NIM: 231232028
Nama: Falmesino Abdul Hamid

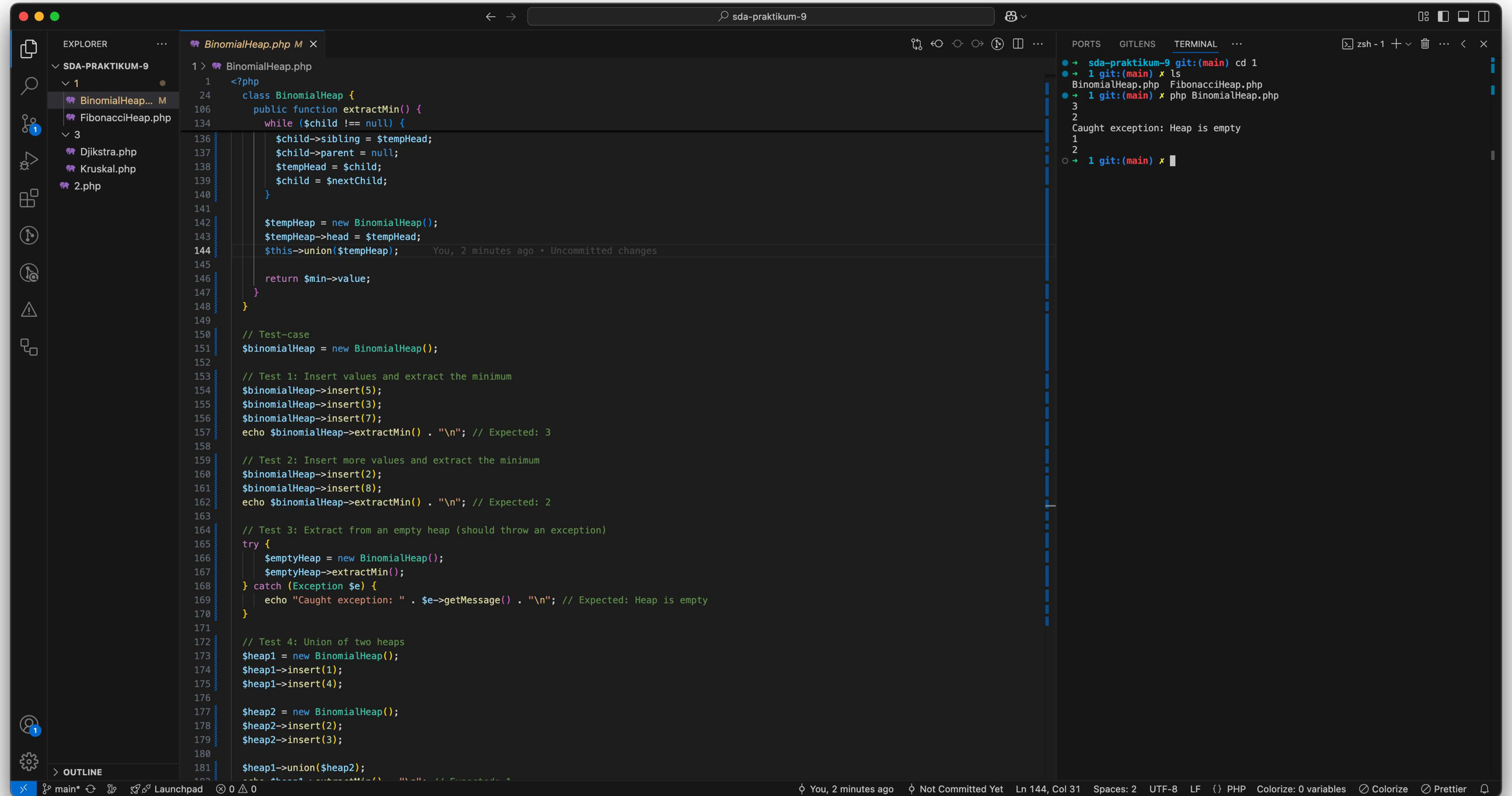
Tugas Pertemuan 9
Struktur Data dan Algoritma (SDA)

URL Source Code: https://github.com/falmesino/sda-praktikum-9

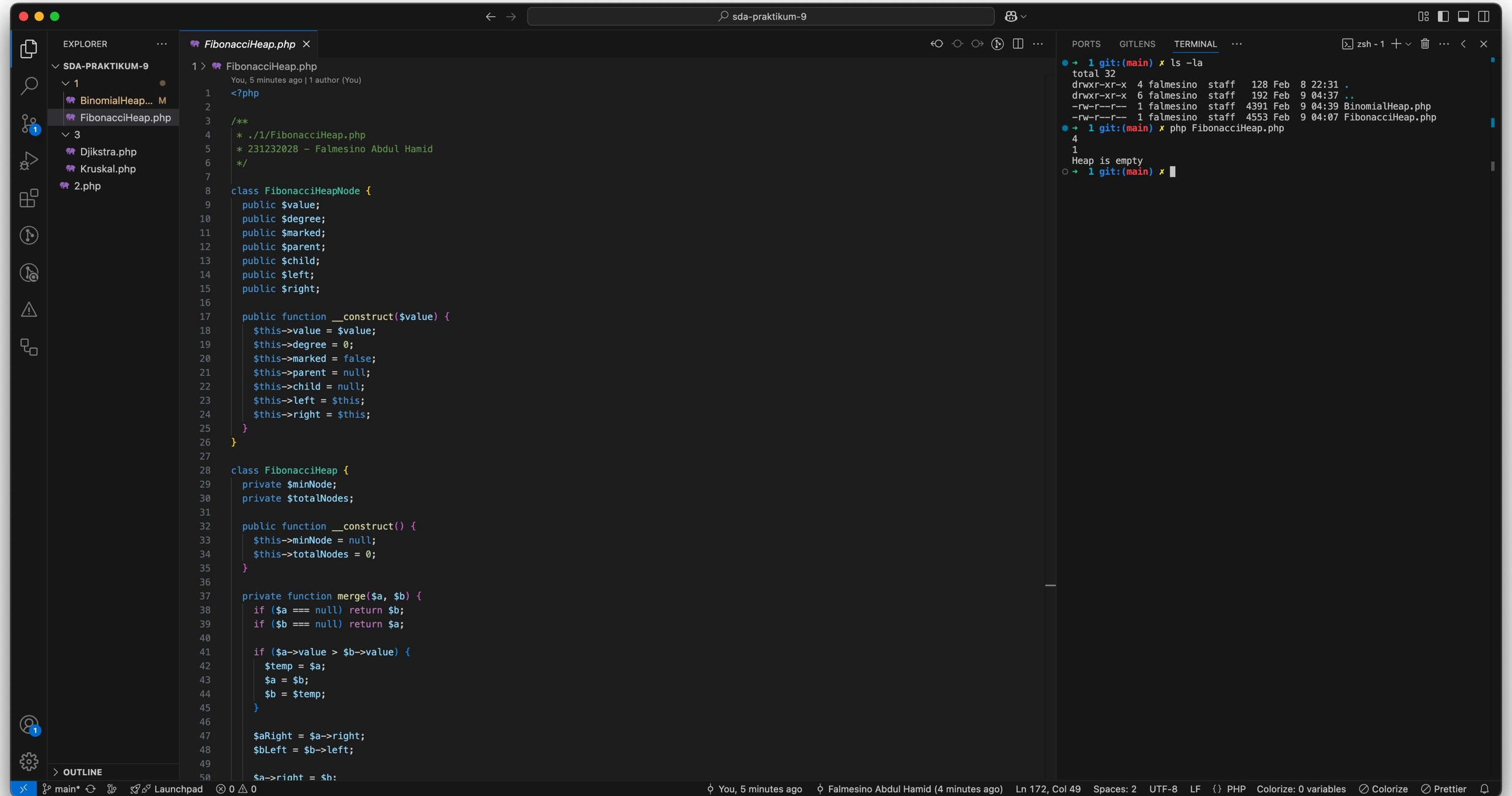
Soal

1. Implementasikan heap binomial dan heap Fibonacci dalam bahasa pemrograman pilihan Anda.
2. Buat priority queue menggunakan heap dan implementasikan operasi enqueue, dequeue, dan peek.
3. Implementasikan algoritma pencarian jalur seperti Dijkstra dan Kruskal menggunakan priority queue.

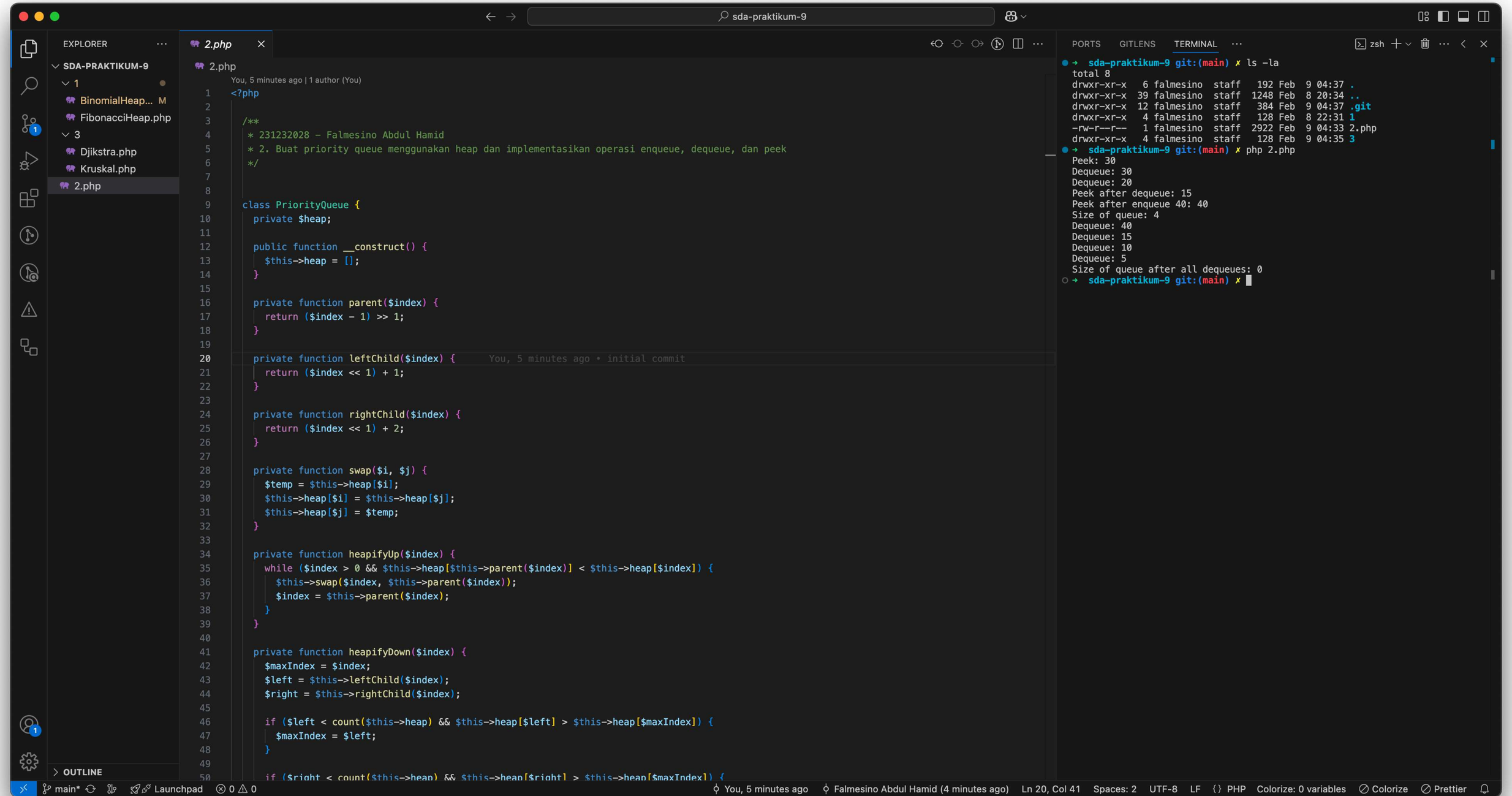
```php
<?php

  /**
   * ./1/BinomialHeap.php
   * 231232028 - Falmesino Abdul Hamid
   */

  class BinomialHeapNode {
    public $value;
    public $degree;
    public $parent;
    public $child;
    public $sibling;

    public function __construct($value) {
      $this->value = $value;
      $this->degree = 0;
      $this->parent = null;
      $this->child = null;
      $this->sibling = null;
    }
  }

  class BinomialHeap {
    private $head;

    public function __construct() {
      $this->head = null;
    }

    private function link(BinomialHeapNode $y, BinomialHeapNode $z) {
      $y->parent = $z;
      $y->sibling = $z->child;
      $z->child = $y;
      $z->degree++;
    }

    private function merge(?BinomialHeapNode $h1, ?BinomialHeapNode $h2): ?BinomialHeapNode {
      // If one of the heaps is empty, return the other
      if ($h1 === null) {
        return $h2;
      }
      if ($h2 === null) {
        return $h1;
      }

      $head = new BinomialHeapNode(null);
      $tail = $head;

      while ($h1 !== null && $h2 !== null) {
        if ($h1->degree <= $h2->degree) {
          $tail->sibling = $h1;
          $h1 = $h1->sibling;
        } else {
          $tail->sibling = $h2;
          $h2 = $h2->sibling;
        }
        $tail = $tail->sibling;
      }

      if ($h1 !== null) {
        $tail->sibling = $h1;
      } else {
        $tail->sibling = $h2;
      }

      return $head->sibling;
    }

    public function union(BinomialHeap $h) {
      $this->head = $this->merge($this->head, $h->head);
      if ($this->head === null) {
        return;
      }

      $prev = null;
      $curr = $this->head;
      $next = $curr->sibling;

      while ($next !== null) {
        if ($curr->degree != $next->degree || ($next->sibling !== null && $next->sibling->degree ==
$curr->degree)) {
          $prev = $curr;
          $curr = $next;
        } elseif ($curr->value <= $next->value) {
          $curr->sibling = $next->sibling;
          $this->link($next, $curr);
        } else {
          if ($prev === null) {
            $this->head = $next;
          } else {
            $this->sibling = $next;
          }
          $this->link($curr, $next);
          $curr = $next;
        }
        $next = $curr->sibling;
      }
    }

    public function insert($value) {
      $temp = new BinomialHeap();
      $temp->head = new BinomialHeapNode($value);
      $this->union($temp);
    }

    public function extractMin() {
      if ($this->head === null) {
        throw new Exception("Heap is empty");
      }

      $min = $this->head;
      $minPrev = null;
      $curr = $this->head;
      $prev = null;

      while ($curr !== null) {
        if ($curr->value < $min->value) {
          $min = $curr;
          $minPrev = $prev;
        }
        $prev = $curr;
        $curr = $curr->sibling;
      }

      if ($minPrev === null) {
        $this->head = $min->sibling;
      } else {
        $minPrev->sibling = $min->sibling;
      }

      $child = $min->child;
      $tempHead = null;

      while ($child !== null) {
        $nextChild = $child->sibling;
        $child->sibling = $tempHead;
        $child->parent = null;
        $tempHead = $child;
        $child = $nextChild;
      }

      $tempHeap = new BinomialHeap();
      $tempHeap->head = $tempHead;
      $this->union($tempHeap);

      return $min->value;
    }
  }

  // Test-case
  $binomialHeap = new BinomialHeap();

  // Test 1: Insert values and extract the minimum
  $binomialHeap->insert(5);
  $binomialHeap->insert(3);
  $binomialHeap->insert(7);
  echo $binomialHeap->extractMin() . "\n"; // Expected: 3

  // Test 2: Insert more values and extract the minimum
  $binomialHeap->insert(2);
  $binomialHeap->insert(8);
  echo $binomialHeap->extractMin() . "\n"; // Expected: 2

  // Test 3: Extract from an empty heap (should throw an exception)
  try {
      $emptyHeap = new BinomialHeap();
      $emptyHeap->extractMin();
  } catch (Exception $e) {
      echo "Caught exception: " . $e->getMessage() . "\n"; // Expected: Heap is empty
  }

  // Test 4: Union of two heaps
  $heap1 = new BinomialHeap();
  $heap1->insert(1);
  $heap1->insert(4);

  $heap2 = new BinomialHeap();
  $heap2->insert(2);
  $heap2->insert(3);

  $heap1->union($heap2);
  echo $heap1->extractMin() . "\n"; // Expected: 1
  echo $heap1->extractMin() . "\n"; // Expected: 2

?>
```


sda-praktikum-9
EXPLORER
SDA-PRAKTIKUM-9
1
BinomialHeap... M
FibonacciHeap.php
3
Djikstra.php
Kruskal.php
2.php
OUTLINE
BinomialHeap.php M
1 > BinomialHeap.php
<?php
class BinomialHeap {
public function extractMin() {
while ($child !== null) {
$child->sibling = $tempHead;
$child->parent = null;
$tempHead = $child;
$child = $nextChild;
}
$tempHeap = new BinomialHeap();
$tempHeap->head = $tempHead;
$this->union($tempHeap); You, 2 minutes ago • Uncommitted changes
return $min->value;
}
}
// Test-case
$binomialHeap = new BinomialHeap();
// Test 1: Insert values and extract the minimum
$binomialHeap->insert(5);
$binomialHeap->insert(3);
$binomialHeap->insert(7);
echo $binomialHeap->extractMin() . "\n"; // Expected: 3
// Test 2: Insert more values and extract the minimum
$binomialHeap->insert(2);
$binomialHeap->insert(8);
echo $binomialHeap->extractMin() . "\n"; // Expected: 2
// Test 3: Extract from an empty heap (should throw an exception)
try {
$emptyHeap = new BinomialHeap();
$emptyHeap->extractMin();
} catch (Exception $e) {
echo "Caught exception: " . $e->getMessage() . "\n"; // Expected: Heap is empty
}
// Test 4: Union of two heaps
$heap1 = new BinomialHeap();
$heap1->insert(1);
$heap1->insert(4);
$heap2 = new BinomialHeap();
$heap2->insert(2);
$heap2->insert(3);
$heap1->union($heap2);
PORTS
GITLENS
TERMINAL
zsh - 1
sda-praktikum-9 git:(main) cd 1
1 git:(main) ✗ ls
BinomialHeap.php FibonacciHeap.php
1 git:(main) ✗ php BinomialHeap.php
3
2
Caught exception: Heap is empty
1
2
1 git:(main) ✗
main*
Launchpad
0 0
You, 2 minutes ago
Not Committed Yet
Ln 144, Col 31
Spaces: 2
UTF-8
LF
PHP
Colorize: 0 variables
Colorize
Prettier

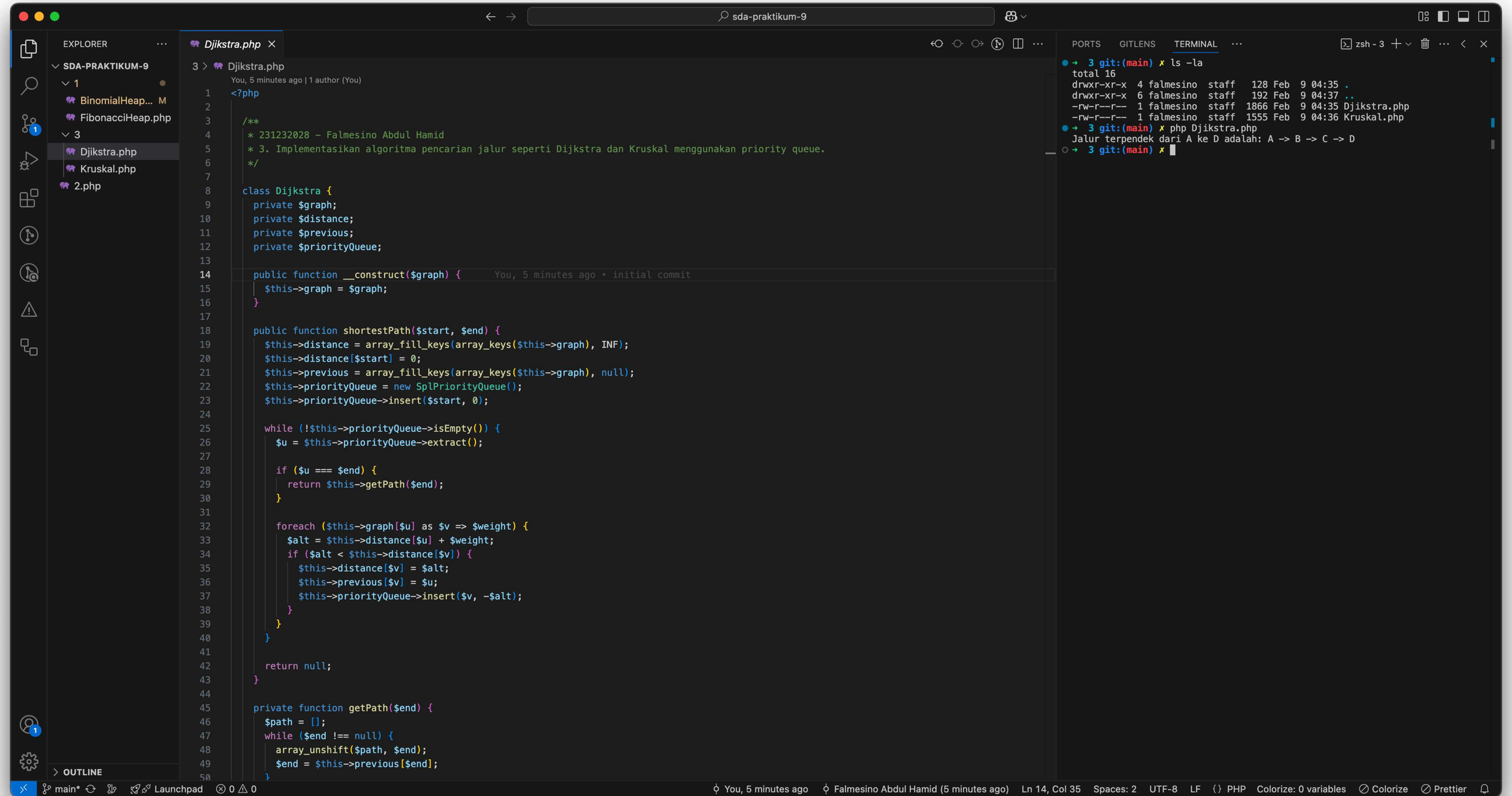
```php
<?php

/**
 * ./1/FibonacciHeap.php
 * 231232028 - Falmesino Abdul Hamid
 */

class FibonacciHeapNode {
  public $value;
  public $degree;
  public $marked;
  public $parent;
  public $child;
  public $left;
  public $right;

  public function __construct($value) {
    $this->value = $value;
    $this->degree = 0;
    $this->marked = false;
    $this->parent = null;
    $this->child = null;
    $this->left = $this;
    $this->right = $this;
  }
}

class FibonacciHeap {
  private $minNode;
  private $totalNodes;

  public function __construct() {
    $this->minNode = null;
    $this->totalNodes = 0;
  }

  private function merge($a, $b) {
    if ($a === null) return $b;
    if ($b === null) return $a;

    if ($a->value > $b->value) {
      $temp = $a;
      $a = $b;
      $b = $temp;
    }

    $aRight = $a->right;
    $bLeft = $b->left;

    $a->right = $b;
    $b->left = $a;
    $aRight->left = $bLeft;
    $bLeft->right = $aRight;

    return $a;
  }

  public function insert($value) {
    $node = new FibonacciHeapNode($value);
    if ($this->minNode !== null) {
      $this->minNode = $this->merge($this->minNode, $node);
    } else {
      $this->minNode = $node;
    }
    $this->totalNodes++;
  }

  public function extractMin() {
    $min = $this->minNode;
    if ($min !== null) {
      if ($min->child !== null) {
        $child = $min->child;
        do {
          $next = $child->right;
          $child->parent = null;
          $this->minNode = $this->merge($this->minNode, $child);
          $child = $next;
        } while ($child !== $min->child);
      }

      $min->left->right = $min->right;
      $min->right->left = $min->left;

      if ($min === $min->right) {
        $this->minNode = null;
      } else {
        $this->minNode = $min->right;
        $this->consolidate();
      }
      $this->totalNodes--;
    }
    return $min ? $min->value : null;
  }

  private function consolidate() {
    $array = array_fill(0, floor(log($this->totalNodes) / log(2)) + 1, null);
    $nodes = array();
    $current = $this->minNode;

    // Collect all root nodes
    if ($current !== null) {
      do {
        $nodes[] = $current;
        $current = $current->right;
      } while ($current !== $this->minNode);
    }

    // Consolidate nodes with the same degree
    foreach ($nodes as $node) {
      $degree = $node->degree;
      while ($array[$degree] !== null) {
        $other = $array[$degree];
        if ($node->value > $other->value) {
          // Swap nodes to ensure $node has the smaller value
          $temp = $node;
          $node = $other;
          $other = $temp;
        }
        $this->link($other, $node); // Link $other under $node
        $array[$degree] = null;
        $degree++;
      }
      $array[$degree] = $node;
    }

    // Rebuild the root list from the array
    $this->minNode = null;
    foreach ($array as $node) {
      if ($node !== null) {
        if ($this->minNode === null) {
          // Initialize the root list with the first node
          $this->minNode = $node;
          $this->minNode->left = $this->minNode;
          $this->minNode->right = $this->minNode;
        } else {
          // Merge the node into the root list
          $this->minNode = $this->merge($this->minNode, $node);
        }
      }
    }
  }

  private function link($y, $x) {
    // Remove $y from the root list
    $y->left->right = $y->right;
    $y->right->left = $y->left;

    // Make $y a child of $x
    $y->parent = $x;
    if ($x->child === null) {
      // If $x has no children, set $y as its only child
      $x->child = $y;
      $y->left = $y;
      $y->right = $y;
    } else {
      // Insert $y into $x's child list
      $y->left = $x->child;
      $y->right = $x->child->right;
      $x->child->right->left = $y;
      $x->child->right = $y;
    }

    // Increment $x's degree and mark $y as not marked
    $x->degree++;
    $y->marked = false;
  }
}

// test-case
$fibonacciHeap = new FibonacciHeap();

// Test 1: Insert values and extract the minimum
$fibonacciHeap->insert(10);
$fibonacciHeap->insert(4);
$fibonacciHeap->insert(15);
echo $fibonacciHeap->extractMin() . "\n"; // Expected: 4

// Test 2: Insert more values and extract the minimum
$fibonacciHeap->insert(1);
$fibonacciHeap->insert(20);
echo $fibonacciHeap->extractMin() . "\n"; // Expected: 1

// Test 3: Extract from an empty heap (should return null)
$emptyHeap = new FibonacciHeap();
echo $emptyHeap->extractMin() === null ? "Heap is empty\n" : "Error\n"; // Expected: Heap is empty

?>
```


sda-praktikum-9
EXPLORER
SDA-PRAKTIKUM-9
1
BinomialHeap... M
FibonacciHeap.php
3
Djikstra.php
Kruskal.php
2.php
OUTLINE
FibonacciHeap.php
1 > FibonacciHeap.php
You, 5 minutes ago | 1 author (You)
<?php
/**
* ./1/FibonacciHeap.php
* 231232028 - Falmesino Abdul Hamid
*/
class FibonacciHeapNode {
public $value;
public $degree;
public $marked;
public $parent;
public $child;
public $left;
public $right;
public function __construct($value) {
$this->value = $value;
$this->degree = 0;
$this->marked = false;
$this->parent = null;
$this->child = null;
$this->left = $this;
$this->right = $this;
}
}
class FibonacciHeap {
private $minNode;
private $totalNodes;
public function __construct() {
$this->minNode = null;
$this->totalNodes = 0;
}
private function merge($a, $b) {
if ($a === null) return $b;
if ($b === null) return $a;
if ($a->value > $b->value) {
$temp = $a;
$a = $b;
$b = $temp;
}
$aRight = $a->right;
$bLeft = $b->left;
PORTS
GITLENS
TERMINAL
zsh - 1
→ 1 git:(main) ✗ ls -la
total 32
drwxr-xr-x 4 falmesino staff 128 Feb 8 22:31 .
drwxr-xr-x 6 falmesino staff 192 Feb 9 04:37 ..
-rw-r--r-- 1 falmesino staff 4391 Feb 9 04:39 BinomialHeap.php
-rw-r--r-- 1 falmesino staff 4553 Feb 9 04:07 FibonacciHeap.php
→ 1 git:(main) ✗ php FibonacciHeap.php
4
1
Heap is empty
→ 1 git:(main) ✗
main*
Launchpad
0 0
You, 5 minutes ago
Falmesino Abdul Hamid (4 minutes ago)
Ln 172, Col 49
Spaces: 2
UTF-8
LF
PHP
Colorize: 0 variables
Colorize
Prettier

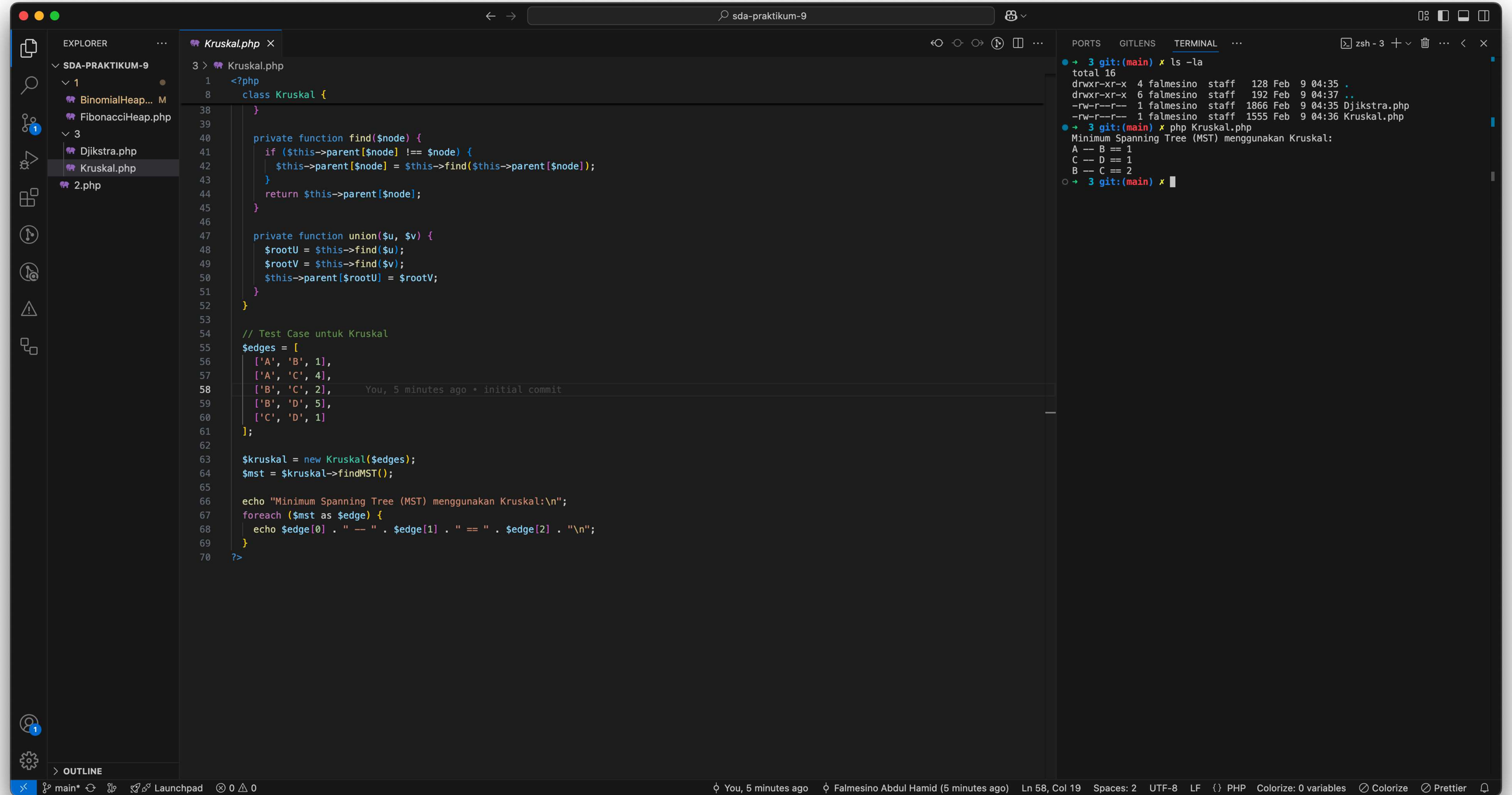
```php
<?php

  /**
   * 231232028 - Falmesino Abdul Hamid
   * 2. Buat priority queue menggunakan heap dan implementasikan operasi enqueue, dequeue, dan peek
   */


  class PriorityQueue {
    private $heap;

    public function __construct() {
      $this->heap = [];
    }

    private function parent($index) {
      return ($index - 1) >> 1;
    }

    private function leftChild($index) {
      return ($index << 1) + 1;
    }

    private function rightChild($index) {
      return ($index << 1) + 2;
    }

    private function swap($i, $j) {
      $temp = $this->heap[$i];
      $this->heap[$i] = $this->heap[$j];
      $this->heap[$j] = $temp;
    }

    private function heapifyUp($index) {
      while ($index > 0 && $this->heap[$this->parent($index)] < $this->heap[$index]) {
        $this->swap($index, $this->parent($index));
        $index = $this->parent($index);
      }
    }

    private function heapifyDown($index) {
      $maxIndex = $index;
      $left = $this->leftChild($index);
      $right = $this->rightChild($index);

      if ($left < count($this->heap) && $this->heap[$left] > $this->heap[$maxIndex]) {
        $maxIndex = $left;
      }

      if ($right < count($this->heap) && $this->heap[$right] > $this->heap[$maxIndex]) {
        $maxIndex = $right;
      }

      if ($index !== $maxIndex) {
        $this->swap($index, $maxIndex);
        $this->heapifyDown($maxIndex);
      }
    }

    public function enqueue($value) {
      array_push($this->heap, $value);
      $this->heapifyUp(count($this->heap) - 1);
    }

    public function dequeue() {
      if (count($this->heap) === 0) {
        throw new Exception("Queue is empty");
      }
      $max = $this->heap[0];
      $last = array_pop($this->heap);

      if (!empty($this->heap)) {
        $this->heap[0] = $last;
        $this->heapifyDown(0);
      }

      return $max;
    }

    public function peek() {
      if (count($this->heap) === 0) {
        throw new Exception("Queue is empty");
      }
      return $this->heap[0];
    }

    public function isEmpty() {
      return count($this->heap) === 0;
    }

    public function size() {
      return count($this->heap);
    }
  }

  // Test cases
  $pq = new PriorityQueue();

  $pq->enqueue(10);
  $pq->enqueue(20);
  $pq->enqueue(15);
  $pq->enqueue(30);
  $pq->enqueue(5);

  echo "Peek: " . $pq->peek() . "\n"; // Should print 30

  echo "Dequeue: " . $pq->dequeue() . "\n"; // Should print 30
  echo "Dequeue: " . $pq->dequeue() . "\n"; // Should print 20

  echo "Peek after dequeue: " . $pq->peek() . "\n"; // Should print 15

  $pq->enqueue(40);
  echo "Peek after enqueue 40: " . $pq->peek() . "\n"; // Should print 40

  echo "Size of queue: " . $pq->size() . "\n"; // Should print 4

  while (!$pq->isEmpty()) {
    echo "Dequeue: " . $pq->dequeue() . "\n";
  }

  echo "Size of queue after all dequeues: " . $pq->size() . "\n"; // Should print 0

  ?>
```


sda-praktikum-9
EXPLORER
SDA-PRAKTIKUM-9
1
BinomialHeap... M
FibonacciHeap.php
3
Djikstra.php
Kruskal.php
2.php
OUTLINE
2.php
You, 5 minutes ago | 1 author (You)
<?php
/**
* 231232028 - Falmesino Abdul Hamid
* 2. Buat priority queue menggunakan heap dan implementasikan operasi enqueue, dequeue, dan peek
*/
class PriorityQueue {
private $heap;
public function __construct() {
$this->heap = [];
}
private function parent($index) {
return ($index - 1) >> 1;
}
private function leftChild($index) {
You, 5 minutes ago • initial commit
return ($index << 1) + 1;
}
private function rightChild($index) {
return ($index << 1) + 2;
}
private function swap($i, $j) {
$temp = $this->heap[$i];
$this->heap[$i] = $this->heap[$j];
$this->heap[$j] = $temp;
}
private function heapifyUp($index) {
while ($index > 0 && $this->heap[$this->parent($index)] < $this->heap[$index]) {
$this->swap($index, $this->parent($index));
$index = $this->parent($index);
}
}
private function heapifyDown($index) {
$maxIndex = $index;
$left = $this->leftChild($index);
$right = $this->rightChild($index);
if ($left < count($this->heap) && $this->heap[$left] > $this->heap[$maxIndex]) {
$maxIndex = $left;
}
PORTS
GITLENS
TERMINAL
zsh
sda-praktikum-9 git:(main) ✗ ls -la
total 8
drwxr-xr-x 6 falmesino staff 192 Feb 9 04:37 .
drwxr-xr-x 39 falmesino staff 1248 Feb 8 20:34 ..
drwxr-xr-x 12 falmesino staff 384 Feb 9 04:37 .git
drwxr-xr-x 4 falmesino staff 128 Feb 8 22:31 1
-rw-r--r-- 1 falmesino staff 2922 Feb 9 04:33 2.php
drwxr-xr-x 4 falmesino staff 128 Feb 9 04:35 3
sda-praktikum-9 git:(main) ✗ php 2.php
Peek: 30
Dequeue: 30
Dequeue: 20
Peek after dequeue: 15
Peek after enqueue 40: 40
Size of queue: 4
Dequeue: 40
Dequeue: 15
Dequeue: 10
Dequeue: 5
Size of queue after all dequeues: 0
sda-praktikum-9 git:(main) ✗
main*
Launchpad
0 0
You, 5 minutes ago
Falmesino Abdul Hamid (4 minutes ago)
Ln 20, Col 41
Spaces: 2
UTF-8
LF
PHP
Colorize: 0 variables
Colorize
Prettier

```php
<?php

  /**
   * 231232028 - Falmesino Abdul Hamid
   * 3. Implementasikan algoritma pencarian jalur seperti Dijkstra dan Kruskal menggunakan priority queue.
   */

  class Dijkstra {
    private $graph;
    private $distance;
    private $previous;
    private $priorityQueue;

    public function __construct($graph) {
      $this->graph = $graph;
    }

    public function shortestPath($start, $end) {
      $this->distance = array_fill_keys(array_keys($this->graph), INF);
      $this->distance[$start] = 0;
      $this->previous = array_fill_keys(array_keys($this->graph), null);
      $this->priorityQueue = new SplPriorityQueue();
      $this->priorityQueue->insert($start, 0);

      while (!$this->priorityQueue->isEmpty()) {
        $u = $this->priorityQueue->extract();

        if ($u === $end) {
          return $this->getPath($end);
        }

        foreach ($this->graph[$u] as $v => $weight) {
          $alt = $this->distance[$u] + $weight;
          if ($alt < $this->distance[$v]) {
            $this->distance[$v] = $alt;
            $this->previous[$v] = $u;
            $this->priorityQueue->insert($v, -$alt);
          }
        }
      }

      return null;
    }

    private function getPath($end) {
      $path = [];
      while ($end !== null) {
        array_unshift($path, $end);
        $end = $this->previous[$end];
      }
      return $path;
    }
  }

  // test-case
  $graph = [
    'A' => ['B' => 1, 'C' => 4],
    'B' => ['A' => 1, 'C' => 2, 'D' => 5],
    'C' => ['A' => 4, 'B' => 2, 'D' => 1],
    'D' => ['B' => 5, 'C' => 1]
  ];

  $dijkstra = new Dijkstra($graph);
  $start = 'A';
  $end = 'D';
  $path = $dijkstra->shortestPath($start, $end);

  if ($path) {
    echo "Jalur terpendek dari $start ke $end adalah: " . implode(' -> ', $path) . "\n";
  } else {
    echo "Tidak ada jalur dari $start ke $end.\n";
  }
?>
```


sda-praktikum-9
EXPLORER
SDA-PRAKTIKUM-9
1
BinomialHeap... M
FibonacciHeap.php
3
Djikstra.php
Kruskal.php
2.php
OUTLINE
Djikstra.php
3 > Djikstra.php
You, 5 minutes ago | 1 author (You)
<?php
/**
* 231232028 - Falmesino Abdul Hamid
* 3. Implementasikan algoritma pencarian jalur seperti Dijkstra dan Kruskal menggunakan priority queue.
*/
class Dijkstra {
private $graph;
private $distance;
private $previous;
private $priorityQueue;
public function __construct($graph) {
You, 5 minutes ago • initial commit
$this->graph = $graph;
}
public function shortestPath($start, $end) {
$this->distance = array_fill_keys(array_keys($this->graph), INF);
$this->distance[$start] = 0;
$this->previous = array_fill_keys(array_keys($this->graph), null);
$this->priorityQueue = new SplPriorityQueue();
$this->priorityQueue->insert($start, 0);
while (!$this->priorityQueue->isEmpty()) {
$u = $this->priorityQueue->extract();
if ($u === $end) {
return $this->getPath($end);
}
foreach ($this->graph[$u] as $v => $weight) {
$alt = $this->distance[$u] + $weight;
if ($alt < $this->distance[$v]) {
$this->distance[$v] = $alt;
$this->previous[$v] = $u;
$this->priorityQueue->insert($v, -$alt);
}
}
}
return null;
}
private function getPath($end) {
$path = [];
while ($end !== null) {
array_unshift($path, $end);
$end = $this->previous[$end];
}
PORTS
GITLENS
TERMINAL
zsh - 3
3 git:(main) ✗ ls -la
total 16
drwxr-xr-x 4 falmesino staff 128 Feb 9 04:35 .
drwxr-xr-x 6 falmesino staff 192 Feb 9 04:37 ..
-rw-r--r-- 1 falmesino staff 1866 Feb 9 04:35 Djikstra.php
-rw-r--r-- 1 falmesino staff 1555 Feb 9 04:36 Kruskal.php
3 git:(main) ✗ php Djikstra.php
Jalur terpendek dari A ke D adalah: A -> B -> C -> D
3 git:(main) ✗
main*
Launchpad
0 0
You, 5 minutes ago
Falmesino Abdul Hamid (5 minutes ago)
Ln 14, Col 35
Spaces: 2
UTF-8
LF
PHP
Colorize: 0 variables
Colorize
Prettier

```php
<?php

  /**
   * 231232028 - Falmesino Abdul Hamid
   * 3. Implementasikan algoritma pencarian jalur seperti Dijkstra dan Kruskal menggunakan priority queue.
   */

  class Kruskal {
    private $edges;
    private $parent;

    public function __construct($edges) {
      $this->edges = $edges;
    }

    public function findMST() {
      usort($this->edges, function($a, $b) {
        return $a[2] - $b[2];
      });

      $this->parent = [];
      foreach ($this->edges as $edge) {
        $this->parent[$edge[0]] = $edge[0];
        $this->parent[$edge[1]] = $edge[1];
      }

      $mst = [];
      foreach ($this->edges as $edge) {
        $u = $edge[0];
        $v = $edge[1];
        if ($this->find($u) !== $this->find($v)) {
          $mst[] = $edge;
          $this->union($u, $v);
        }
      }

      return $mst;
    }

    private function find($node) {
      if ($this->parent[$node] !== $node) {
        $this->parent[$node] = $this->find($this->parent[$node]);
      }
      return $this->parent[$node];
    }

    private function union($u, $v) {
      $rootU = $this->find($u);
      $rootV = $this->find($v);
      $this->parent[$rootU] = $rootV;
    }
  }

  // Test Case untuk Kruskal
  $edges = [
    ['A', 'B', 1],
    ['A', 'C', 4],
    ['B', 'C', 2],
    ['B', 'D', 5],
    ['C', 'D', 1]
  ];

  $kruskal = new Kruskal($edges);
  $mst = $kruskal->findMST();

  echo "Minimum Spanning Tree (MST) menggunakan Kruskal:\n";
  foreach ($mst as $edge) {
    echo $edge[0] . " -- " . $edge[1] . " == " . $edge[2] . "\n";
  }
?>
```


sda-praktikum-9
EXPLORER
SDA-PRAKTIKUM-9
1
BinomialHeap... M
FibonacciHeap.php
3
Djikstra.php
Kruskal.php
2.php
OUTLINE
Kruskal.php
3 > Kruskal.php
<?php
class Kruskal {
}
private function find($node) {
if ($this->parent[$node] !== $node) {
$this->parent[$node] = $this->find($this->parent[$node]);
}
return $this->parent[$node];
}
private function union($u, $v) {
$rootU = $this->find($u);
$rootV = $this->find($v);
$this->parent[$rootU] = $rootV;
}
}
// Test Case untuk Kruskal
$edges = [
['A', 'B', 1],
['A', 'C', 4],
['B', 'C', 2],
You, 5 minutes ago • initial commit
['B', 'D', 5],
['C', 'D', 1]
];
$kruskal = new Kruskal($edges);
$mst = $kruskal->findMST();
echo "Minimum Spanning Tree (MST) menggunakan Kruskal:\n";
foreach ($mst as $edge) {
echo $edge[0] . " -- " . $edge[1] . " == " . $edge[2] . "\n";
}
?>
PORTS
GITLENS
TERMINAL
zsh - 3
→ 3 git:(main) ✗ ls -la
total 16
drwxr-xr-x 4 falmesino staff 128 Feb 9 04:35 .
drwxr-xr-x 6 falmesino staff 192 Feb 9 04:37 ..
-rw-r--r-- 1 falmesino staff 1866 Feb 9 04:35 Djikstra.php
-rw-r--r-- 1 falmesino staff 1555 Feb 9 04:36 Kruskal.php
→ 3 git:(main) ✗ php Kruskal.php
Minimum Spanning Tree (MST) menggunakan Kruskal:
A -- B == 1
C -- D == 1
B -- C == 2
→ 3 git:(main) ✗
main*
Launchpad
0 0
You, 5 minutes ago
Falmesino Abdul Hamid (5 minutes ago)
Ln 58, Col 19
Spaces: 2
UTF-8
LF
PHP
Colorize: 0 variables
Colorize
Prettier