Structures de données et algorithmes

Répétition 7: Résolution de problèmes

Jean-Michel Begon

10 avril 2020

1 Recherche exhaustive

1.1 Voyageur de commerce

Soient n villes et D, une matrice de distances telle que $D_{i,j} > 0$ est la distance entre la ville i et la ville j.

Le problème du voyageur de commerce (*travel salesman problem, TSP*) consiste à trouver le chemin minimum pour passer par n villes et revenir à la ville de départ.

Quelle serait la complexité d'une recherche exhaustive, en fonction de n pour résoudre ce problème ?

Notons que $D_{i,j}$ n'est pas nécessairement égal à $D_{j,i}$ et qu'il n'existe pas nécessairement un chemin entre i et j (ce qu'on notera $D_{i,j} = \infty$), même s'il est possible d'atteindre toutes les villes (le graphe des villes est connexe).

1.2 Problème SAT

Soit $\Pi = \{p_1, \dots p_n\}$ un ensemble de n symboles propositionnels, et soit Φ une formule logique (grammaticalement correcte) construite sur ces symboles et contenant m ($n \leq m$) propositions.

Le problème SAT consiste à déterminer s'il existe une interprétation qui satisfasse la formule Φ. C'est-à-dire, s'il est possible de rendre la formule "vraie" pour au moins une assignation de valeurs de vérité des propositions.

Par exemple, la formule

$$(p_1 \wedge p_2) \vee (p_1 \wedge p_3) \tag{1}$$

qui se lit "p_1 et p_2 ou p_1 et p_3 est satisfaisable, par exemple si p_1 et p_3 sont vrais. Ici $n = 3$ et $m = 4$

En revanche, la formule

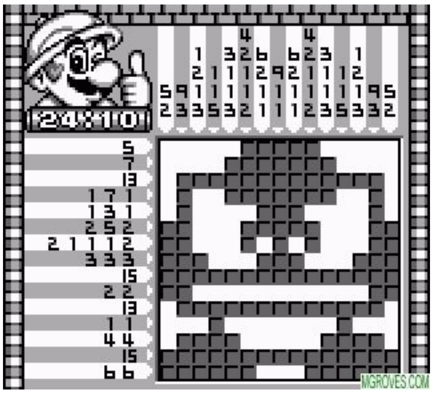
$$p_1 \wedge (\neg p_1) \tag{2}$$

qui se lit "p_1 et non p_1" n'est pas satisfaisable. Ici $n = 1$ et $m = 2$

Quelle serait la complexité d'une recherche exhaustive, en fonction de n pour résoudre ce problème ?

1.3 Logimage (aussi appelé picross)

Proposez un algorithme par recherche exhaustive pour résoudre un logimage (`https://fr.wikipedia.org/wiki/Picross`) tel que celui-ci :



Cette solution est-elle envisageable en pratique ? Par exemple pour une grille 15×15 ?

2 Memoïsation

2.1 Nombres de Catalan

Dans une répétition précédente, nous avions solutionné l'exercice :

> *Soit un ensemble de N valeurs entières distinctes. Ecrivez une fonction calculant le nombre d'arbres binaires de recherche distincts qu'il est possible de construire à partir de ces N valeurs ($N \geq 1$).*

par le pseudo-code suivant :

NbTree(N)
if $N \leq 1$
    return 1
$Nb = 0$
for $i = 1$ **to** N
    $Nb = Nb + \text{NbTree}(i-1) * \text{NbTree}(N-i)$
return Nb

dont la complexité est importante à cause des appels à des sous-problèmes déjà résolus.

1. Dessinez le graphe des sous-problèmes pour une taille de $N = 4$.
2. Utilisez la mémoïsation pour faire baisser la complexité de l'algorithme (donnez le pseudo-code d'une version ascendante et d'une version descendante).
3. Quelles sont ces complexités ?

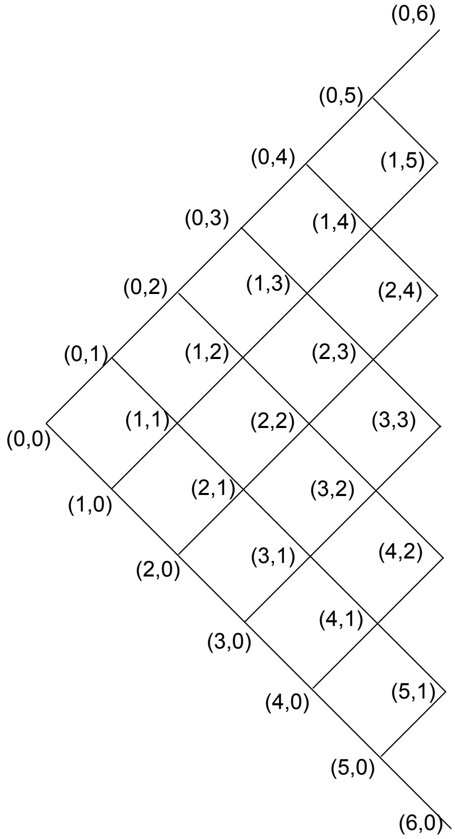


FIGURE 1 – Treillis de Launay

2.2 Micmaths

Dans une vidéo intitulée "La puissance organisatrice du hasard - Micmaths" (2017), Mickaël Launay exprime qu'une bille qui se déplace au hasard mais strictement vers la droite sur un treillis similaire à celui de la figure 1 va adopter une trajectoire presque droite car il y a plus de chemins qui mènent vers les positions centrales (par exemple la position $(3, 3)$, au départ de la position $(0, 0)$).

1. Soit $T(i, j)$ $(i, j = 0, \ldots, n-1)$ le nombre de chemins menant à la position (i, j) au départ de la position $(0, 0)$. Formulez récursivement T.
2. A l'aide de la mémoïsaiton, donnez le pseudo-code d'une fonction *efficace* pour calculer $T(i, j)$ $(i, j = 0, \ldots, n-1)$.

Remarque : si on suppose que tous les chemins sont équiprobables, on s'aperçoit que certains états "macroscopiques" (*i.e.* la position d'arrivée) sont, eux, plus probables. C'est la seconde loi de la thermodynamique.