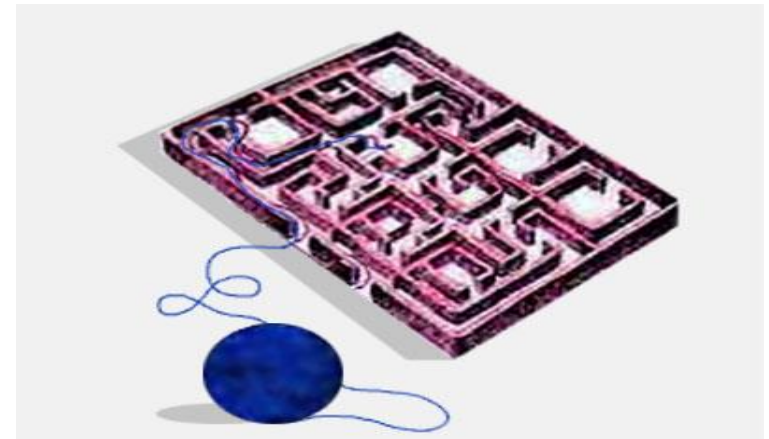


RECHERCHE AVEUGLE ET HEURISTIQUE

La recherche exploratoire est une vieille idée

Le labyrinthe et le Fil d'Ariane

Selon la mythologie grecque, Thésée vint en Crête pour tuer le Minotaure, un monstre qui vivait dans un labyrinthe. Ariane donna à Thésée un fil qu'il déroula en parcourant le labyrinthe. Après avoir tué le Minotaure, Thésée retrouva son chemin vers l'entrée en suivant le fil, rejoignit Ariane et s'échappa de Crête.

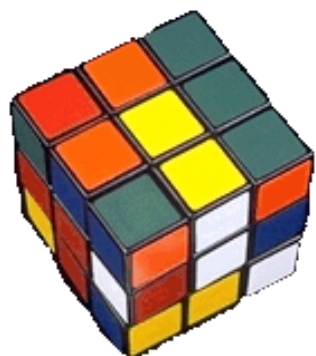


L'histoire des civilisations montre que les puzzles et les jeux qui demandent d'explorer des solutions alternatives ont fasciné les hommes et ont été considérés comme un défi pour l'intelligence humaine:

- Les échecs: originaire de Perse et des Indes il y a environ 4000 ans
- Le jeu de dames apparaît il y a 3600 ans sur des fresques égyptiennes
- Le jeu de Go apparaît en Chine il y a plus de 3000 ans



	M	A	S	H		S	O	M	A	T	I	C		M	A	P	S							
P	I	X	I	E		W	A	Y	S	I	D	E		E	S	A	U							
O	D	E	L	L		A	R	T	I	C	L	E		L	I	T								
W	I	L	L	I	A	M	S	H	A	K	E	S	P	E	A	R	E							
			X	S										S	E	N	I	O	R					
A	M	F	M		I	R	C		R	I	G	H	T		A	N	Y							
S	A	L	M	A	N	R	U	S	H	D	I	E			A	R	S	E						
A	L	U	M	N	I		B	U	Y	I	N				R	C								
					O	N	E	I		M	O				P	H								
			E	R	N	E	S	T	H	E	M	I	N	G	W	A	Y							
	A	B	E					E	S		M	O	A											
S	M	B	D					U	L	T	R	A		Y	O	N	D	E	R					
M	I	S	S					A	L	L	E	N	G	I	N	S	B	E	R	G				
U	N							O	D	E	A	R		E	V	E		C	R	A	B			
T	O	I	L	E	R									S	O									
								B	A	R	B	A	R	A	K	I	N	G	S	O	L	V	E	R
	E	L	E	N	A			M	A	L	A	R	I	A		M	O	I	R	E				
	R	E	A	C	T			O	N	A	N	I	S	T		P	L	A	S	M				
	R	O	M	E	O			K	I	N	E	S	I	S		H	A	L	T					



Exemple: puzzle-8

8	2	
3	4	7
5	1	6

État initial

1	2	3
4	5	6
7	8	

État final

État: n'importe quel arrangement des 8 plaquettes numérotées et de la case vide sur un damier 3x3

Espace d'états pour le puzzle-8

8	2	
3	4	7
5	1	6

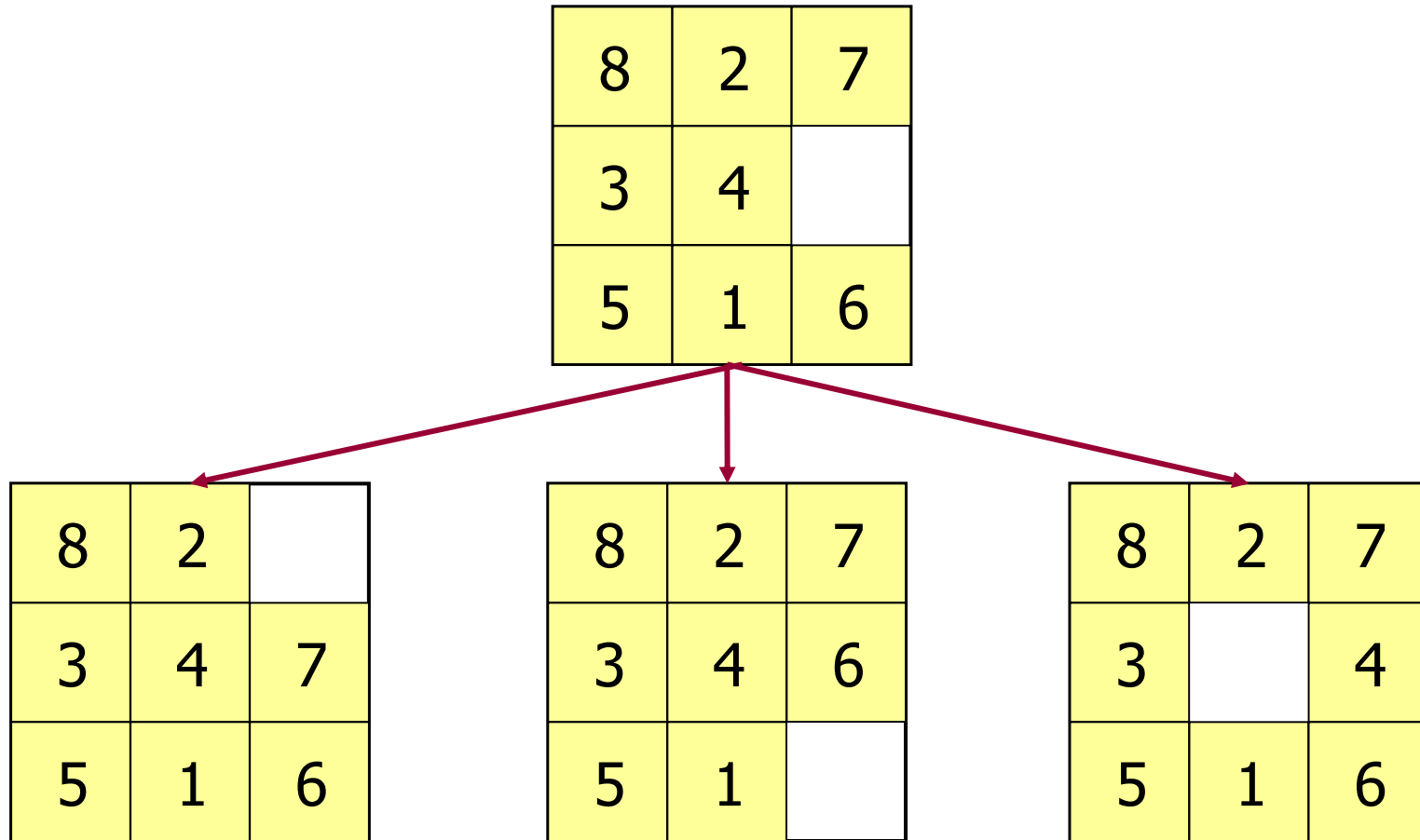
8	2	7
3	4	
5	1	6

...

8		2
3	4	7
5	1	6

	8	2
3	4	7
5	1	6

Fonction "successeur" du puzzle-8



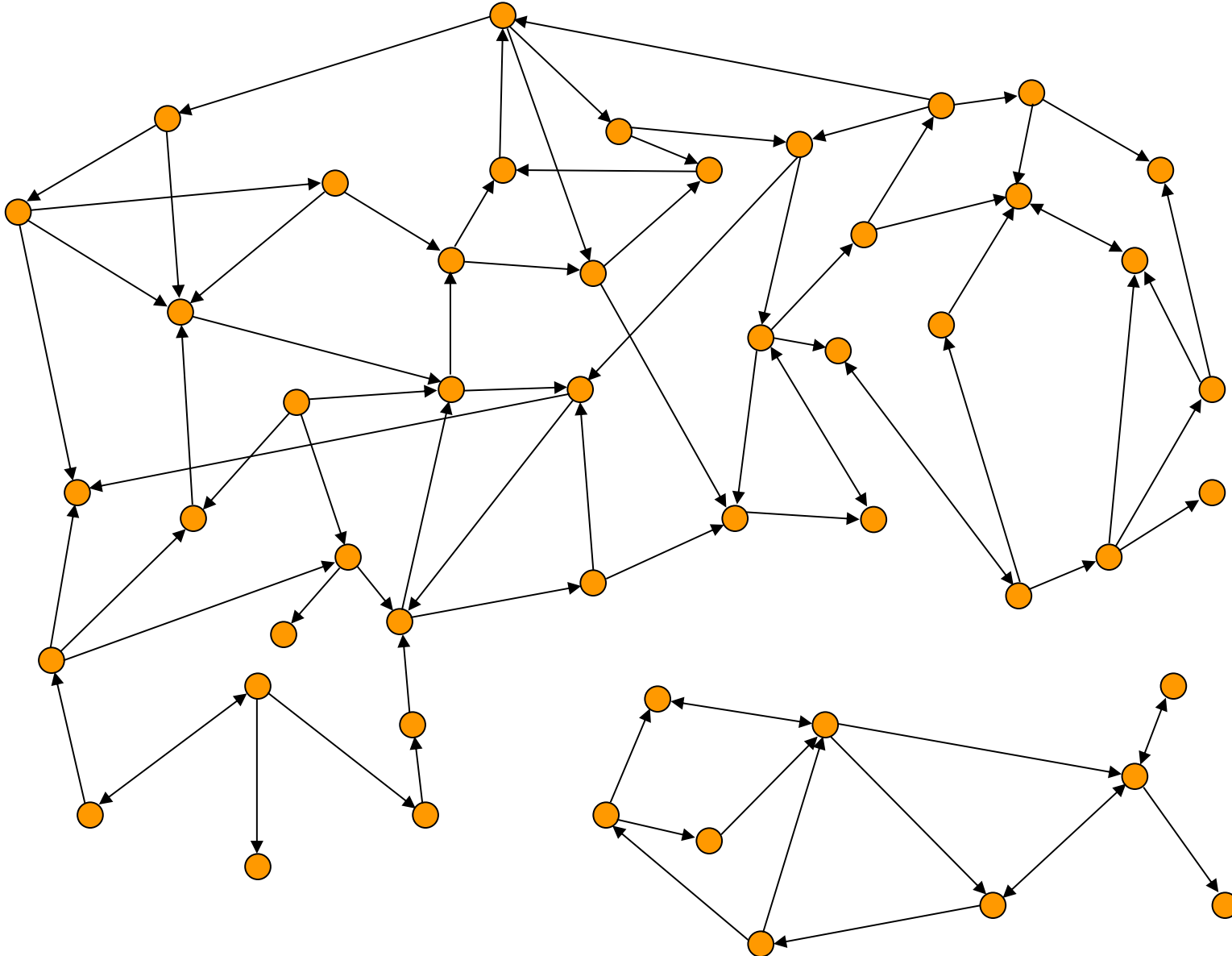
- Les algorithmes de recherche sont une technique générale de résolution de problèmes qui:
 - évolue dans un espace appelé *espace d'états*
 - explore systématiquement toutes les *alternatives*
 - trouve la *séquence d'étapes* menant à la solution
- "Problem Space Hypothesis"
 - (Allen Newell, SOAR: An Architecture for General Intelligence, 1987)
 - toute recherche orientée vers un but se déroule dans *l'espace du problème (espace d'états)*

- Défini comme:
 - Chaque état est représenté par un nœud distinct
 - Un arc (ou arrête) relie un nœud s à un nœud s' si

$$s' \in \text{SUCCESSEURS}(s)$$

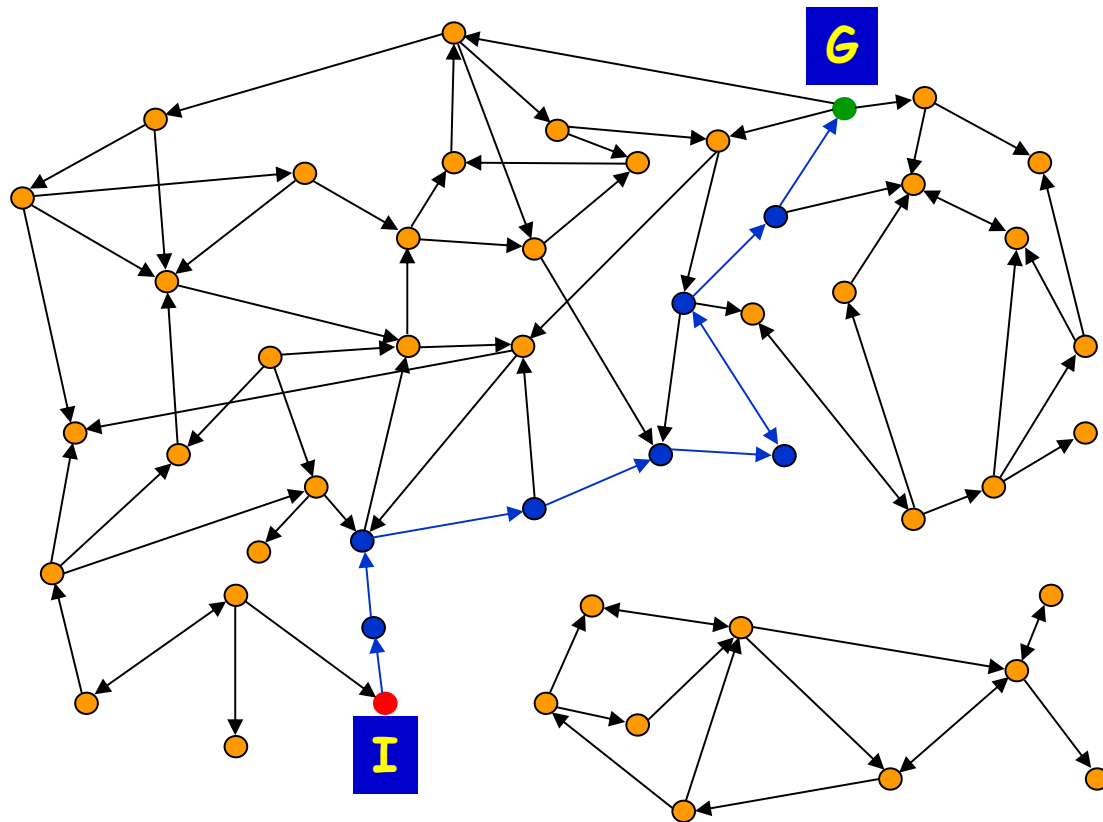
- Le graphe d'états peut contenir plus d'une composante connexe

Graphe d'états



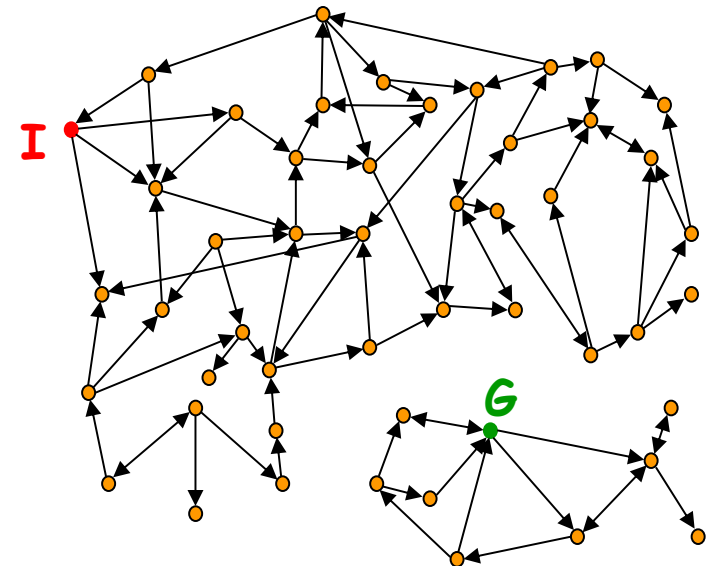
Solution à un problème de recherche

- Une **solution** est un chemin reliant l'état initial **I** à un état solution **G** (n'importe lequel)



Solution à un problème de recherche

- Une **solution** est un chemin reliant l'état initial I à un état solution G (n'importe lequel)
- Le **coût** d'un chemin est la somme des coûts des arcs qui le constituent
- Une solution **optimale** est un chemin-solution de coût minimum (la plupart du temps) ou maximum
- Il peut ne pas y avoir de solution !!!



Quelle est la taille de l'espace de recherche ?

Exemple pour un puzzle- (n^2-1)

- puzzle-8 → **combien d'états ???**

Quelle est la taille de l'espace de recherche ?

Exemple pour un puzzle

- puzzle-8 $\rightarrow 9! = 362'880$ états
- puzzle-15 $\rightarrow 16! \sim 2.09 \times 10^{13}$ états
- puzzle-24 $\rightarrow 25! \sim 10^{25}$ états

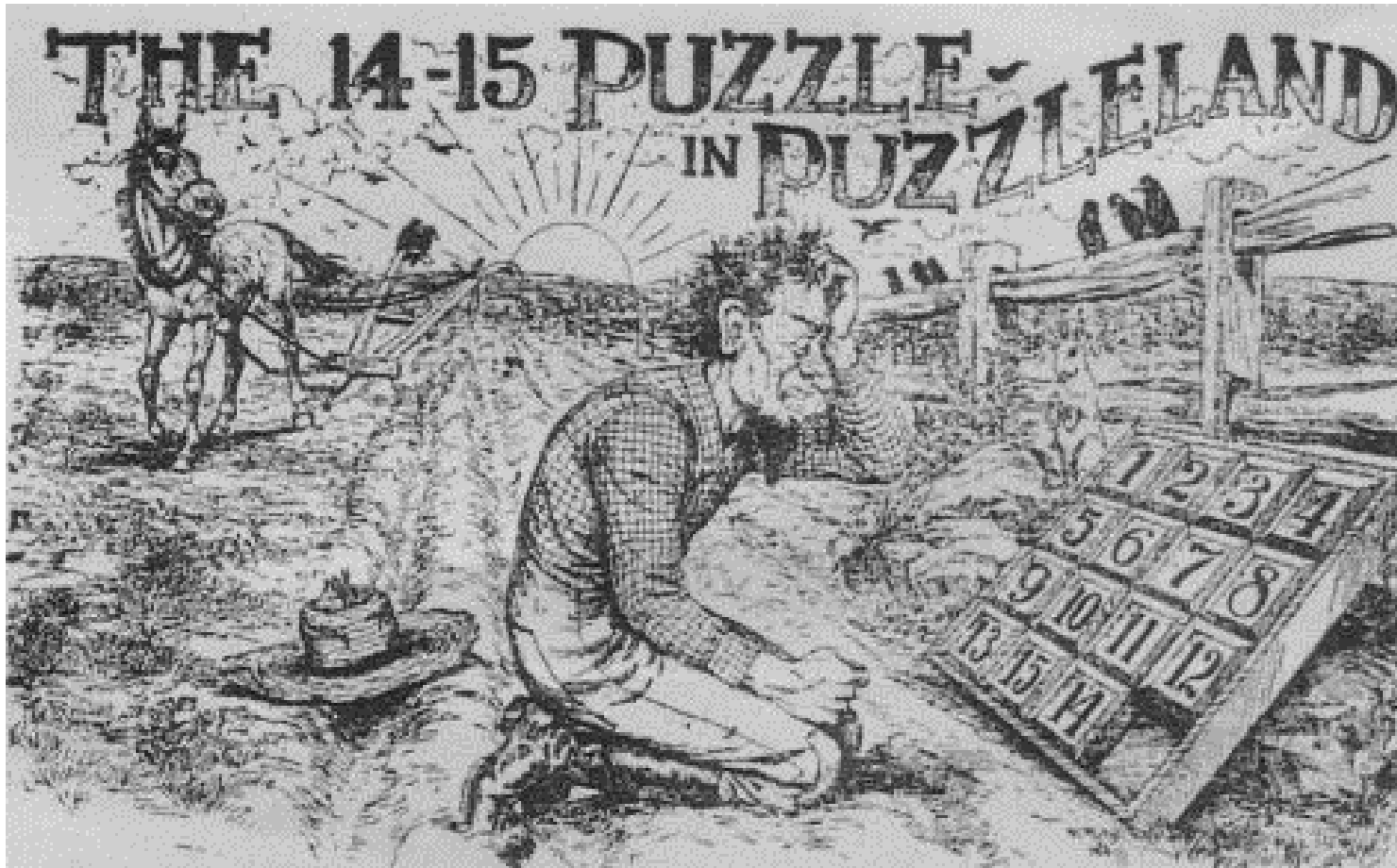
**Mais seule la moitié de ces états est accessible à partir de
n'importe quel état donné !
(mais on peut ne pas savoir ça d'avance)**

Puzzle

8	2	
3	4	7
5	1	6

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

■ ■ ■ ■



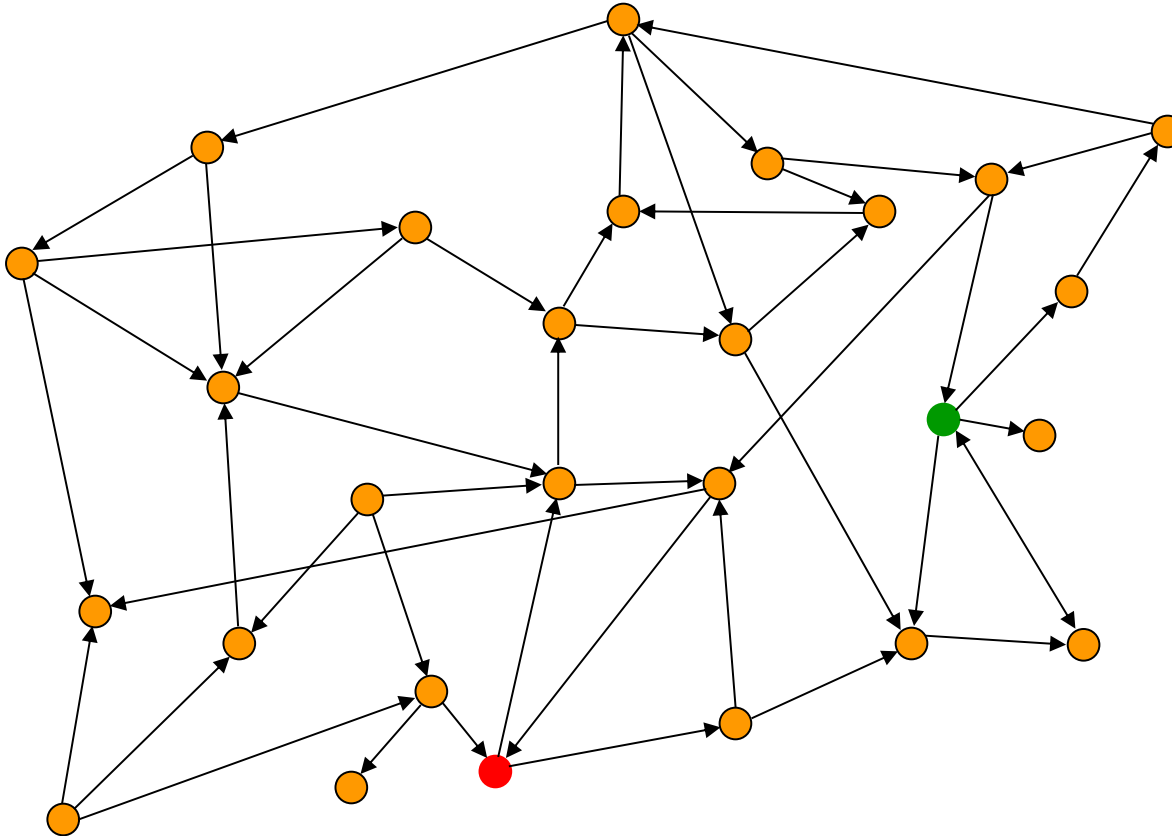
Personne n'a jamais remporté le prix !!

Quel est alors l'espace de recherche ?

- a) L'ensemble de tous les états?
[càd, un ensemble de $16!$ états pour le puzzle-15]
- b) L'ensemble de tous les états accessibles à partir d'un état initial donné?
[càd, un ensemble de $16!/2$ états pour le puzzle-15]

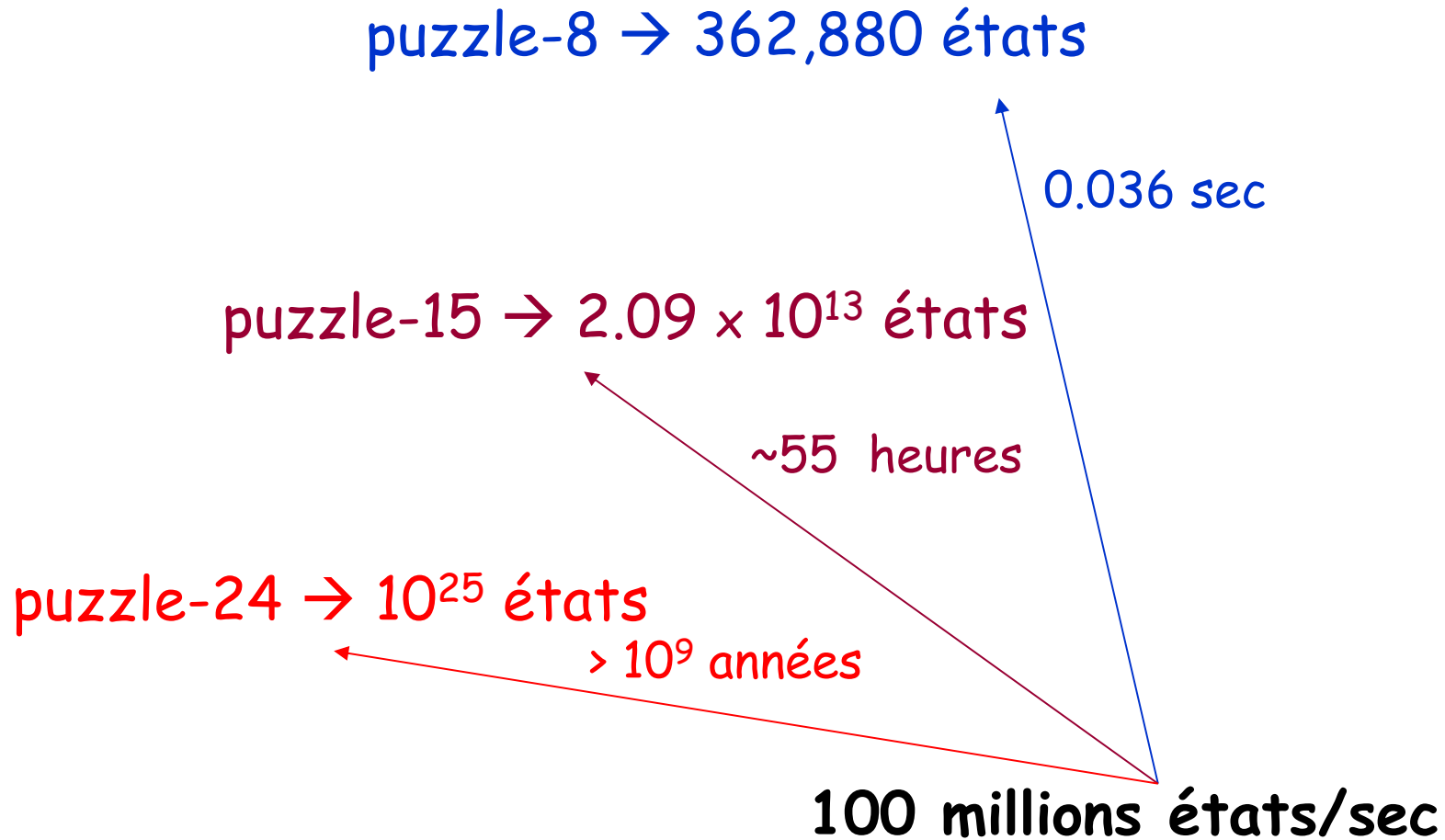
En général la réponse est a)
[car on ne connaît pas à l'avance quels sont les états accessibles]

Recherche dans un espace d'états

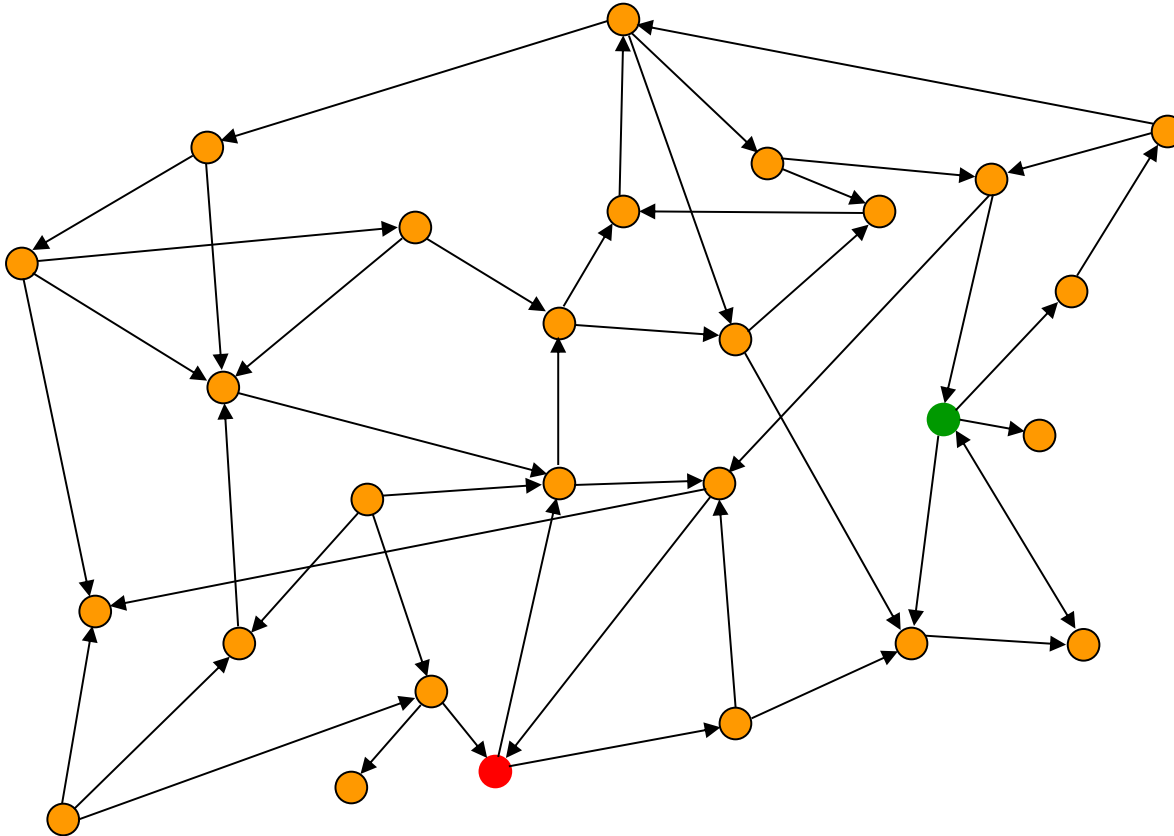


- En général, il est souvent impossible (ou trop coûteux) de construire une représentation complète du graphe d'états

Puzzle-8, -15, -24

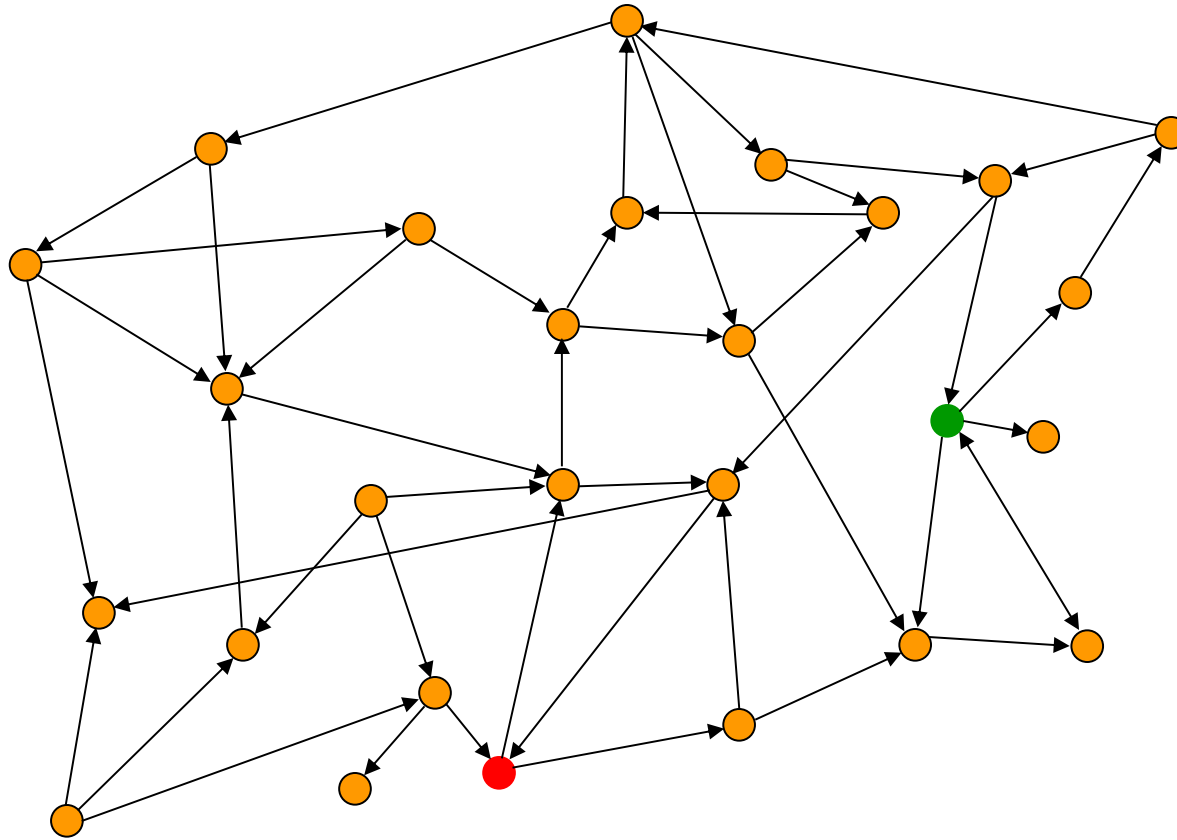


Recherche dans un espace d'états

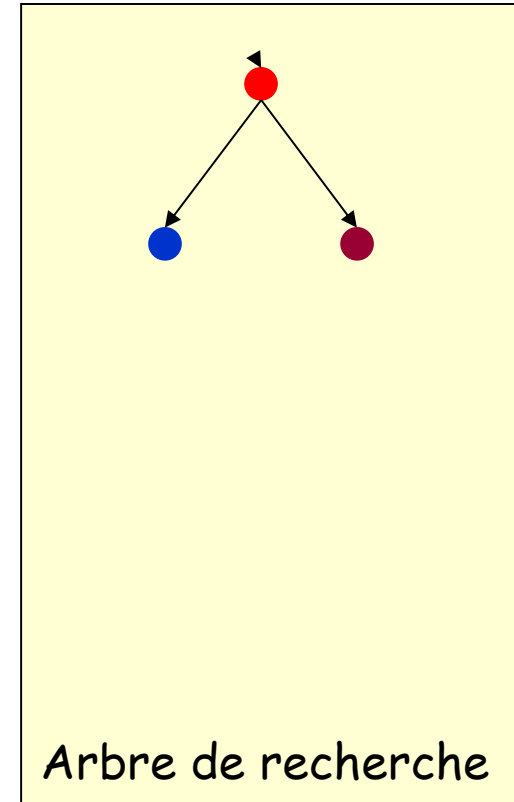
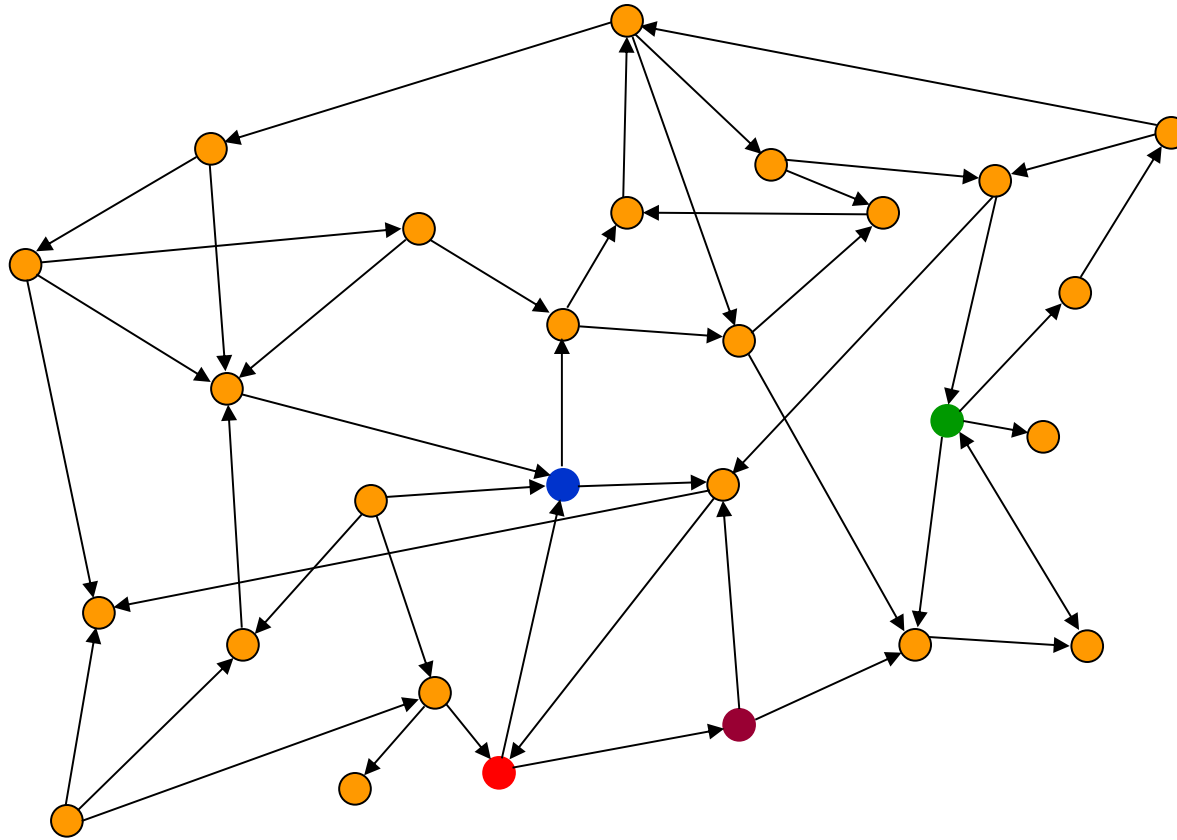


- Souvent il est impossible (ou trop coûteux) de construire une représentation complète du graphe d'états
- On doit donc construire une solution en n'explorant qu'une petite portion du graphe

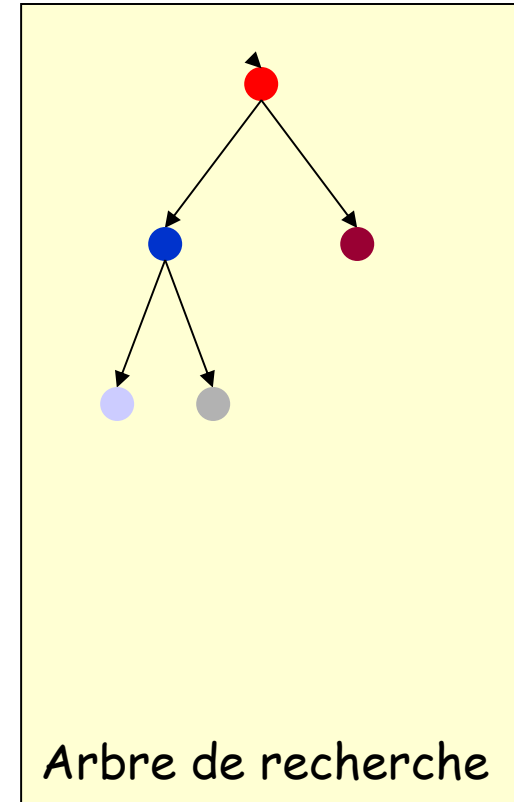
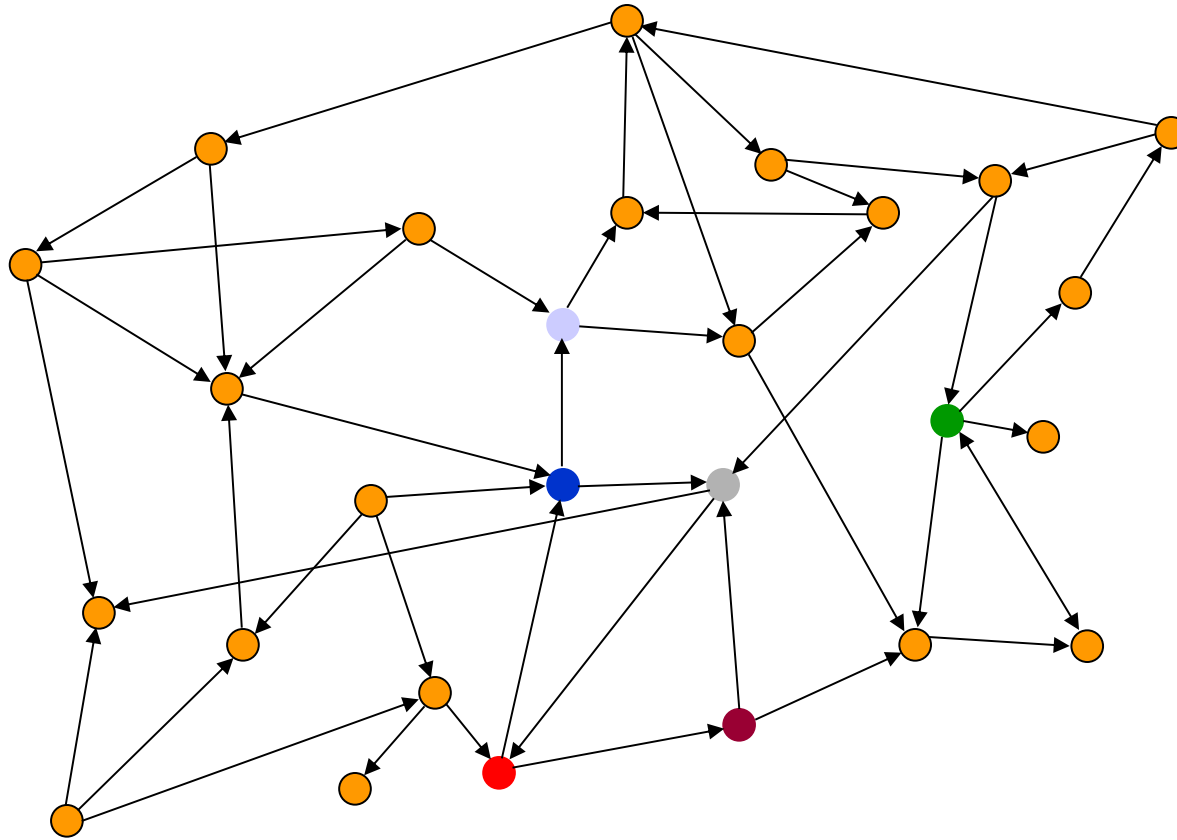
Recherche dans un espace d'états



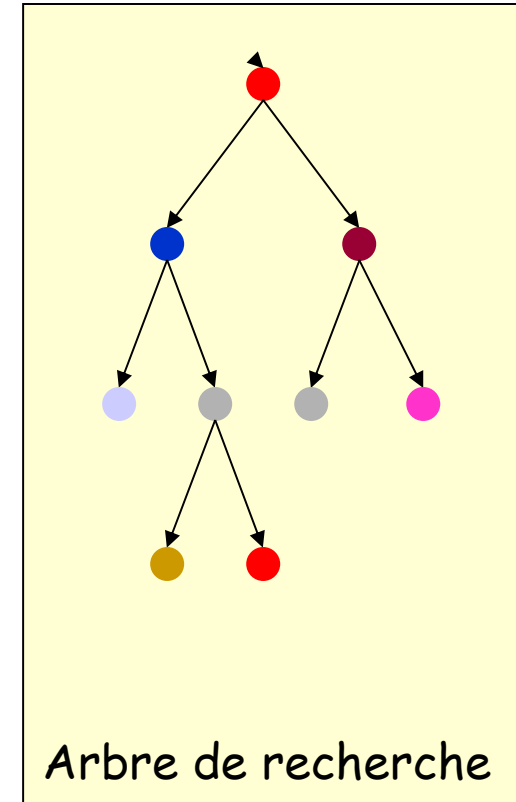
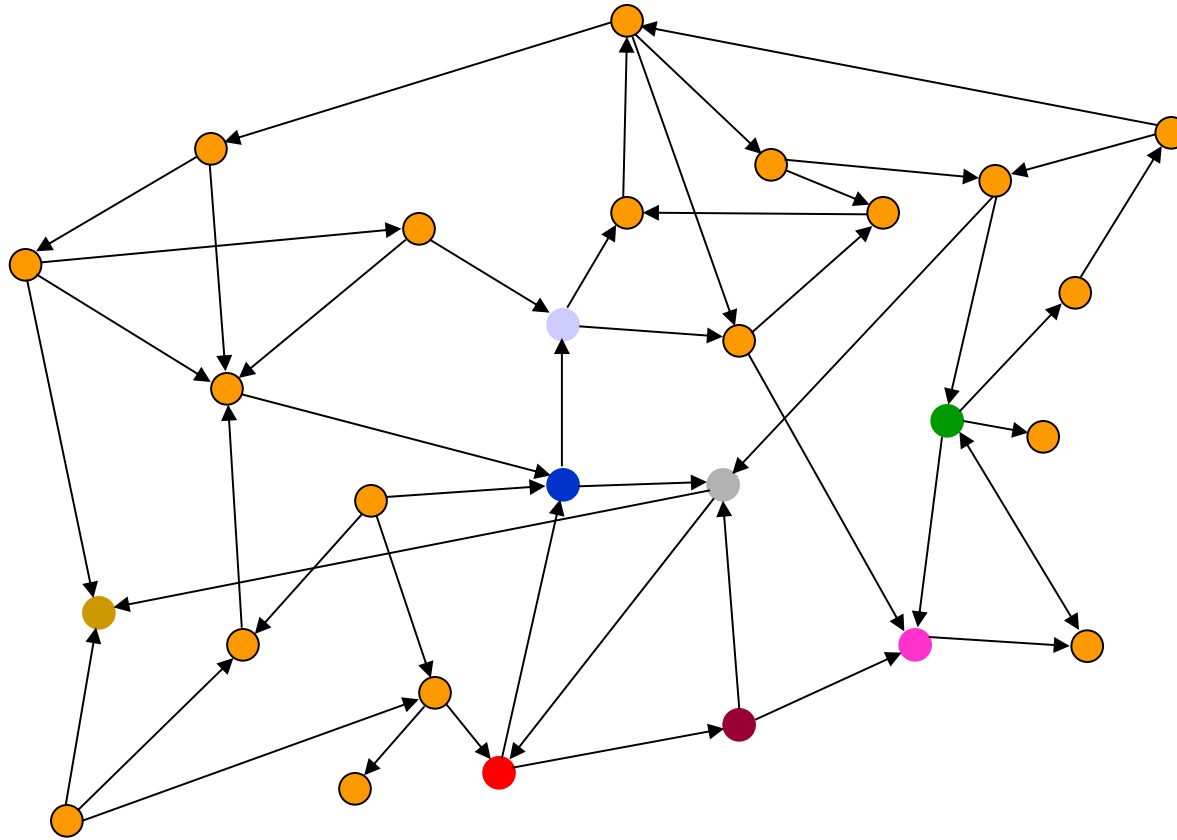
Recherche dans un espace d'états



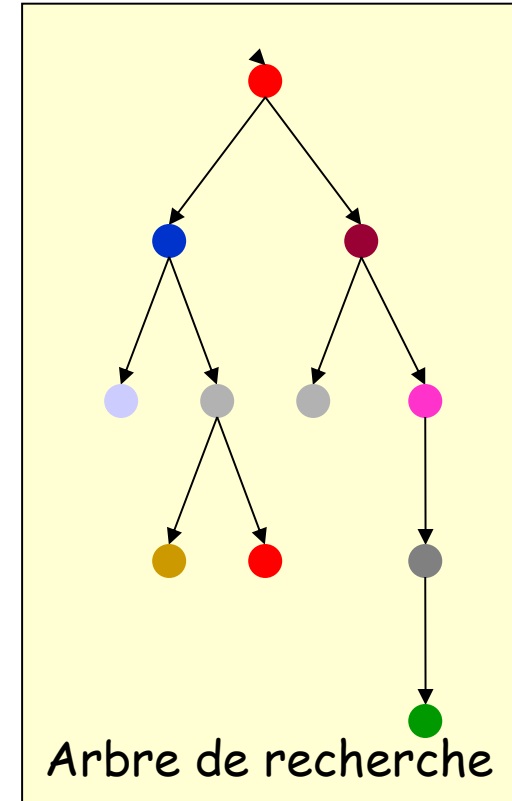
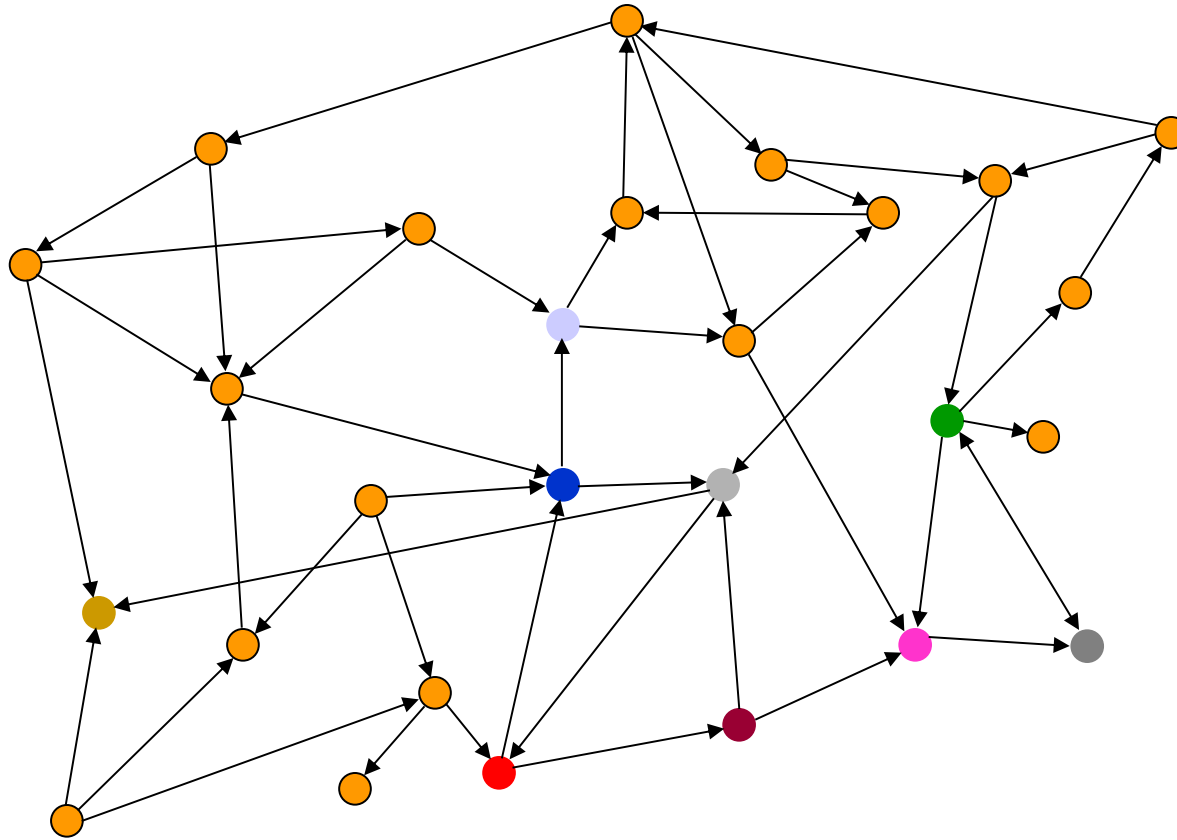
Recherche dans un espace d'états



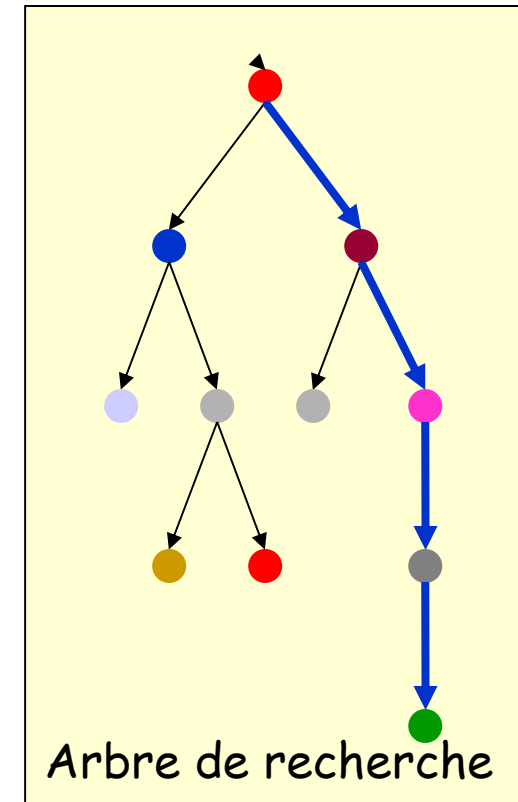
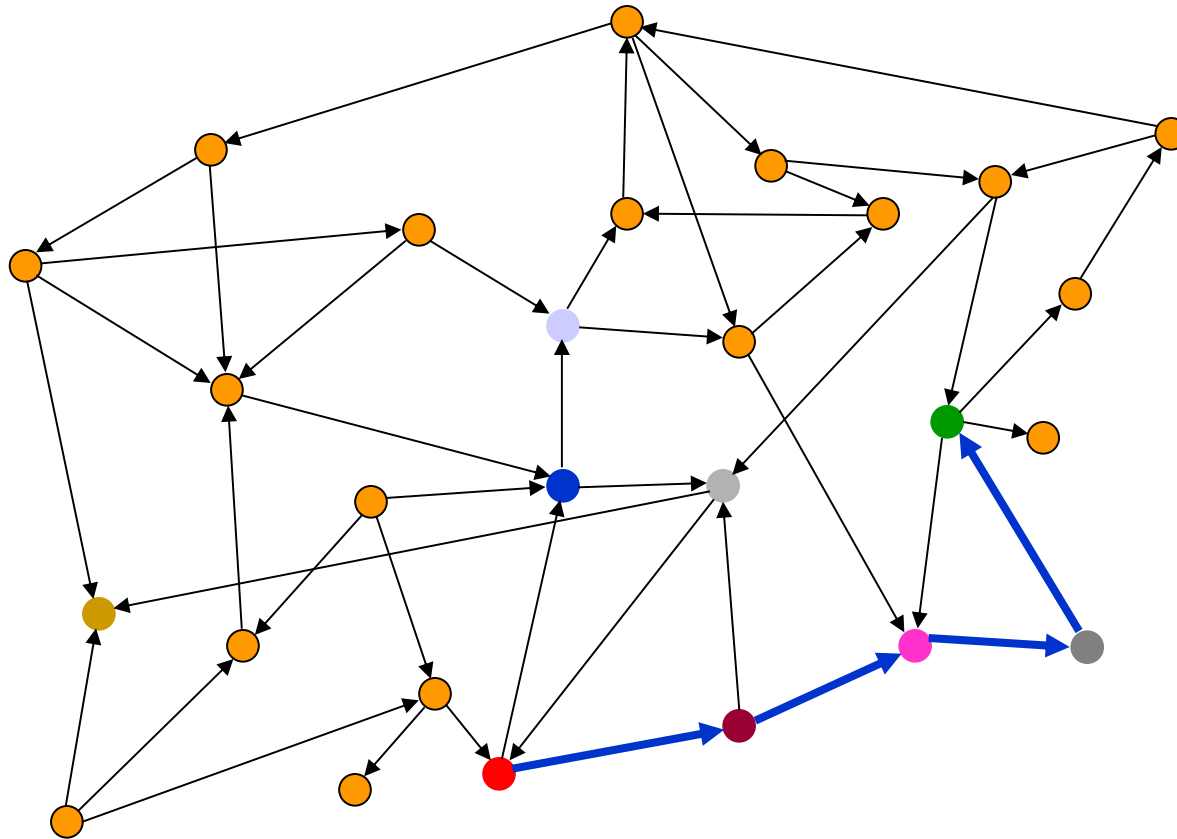
Recherche dans un espace d'états



Recherche dans un espace d'états



Recherche dans un espace d'états



Algorithme général de recherche (#1)

1. Si SOLUTION?(état-initial) alors retourner état-initial
2. INSERER(noeud-initial, FILE)
3. Répéter:
 - a. Si vide(FILE) alors retourner **échec**
 - b. $n \leftarrow \text{RETIRER}(\text{FILE})$ *Prolongement de n*
 - c. $s \leftarrow \text{ETAT}(n)$
 - d. Pour chaque état s' SUCCESSEUR(s)
 - i. Créer un nouveau noeud n' comme "enfant" de n
 - ii. Si SOLUTION?(s') alors retourner **chemin ou état-solution**
 - iii. INSERER(n' , FILE)

Les méthodes de recherche diffèrent les unes des autres selon l'ordre dans lequel les nœuds sont insérés et extraits de la file d'attente.

Définition formelle d'un problème de recherche

- Un problème de recherche est défini par les éléments suivants:
 - Q un ensemble fini non-vide d'états
 - $S \subseteq Q$ un ensemble non-vide d'états initiaux
 - $G \subseteq Q$ un ensemble non-vide d'états-solutions (pouvant être définis explicitement ou implicitement par un test-solution)
 - A un ensemble d'actions
 - Successeurs: $Q \times A \rightarrow Q$
 - Fonction d'évaluation: $Q \times A \times Q \rightarrow \mathbb{R}^+$
(défini uniquement pour des états qui sont successeurs l'un de l'autre)

Définir un problème de recherche

- Espace d'états
- État initial
- Fonction "successeur"
- Etat-solution
- Coût du chemin

Définir un problème de recherche

- Espace d'états
 - chaque état est une représentation abstraite de l'environnement
 - l'espace d'état est discret, il peut être fini ou infini
- État initial
- Fonction "successeur"
- Etat-solution
- Coût du chemin

Définir un problème de recherche

- Espace d'états
- État initial
 - habituellement l'état courant
 - parfois un ou plusieurs états hypothétiques
- Fonction "successeur"
- Etat-solution
- Coût du chemin

Définir un problème de recherche

- Espace d'états
- État initial
- Fonction "successeur"
 - fonction : [état \rightarrow sous-ensemble d'états]
 - une représentation abstraite des actions possibles
- Etat-solution
- Coût du chemin

Définir un problème de recherche

- Espace d'états
- État initial
- Fonction "successeur"
- Etat-solution
 - habituellement une condition à satisfaire
 - parfois la description explicite d'un état
- Coût du chemin

Définir un problème de recherche

- Espace d'états
- État initial
- Fonction "successeur"
- Etat-solution
- Coût du chemin
 - fonction : [chemin \rightarrow nombre positif]
 - habituellement: coût du chemin = somme des coûts de ses étapes
 - Ex: # déplacements de la plaquette "vide"

Fonction successeur

- Elle représente implicitement toutes les actions possibles dans chaque état
- Seuls les résultats des actions (les états successeurs) et leurs coûts sont retournés par la fonction
- La fonction successeur est une "boîte noire", son contenu est (le plus souvent) inconnu.
Par ex., en planification d'assemblage, la fonction successeur peut être particulièrement complexe (collision, stabilité, prise, ...)

Coût d'un chemin

- Le coût d'un arc est un nombre positif mesurant le "coût" de la réalisation de l'action représentée par l'arc, exemples:
 - 1 pour le puzzle-8
 - le temps nécessaire à assembler deux éléments d'un dispositif
- On suppose que pour n'importe quel problème le coût c d'un arc vérifie toujours la relation

$$c \geq \varepsilon > 0 \quad \text{où } \varepsilon \text{ est une constante}$$

[cette condition garantit que, si le chemin devient arbitrairement long, son coût devient aussi arbitrairement grand]

Pourquoi est-ce nécessaire?

Etat solution

- Il peut être explicitement décrit

1	2	3
4	5	6
7	8	

- ou partiellement décrit

1	~	~
~	5	~
~	8	~

("~" signifie "quelconque"
sauf 1, 5 et 8)

- ou défini par une condition
ex: la somme de chaque ligne, chaque
colonne et chaque diagonale vaut 30

15	1	2	12
4	10	9	7
8	6	5	11
3	13	14	


- Les méthodes de recherche jouent un rôle-clé dans beaucoup d'applications ex.:
 - Recherche de chemin : lignes aériennes, réseaux téléphoniques et d'ordinateurs,
 - Distribution courrier, colis
 - Routage dans oléoducs, routage circuits VLSI,
 - Conception de médicaments,
 - Bioinformatique (classification et comparaison de structures de protéines)
 - Planification des déplacements d'un robot autonome,
 - Jeux vidéo.

Recherche heuristique: généralités

- Les algorithmes de **recherche aveugle** n'exploitent **aucune information** concernant la structure de l'arbre de recherche ou la présence potentielle de nœuds-solution pour **optimiser la recherche**.
- Recherche "rustique" à travers l'espace jusqu'à trouver une solution.
- La plupart des problèmes réels sont susceptibles de provoquer une **explosion combinatoire** du nombre d'états possibles.
- Un algorithme de **recherche heuristique** utilise toute l'information disponible pour rendre le processus de recherche **plus efficace**.
- Une **information heuristique** est une règle ou une méthode qui **améliore presque toujours** le processus de recherche.

Rappel

L'ordre des éléments
dans la file d'attente
définit la stratégie de
recherche



Algorithme de recherche modifié

1. INSERER(noeud-initial, FILE)
2. Répéter:
 - a. Si vide(FILE) alors retourner **échec**
 - b. $n \leftarrow$ RETIRER(FILE)
 - c. $s \leftarrow$ ETAT(n)
 - d. Si SOLUTION?(s) alors retourner **chemin ou état-solution**
 - e. Pour chaque état s' SUCCESEURS(s)
 - i. Créer un noeud n' comme successeur de n
 - ii. INSERER(n' , FILE)

14 2005-2006 - © C. Pellegrin

63

Fonction heuristique

- La **fonction heuristique** $h(N) \geq 0$ estime la distance séparant ETAT(N) d'un état-solution
- Sa valeur est **indépendante de l'arbre de recherche**, elle ne dépend que de ETAT(N) et du test SOLUTION?
- Exemple

5		8
4	2	1
7	3	6

ETAT(N)

1	2	3
4	5	6
7	8	

état solution

$$h_1(N) = \text{nombre de plaquettes mal placées} = 6$$

Autres exemples

5		8
4	2	1
7	3	6

ETAT(N)

1	2	3
4	5	6
7	8	

état solution

$h_1(N)$ = nombre de plaquettes mal placées = 6

$h_2(N)$ = somme des distances (Manhattan) de chaque
plaquette numérotée à sa position finale
= $2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13$

- Si l'espace d'états est infini, alors la recherche n'est en général pas complète (= n'aboutit pas à une solution)
- Si l'espace d'états est fini et si on n'écarte pas les nœuds qui revisitent des états déjà rencontrés, alors la recherche n'est en général pas complète
- Si l'espace d'états est fini et si on écarte les nœuds qui revisitent des états déjà rencontrés, alors la recherche est complète, mais en général elle ne sera pas optimale