Algorithmique

11 avril 2022

Table of Contents

Recherche par force brute

Définition

La **recherche exhaustive** ou **recherche par force brute** consiste à énumérer et tester l'ensemble des cas existants, jusqu'à répondre au problème posé.

Recherche par force brute

Définition

La **recherche exhaustive** ou **recherche par force brute** consiste à énumérer et tester l'ensemble des cas existants, jusqu'à répondre au problème posé.

Recherche par force brute

Définition

La **recherche exhaustive** ou **recherche par force brute** consiste à énumérer et tester l'ensemble des cas existants, jusqu'à répondre au problème posé.

Exemple

- Résolution Rubik's cube
- Trouver une configuration pour placer n reines sur un échiquier $n \times n$
- Pour trouver un mot de passe, on peut les vérifier tous jusqu'à trouver celui que l'on cherche.

Recherche par force brute

Définition

La **recherche exhaustive** ou **recherche par force brute** consiste à énumérer et tester l'ensemble des cas existants, jusqu'à répondre au problème posé.

Exemple

- Résolution Rubik's cube
- Trouver une configuration pour placer n reines sur un échiquier $n \times n$
- Pour trouver un mot de passe, on peut les vérifier tous jusqu'à trouver celui que l'on cherche.

Une recherche exhaustive n'est pas toujours une stratégie optimale !

Recherche par force brute

Définition

La **recherche exhaustive** ou **recherche par force brute** consiste à énumérer et tester l'ensemble des cas existants, jusqu'à répondre au problème posé.

Exemple

- Résolution Rubik's cube
- Trouver une configuration pour placer n reines sur un échiquier $n \times n$
- Pour trouver un mot de passe, on peut les vérifier tous jusqu'à trouver celui que l'on cherche.

Une recherche exhaustive n'est pas toujours une stratégie optimale !
Le temps de ce type d'algorithmes dépend du problème considéré.
Dans le pire cas, on doit énumérer l'ensemble $T(n)$ des objets de taille n ; en moyenne, on s'arrête après $\frac{\sum_{i \leq |T(n)|} i}{|T(n)|} \sim \frac{|T(n)|}{2}$.

Recherche exhaustive - en pratique

Lorsque ce n'est pas précisé, les algorithmes sont à écrire en Caml et en C

- Écrire une fonction recherchant un élément pair d'une liste (puis d'un vecteur)
- Écrire une fonction recherchant l'élément le plus grand d'un tableau
- Caml - Écrire une fonction prenant une liste de liste, en renvoyant `Some(l)` si `l` contient un 0 et `None` sinon
- Renvoyer la plus longue des listes d'une liste, parmi celles contenant 0. Renvoyer la liste vide si aucune n'a été trouvée.

```
let rec trouve0 = function
[]->false
|a::s-> (a=0)||(trouve0 s);;

let rec listeAvec0  = function
[]->None
|a::s-> if trouve0 a then Some(a) else listeAvec0 s;;

let rec listIt f liste x = match liste with
[]->x
|a::s-> f a (listIt f s x);;

let trouve0bis = function liste->(listIt (fun x y-> (x=0)||y) liste false);;
let listeAvec0bis =
    function liste->(listIt (fun x y-> if trouve0bis x then Some x else y) liste None);;


let listeAvec0Ter =
    function liste->
        (listIt (fun x y->
             if (function liste->(listIt (fun x y-> (x=0)||y) liste false)) x
                then Some x
                else y)
             liste None);;
```

Récursif ou impératif ?

Si l'on veut procéder par impératif, on peut utiliser des exceptions pour stopper la recherche au cours de l'énumération :

```
essayer :
    pour chaque élément:
        si élément trouvé :
            lever exception (trouvé(élément))
        lever exception (non trouvé (élément))
avec : trouvé(élément)-> renvoyer élément
```

Table of Contents

Backtracking

Définition

Les algorithmes de **backtracking** ou de **retour sur trace** énumèrent les solutions potentielles en sélectionnant une des variables du problème, en lui affectant l'une après l'autre les valeurs possibles, et pour chaque cas, en répondant récursivement au problème, avant de passer au suivant si aucune solution n'a été trouvée.

Backtracking

Définition

Les algorithmes de **backtracking** ou de **retour sur trace** énumèrent les solutions potentielles en sélectionnant une des variables du problème, en lui affectant l'une après l'autre les valeurs possibles, et pour chaque cas, en répondant récursivement au problème, avant de passer au suivant si aucune solution n'a été trouvée.

Le backtracking correspond à un parcours en profondeur sur l'arbre de décision du problème. Augmenter la hauteur du noeud visité, c'est améliorer une solution partielle dans l'optique de parvenir à une solution ; si une solution partielle ne peut être améliorée, l'algorithme l'abandonne, ainsi que le choix fait précédemment et *revient en arrière* pour envisager d'autres solutions partielles.
On peut améliorer un algorithme de backtracking si l'on dispose d'informations affirmant que dans un sous-arbre ne se trouve aucune solution : le parcours récursif de ce sous-arbre est ainsi écarté.

Backtracking

- Dans un arbre étiqueté, trouver, si c'est possible, une feuille étiquetée par un nombre pair.
- Étant donné un arbre étiqueté et une fonction f, renvoyer celle des feuilles dont l'image de l'étiquette par f est maximale.

Backtracking

- Dans un arbre étiqueté, trouver, si c'est possible, une feuille étiquetée par un nombre pair.
- Étant donné un arbre étiqueté et une fonction f, renvoyer celle des feuilles dont l'image de l'étiquette par f est maximale.

La structure de l'arbre colle à celle du backtracking, mais l'approche de backtracking peut être utilisée dans des contextes apparemment très différents. Le rapport aux arbres se fait en observant les arbres de décisions associés.

Backtracking

- Dans un arbre étiqueté, trouver, si c'est possible, une feuille étiquetée par un nombre pair.
- Étant donné un arbre étiqueté et une fonction f, renvoyer celle des feuilles dont l'image de l'étiquette par f est maximale.

La structure de l'arbre colle à celle du backtracking, mais l'approche de backtracking peut être utilisée dans des contextes apparemment très différents. Le rapport aux arbres se fait en observant les arbres de décisions associés.
Selon que l'on cherche un élément satisfaisant une propriété, l'ensemble de tels éléments ou un élément maximisant une fonction, l'approche diffère.

- existe-t-il un élément qui [...] ?

Backtracking

- Dans un arbre étiqueté, trouver, si c'est possible, une feuille étiquetée par un nombre pair.
- Étant donné un arbre étiqueté et une fonction f, renvoyer celle des feuilles dont l'image de l'étiquette par f est maximale.

La structure de l'arbre colle à celle du backtracking, mais l'approche de backtracking peut être utilisée dans des contextes apparemment très différents. Le rapport aux arbres se fait en observant les arbres de décisions associés.
Selon que l'on cherche un élément satisfaisant une propriété, l'ensemble de tels éléments ou un élément maximisant une fonction, l'approche diffère.

- existe-t-il un élément qui [...] ? utilisation du type `Option`

Backtracking

- Dans un arbre étiqueté, trouver, si c'est possible, une feuille étiquetée par un nombre pair.
- Étant donné un arbre étiqueté et une fonction f, renvoyer celle des feuilles dont l'image de l'étiquette par f est maximale.

La structure de l'arbre colle à celle du backtracking, mais l'approche de backtracking peut être utilisée dans des contextes apparemment très différents. Le rapport aux arbres se fait en observant les arbres de décisions associés.
Selon que l'on cherche un élément satisfaisant une propriété, l'ensemble de tels éléments ou un élément maximisant une fonction, l'approche diffère.

- existe-t-il un élément qui [...] ? utilisation du type `Option`
- donner l'ensemble des éléments qui [...]

Backtracking

- Dans un arbre étiqueté, trouver, si c'est possible, une feuille étiquetée par un nombre pair.
- Étant donné un arbre étiqueté et une fonction f, renvoyer celle des feuilles dont l'image de l'étiquette par f est maximale.

La structure de l'arbre colle à celle du backtracking, mais l'approche de backtracking peut être utilisée dans des contextes apparemment très différents. Le rapport aux arbres se fait en observant les arbres de décisions associés.
Selon que l'on cherche un élément satisfaisant une propriété, l'ensemble de tels éléments ou un élément maximisant une fonction, l'approche diffère.

- existe-t-il un élément qui [...] ? utilisation du type `Option`
- donner l'ensemble des éléments qui [...] : on concatène des listes de résultats en provenance des sous-arbres

Backtracking

- Dans un arbre étiqueté, trouver, si c'est possible, une feuille étiquetée par un nombre pair.
- Étant donné un arbre étiqueté et une fonction f, renvoyer celle des feuilles dont l'image de l'étiquette par f est maximale.

La structure de l'arbre colle à celle du backtracking, mais l'approche de backtracking peut être utilisée dans des contextes apparemment très différents. Le rapport aux arbres se fait en observant les arbres de décisions associés.
Selon que l'on cherche un élément satisfaisant une propriété, l'ensemble de tels éléments ou un élément maximisant une fonction, l'approche diffère.

- existe-t-il un élément qui [...] ? utilisation du type `Option`
- donner l'ensemble des éléments qui [...] : on concatène des listes de résultats en provenance des sous-arbres
- trouver le "meilleur" élément qui [...]

Backtracking

- Dans un arbre étiqueté, trouver, si c'est possible, une feuille étiquetée par un nombre pair.
- Étant donné un arbre étiqueté et une fonction f, renvoyer celle des feuilles dont l'image de l'étiquette par f est maximale.

La structure de l'arbre colle à celle du backtracking, mais l'approche de backtracking peut être utilisée dans des contextes apparemment très différents. Le rapport aux arbres se fait en observant les arbres de décisions associés.
Selon que l'on cherche un élément satisfaisant une propriété, l'ensemble de tels éléments ou un élément maximisant une fonction, l'approche diffère.

- existe-t-il un élément qui [...] ? utilisation du type `Option`
- donner l'ensemble des éléments qui [...] : on concatène des listes de résultats en provenance des sous-arbres
- trouver le "meilleur" élément qui [...] : utilisation du type `Option`, et comparaison des résultats des sous-arbres

TD : Labyrinthe

Sur une grille de hauteur n et de largeur n, on trouve des obstacles en certains points en coordonnées entières. Trouver un moyen (s'il existe) de rejoindre le point B à partir du point A.

TD : Labyrinthe

Sur une grille de hauteur n et de largeur n, on trouve des obstacles en certains points en coordonnées entières. Trouver un moyen (s'il existe) de rejoindre le point B à partir du point A.

- Comment est représenté en machine un tel "labyrinthe" ?
- Comment s'assurer que l'algorithme termine ?
- L'implémenter.

D'autres exemples ?

- SUBSET : On cherche dans un ensemble P inclus dans $\mathbb{N}$ un sous-ensemble de somme donnée.

D'autres exemples ?

- SUBSET : On cherche dans un ensemble P inclus dans $\mathbb{N}$ un sous-ensemble de somme donnée. L'ensemble des possibilités

D'autres exemples ?

- SUBSET : On cherche dans un ensemble P inclus dans $\mathbb{N}$ un sous-ensemble de somme donnée. L'ensemble des possibilités est de taille 2^n, avec n la taille de P.

D'autres exemples ?

- SUBSET : On cherche dans un ensemble P inclus dans $\mathbb{N}$ un sous-ensemble de somme donnée. L'ensemble des possibilités est de taille 2^n, avec n la taille de P.
- Pavages : on cherche à paver une partie du plan avec des tuiles, dont on dispose en autant d'exemplaires que l'on souhaite, qui appartiennent à un ensemble de modèles finis.

D'autres exemples ?

- SUBSET : On cherche dans un ensemble P inclus dans $\mathbb{N}$ un sous-ensemble de somme donnée. L'ensemble des possibilités est de taille 2^n, avec n la taille de P.
- Pavages : on cherche à paver une partie du plan avec des tuiles, dont on dispose en autant d'exemplaires que l'on souhaite, qui appartiennent à un ensemble de modèles finis.
- Un cavalier se déplace sur un échiquier sans jamais repasser par une cas qu'il a déjà visitée. Peut-il visiter l'ensemble des cases de l'échiquier ?