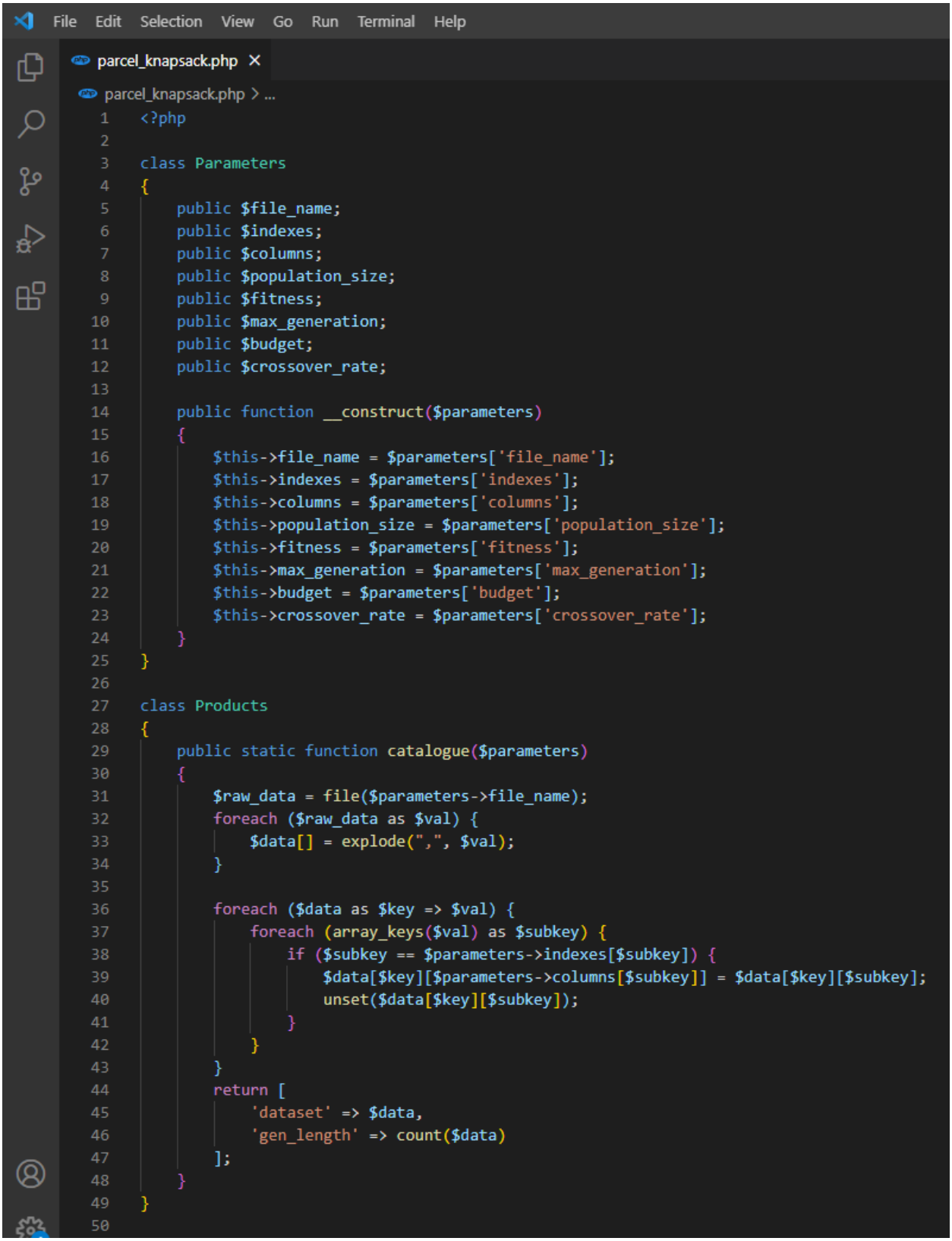
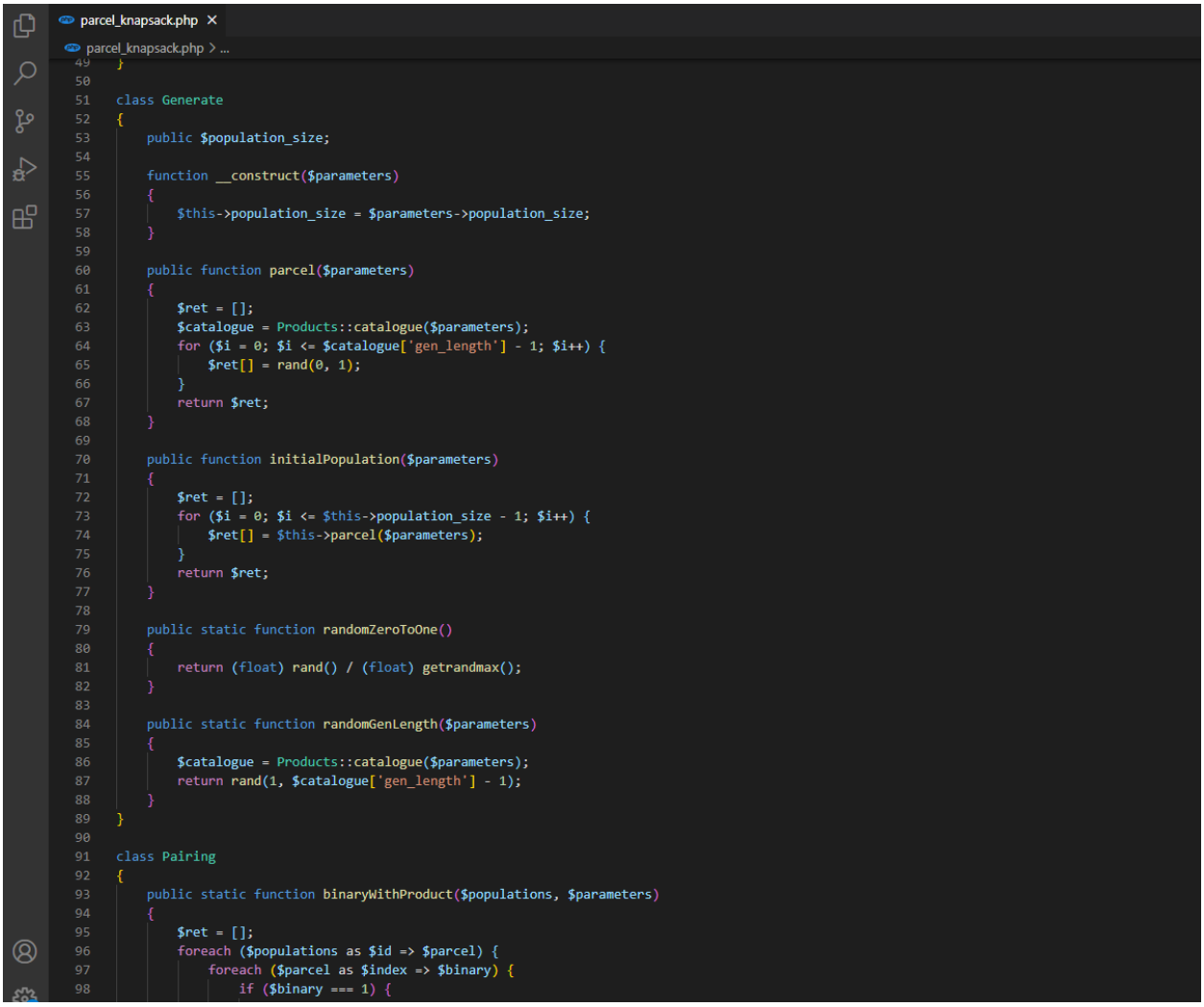
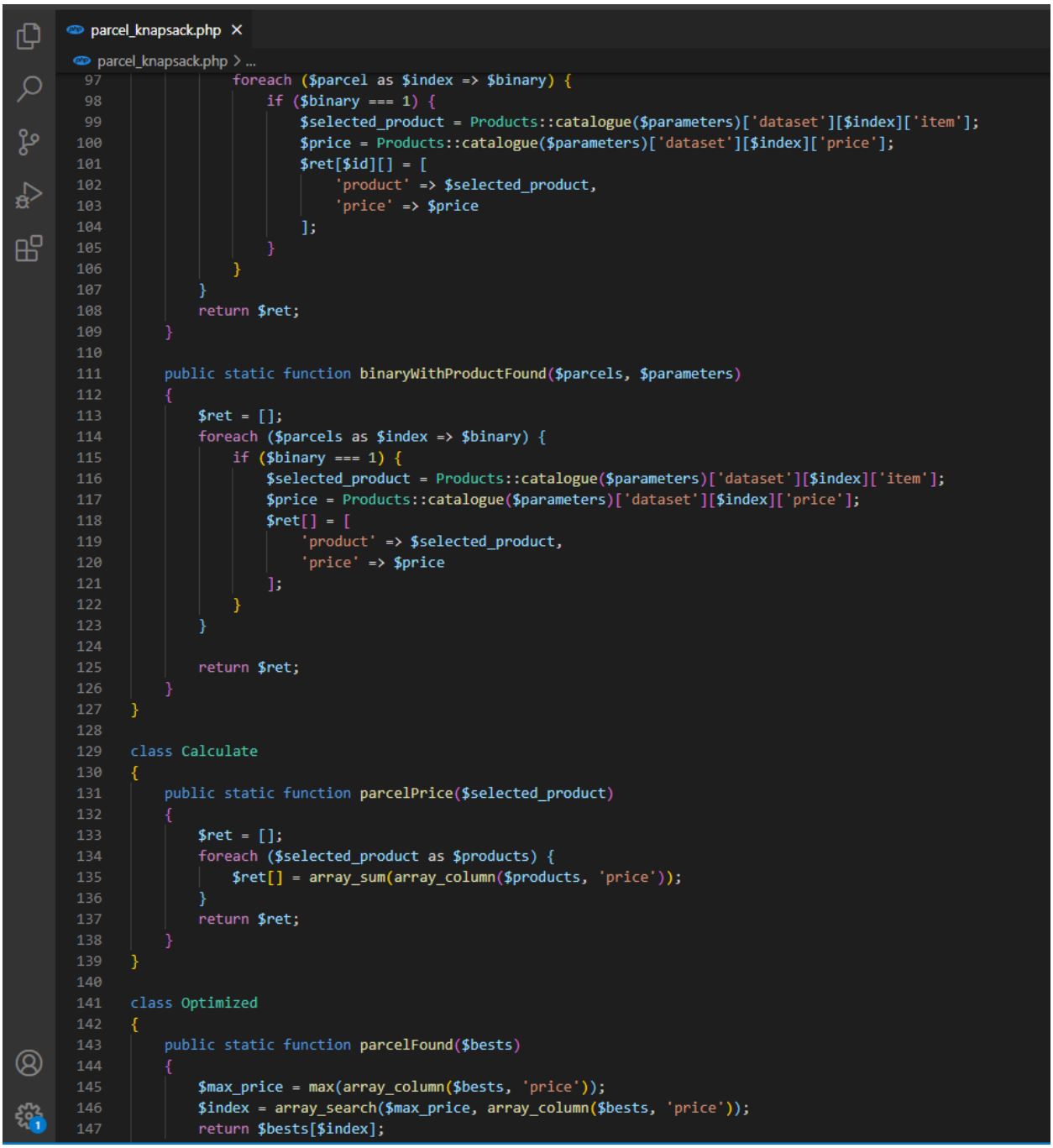

File Edit Selection View Go Run Terminal Help
parcel_knapsack.php
parcel_knapsack.php > ...
<?php
class Parameters
{
public $file_name;
public $indexes;
public $columns;
public $population_size;
public $fitness;
public $max_generation;
public $budget;
public $crossover_rate;
public function __construct($parameters)
{
$this->file_name = $parameters['file_name'];
$this->indexes = $parameters['indexes'];
$this->columns = $parameters['columns'];
$this->population_size = $parameters['population_size'];
$this->fitness = $parameters['fitness'];
$this->max_generation = $parameters['max_generation'];
$this->budget = $parameters['budget'];
$this->crossover_rate = $parameters['crossover_rate'];
}
}
class Products
{
public static function catalogue($parameters)
{
$raw_data = file($parameters->file_name);
foreach ($raw_data as $val) {
$data[] = explode(",", $val);
}
foreach ($data as $key => $val) {
foreach (array_keys($val) as $subkey) {
if ($subkey == $parameters->indexes[$subkey]) {
$data[$key][$parameters->columns[$subkey]] = $data[$key][$subkey];
unset($data[$key][$subkey]);
}
}
}
return [
'dataset' => $data,
'gen_length' => count($data)
];
}
}

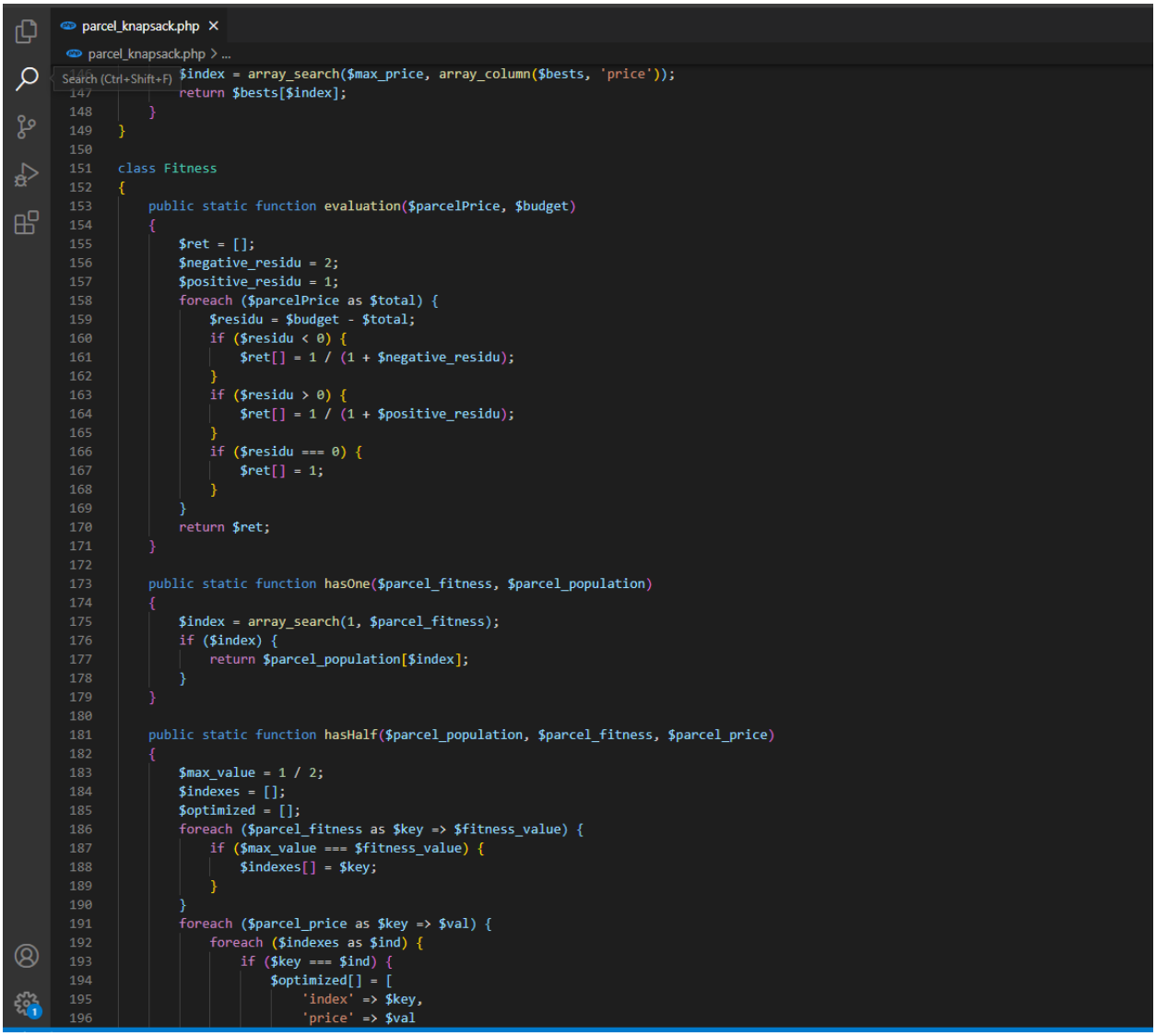


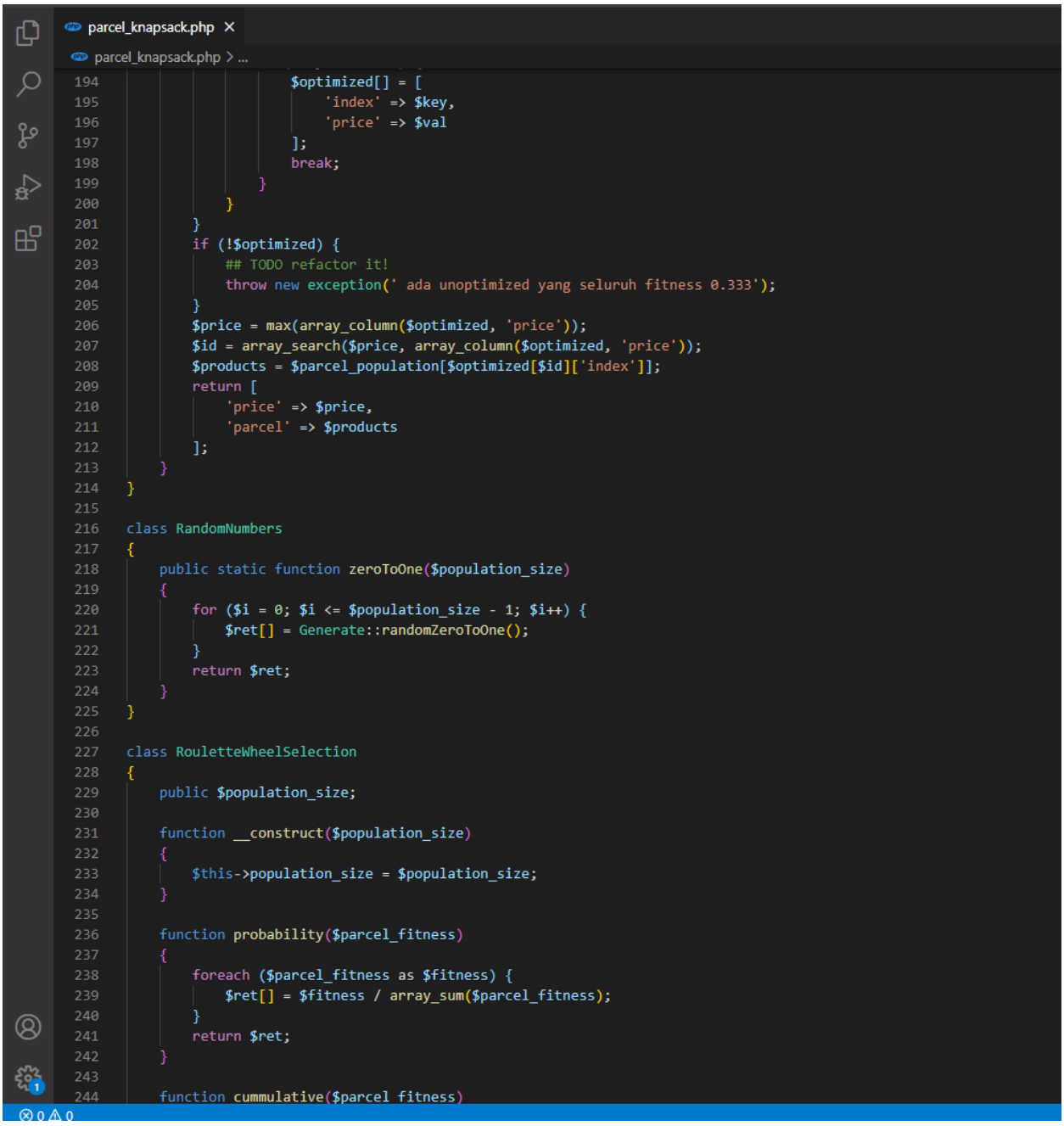
parcel_knapsack.php

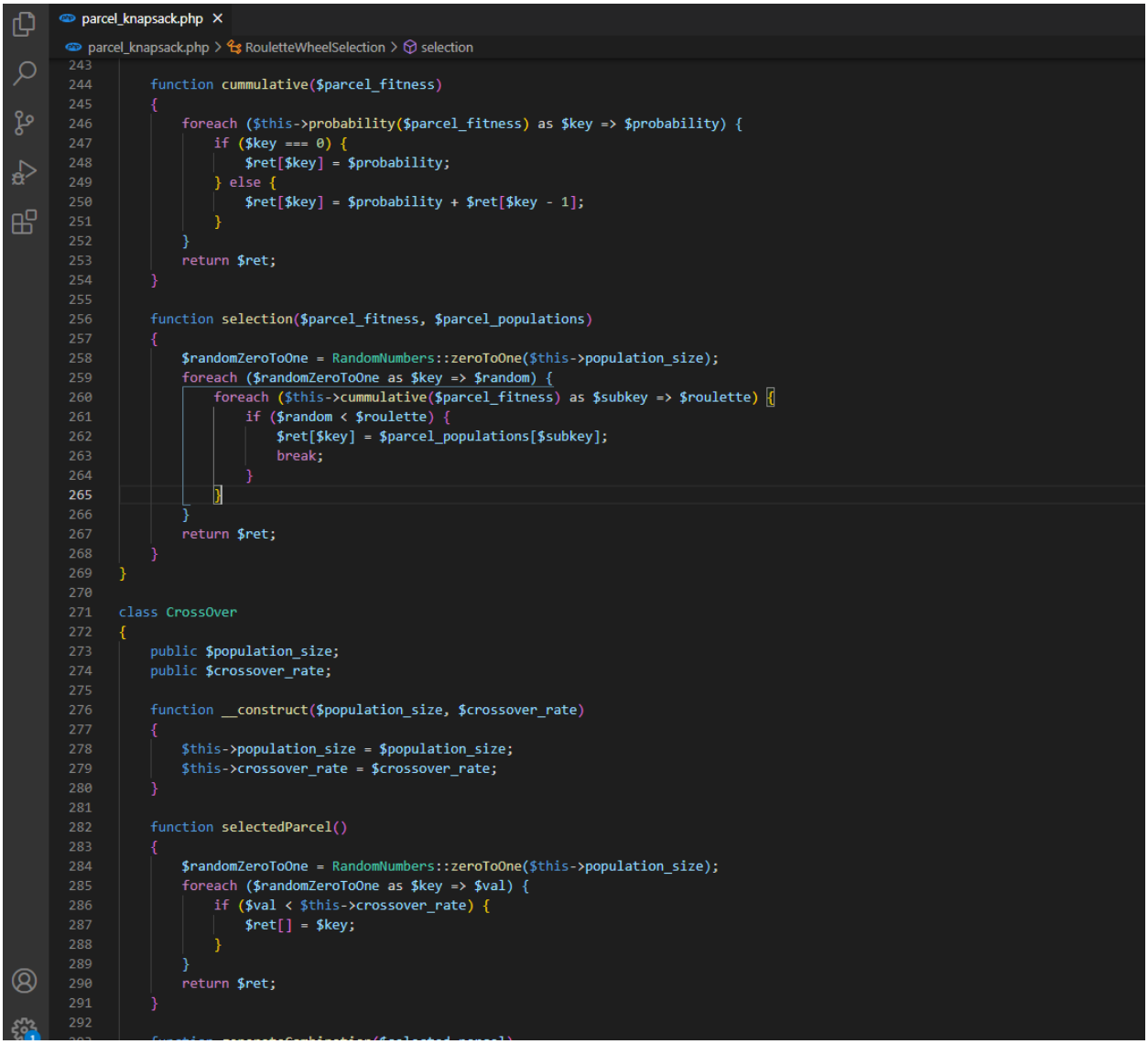
parcel_knapsack.php > ...

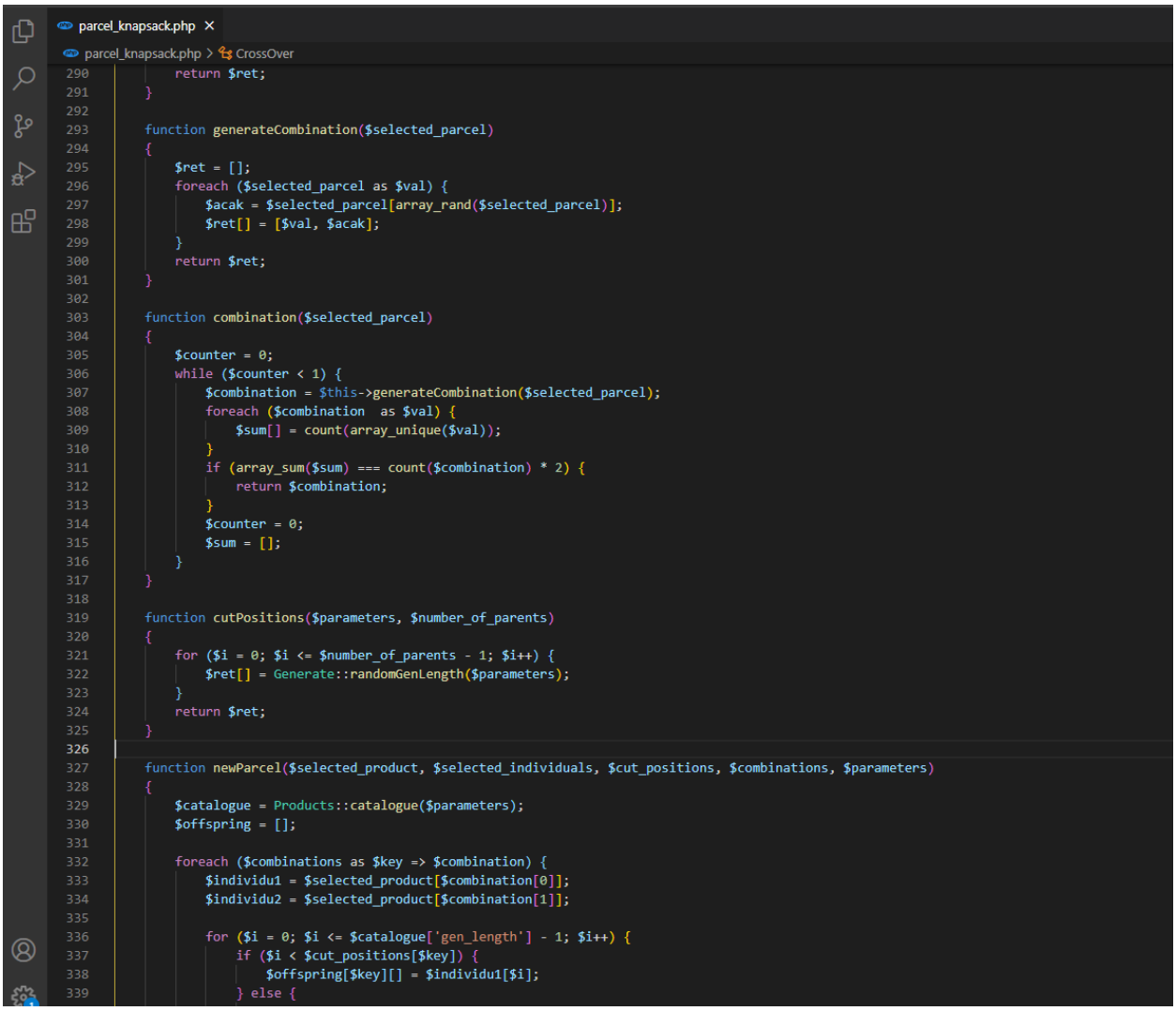
```php
}

class Generate
{
    public $population_size;

    function __construct($parameters)
    {
        $this->population_size = $parameters->population_size;
    }

    public function parcel($parameters)
    {
        $ret = [];
        $catalogue = Products::catalogue($parameters);
        for ($i = 0; $i <= $catalogue['gen_length'] - 1; $i++) {
            $ret[] = rand(0, 1);
        }
        return $ret;
    }

    public function initialPopulation($parameters)
    {
        $ret = [];
        for ($i = 0; $i <= $this->population_size - 1; $i++) {
            $ret[] = $this->parcel($parameters);
        }
        return $ret;
    }

    public static function randomZeroToOne()
    {
        return (float) rand() / (float) getrandmax();
    }

    public static function randomGenLength($parameters)
    {
        $catalogue = Products::catalogue($parameters);
        return rand(1, $catalogue['gen_length'] - 1);
    }
}

class Pairing
{
    public static function binaryWithProduct($populations, $parameters)
    {
        $ret = [];
        foreach ($populations as $id => $parcel) {
            foreach ($parcel as $index => $binary) {
                if ($binary === 1) {
```

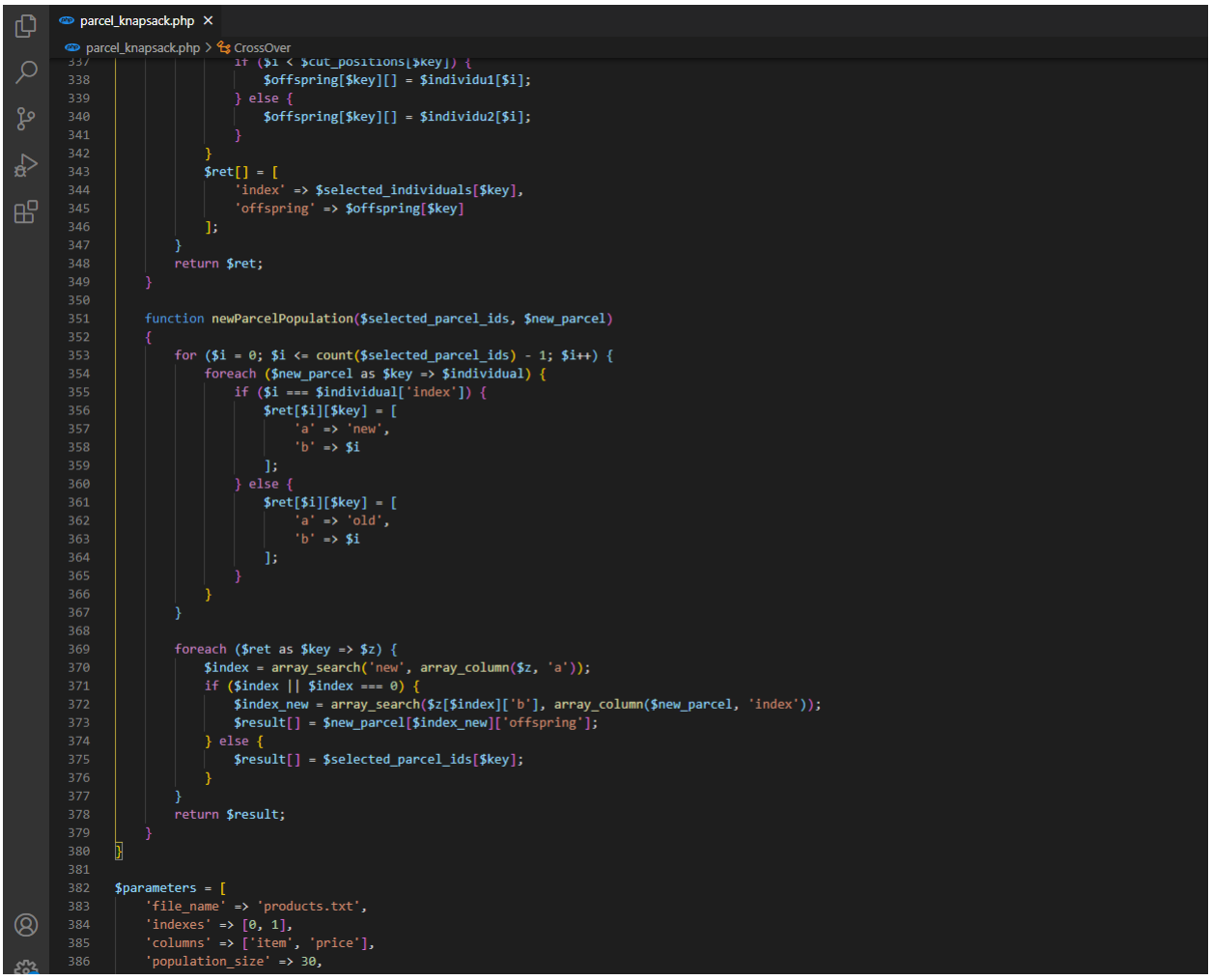

parcel_knapsack.php
parcel_knapsack.php > ...
97 foreach ($parcel as $index => $binary) {
98 if ($binary === 1) {
99 $selected_product = Products::catalogue($parameters)['dataset'][$index]['item'];
100 $price = Products::catalogue($parameters)['dataset'][$index]['price'];
101 $ret[$id][] = [
102 'product' => $selected_product,
103 'price' => $price
104];
105 }
106 }
107 }
108 return $ret;
109 }
110
111 public static function binaryWithProductFound($parcels, $parameters)
112 {
113 $ret = [];
114 foreach ($parcels as $index => $binary) {
115 if ($binary === 1) {
116 $selected_product = Products::catalogue($parameters)['dataset'][$index]['item'];
117 $price = Products::catalogue($parameters)['dataset'][$index]['price'];
118 $ret[] = [
119 'product' => $selected_product,
120 'price' => $price
121];
122 }
123 }
124
125 return $ret;
126 }
127 }
128
129 class Calculate
130 {
131 public static function parcelPrice($selected_product)
132 {
133 $ret = [];
134 foreach ($selected_product as $products) {
135 $ret[] = array_sum(array_column($products, 'price'));
136 }
137 return $ret;
138 }
139 }
140
141 class Optimized
142 {
143 public static function parcelFound($bests)
144 {
145 $max_price = max(array_column($bests, 'price'));
146 $index = array_search($max_price, array_column($bests, 'price'));
147 return $bests[$index];

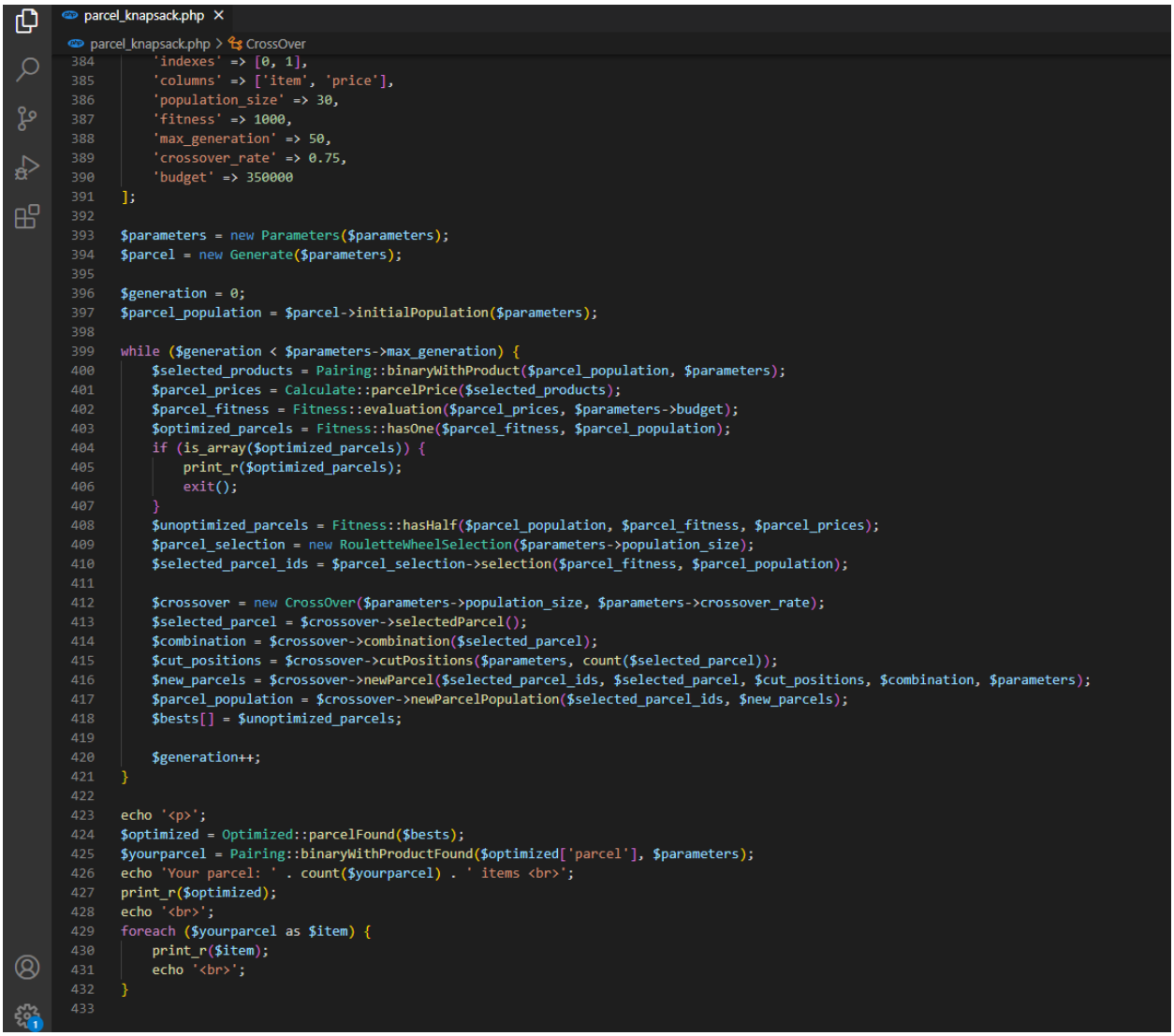

parcel_knapsack.php
parcel_knapsack.php > ...
Search (Ctrl+Shift+F)
$index = array_search($max_price, array_column($bests, 'price'));
return $bests[$index];
}
}
class Fitness
{
public static function evaluation($parcelPrice, $budget)
{
$ret = [];
$negative_residu = 2;
$positive_residu = 1;
foreach ($parcelPrice as $total) {
$residu = $budget - $total;
if ($residu < 0) {
$ret[] = 1 / (1 + $negative_residu);
}
if ($residu > 0) {
$ret[] = 1 / (1 + $positive_residu);
}
if ($residu === 0) {
$ret[] = 1;
}
}
return $ret;
}
public static function hasOne($parcel_fitness, $parcel_population)
{
$index = array_search(1, $parcel_fitness);
if ($index) {
return $parcel_population[$index];
}
}
public static function hasHalf($parcel_population, $parcel_fitness, $parcel_price)
{
$max_value = 1 / 2;
$indexes = [];
$optimized = [];
foreach ($parcel_fitness as $key => $fitness_value) {
if ($max_value === $fitness_value) {
$indexes[] = $key;
}
}
foreach ($parcel_price as $key => $val) {
foreach ($indexes as $ind) {
if ($key === $ind) {
$optimized[] = [
'index' => $key,
'price' => $val


parcel_knapsack.php
parcel_knapsack.php > ...
194 $optimized[] = [
195 'index' => $key,
196 'price' => $val
197];
198 break;
199 }
200 }
201 }
202 if (!$optimized) {
203 ## TODO refactor it!
204 throw new exception(' ada unoptimized yang seluruh fitness 0.333');
205 }
206 $price = max(array_column($optimized, 'price'));
207 $id = array_search($price, array_column($optimized, 'price'));
208 $products = $parcel_population[$optimized[$id]['index']];
209 return [
210 'price' => $price,
211 'parcel' => $products
212];
213 }
214 }
215
216 class RandomNumbers
217 {
218 public static function zeroToOne($population_size)
219 {
220 for ($i = 0; $i <= $population_size - 1; $i++) {
221 $ret[] = Generate::randomZeroToOne();
222 }
223 return $ret;
224 }
225 }
226
227 class RouletteWheelSelection
228 {
229 public $population_size;
230
231 function __construct($population_size)
232 {
233 $this->population_size = $population_size;
234 }
235
236 function probability($parcel_fitness)
237 {
238 foreach ($parcel_fitness as $fitness) {
239 $ret[] = $fitness / array_sum($parcel_fitness);
240 }
241 return $ret;
242 }
243
244 function cummulative($parcel_fitness)
⊗ 0 ⚠ 0

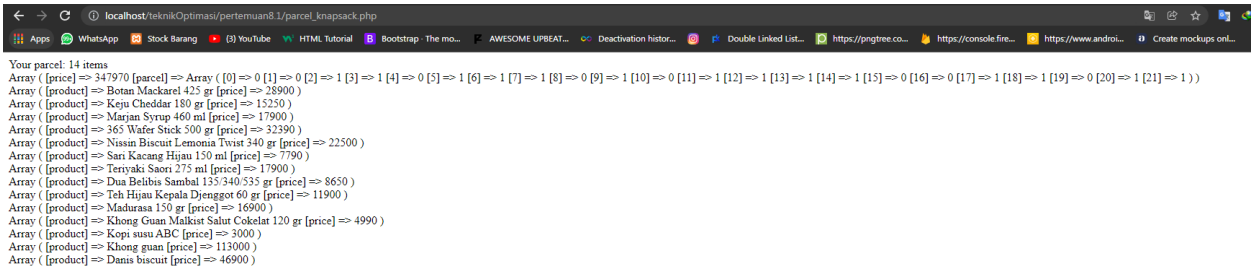

parcel_knapsack.php
parcel_knapsack.php > RouletteWheelSelection > selection
function cummulative($parcel_fitness)
{
foreach ($this->probability($parcel_fitness) as $key => $probability) {
if ($key === 0) {
$ret[$key] = $probability;
} else {
$ret[$key] = $probability + $ret[$key - 1];
}
}
return $ret;
}
function selection($parcel_fitness, $parcel_populations)
{
$randomZeroToOne = RandomNumbers::zeroToOne($this->population_size);
foreach ($randomZeroToOne as $key => $random) {
foreach ($this->cummulative($parcel_fitness) as $subkey => $roulette) {
if ($random < $roulette) {
$ret[$key] = $parcel_populations[$subkey];
break;
}
}
}
return $ret;
}
}
class CrossOver
{
public $population_size;
public $crossover_rate;
function __construct($population_size, $crossover_rate)
{
$this->population_size = $population_size;
$this->crossover_rate = $crossover_rate;
}
function selectedParcel()
{
$randomZeroToOne = RandomNumbers::zeroToOne($this->population_size);
foreach ($randomZeroToOne as $key => $val) {
if ($val < $this->crossover_rate) {
$ret[] = $key;
}
}
return $ret;
}


parcel_knapsack.php
parcel_knapsack.php > CrossOver
290 return $ret;
291 }
292
293 function generateCombination($selected_parcel)
294 {
295 $ret = [];
296 foreach ($selected_parcel as $val) {
297 $acak = $selected_parcel[array_rand($selected_parcel)];
298 $ret[] = [$val, $acak];
299 }
300 return $ret;
301 }
302
303 function combination($selected_parcel)
304 {
305 $counter = 0;
306 while ($counter < 1) {
307 $combination = $this->generateCombination($selected_parcel);
308 foreach ($combination as $val) {
309 $sum[] = count(array_unique($val));
310 }
311 if (array_sum($sum) === count($combination) * 2) {
312 return $combination;
313 }
314 $counter = 0;
315 $sum = [];
316 }
317 }
318
319 function cutPositions($parameters, $number_of_parents)
320 {
321 for ($i = 0; $i <= $number_of_parents - 1; $i++) {
322 $ret[] = Generate::randomGenLength($parameters);
323 }
324 return $ret;
325 }
326
327 function newParcel($selected_product, $selected_individuals, $cut_positions, $combinations, $parameters)
328 {
329 $catalogue = Products::catalogue($parameters);
330 $offspring = [];
331
332 foreach ($combinations as $key => $combination) {
333 $individu1 = $selected_product[$combination[0]];
334 $individu2 = $selected_product[$combination[1]];
335
336 for ($i = 0; $i <= $catalogue['gen_length'] - 1; $i++) {
337 if ($i < $cut_positions[$key]) {
338 $offspring[$key][] = $individu1[$i];
339 } else {


parcel_knapsack.php
parcel_knapsack.php > CrossOver
if ($i < $cut_positions[$key]) {
$offspring[$key][] = $individu1[$i];
} else {
$offspring[$key][] = $individu2[$i];
}
}
$ret[] = [
'index' => $selected_individuals[$key],
'offspring' => $offspring[$key]
];
}
return $ret;
}
function newParcelPopulation($selected_parcel_ids, $new_parcel)
{
for ($i = 0; $i <= count($selected_parcel_ids) - 1; $i++) {
foreach ($new_parcel as $key => $individual) {
if ($i === $individual['index']) {
$ret[$i][$key] = [
'a' => 'new',
'b' => $i
];
} else {
$ret[$i][$key] = [
'a' => 'old',
'b' => $i
];
}
}
}
foreach ($ret as $key => $z) {
$index = array_search('new', array_column($z, 'a'));
if ($index || $index === 0) {
$index_new = array_search($z[$index]['b'], array_column($new_parcel, 'index'));
$result[] = $new_parcel[$index_new]['offspring'];
} else {
$result[] = $selected_parcel_ids[$key];
}
}
return $result;
}
}
$parameters = [
'file_name' => 'products.txt',
'indexes' => [0, 1],
'columns' => ['item', 'price'],
'population_size' => 30,


parcel_knapsack.php
parcel_knapsack.php > CrossOver
'indexes' => [0, 1],
'columns' => ['item', 'price'],
'population_size' => 30,
'fitness' => 1000,
'max_generation' => 50,
'crossover_rate' => 0.75,
'budget' => 350000
];
$parameters = new Parameters($parameters);
$parcel = new Generate($parameters);
$generation = 0;
$parcel_population = $parcel->initialPopulation($parameters);
while ($generation < $parameters->max_generation) {
$selected_products = Pairing::binaryWithProduct($parcel_population, $parameters);
$parcel_prices = Calculate::parcelPrice($selected_products);
$parcel_fitness = Fitness::evaluation($parcel_prices, $parameters->budget);
$optimized_parcels = Fitness::hasOne($parcel_fitness, $parcel_population);
if (is_array($optimized_parcels)) {
print_r($optimized_parcels);
exit();
}
$unoptimized_parcels = Fitness::hasHalf($parcel_population, $parcel_fitness, $parcel_prices);
$parcel_selection = new RouletteWheelSelection($parameters->population_size);
$selected_parcel_ids = $parcel_selection->selection($parcel_fitness, $parcel_population);
$crossover = new CrossOver($parameters->population_size, $parameters->crossover_rate);
$selected_parcel = $crossover->selectedParcel();
$combination = $crossover->combination($selected_parcel);
$cut_positions = $crossover->cutPositions($parameters, count($selected_parcel));
$new_parcels = $crossover->newParcel($selected_parcel_ids, $selected_parcel, $cut_positions, $combination, $parameters);
$parcel_population = $crossover->newParcelPopulation($selected_parcel_ids, $new_parcels);
$bests[] = $unoptimized_parcels;
$generation++;
}
echo '<p>';
$optimized = Optimized::parcelFound($bests);
$yourparcel = Pairing::binaryWithProductFound($optimized['parcel'], $parameters);
echo 'Your parcel: ' . count($yourparcel) . ' items
';
print_r($optimized);
echo '
';
foreach ($yourparcel as $item) {
print_r($item);
echo '
';
}

Hasil:


localhost/teknikOptimasi/pertemuan8.1/parcel_knapsack.php
Apps
WhatsApp
Stock Barang
(3) YouTube
HTML Tutorial
Bootstrap · The mo...
AWESOME UPBEAT...
Deactivation histor...
Double Linked List...
https://pngtree.co...
https://console.fire...
https://www.androi...
Create mockups onl...
Your parcel: 14 items
Array ([price] => 347970 [parcel] => Array ([0] => 0 [1] => 0 [2] => 1 [3] => 1 [4] => 0 [5] => 1 [6] => 1 [7] => 1 [8] => 0 [9] => 1 [10] => 0 [11] => 1 [12] => 1 [13] => 1 [14] => 1 [15] => 0 [16] => 0 [17] => 1 [18] => 1 [19] => 0 [20] => 1 [21] => 1))
Array ([product] => Botan Mackarel 425 gr [price] => 28900)
Array ([product] => Keju Cheddar 180 gr [price] => 15250)
Array ([product] => Marjan Syrup 460 ml [price] => 17900)
Array ([product] => 365 Wafer Stick 500 gr [price] => 32390)
Array ([product] => Nissin Biscuit Lemonia Twist 340 gr [price] => 22500)
Array ([product] => Sari Kacang Hijau 150 ml [price] => 7790)
Array ([product] => Teriyaki Saori 275 ml [price] => 17900)
Array ([product] => Dua Belibis Sambal 135/340/535 gr [price] => 8650)
Array ([product] => Teh Hijau Kepala Djenggot 60 gr [price] => 11900)
Array ([product] => Madurasa 150 gr [price] => 16900)
Array ([product] => Khong Guan Malkist Salut Cokelat 120 gr [price] => 4990)
Array ([product] => Kopi susu ABC [price] => 3000)
Array ([product] => Khong guan [price] => 113000)
Array ([product] => Danis biscuit [price] => 46900)