.

Contoh Kode PHP:

```
function bfs($graph, $target) {
$visited = array();
$queue = array($graph[0]);
while (!empty($queue)) {
$node = array_shift($queue);
if ($node == $target) {
return true;
}
if (!in_array($node, $visited)) {
$visited[] = $node;
foreach ($graph[$node] as $neighbor) {
$queue[] = $neighbor;
}
}
}
return false;
}
```

6. _Dijkstra's Algorithm_:
Pencarian jalur terpendek dalam graf.

- Kelebihan: Efisien untuk data graf besar.
- Kekurangan: Kurang efisien untuk data graf kecil.

Aplikasi: Pengurutan data graf, pencarian jalur terpendek dalam jaringan sosial.
Contoh Kode PHP:

```
function dijkstra($graph, $start) {
 $distance = array_fill(0, count($graph), INF);
 $distance[$start] = 0;
 $queue = array($start);
 while (!empty($queue)) {
 $node = array_shift($queue);
 foreach ($graph[$node] as $neighbor => $weight) {
 $newDistance = $distance[$node] + $weight;
 if ($newDistance < $distance[$neighbor]) {
 $distance[$neighbor] = $newDistance;
 $queue[] = $neighbor;
 }
 }
 }
 return $distance;
}
```

7. _A* Algorithm_:
Pencarian jalur terpendek dengan heuristik.

- Kelebihan: Efisien untuk data graf besar dengan heuristik yang baik.
- Kekurangan: Kurang efisien untuk data graf kecil.

Aplikasi: Pengurutan data graf, pencarian jalur terpendek dalam jaringan sosial.
Contoh Kode PHP:

```
function aStar($graph, $start, $goal, $heuristic) {
 $distance = array_fill(0, count($graph), INF);
 $distance[$start] = 0;
 $queue = array($start);
 while (!empty($queue)) {
 $node = array_shift($queue);
 if ($node == $goal) {
 return $distance[$node];
 }
```

```
 foreach ($graph[$node] as $neighbor => $weight) {
 $newDistance = $distance[$node] + $weight;
 if ($newDistance < $distance[$neighbor]) {
 $distance[$neighbor] = $newDistance;
 $queue[] = $neighbor;
 }
 }
 }
 return -1;
}
```

8. _Floyd-Warshall Algorithm_:
Pencarian jalur terpendek dalam graf berbobot.

- Kelebihan: Efisien untuk data graf besar.
- Kekurangan: Kurang efisien untuk data graf kecil.

Aplikasi: Pengurutan data graf, pencarian jalur terpendek dalam jaringan sosial.
Contoh Kode PHP:

```
function floydWarshall($graph) {
 $n = count($graph);
 for ($k = 0; $k < $n; $k++) {
 for ($i = 0; $i < $n; $i++) {
 for ($j = 0; $j < $n; $j++) {
 $graph[$i][$j] = min($graph[$i][$j], $graph[$i][$k] + $graph[$k][$j]);
 }
 }
 }
 return $graph;
}
```

9. _Bellman-Ford Algorithm_:
Pencarian jalur terpendek dalam graf berbobot.

- Kelebihan: Efisien untuk data graf besar.
- Kekurangan: Kurang efisien untuk data graf kecil.

Aplikasi: Pengurutan data graf, pencarian jalur terpendek dalam jaringan sosial.
Contoh Kode PHP:

```
function bellmanFord($graph, $start) {
 $n = count($graph);
 $distance = array_fill(0, $n, INF);
 $distance[$start] = 0;
 for ($i = 0; $i < $n - 1; $i++) {
 for ($j = 0; $j < $n; $j++) {
 foreach ($graph[$j] as $neighbor => $weight) {
 $distance[$neighbor] = min($distance[$neighbor], $distance[$j] + $weight);
 }
 }
 }
 return $distance;
}
```

10. _K-Mean Clustering_:
Algoritma clustering untuk mengelompokkan data.

- Kelebihan: Efisien untuk data besar.
- Kekurangan: Kurang efisien untuk data kecil.

Aplikasi: Analisis data pelanggan, pengelompokkan data sensor IoT.

Contoh Listing K-Mean Clustering dengan PHP

```
class KMeanClustering {
    private $data;
    private $centroid;
    private $cluster;
    private $k;

    public function __construct($data, $k) {
        $this->data = $data;
        $this->k = $k;
        $this->centroid = array();
        $this->cluster = array();
```

```
}

public function initCentroid() {
    for ($i = 0; $i < $this->k; $i++) {
        $this->centroid[] = $this->data[rand(0, count($this->data) - 1)];
    }
}

public function assignCluster() {
    foreach ($this->data as $point) {
        $minDistance = INF;
        $clusterIndex = -1;
        foreach ($this->centroid as $index => $centroid) {
            $distance = $this->calculateDistance($point, $centroid);
            if ($distance < $minDistance) {
                $minDistance = $distance;
                $clusterIndex = $index;
            }
        }
        $this->cluster[$clusterIndex][] = $point;
    }
}

    public function updateCentroid() {
        foreach ($this->cluster as $index => $cluster) {
            $sumX = 0;
            $sumY = 0;
            foreach ($cluster as $point) {
                $sumX += $point[0];
                $sumY += $point[1];
            }
            $this->centroid[$index] = array($sumX / count($cluster), $sumY /
count($cluster));
        }
    }

    public function calculateDistance($point1, $point2) {
        return sqrt(pow($point1[0] - $point2[0], 2) + pow($point1[1] - $point2[1], 2));
    }

    public function run() {
```

```
        $this->initCentroid();
        while (true) {
            $this->assignCluster();
            $oldCentroid = $this->centroid;
            $this->updateCentroid();
            if ($this->centroid == $oldCentroid) {
                break;
            }
        }
        return $this->cluster;
    }
}

// Contoh data
$data = array(
    array(1, 2),
    array(2, 1),
    array(3, 3),
    array(4, 4),
    array(5, 5),
    array(6, 6),
    array(7, 7),
    array(8, 8),
    array(9, 9)
);

$k = 3;

$kMean = new KMeanClustering($data, $k);
$cluster = $kMean->run();

print_r($cluster);
```

Penjelasan :

1. Kelas `KMeanClustering` memiliki properti `$data`, `$centroid`, `$cluster`, dan `$k`.
2. Metode `initCentroid()` menginisialisasi centroid secara acak.
3. Metode `assignCluster()` mengassign setiap titik data ke cluster terdekat.
4. Metode `updateCentroid()` memperbarui centroid berdasarkan titik-titik dalam cluster.

5. Metode `calculateDistance()` menghitung jarak antara dua titik.
6. Metode `run()` menjalankan algoritma K-Mean Clustering.

Hasil :
Contoh di atas akan menghasilkan clustering data menjadi 3 kelompok. Hasilnya akan berbeda-beda karena centroid diinisialisasi secara acak.

11. _K-Nearest Neighbor (KNN)_:
Algoritma pencarian tetangga terdekat.

- Kelebihan: Efisien untuk data besar.
- Kekurangan: Kurang efisien untuk data kecil.

Aplikasi: Rekomendasi produk, pengenalan pola.

Berikut adalah contoh listing program PHP untuk implementasi K-Nearest Neighbor (KNN) untuk rekomendasi produk:

KNN.php

```
class KNN {
    private $data;
    private $k;

    public function __construct($data, $k) {
        $this->data = $data;
        $this->k = $k;
    }

    public function calculateDistance($point1, $point2) {
        return sqrt(pow($point1['rating'] - $point2['rating'], 2) + pow($point1['harga'] -
$point2['harga'], 2));
    }

    public function findKNN($input) {
        $distances = array();
        foreach ($this->data as $index => $product) {
            $distance = $this->calculateDistance($input, $product);
            $distances[] = array('index' => $index, 'distance' => $distance);
        }
        usort($distances, function($a, $b) {
            return $a['distance'] - $b['distance'];
```

```
        });
        return array_slice($distances, 0, $this->k);
    }

    public function predict($input) {
        $knn = $this->findKNN($input);
        $sumRating = 0;
        foreach ($knn as $neighbor) {
            $sumRating += $this->data[$neighbor['index']]['rating'];
        }
        return $sumRating / $this->k;
    }
}

// Contoh data produk
$data = array(
    array('nama' => 'Produk A', 'rating' => 4.5, 'harga' => 100),
    array('nama' => 'Produk B', 'rating' => 4.2, 'harga' => 80),
    array('nama' => 'Produk C', 'rating' => 4.8, 'harga' => 120),
    array('nama' => 'Produk D', 'rating' => 4.1, 'harga' => 90),
    array('nama' => 'Produk E', 'rating' => 4.6, 'harga' => 110)
);

// Input pengguna
$input = array('rating' => 4.3, 'harga' => 105);

$k = 3;

$knn = new KNN($data, $k);
$rekomendasi = $knn->predict($input);

echo "Rekomendasi produk: ";
foreach ($knn->findKNN($input) as $neighbor) {
    echo $data[$neighbor['index']]['nama'] . " ";
}
echo "\nRating prediksi: $rekomendasi";
```

Penjelasan :

1. Kelas `KNN` memiliki properti `$data` dan `$k`.
2. Metode `calculateDistance()` menghitung jarak antara dua titik (produk).
3. Metode `findKNN()` mencari K tetangga terdekat dari input pengguna.

4. Metode `predict()` memprediksi rating produk berdasarkan K tetangga terdekat.
5. Contoh data produk dan input pengguna digunakan untuk demonstrasi.

Aplikasi :
1. Rekomendasi produk e-commerce.
2. Pengenalan pola perilaku pengguna.
3. Analisis sentimen opini publik.

12. _Support Vector Machine (SVM)_:
Algoritma klasifikasi data.

- Kelebihan: Efisien untuk data besar.
- Kekurangan: Kurang efisien untuk data kecil.

Aplikasi: Klasifikasi data pelanggan, pengenalan pola.

Contoh listing program PHP untuk implementasi Support Vector Machine (SVM) menggunakan library PHP-ML:

Instalasi PHP-ML
Sebelum menjalankan contoh program, pastikan Anda telah menginstal library PHP-ML melalui Composer:

bash
composer require php-ai/php-ml

Contoh Program SVM
```
use Phpml\Classification\SupportVectorMachine;
use Phpml\SupportVectorMachine\Kernel;

// Data pelanggan
$data = [
    ['nama' => 'John', 'umur' => 25, 'penghasilan' => 5000000, 'label' => 1], // Pelanggan loyal
    ['nama' => 'Jane', 'umur' => 30, 'penghasilan' => 3000000, 'label' => 0], // Bukan pelanggan loyal
    ['nama' => 'Bob', 'umur' => 20, 'penghasilan' => 4000000, 'label' => 1],
    ['nama' => 'Alice', 'umur' => 35, 'penghasilan' => 6000000, 'label' => 1],
    ['nama' => 'Mike', 'umur' => 40, 'penghasilan' => 2000000, 'label' => 0]
];
```

```
// Konversi data ke format SVM
$dataset = [];
foreach ($data as $pelanggan) {
    $dataset[] = [$pelanggan['umur'], $pelanggan['penghasilan'], $pelanggan['label']];
}

// Pembagian data menjadi training dan testing
$trainData = array_slice($dataset, 0, 3);
$testData = array_slice($dataset, 3);

// Inisialisasi SVM
$sdk = new SupportVectorMachine(Kernel::RBF, 1000);
$sdk->train($trainData);

// Prediksi
$prediksi = $sdk->predict([$testData[0][0], $testData[0][1]]);

// Hasil prediksi
echo "Prediksi: " . ($prediksi == 1 ? "Pelanggan Loyal" : "Bukan Pelanggan Loyal");
```

Penjelasan :

1. Data pelanggan disimpan dalam array `$data`.
2. Data dikonversi ke format SVM menggunakan `$dataset`.
3. Data dibagi menjadi training (`$trainData`) dan testing (`$testData`).
4. SVM diinisialisasi dengan kernel RBF dan parameter C=1000.
5. Model SVM dilatih menggunakan data training.
6. Prediksi dilakukan menggunakan data testing.
7. Hasil prediksi ditampilkan.

Aplikasi :

1. Klasifikasi data pelanggan.
2. Pengenalan pola perilaku pengguna.
3. Analisis sentimen opini publik.

Aplikasi Algoritma Searching dalam Berbagai Bidang :

1. Pengurutan data transaksi keuangan (Binary Search)
2. Pengurutan data sensor IoT (Hashing)
3. Pengurutan data IP address (Linear Search)
4. AI: Pengurutan data pengguna dalam aplikasi rekomendasi konten (DFS, BFS).

5. Data Mining: Pengurutan data transaksi keuangan (Binary Search).
6. Big Data: Pengurutan data sensor IoT (Hashing).
7. IoT: Pengurutan data sensor suhu (Linear Search).
8. Jaringan Sosial: Pengurutan data pertemanan (Floyd-Warshall, Bellman-Ford).