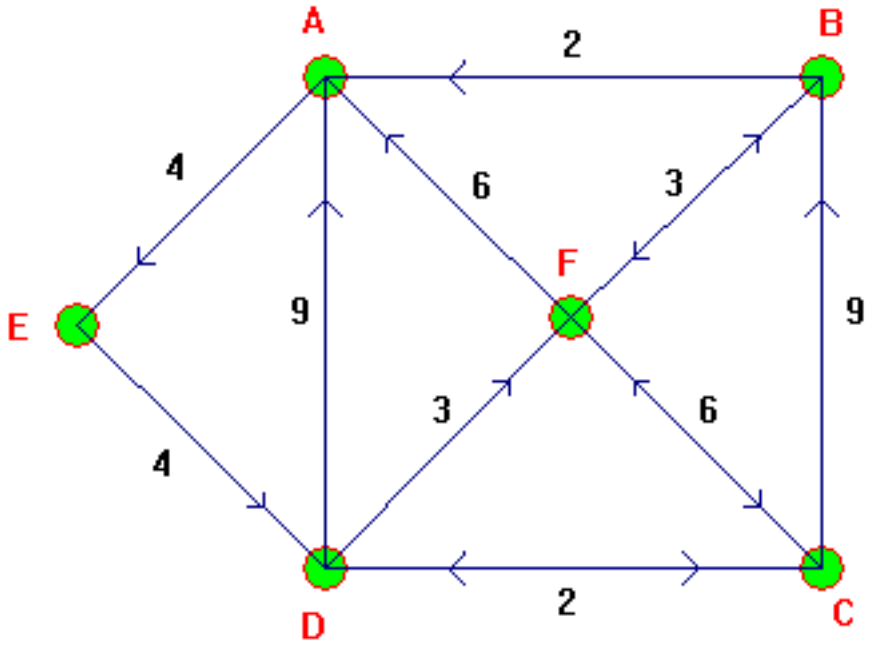
L'algorithme du plus court chemin de Dijkstra

L'algorithme de Dijkstra (mathématicien et informaticien néerlandais du XXème siècle) est utilisé notamment par les GPS pour déterminer **le plus court chemin** pour se rendre d'une ville à une autre connaissant le réseau routier d'une région.

I. Exemples

Exemple 1 : On dispose du graphe ci-dessous, orienté et pondéré (en km, en temps, en consommation d'essence…).
En prenant ici la distance comme poids du graphe, on se propose de rechercher le plus court chemin pour aller de C à A.



On réalise pour cela un tableau avec autant de colonnes que de sommets, une colonne où seront indiqués les sommets fixés, une colonne pour la retenue et une colonne pour la provenance.

C	B	F	D	E	A	Fixés	Retenues	Provenances

Partant du point de départ C, le premier point fixé est le point C (on ferme la colonne avec des ∞ dans celle-ci) ; on note la distance du point C à tous les sommets adjacents à celui-ci, sinon on note 100.

C	B	F	D	E	A	Fixés	Retenues	Provenances
100C	9C	6C	2C	100C	100C	C	0	C

On repère le sommet de plus faible poids, ici D, qui donne la retenue de 2, le sommet D sera le prochain sommet fixé. Pour les sommets adjacents à D, on ajoute la retenue et on écrit le nouveau

résultat s'il est strictement plus petit que le précédent ou sinon on laisse le précédent, sinon si le sommet n'est pas adjacent on écrit la distance précédente si elle existe, sinon ∞.

C	B	F	D	E	A	Fixés	Retenues	Provenances
100C	9C	6C	2C	100C	100C	C	0	C
∞	9C	5D	∞	100C	11D	D	2	C

On continue le procédé :

C	B	F	D	E	A	Fixés	Retenues	Provenances
100C	9C	6C	2C	100C	100C	C	0	C
∞	9C	5D	∞	100C	11D	D	2	C
∞	8F	∞	∞	100C	11D	F	5	D

Une fois de plus :

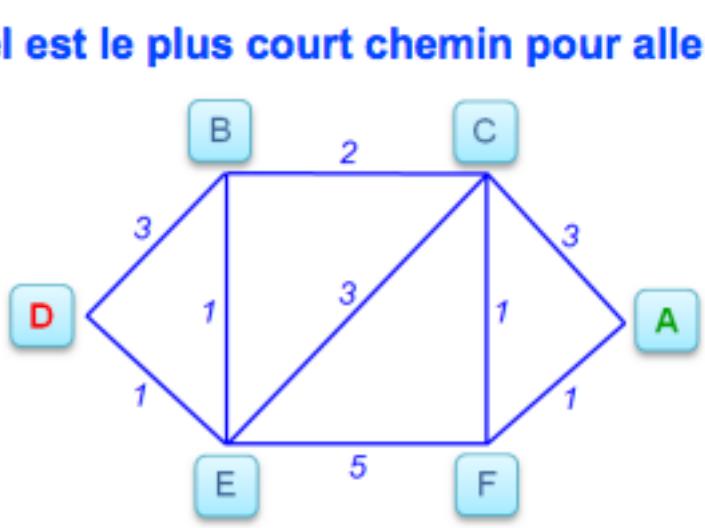
C	B	F	D	E	A	Fixés	Retenues	Provenances
100C	9C	6C	2C	100C	100C	C	0	C
∞	9C	5D	∞	100C	11D	D	2	C
∞	8F	∞	∞	100C	11D	F	5	D
∞	∞	∞	∞	100C	10B	B	8	F

On s'arrête car on rencontre le point d'arrivée A

C	B	F	D	E	A	Fixés	Retenues	Provenances
100C	9C	6C	2C	100C	100C	C	0	C
∞	9C	5D	∞	100C	11D	D	2	C
∞	8F	∞	∞	100C	11D	F	5	D
∞	∞	∞	∞	100C	10B	B	8	F
∞	∞	∞	∞	14A	∞	A	10	B

On part du point d'arrivée A et on remonte en passant par les provenances jusqu'au point de départ.
A provient de B, qui provient de F, qui provient de D, qui provient de C.
Le chemin le plus court est le chemin C → D → F → B → A de distance totale 2+3+3+2=10.
N.B : On pourrait croire ici que le chemin le plus court est donné par les points fixés, il n'en est rien.

Exemple 2 :



Travail n°1 : Montrer, en réalisant le tableau sur une feuille, qu'il s'agit de D → E → C → F → A avec une distance totale de 6.

II. Recherche de l'algorithme

Nous allons d'abord synthétiser les informations du graphe à l'aide d'une matrice.

		0	1	2	3	4	5
		D	B	E	C	F	A
0	D	0	3	1	0	0	0
1	B	3	0	1	2	0	0
2	E	1	1	0	3	5	0
3	C	0	2	3	0	1	3
4	F	0	0	5	1	0	1
5	A	0	0	0	3	1	0

On note dans cette matrice la distance au point adjacent si celle-ci est non nulle sinon on note 0.

On est amené à créer un tableau dont les lignes sont des listes de 8 valeurs.

D	B	E	C	F	A	Fixés	Retenues

On initialise avec comme sommet fixé le point de départ D. La retenue est nulle.

Pour chaque nouvelle ligne écrite, on écrit la distance de chaque sommet au nœud fixé à laquelle on ajoute la retenue, on garde la valeur de la ligne précédente si celle-ci est plus petite.
On recherche le sommet qui a la plus petite valeur non nulle de distance au sommet fixé, ce sommet constitue le sommet fixé pour la ligne suivante; on s'arrête lorsque le nœud fixé est le point d'arrivée A.

III. Ecriture de l'algorithme de Dijkstra en JavaScript (utilisation de la console)

- 1ière étape : savoir écrire la matrice et savoir récupérer une valeur de celle-ci.

```
<html>
<head>
<meta charset="utf-8" />
<title>Algorithme de Dijkstra</title>
</head>
<body>
<script type="text/javascript">
var matrice=[
[0,3,1,0,0,0],
[3,0,1,2,0,0],
[1,1,0,3,5,0],
[0,2,3,0,1,3],
[0,0,5,1,0,1],
[0,0,0,3,1,0]
]
console.log(matrice[3][1]);
</script>
</body>
</html>
```

- 2ième étape : savoir créer des lignes comme des listes et savoir faire référence à une valeur de la liste.

```
<html>
<head>
<meta charset="utf-8" />
<title>Algorithme de Dijkstra</title>
</head>
<body>

<script type="text/javascript">
var ligne=[];
for (i=1;i<=5;i++) {

ligne.push([0,1,2,3]);

}
console.log(ligne);
console.log(ligne[2]);
console.log(ligne[2][3]);

</script>
</body>
</html>
```

- 3ième étape : écrire la première ligne du tableau de l'algorithme pour l'initialiser.

```
<html>
<head>
<meta charset="utf-8" />
<title>Algorithme de Dijkstra</title>
</head>
<body>
<script type="text/javascript">
var matrice=[
[0,3,1,0,0,0],
[3,0,1,2,0,0],
[1,1,0,3,5,0],
[0,2,3,0,1,3],
[0,0,5,1,0,1],
[0,0,0,3,1,0]
];

var ligne=[];
ligne.push([0,0,0,0,0,0,"D",0]);

for (i=0;i<=5;i++) {

ligne[0][i]=matrice[0][i];
}
console.log(ligne[0]);

</script>
</body>
</html>
```

- 4ième étape : écriture de la deuxième ligne du tableau.

→ Pour faciliter la recherche du minimum, nous allons placer 100 à la place de 0 dans la matrice.
→ Comprenez bien le rôle de **liste.IndexOf('valeur')** qui détermine l'indice de position de l'élément 'valeur' à l'intérieur de la liste.

```
<html>
<head>
<meta charset="utf-8" />
<title>Algorithme de Dijkstra</title>
</head>
<body>

<script type="text/javascript">
var matrice=[
[100,3,1,100,100,100],
[3,100,1,2,100,100],
[1,1,100,3,5,100],
[100,2,3,100,1,3],
[100,100,5,1,100,1],
[100,100,100,3,1,100]
];

var ligne=[];

ligne.push([100,0,0,0,0,0,"D",0]);

for (i=0;i<=5;i++) {

ligne[0][i]=matrice[0][i];
}
console.log(ligne[0]);

var liste=["D","B","E","C","F","A"];
var etat=["sel","non","non","non","non","non"];

var retenue=Math.min(ligne[0][0],ligne[0][1],ligne[0][2],ligne[0][3],ligne[0][4],ligne[0][5]);

var position=ligne[0].indexOf(retenue);
var temp=[100,100,100,0,0,0,"D",0];

etat[position]="sel";

for (i=0;i<=5;i++) {
temp[i]=matrice[position][i]+retenue;
if (etat[i]=="sel") {
temp[i]=100;
}
}
temp[7]=retenue;
temp[6]=liste[position];
console.log(temp);
</script>
</body>
</html>
```

- 5ième étape : utilisation d'une boucle for pour écrire toutes les lignes du tableau en sachant s'arrêter avec un break.

```
<html>
<head>
<meta charset="utf-8" />
<title>Algorithme de Dijkstra</title>
</head>
<body>
<script type="text/javascript">
var matrice=[
[100,3,1,100,100,100],
[3,100,1,2,100,100],
[1,1,100,3,5,100],
[100,2,3,100,1,3],
[100,100,5,1,100,1],
[100,100,100,3,1,100]
];
var chemin="";
var ligne=[];
ligne.push([100,0,0,0,0,0,"D",0]);
for (i=0;i<=5;i++) {
ligne[0][i]=matrice[0][i];
}
console.log(ligne[0]);
var liste=["D","B","E","C","F","A"];
var etat=["sel","non","non","non","non","non"];
for (k=0;k<=20;k++) {
var retenue=Math.min(ligne[k][0],ligne[k][1],ligne[k][2],ligne[k][3],ligne[k][4],ligne[k][5]);
var position=ligne[k].indexOf(retenue);
var temp=[100,100,100,0,0,0,"D",0];
etat[position]="sel";
for (i=0;i<=5;i++) {
temp[i]=matrice[position][i]+retenue;
if (etat[i]=="sel") {
temp[i]=100;
}
}
temp[7]=retenue;
temp[6]=liste[position];
ligne.push(temp);
chemin+="_"+ligne[k][6]+"_";
if (temp[6]=="A") {
console.log(k);
break;
}
}
console.log(ligne);
console.log("Le chemin le plus court est : "+chemin+"_A");
</script>
</body>
</html>
```

- 6ième étape : Amélioration du code avec une boucle while.

```
<html>
<head>
<meta charset="utf-8" />
<title>Algorithme de Dijkstra</title>
</head>
<body>
<script type="text/javascript">
var matrice=[
[100,3,1,100,100,100],
[3,100,1,2,100,100],
[1,1,100,3,5,100],
[100,2,3,100,1,3],
[100,100,5,1,100,1],
[100,100,100,3,1,100]
];
var chemin="";
var ligne=[];
var k=0;
var temp=[];
ligne.push([100,0,0,0,0,0,"D",0]);
for (i=0;i<=5;i++) {
ligne[0][i]=matrice[0][i];
}
console.log(ligne[0]);
var liste=["D","B","E","C","F","A"];
var etat=["sel","non","non","non","non","non"];
while (temp[6]!="A") {
var retenue=Math.min(ligne[k][0],ligne[k][1],ligne[k][2],ligne[k][3],ligne[k][4],ligne[k][5]);-
var position=ligne[k].indexOf(retenue);
var temp=[100,100,100,0,0,0,"D",0];
etat[position]="sel";
for (i=0;i<=5;i++) {
temp[i]=matrice[position][i]+retenue;
if (etat[i]=="sel") {
temp[i]=100;
}
}
temp[7]=retenue;
temp[6]=liste[position];
ligne.push(temp);
chemin+="_"+ligne[k][6]+"_";
k++;
}
console.log(ligne);
console.log("Le chemin le plus court est : "+chemin+"_A");
</script>
</body>
</html>
```

- 7ième étape : Ajout d'une condition pour respecter l'algorithme.

```
<html>
<head>
<meta charset="utf-8" />
<title>Algorithme de Dijkstra</title>
</head>
<body>
<script type="text/javascript">
var matrice=[
[100,3,1,100,100,100],
[3,100,1,2,100,100],
[1,1,100,3,5,100],
[100,2,3,100,1,3],
[100,100,5,1,100,1],
[100,100,100,3,1,100]
];
var chemin="";
var ligne=[];
var k=0;
var temp=[];
ligne.push([100,0,0,0,0,0,"D",0]);
for (i=0;i<=5;i++) {
ligne[0][i]=matrice[0][i];
}
console.log(ligne[0]);
var liste=["D","B","E","C","F","A"];
var etat=["sel","non","non","non","non","non"];
while (temp[6]!="A") {
var retenue=Math.min(ligne[k][0],ligne[k][1],ligne[k][2],ligne[k][3],ligne[k][4],ligne[k][5]);
var position=ligne[k].indexOf(retenue);
var temp=[100,100,100,0,0,0,"D",0];
etat[position]="sel";
for (i=0;i<=5;i++) {
var valeur=matrice[position][i]+retenue;
temp[i]=valeur;
if (k>=1) {
if (valeur>ligne[k][i] && valeur<100) {temp[i]=ligne[k][i]};
}
if (etat[i]=="sel") {
temp[i]=100;
}
}
temp[7]=retenue;
temp[6]=liste[position];
ligne.push(temp);
chemin+="_"+ligne[k][6]+"_";
k++;
}
console.log(ligne);
console.log("Le chemin le plus court est : "+chemin+"_A");
</script>
</body>
</html>
```

- 8ème étape : Présentation du tableau dans la page html, on commence par voir comment on fait un tableau statique.

```
<html>
<head>
<meta charset="utf-8" />
<title>Algorithme de Dijkstra</title>
<table align=center width 100% cellpadding=2 cellspacing=2 border=4 bordercolor="0000FF">
<tr>
<td align=center>D</td>
<td align=center>B</td>
<td align=center>E</td>
<td align=center>C</td>
<td align=center>F</td>
<td align=center>A</td>
<td align=center>Fixés</td>
<td align=center>Retenues</td>
<tr>
<tr>
<td align=center>100</td>
<td align=center>3</td>
<td align=center>1</td>
<td align=center>100</td>
<td align=center>100</td>
<td align=center>100</td>
<td align=center>D</td>
<td align=center>0</td>
<tr>
</table>
</head>
<body>
```

- 9ième étape : Présentation du tableau de façon dynamique dans la page html. On insère dans le DOM de la page html (Document Object Model), pour chaque nouvelle ligne, un élément TR et ses éléments TD fils qui sont les cellules du tableau, les éléments TR sont eux-mêmes des fils du tableau.

```
<html>
<head>
<meta charset="utf-8" />
<title>Algorithme de Dijkstra</title>
<table id="tableau" align=center width 100% cellpadding=2 cellspacing=2 border=4
bordercolor="0000FF">
<tr>
<td align=center>D</td>
<td align=center>B</td>
<td align=center>E</td>
<td align=center>C</td>
<td align=center>F</td>
<td align=center>A</td>
<td align=center>Fixés</td>
<td align=center>Retenues</td>
<tr>
</table>
</head>
```

```
<body>
<script type="text/javascript">
var tableau=document.getElementById('tableau');
var matrice=[
[100,3,1,100,100,100],
[3,100,1,2,100,100],
[1,1,100,3,5,100],
[100,2,3,100,1,3],
[100,100,5,1,100,1],
[100,100,100,3,1,100]
];
var chemin="";
var ligne=[];
var k=0;
var temp=[];
ligne.push([100,0,0,0,0,0,"D",0]);
for (i=0;i<=5;i++) {
ligne[0][i]=matrice[0][i];
}
var liste=["D","B","E","C","F","A"];
var etat=["sel","non","non","non","non","non"];
while (temp[6]!="A") {
var retenue=Math.min(ligne[k][0],ligne[k][1],ligne[k][2],ligne[k][3],ligne[k][4],ligne[k][5]);
var position=ligne[k].indexOf(retenue);
var temp=[100,100,100,0,0,0,"D",0];
etat[position]="sel";
for (i=0;i<=5;i++) {
var valeur=matrice[position][i]+retenue;
temp[i]=valeur;
if (k>=1) {
if (valeur>ligne[k][i] && valeur<100) {
temp[i]=ligne[k][i];
};
}
if (etat[i]=="sel") {
temp[i]=100;
}
}
temp[7]=retenue;
temp[6]=liste[position];
ligne.push(temp);
k++;
}
for (i=0;i<=k;i++) {
var tr = document.createElement("TR");
for (j=0;j<=7;j++) {
var td = document.createElement("TD");
td.setAttribute("align","center");
td.innerHTML = ligne[i][j];
tr.appendChild(td);
}
tableau.appendChild(tr);
}
</script>
</body>
</html>
```

- 10ième étape : Le chemin le plus court n'est pas donné par la suite des sommets fixés mais il faut passer par les provenances des sommets en partant du point d'arrivée.

```
<html>
<head>
<meta charset="utf-8" />
<title>Algorithme de Dijkstra</title>
<table id="tableau" align=center width 100% cellpadding=2 cellspacing=2 border=4
bordercolor="0000FF">
<tr>
<td align=center>D</td>
<td align=center>B</td>
<td align=center>E</td>
<td align=center>C</td>
<td align=center>F</td>
<td align=center>A</td>
<td align=center>Fixés</td>
<td align=center>Retenues</td>
<td align=center>Provenances</td>
<tr>
</table>
<br>
<div style="text-align: center;">
<span  id="chemin"></span>
</div>
</head>
<body>
<script type="text/javascript">

var k=0;
var distance=0;
var total=0;
var parcours="";
var matrice=[
[100,3,1,100,100,100],
[3,100,1,2,100,100],
[1,1,100,3,5,100],
[100,2,3,100,1,3],
[100,100,5,1,100,1],
[100,100,100,3,1,100]
];

var liste=["D","B","E","C","F","A"];
var etat=["sel","non","non","non","non","non"];

var liste_fixe=[];
var liste_provenance=[];

function num(texte) {
        var x=texte.indexOf("/");
        return Number(texte.slice(0,x));
}

function lit(texte) {

        var x=texte.indexOf("/");
```

```
        return texte.slice(x+1,texte.length);
}

var ligne=[];
var temp=[];
ligne.push(["","","","","","","D",0,"D"]);
for (i=0;i<=5;i++) {
ligne[0][i]=matrice[0][i]+"/D";
}

while (temp[8]!="A") {
var
retenue=Math.min(num(ligne[k][0]),num(ligne[k][1]),num(ligne[k][2]),num(ligne[k][3]),num(ligne[k][4]),n
um(ligne[k][5]));
var
suite=[num(ligne[k][0]),num(ligne[k][1]),num(ligne[k][2]),num(ligne[k][3]),num(ligne[k][4]),num(ligne[k][
5])];
var position=suite.indexOf(retenue);
var temp=["","","","","","","",0,"",""];
etat[position]="sel";
for (i=0;i<=5;i++) {
var valeur=matrice[position][i]+retenue;
temp[i]=valeur+"/"+liste[position];
if (k>=0) {
if (valeur>=num(ligne[k][i]) && valeur<10000) {
temp[i]=ligne[k][i];
};
}
if (etat[i]=="sel") {
temp[i]=10000+"/"+liste[position];
}
}

temp[7]=retenue;
temp[6]=liste[position];
temp[8]=lit(ligne[k][position]);

ligne.push(temp);

k++;
}

for (i=1;i<=k-1;i++){
liste_fixe.push(ligne[i][6]);
liste_provenance.push(ligne[i][8]);
}
console.log(liste_fixe);
console.log(liste_provenance);

var chemin_inverse=[];
var pas=liste_fixe[liste_fixe.length-1];

chemin_inverse.push(pas);

for (i=1;i<=liste_fixe.length+1;i++) {
```

```
if (pas==liste_fixe[0]) {

pas=liste_provenance[liste_fixe.indexOf(pas)];

if (chemin_inverse.indexOf(pas) ==-1) {
chemin_inverse.push(pas);
}
        break;
}

pas=liste_provenance[liste_fixe.indexOf(pas)];
chemin_inverse.push(pas);
}

var chemin=chemin_inverse.reverse();
console.log(chemin);

for (i=0;i<=k-1;i++) {
var tr = document.createElement("TR");
for (j=0;j<=8;j++) {
var td = document.createElement("TD");
td.setAttribute("align","center");
td.innerHTML = ligne[i][j];
tr.appendChild(td);
}
tableau.appendChild(tr);
}

for (i=0;i<=chemin.length-2;i++) {
distance=matrice[liste.indexOf(chemin[i])][liste.indexOf(chemin[i+1])];
parcours+=chemin[i]+"--"+distance+"-->";
total+=distance;
}

document.getElementById("chemin").innerHTML = "Le chemin le plus court est :
"+parcours+chemin[chemin.length-1]+" avec une distance totale de "+total;

</script>
</body>
</html>
```

Le programme est ici : http://isnangellier.alwaysdata.net/php/algosolution.html

Remarque : Le programme permet d'avoir la liste des sommets fixés : ["E", "B", "C", "F", "A"]
Ainsi que la liste des provenances associées à ces sommets: ["D", "E", "E", "C", "F"]

Pour trouver le chemin le plus court de D à A, on part du point d'arrivée qui est le point A, qui provient du point F, lequel provient du point C, lequel provient du point E, lequel provient du point D.

On trouve, après inversion, la chaîne donnant le chemin le plus court : ["D", "E", "C", "F", "A"]

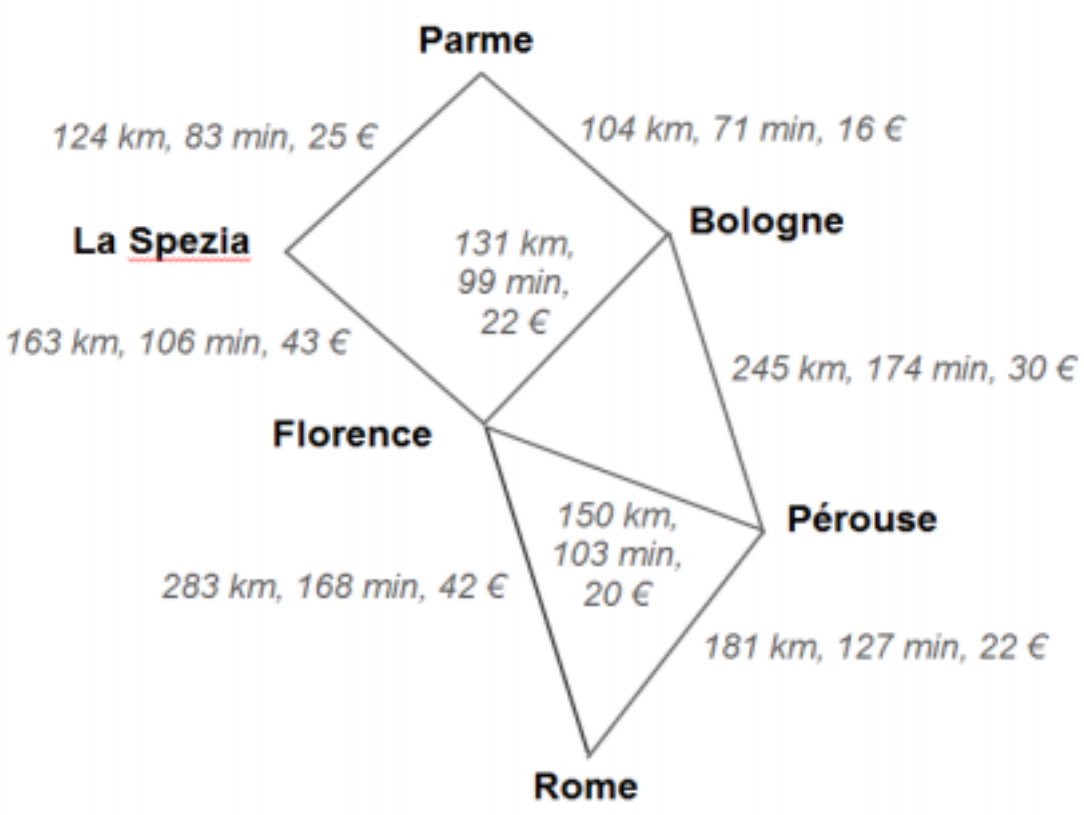
On observera la ligne de code particulière employée : `pas=liste_provenance[liste_fixe.indexOf(pas)];`

Cette ligne utilise une méthode dite **récursive** qui permet d'obtenir le résultat.

IV. Applications

Travail n°2 : Utiliser le programme précédent pour répondre à l'une des questions suivantes :

"Tous les chemins mènent à Rome" dit-on mais pour un Parmesan (un habitant de Parme), quel est le chemin le plus court ? Le plus rapide ? Le moins cher ?



Travail n°3 : Utiliser le programme précédent pour retrouver le résultat de l'exemple 1.

Travail n°4 : Adapter le programme pour déterminer le chemin le plus court du point d'entrée E au point de sortie S.