```
<?PHP
class Dijkstra {

	var $visited = array();
	var $distance = array();
	var $previousNode = array();
	var $startnode =null;
	var $map = array();
	var $infiniteDistance = 0;
	var $bestPath = 0;
	var $matrixWidth = 0;

	function Dijkstra(&$ourMap, $infiniteDistance) {
		$this -> infiniteDistance = $infiniteDistance;
		$this -> map = &$ourMap;
		$this -> bestPath = 0;
	}

	function findShortestPath($start,$to = null) {
		$this -> startnode = $start;
		foreach (array_keys($this->map) as $i) {
			if ($i == $this -> startnode) {
				$this -> visited[$i] = true;
				$this -> distance[$i] = 0;
			} else {
				$this -> visited[$i] = false;
				$this -> distance[$i] = isset($this -> map[$this ->
startnode][$i])
					? $this -> map[$this -> startnode][$i]
					: $this -> infiniteDistance;
			}
			$this -> previousNode[$i] = $this -> startnode;
		}

		$maxTries = count($this->map);
		for ($tries = 0; in_array(false,$this -> visited,true) && $tries <=
$maxTries; $tries++) {
			$this -> bestPath = $this->findBestPath($this-
>distance,array_keys($this -> visited,false,true));
			if($to !== null && $this -> bestPath === $to) {
				break;
			}
			$this -> updateDistanceAndPrevious($this -> bestPath);

			$this -> visited[$this -> bestPath] = true;
		}
	}

	function findBestPath($ourDistance, $ourNodesLeft) {
		$bestPath = $this -> infiniteDistance;
		$bestNode = 0;
		foreach ($ourNodesLeft as $node) {
			if($ourDistance[$node] < $bestPath) {
				$bestPath = $ourDistance[$node];
				$bestNode = $node;
			}
		}
```

```
                return $bestNode;
        }

        function updateDistanceAndPrevious($obp) {
                foreach (array_keys($this->map) as $i) {
                        if(     isset($this->map[$obp][$i])
                           &&  ($this->map[$obp][$i] != $this->infiniteDistance || $this-
>map[$obp][$i] == 0 )
                                &&      ($this->distance[$obp] + $this->map[$obp][$i] <
$this -> distance[$i])
                        )
                        {
                                $this -> distance[$i] = $this -> distance[$obp] +
$this -> map[$obp][$i];
                                $this -> previousNode[$i] = $obp;
                        }
                }
        }

        function printMap(&$map) {
                $placeholder = ' %' . strlen($this -> infiniteDistance) .'d';
                $foo = '';
                for($i=0,$im=count($map);$i<$im;$i++) {
                        for ($k=0,$m=$im;$k<$m;$k++) {
                                $foo.= sprintf($placeholder, isset($map[$i][$k]) ?
$map[$i][$k] : $this -> infiniteDistance);
                        }
                        $foo.= "\n";
                }
                return $foo;
        }

        function getResults($to = null) {
                $ourShortestPath = array();
                $foo = '';
                foreach (array_keys($this->map) as $i) {
                        if($to !== null && $to !== $i) {
                                continue;
                        }
                        $ourShortestPath[$i] = array();
                        $endNode = null;
                        $currNode = $i;
                        $ourShortestPath[$i][] = $i;
                        while ($endNode === null || $endNode != $this -> startnode) {
                                $ourShortestPath[$i][] = $this -> previousNode[$currNode];
                                $endNode = $this -> previousNode[$currNode];
                                $currNode = $this -> previousNode[$currNode];
                        }
                        $ourShortestPath[$i] = array_reverse($ourShortestPath[$i]);
                        if ($to === null || $to === $i) {
                        if($this -> distance[$i] >= $this -> infiniteDistance) {
                                $foo .= sprintf("no route from %d to %d. \n",$this ->
startnode,$i);
                        } else {
                                $foo .= sprintf('%d => %d = %d [%d]: (%s).'"\n" ,
                                                $this -> startnode,$i,$this -> distance[$i],
                                                count($ourShortestPath[$i]),
```

```
                                        implode('-',$ourShortestPath[$i]));
                        }
                        $foo .= str_repeat('-',20) . "\n";
                                if ($to === $i) {
                                        break;
                                }
                        }
                }
                return $foo;
        }
} // end class
?>
<?php
$pt=$_POST['pt'];
echo "$pt<br>";
// I is the infinite distance.
define('I',1000);

// Size of the matrix
$matrixWidth = 20;

// $points is an array in the following format: (router1,router2,distance-between-them)
$points = array(
        array(0,1,4),
        array(0,2,I),
        array(1,2,10),
        array(1,3,15),
        array(1,4,25),
        array(3,4,15),
        array(4,5,5),
        array(4,7,35),
        array(5,16,30),
        array(6,16,40),
        array(4,19,20),
        array(10,11,20),
        array(12,13,20),
);

$ourMap = array();


// Read in the points and push them into the map

foreach ($points as $point) {
        $x = $point[0];
        $y = $point[1];
        $c = $point[2];
        $ourMap[$x][$y] = $c;
        $ourMap[$y][$x] = $c;
}

// ensure that the distance from a node to itself is always zero
// Purists may want to edit this bit out.

for ($i=0; $i < $matrixWidth; $i++) {
   $ourMap[$i][$i] = 0;
}
```

```
// initialize the algorithm class
$dijkstra = new Dijkstra($ourMap, I,$matrixWidth);

// $dijkstra->findShortestPath(0,13); to find only path from field 0 to field 13...
$dijkstra->findShortestPath(0);

// Display the results
$pt=$_POST['pt'];
echo $dijkstra->getResults(16);
/*
echo '<pre>';
echo "the map looks like:\n\n";
echo $dijkstra -> printMap($ourMap);
echo "\n\nthe shortest paths from point 0:\n";
echo $dijkstra -> getResults();
echo '</pre>';
*/
?>
<form action="dijkstra.php" method="POST">
        vul het punt in: <input type=text name='pt'><br>
        <input type=submit value=stuur>
</form>
```