Résolution de parcours dans un graph avec l'algorithme de Dijkstra.

Initialisation

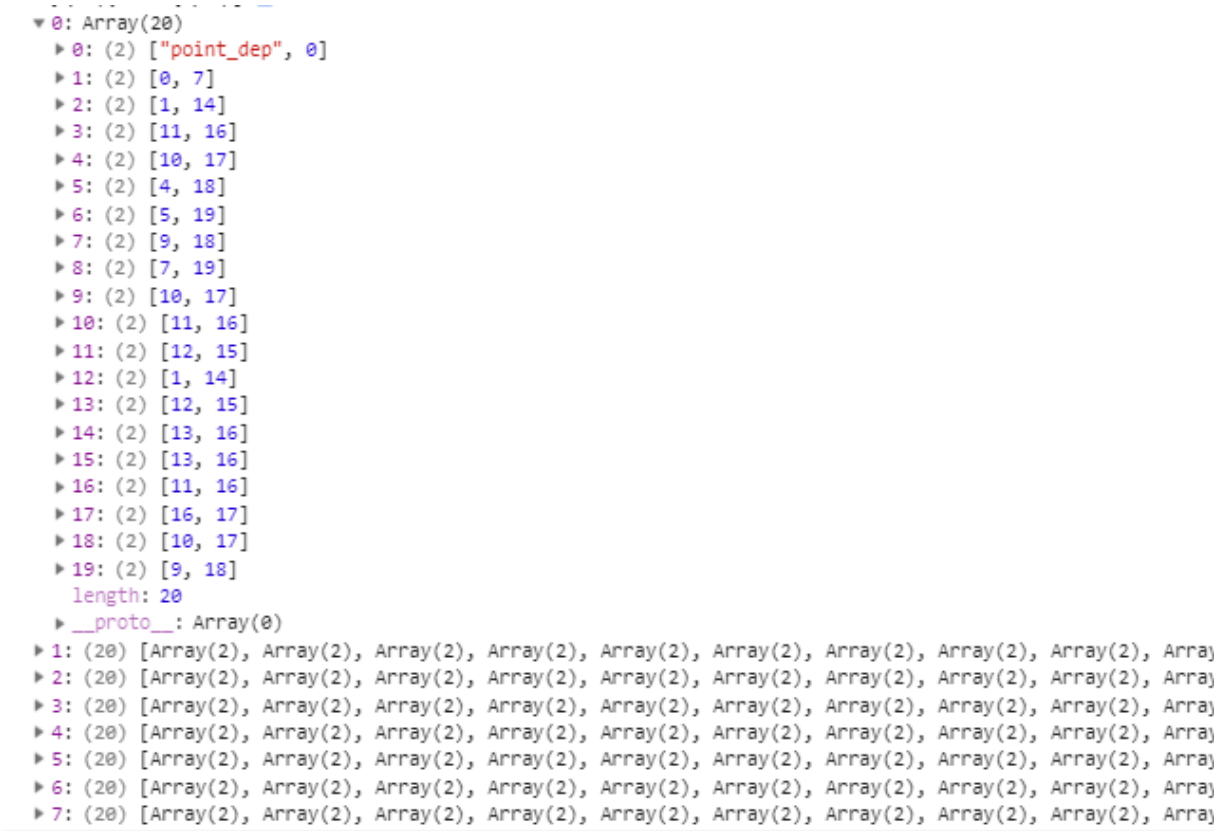
```
var graph = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19];//creation des points
var graph_vec = new Array();//creation des vecteurs Point1,Point2,coup

graph_vec.push([0,1,7]);
graph_vec.push([1,2,7]);
graph_vec.push([1,12,7]);
graph_vec.push([2,3,7]);
graph_vec.push([3,4,7]);
graph_vec.push([3,11,1]);
graph_vec.push([4,5,1]);
graph_vec.push([4,10,1]);
graph_vec.push([5,6,1]);
graph_vec.push([5,7,1]);
graph_vec.push([7,8,1]);
graph_vec.push([7,9,1]);
graph_vec.push([9,10,1]);
graph_vec.push([9,19,1]);
graph_vec.push([10,11,1]);
graph_vec.push([10,18,1]);
graph_vec.push([11,12,1]);
graph_vec.push([11,16,1]);
graph_vec.push([12,13,1]);
graph_vec.push([13,14,1]);
graph_vec.push([13,15,1]);
graph_vec.push([15,16,1]);
graph_vec.push([16,17,1]);
graph_vec.push([16,18,1]);
graph_vec.push([17,18,1]);
graph_vec.push([18,19,1]);
```

Dans le programme, on initialise un tableau « graph » qui contient les points du graph, puis on initialise un autre tableau qui lui contient les arrêtes du graph avec le point1, le point2 et le coup de l'arrête.

```
0: (20) [null, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
1: (20) [7, null, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, 7, -1, -1, -1, -1, -1, -1, -1]
2: (20) [-1, 7, null, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
3: (20) [-1, -1, 7, null, 7, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1]
4: (20) [-1, -1, -1, 7, null, 1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
5: (20) [-1, -1, -1, -1, 1, null, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
6: (20) [-1, -1, -1, -1, -1, 1, null, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
7: (20) [-1, -1, -1, -1, -1, 1, -1, null, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
8: (20) [-1, -1, -1, -1, -1, -1, -1, 1, null, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
9: (20) [-1, -1, -1, -1, -1, -1, -1, 1, -1, null, 1, -1, -1, -1, -1, -1, -1, -1, -1, 1]
10: (20) [-1, -1, -1, -1, 1, -1, -1, -1, -1, 1, null, 1, -1, -1, -1, -1, -1, -1, 1, -1]
11: (20) [-1, -1, -1, 1, -1, -1, -1, -1, -1, -1, 1, null, 1, -1, -1, -1, 1, -1, -1, -1]
12: (20) [-1, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, null, 1, -1, -1, -1, -1, -1, -1]
13: (20) [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, null, 1, 1, -1, -1, -1, -1]
14: (20) [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, null, -1, -1, -1, -1, -1]
15: (20) [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, null, 1, -1, -1, -1]
16: (20) [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, 1, null, 1, 1, -1]
17: (20) [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, null, 1, -1]
18: (20) [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, 1, 1, null, 1]
19: (20) [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1, 1, null]
```

On obtient dans la console une matrice avec chacune des arrêtes (-1 valeur si il n'y a pas d'arrête).



En résultat on obtient un tableau pour chaque point. Dans ce tableau pour chaque point on a le point d'avant par lequel le chemin le plus court passe et le coups entre le point initiale et le point d'arriver.