Programmation Effective – TD 05 : Recherche exhaustive et *backtracking*

matthieu.gallet@ens-lyon.fr

mardi 11 mars 2008

1 Recherche exhaustive

Comme son nom l'indique, la recherche exhaustive (ou *brute-force*) consiste à explorer l'ensemble de l'espace des solutions à un problème, pour éliminer les solutions incorrectes et sélectionner la meilleure. Naturellement, cette méthode est généralement extrêmement coûteuse en temps et doit donc être réservée pour les petites instances des problèmes, ou quand il n'y a réellement pas d'autre solution. Cette méthode est souvent la seule possible quand on cherche à casser une clef de chiffrement. On peut souvent représenter les différentes possibilités sous forme d'un arbre. Prenons comme exemple la résolution d'un sac-à-dos de taille 10, avec des objets de couples (tailles, valeurs) respectives $(3, 8)$, $(12, 9)$, $(4, 7)$, $(5, 12)$. L'arbre (voir Figure 1) permet de visualiser simplement l'ensemble des choix à faire pour obtenir n'importe quelle solution, et, finalement, de choisir la meilleure. Cependant, ce n'est pas parce qu'on se résout à utiliser une méthode de recherche exhaustive qu'il ne faut pas être intelligent ! Par exemple, on peut organiser l'espace des solutions de façon à mettre les solutions les plus probables en premier. Ainsi, si on cherche à casser une clef de chiffrement, on testera en premier les mots du dictionnaire qui ont plus de chances d'apparaître.

2 *backtracking*

Dans l'exemple précédent, nous avons généré l'ensemble des solutions, mais on peut facilement se rendre compte que dès qu'on prend l'objet de taille 12, on explose le sac-à-dos et on obtient une solution irréalisable. On peut donc s'arrêter de chercher dans cette direction. On parle alors d'élagage de l'arbre, et on fait du *backtracking*. La Figure 1 montre cet élagage. Le code général pour une recherche de solution utilisant de l'élagage peut être donné par l'Algorithme 1.

Comme pour la recherche exhaustive, l'ordre de traitement des variables peut avoir de l'importance sur le temps de traitement. Si on reprend l'exemple du sac-à-dos, il vaut mieux traiter l'objet de taille 12 en premier plutôt qu'en dernier, afin d'éliminer au plus tôt une grande partie des solutions potentielles. On traite en général les variables les plus contraignantes en premier.

3 Applications

Résolvez les problèmes suivants : `http://acm.uva.es/p/v1/102.html`, `http://acm.uva.es/p/v1/112.html`, `http://acm.uva.es/p/v1/131.html`, `http://acm.uva.es/p/v1/140.html`, `http://acm.uva.es/p/v104/10483.html`, `http://acm.uva.es/p/v107/10776.html`.

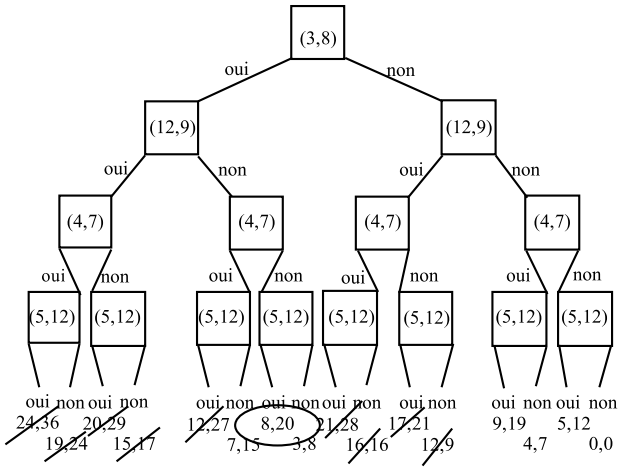

FIG. 1 – Ensemble des possibilités pour un sac-à-dos

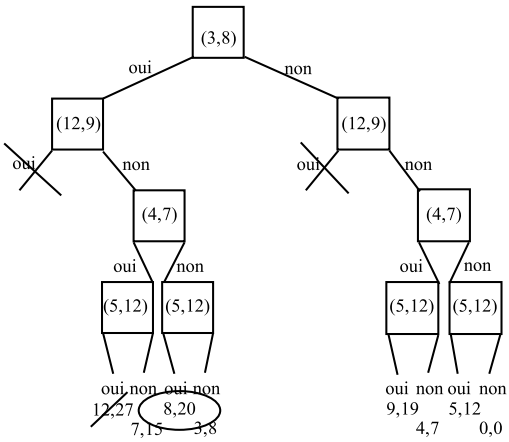

FIG. 2 – Élagage des possibilités

Algorithm 1 backtrackingSearch()

```
si toutes les variables sont choisies alors
  si la solution est valide alors
    renvoyer VRAI
  sinon
    renvoyer FAUX
  finsi
sinon
  soit v une variable libre
  pour tout x valeur possible de v faire
    v ← x
    si la solution partielle est faisable alors
      si backtrackingSearch() alors
        renvoyer VRAI
      finsi
    finsi
  fin pour
  libérer v
  renvoyer FAUX
finsi
```