

```
algen_knapsack.php X
algen_knapsack.php > ...
1  k?php
2
3  class Parameters
4  {
5      const FILE_NAME = 'products.txt';
6      const COLUMNS = ['item', 'price'];
7      const POPULATION_SIZE = 30;
8      const BUDGET = 280000;
9      const STOPPING_VALUE = 10000;
10     const CROSOVER_RATE = 0.8;
11 }
12
13 class Catalogue
14 {
15     function createProductColumn($listOfRawProduct)
16     {
17         foreach (array_keys($listOfRawProduct) as $listOfRawProductKey) {
18             $listOfRawProduct[Parameters::COLUMNS[$listOfRawProductKey]] = $listOfRawProduct[$listOfRawProductKey];
19             unset($listOfRawProduct[$listOfRawProductKey]);
20         }
21         return $listOfRawProduct;
22     }
23
24     function product()
25     {
26         $collectionOfListProduct = [];
27         $raw_data = file(Parameters::FILE_NAME);
28         foreach ($raw_data as $listOfRawProduct) {
29             $collectionOfListProduct[] = $this->createProductColumn(explode(",", $listOfRawProduct));
30         }
31         return $collectionOfListProduct;
32     }
33 }
34
35 class Individu
36 {
37     function countNumberOfGen()
38     {
39         $catalogue = new Catalogue;
40         return count($catalogue->product());
41     }
42
43     function createRandomIndividu()
44     {
45         for ($i = 0; $i <= $this->countNumberOfGen() - 1; $i++) {
46             $ret[] = rand(0, 1);
47         }
48         return $ret;
49     }
50 }
51
```

```
algen_knapsack.php X
algen_knapsack.php > ...
48     return $ret;
49 }
50 }
51
52 class Population
53 {
54     function createRandomPopulation()
55     {
56         $individu = new Individu;
57         for ($i = 0; $i <= Parameters::POPULATION_SIZE - 1; $i++) {
58             $ret[] = $individu->createRandomIndividu();
59         }
60         return $ret;
61     }
62 }
63
64 class Fitness
65 {
66     function selectingItem($individu)
67     {
68         $catalogue = new Catalogue;
69         foreach ($individu as $individuKey => $binaryGen) {
70             if ($binaryGen == 1) {
71                 $ret[] = [
72                     'selectedKey' => $individuKey,
73                     'selectedPrice' => $catalogue->product()[$individuKey]['price']
74                 ];
75             }
76         }
77         return $ret;
78     }
79
80     function calculateFitnessValue($individu)
81     {
82         return array_sum(array_column($this->selectingItem($individu), 'selectedPrice'));
83     }
84
85     function countSelectedItem($individu)
86     {
87         return count($this->selectingItem($individu));
88     }
89
90     function searchBestIndividu($fits, $maxItem, $numberOfIndividuHasMaxItem)
91     {
92         if ($numberOfIndividuHasMaxItem == 1) {
93             $index = array_search($maxItem, array_column($fits, 'numberOfSelectedItem'));
94             return $fits[$index];
95         } else {
96             foreach ($fits as $key => $val) {
97                 if ($val['numberOfSelectedItem'] == $maxItem) {
```

```

87         return count($this->selectingItem($individu));
88     }
89
90     function searchBestIndividu($fits, $maxItem, $numberOfIndividuHasMaxItem)
91     {
92         if ($numberOfIndividuHasMaxItem === 1) {
93             $index = array_search($maxItem, array_column($fits, 'numberOfSelectedItem'));
94             return $fits[$index];
95         } else {
96             foreach ($fits as $key => $val) {
97                 if ($val['numberOfSelectedItem'] === $maxItem) {
98                     echo $key . ' ' . $val['fitnessValue'] . '<br>';
99                     $ret[] = [
100                         'individuKey' => $key,
101                         'fitnessValue' => $val['fitnessValue']
102                     ];
103                 }
104             }
105             if (count(array_unique(array_column($ret, 'fitnessValue'))) === 1) {
106                 $index = rand(0, count($ret) - 1);
107             } else {
108                 $max = max(array_column($ret, 'fitnessValue'));
109                 $index = array_search($max, array_column($ret, 'fitnessValue'));
110             }
111             echo 'Hasil';
112             return $ret[$index];
113         }
114     }
115
116     function isFound($fits)
117     {
118         $countedMaxItems = array_count_values(array_column($fits, 'numberOfSelectedItem'));
119         //print_r($countedMaxItems);
120         //echo '<br>';
121         $maxItem = max(array_keys($countedMaxItems));
122         //echo $maxItem;
123         //echo '<br>';
124         //echo $countedMaxItems[$maxItem];
125         $numberOfIndividuHasMaxItem = $countedMaxItems[$maxItem];
126
127         $bestFitnessValue = $this->searchBestIndividu($fits, $maxItem, $numberOfIndividuHasMaxItem)['fitnessValue'];
128         echo '<br>';
129         echo '<br>Best fitness value: ' . $bestFitnessValue;
130
131         $residual = Parameters::BUDGET - $bestFitnessValue;
132         echo ' Residual: ' . $residual;
133
134         if ($residual <= Parameters::STOPPING_VALUE && $residual > 0) {
135             return TRUE;
136         }

```

```

algen_knapsack.php X
algen_knapsack.php > ...
$residual = Parameters::BUDGET - $bestFitnessValue;
echo ' Residual: ' . $residual;

133
134     if ($residual <= Parameters::STOPPING_VALUE && $residual > 0) {
135         return TRUE;
136     }
137 }
138
139 function isFit($fitnessValue)
140 {
141     if ($fitnessValue <= Parameters::BUDGET) {
142         return TRUE;
143     }
144 }
145
146 function fitnessEvaluation($population)
147 {
148     $catalogue = new Catalogue;
149     foreach ($population as $listOfIndividuKey => $listOfIndividu) {
150         echo 'Individu-' . $listOfIndividuKey . '<br>';
151         foreach ($listOfIndividu as $individuKey => $binaryGen) {
152             //echo $binaryGen . '&nbsp;&nbsp;&nbsp;';
153             //print_r($catalogue->product()[$individuKey]);
154             //echo '<br>';
155         }
156         $fitnessValue = $this->calculateFitnessValue($listOfIndividu);
157         $numberOfSelectedItem = $this->countSelectedItem($listOfIndividu);
158         echo 'Max. Item: ' . $numberOfSelectedItem;
159         echo ' Fitness value: ' . $fitnessValue;
160         if ($this->isFit($fitnessValue)) {
161             echo ' (Fit)';
162             $fits[] = [
163                 'selectedIndividuKey' => $listOfIndividuKey,
164                 'numberOfSelectedItem' => $numberOfSelectedItem,
165                 'fitnessValue' => $fitnessValue
166             ];
167             //echo '<p>';
168             //print_r($fits);
169         } else {
170             echo ' (Not Fit)';
171         }
172         echo '<p>';
173     }
174
175     if ($this->isFound($fits)) {
176         echo ' Found';
177     } else {
178         echo ' >> Next generation';
179     }
180 }
181 }

```

```
algen_knapsack.php ×
algen_knapsack.php > ...

180     }
181 }
182
183 class Crossover
184 {
185     public $populations;
186
187     function __construct($populations)
188     {
189         $this->populations = $populations;
190     }
191
192     function randomZeroToOne()
193     {
194         return (float) rand() / (float) getrandmax();
195     }
196
197     function generateCrossover()
198     {
199         for ($i = 0; $i <= Parameters::POPULATION_SIZE - 1; $i++) {
200             $randomZeroToOne = $this->randomZeroToOne();
201             if ($randomZeroToOne < Parameters::CROSOVER_RATE) {
202                 $parents[$i] = $randomZeroToOne;
203             }
204         }
205         foreach (array_keys($parents) as $key) {
206             foreach (array_keys($parents) as $subkey) {
207                 if ($key != $subkey) {
208                     $ret[] = [$key, $subkey];
209                 }
210             }
211             array_shift($parents);
212         }
213         return $ret;
214     }
215
216     function offspring($parent1, $parent2, $cutPointIndex, $offspring)
217     {
218         $lengthOfGen = new Individu;
219         if ($offspring == 1) {
220             for ($i = 0; $i <= $lengthOfGen->countNumberOfGen() - 1; $i++) {
221                 if ($i <= $cutPointIndex) {
222                     $ret[] = $parent1[$i];
223                 }
224                 if ($i > $cutPointIndex) {
225                     $ret[] = $parent2[$i];
226                 }
227             }
228         }
229     }
230 }
```

```
algen_knapsack.php X
algen_knapsack.php > ...
227     }
228 }
229
230 if ($offspring == 2) {
231     for ($i = 0; $i <= $lengthOfGen->countNumberOfGen() - 1; $i++) {
232         if ($i <= $cutPointIndex) {
233             $ret[] = $parent2[$i];
234         }
235         if ($i > $cutPointIndex) {
236             $ret[] = $parent1[$i];
237         }
238     }
239 }
240 return $ret;
241 }
242
243 function cutPointRandom()
244 {
245     $lengthOfGen = new Individu;
246     return rand(0, $lengthOfGen->countNumberOfGen() - 1);
247 }
248
249 function crossover()
250 {
251     $cutPointIndex = $this->cutPointRandom();
252     //echo $cutPointIndex;
253     foreach ($this->generateCrossover() as $listOfCrossover) {
254         $parent1 = $this->populations[$listOfCrossover[0]];
255         $parent2 = $this->populations[$listOfCrossover[1]];
256         // echo '<p></p>';
257         // echo 'Parents :<br>';
258         // foreach ($parent1 as $gen) {
259             // echo $gen;
260         // }
261         // echo ' >< ';
262         // foreach ($parent2 as $gen) {
263             // echo $gen;
264         // }
265         // echo '<br>';
266
267         // echo 'Offspring<br>';
268         $offspring1 = $this->offspring($parent1, $parent2, $cutPointIndex, 1);
269         $offspring2 = $this->offspring($parent1, $parent2, $cutPointIndex, 2);
270         // foreach ($offspring1 as $gen) {
271             // echo $gen;
272         // }
273         // echo ' >< ';
274         // foreach ($offspring2 as $gen) {
275             // echo $gen;
276         // }
277     }
}
```

```
algen_knapsack.php X
algen_knapsack.php > ...
275         // echo $gen;
276         // }
277         // echo '<br>';
278         $offsprings[] = $offspring1;
279         $offsprings[] = $offspring2;
280     }
281     return $offsprings;
282 }
283 }
284
285 class Randomizer
286 {
287     static function getRandomIndexOfGen()
288     {
289         return rand(0, (new Individu())->countNumberOfGen() - 1);
290     }
291
292     static function getRandomIndexOfIndividu()
293     {
294         return rand(0, Parameters::POPULATION_SIZE - 1);
295     }
296 }
297
298 class Mutation
299 {
300     function __construct($population)
301     {
302         $this->population = $population;
303     }
304
305     function calculateMutationRate()
306     {
307         return 1 / (new Individu())->countNumberOfGen();
308     }
309
310     function calculateNumOfMutation()
311     {
312         return round($this->calculateMutationRate() * Parameters::POPULATION_SIZE);
313     }
314
315     function isMutation()
316     {
317         if ($this->calculateNumOfMutation() > 0){
318             return TRUE;
319         }
320     }
321
322     function generateMutation($valueOfGen)
323     {
324         if ($valueOfGen == 0){
325             return 1;
326         }
327     }
328 }
```

```
Explorer (Ctrl+Shift+E) > X
algen_knapsack.php > ...
318         return TRUE;
319     }
320 }
321
322 function generateMutation($valueOfGen)
323 {
324     if ($valueOfGen === 0){
325         return 1;
326     } else {
327         return 0;
328     }
329 }
330
331 function mutation()
332 {
333     if ($this->isMutation()){
334         for ($i = 0; $i <= $this->calculateNumOfMutation() - 1; $i++) {
335             $indexOfIndividu = Randomizer::getRandomIndexofIndividu();
336             $indexOfGen = Randomizer::getRandomIndexofGen();
337             $selectedIndividu = $this->population[$indexOfIndividu];
338
339             //echo 'Before mutation: ';
340             //print_r($selectedIndividu);
341             //echo '<br>';
342             $valueOfGen = $selectedIndividu[$indexOfGen];
343             $mutatedGen = $this->generateMutation($valueOfGen);
344             $selectedIndividu[$indexOfGen] = $mutatedGen;
345             //echo 'After mutation: ';
346             //print_r($selectedIndividu);
347             $ret[] = $selectedIndividu;
348         }
349         return $ret;
350     }
351 }
352 }
353
354 class Selection
355 {
356     function __construct($population, $combinedOffsprings)
357     {
358         $this->population = $population;
359         $this->combinedOffsprings = $combinedOffsprings;
360     }
361
362     function createTemporaryPopulation()
363     {
364         foreach ($this->combinedOffsprings as $offspring){
365             $this->population[] = $offspring;
366         }
367         return $this->population;
368     }
369 }
```



```
algen_knapsack.php X
algen_knapsack.php > ...
354 class Selection
355 {
356     function __construct($population, $combinedOffsprings)
357     {
358         $this->population = $population;
359         $this->combinedOffsprings = $combinedOffsprings;
360     }
361
362     function createTemporaryPopulation()
363     {
364         foreach ($this->combinedOffsprings as $offspring){
365             $this->population[] = $offspring;
366         }
367         return $this->population;
368     }
369
370     function getVariableValue($basePopulation, $fitTemporaryPopulation)
371     {
372         foreach ($fitTemporaryPopulation as $val){
373             $ret[] = $basePopulation[$val[1]];
374         }
375         return $ret;
376     }
377
378     function sortFitTemporaryPopulation()
379     {
380         $tempPopulation = $this->createTemporaryPopulation();
381         $fitness = new Fitness;
382         foreach ($tempPopulation as $key => $individu){
383             $fitnessValue = $fitness->calculateFitnessValue($individu);
384             if ($fitness->isFit($fitnessValue)){
385                 $fitTemporaryPopulation[] = [
386                     $fitnessValue,
387                     $key
388                 ];
389             }
390         }
391         rsort($fitTemporaryPopulation);
392         $fitTemporaryPopulation = array_slice($fitTemporaryPopulation, 0, Parameters::POPULATION_SIZE);
393         return $this->getVariableValue($tempPopulation, $fitTemporaryPopulation);
394     }
395
396     function selectingIndividus()
397     {
398         $selected = $this->sortFitTemporaryPopulation();
399         echo '<p></p>';
400         print_r($selected);
401     }
402
403 }
404
```



Explorer (Ctrl+Shift+E) X



```
algen_knapsack.php > ...
399         echo '<p></p>';
400         print_r($selected);
401     }
402
403 }
404
405 $initialPopulation = new Population;
406 $population = $initialPopulation->createRandomPopulation();
407
408 $fitness = new Fitness;
409 $fitness->fitnessEvaluation($population);
410
411 $crossover = new Crossover($population);
412 $crossoverOffsprings = $crossover->crossover();
413
414 //echo 'Crossover offsprings:<br>';
415 //print_r($crossoverOffsprings);
416
417 echo '<p></p>';
418 //(new Mutation($population))->mutation();
419 $mutation = new Mutation($population);
420 if ($mutation->mutation()){
421     $mutationOffsprings = $mutation->mutation();
422     //echo 'Mutation offspring<br>';
423     //print_r($mutationOffsprings);
424     //echo '<p></p>';
425     foreach ($mutationOffsprings as $mutationOffspring){
426         $crossoverOffsprings[] = $mutationOffspring;
427     }
428 }
429 //echo 'Mutation offsprings <br>';
430 //print_r($crossoverOffsprings);
431 $fitness->fitnessEvaluation($crossoverOffsprings);
432
433 $selection = new Selection($population, $crossoverOffsprings);
434 $selection->selectingIndividus();
435
436
437 // $individu = new Individu;
438 // print_r($individu->createRandomIndividu());
```

← → ↻ ⓘ localhost/teknikOptimasi/pertemuan9/algen_knapsack.php

📱 Apps 🟢 WhatsApp 📦 Stock Barang 📺 (3) YouTube 📄 HTML Tutorial 📄 Bootstrap · The mo... 🎵 AWESOME UPBEAT...

Individu-0

Max. Item: 8 Fitness value: 167890 (Fit)

Individu-1

Max. Item: 12 Fitness value: 317540 (Not Fit)

Individu-2

Max. Item: 9 Fitness value: 233350 (Fit)

Individu-3

Max. Item: 9 Fitness value: 129790 (Fit)

Individu-4

Max. Item: 11 Fitness value: 217980 (Fit)

Individu-5

Max. Item: 11 Fitness value: 147690 (Fit)

Individu-6

Max. Item: 14 Fitness value: 361940 (Not Fit)

Individu-7

Max. Item: 9 Fitness value: 130880 (Fit)

Individu-8

Max. Item: 14 Fitness value: 326690 (Not Fit)

Individu-9

Max. Item: 11 Fitness value: 327920 (Not Fit)

Individu-10

Max. Item: 10 Fitness value: 186770 (Fit)

Individu-11

Max. Item: 14 Fitness value: 365690 (Not Fit)

Individu-12

Max. Item: 9 Fitness value: 228990 (Fit)

Individu-13

Max. Item: 13 Fitness value: 275690 (Fit)

Individu-14

Max. Item: 8 Fitness value: 201690 (Fit)

Individu-15

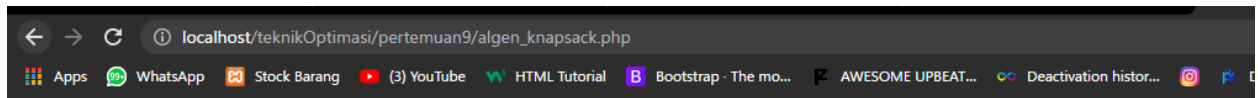
Max. Item: 11 Fitness value: 262640 (Fit)

Individu-16

Max. Item: 12 Fitness value: 247630 (Fit)

Individu-17

Max. Item: 12 Fitness value: 251720 (Fit)



Max. Item: 9 Fitness value: 235950 (Fit)

Individu-26

Max. Item: 12 Fitness value: 218940 (Fit)

Individu-27

Max. Item: 10 Fitness value: 270130 (Fit)

Individu-28

Max. Item: 8 Fitness value: 266130 (Fit)

Individu-29

Max. Item: 9 Fitness value: 176140 (Fit)

Best fitness value: 275690 Residual: 4310 Found

Individu-0

Max. Item: 10 Fitness value: 157640 (Fit)

Individu-1

Max. Item: 11 Fitness value: 289690 (Not Fit)

Individu-2

Max. Item: 9 Fitness value: 154640 (Fit)

Individu-3

Max. Item: 14 Fitness value: 380880 (Not Fit)

Individu-4

Max. Item: 10 Fitness value: 157640 (Fit)

Individu-5

Max. Item: 13 Fitness value: 307590 (Not Fit)

Individu-6

Max. Item: 13 Fitness value: 332540 (Not Fit)

Individu-7

Max. Item: 13 Fitness value: 346940 (Not Fit)

```
localhost/teknikOptimasi/pertemuan9/algen_knapsack.php
Apps WhatsApp Stock Barang (3) YouTube HTML Tutorial Bootstrap - The mo...
Individu-668
Max. Item: 10 Fitness value: 258130 (Fit)

Individu-669
Max. Item: 12 Fitness value: 329540 (Not Fit)

Individu-670
Max. Item: 10 Fitness value: 226230 (Fit)

Individu-671
Max. Item: 9 Fitness value: 277250 (Fit)

Individu-672
Max. Item: 8 Fitness value: 200030 (Fit)

Individu-673
Max. Item: 8 Fitness value: 233990 (Fit)

Individu-674
Max. Item: 9 Fitness value: 313030 (Not Fit)

Individu-675
Max. Item: 11 Fitness value: 270640 (Fit)

Individu-676
Max. Item: 9 Fitness value: 208040 (Fit)

Individu-677
Max. Item: 8 Fitness value: 135990 (Fit)

Individu-678
Max. Item: 9 Fitness value: 278030 (Fit)

62 267880
71 267880
83 267880
85 279880
92 235980
96 235980
101 275690
117 275690
158 267880
164 201690
184 218430
189 201690
194 201690
197 267790
199 267790
200 201690
201 213690
263 267790
274 275690
280 275690
320 267790
332 279880
338 213690
```