INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

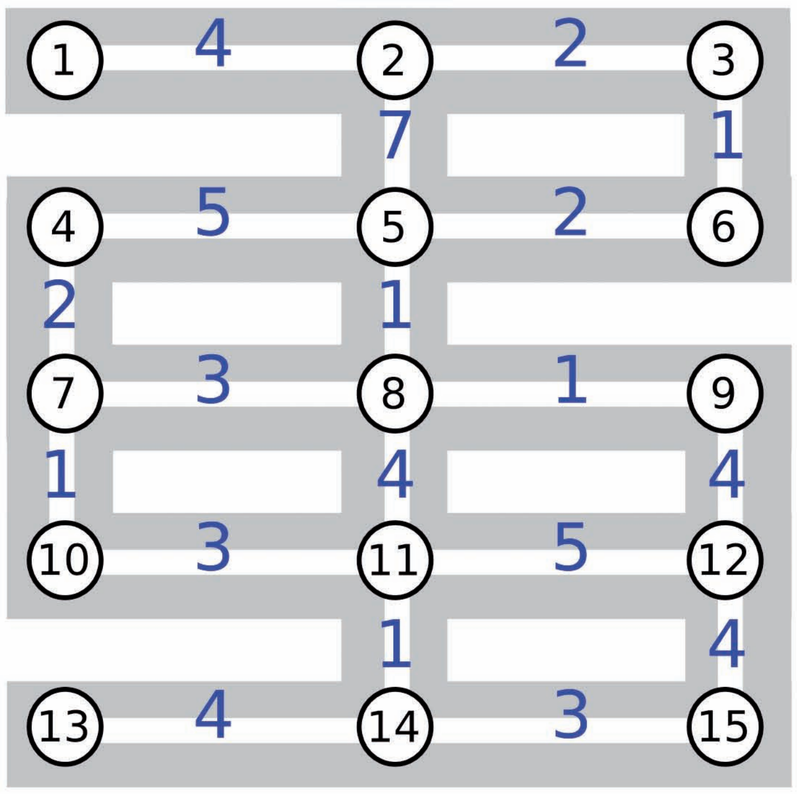
INRIA

Programmer un système de navigation ?

Représentation des **données** du problème

La carte

```
1   [ (. , .); (. , .); (2 , 4); (. , .) ]
2   [ (1 , 4); (. , .); (3 , 2); (5 , 7) ]
3   [ (2 , 2); (. , .); (. , .); (6 , 1) ]
4   [ (. , .); (. , .); (5 , 5); (7 , 2) ]
5   [ (4 , 5); (2 , 7); (6 , 2); (8 , 1) ]
6   [ (5 , 2); (3 , 1); (. , .); (. , .) ]
7   [ (. , .); (4 , 2); (8 , 3); (10, 1) ]
8   [ (7 , 3); (5 , 1); (9 , 1); (11, 4) ]
9   [ (8 , 1); (. , .); (. , .); (12, 4) ]
10  [ (. , .); (7 , 1); (11, 3); (. , .) ]
11  [ (10, 3); (8 , 4); (12, 5); (14, 1) ]
12  [ (11, 5); (9 , 4); (15, 4); (. , .) ]
13  [ (. , .); (. , .); (14, 4); (. , .) ]
14  [ (13, 4); (11, 1); (15, 3); (. , .) ]
15  [ (14, 3); (12, 4); (. , .); (. , .) ]
```



L'itinéraire

Un itinéraire de 1 à 9 :
[1; 2; 5; 8; 9]

Un autre itinéraire de 1 à 9 :
[1; 2; 3; 6; 5; 8; 9]

Un itinéraire de 5 à 15 :
[5; 8; 9; 12; 15]

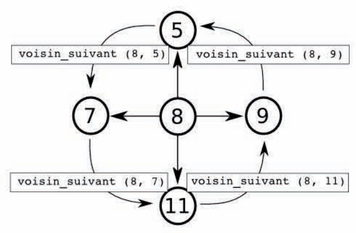
Un autre itinéraire de 5 à 15 :
[5; 4; 7; 10; 11; 14; 15]

Des **algorithmes** pour résoudre le problème

Suivre un itinéraire

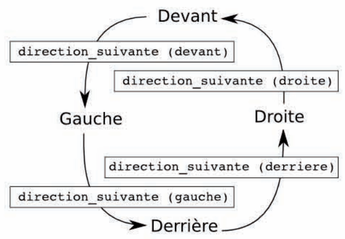
Sous-problème : **"Choisir la bonne direction."**
Entrées :
- Le robot vient d'une position d'`origine`.
- Il se trouve maintenant à une certaine `position`.
- On doit se diriger vers une position de `destination`.

Sortie :
- Le robot doit prendre la `direction` renvoyée.



Programme "`choisir_direction`" *:*

```
direction := derriere;
noeud := origine;
repeter toujours
 noeud := voisin_suivant (position, noeud);
 direction := direction_suivante (direction);
 si noeud = destination
 alors renvoyer (direction);
 si noeud = origine
 alors erreur
```



Problème : **"Suivre un itinéraire."**
Entrées :
- Une position d'`origine`.
- Un `itineraire`.

Sortie :
- Le robot se trouve à la position finale de l'itinéraire.

Programme "`suivre_itineraire`"*:*

```
position := origine;
position_precedente := origine;
tant que non (position = position_finale (itineraire))
 direction := choisir_direction (position_precedente,
                                 position,
                                 etape_suivante (position, itineraire));
 position_precedente := position;
 position := suivre_direction (position, direction)
```

Trouver le meilleur itinéraire

En essayant toutes les solutions ...

```
[1; 2; 3; 6; 5; 8; 9; 12; 15; 14; 11; 10; 7]     27
[1; 2; 5; 8; 9; 12; 15; 14; 11; 10; 7]           29
[1; 2; 3; 6; 5; 8; 9; 12; 11; 10; 7]             24
[1; 2; 5; 8; 9; 12; 11; 10; 7]                   26
[1; 2; 3; 6; 5; 8; 11; 10; 7]                    18
[1; 2; 5; 8; 11; 10; 7]                          20
[1; 2; 3; 6; 5; 8; 7]                            13
[1; 2; 5; 8; 7]                                  15
[1; 2; 3; 6; 5; 4; 7]                            16
[1; 2; 5; 4; 7]                                  18
```

} 10 chemins possibles du point 1 au point 7

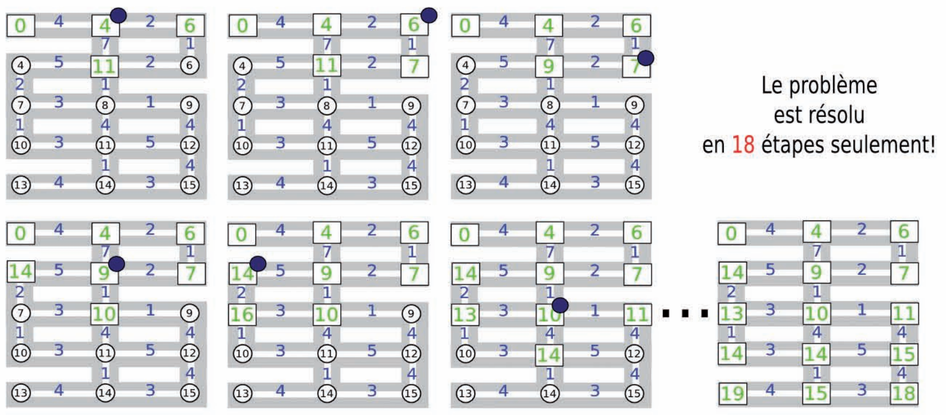
Nous avons une carte avec 15 emplacements. Le nombre de chemins de l'emplacement numéro 1 à 15 est 26.
Si nous avions une carte avec 25 emplacements, le nombre de chemins de l'emplacement 1 à 25 serait à peu près 5000.
Si nous avions une carte avec 36 emplacements, le nombre de chemins de l'emplacement 1 à 36 serait à peu près 1 000 000!

Cette méthode ne fonctionne donc pas pour les cartes que nous utilisons tous les jours!

En étant plus malin ...

1. Je note sur chaque emplacement le meilleur temps connu jusqu'à maintenant pour s'y rendre depuis l'emplacement 1.
2. Je parcours les emplacements et pour chaque voisin, je calcule le temps que j'ai mis pour m'y rendre.
3. Si ce temps est meilleur que celui déjà noté sur le voisin, alors j'inscris ce nouveau temps sur cet emplacement.

(Je continue tant que j'arrive à améliorer les meilleurs temps connus.)



Le problème est résolu en 18 étapes seulement!