Cours 1 : Méthodes d'analyse des algorithmes - Illustration sur les algorithmes de tri

Jean-Stéphane Varré

Université Lille 1

jean-stephane.varre@lifl.fr

Au menu

- exemples introductifs
- notions de pire des cas et meilleur des cas
- les tris bulle, sélection, insertion
- comportement asymptotique
- l'insertion récursive et dichotomique
- résolution des équations de partition
- les tris fusion, rapide

? Question

Comment évaluer à l'avance le comportement d'un algorithme ?

- combien de temps va-t-il mettre à s'éxécuter ?
- combien d'espace mémoire va-t-il utiliser en plus des données initiales ?
- a-t-il toujours le même comportement ou bien existe-t-il des données plus ou moins favorables ?

Préliminaires

- un algorithme est une suite finie d'instructions qui résoud un problème
- un algorithme doit fonctionner sur tous les exemplaires du problème qu'il résoud
- la taille de l'exemplaire est formellement le nombre de bits mais on pourra réduire cette définition au nombre de composantes (cases d'un tableau)
- l'ensemble des exemplaires est le domaine de définition

Question

un algorithme A qui nécessite $10^{-4} \times 2^n$ instructions est-il moins performant qu'un algorithme B qui requiert $10^{-2} \times n^5$ instructions pour traiter une donnée de taille n ?

De combien de manières différentes peut-on rechercher la présence d'un élément dans un tableau ?

```
est_present : 'a array -> 'a -> bool
```


Présence d'un élément dans un tableau (v1)

```
1  let est_present_v1 t v =
2    let trouve = ref false in
3    for i = 0 to (Array.length t) - 1 do
4      if t.(i) = v then
5        trouve := true;
6    done;
7    !trouve
```

- temps d'exécution ?
- espace mémoire occupé ?

Présence d'un élément dans un tableau (v2)

```
1  let est_present_v2 t v =
2    let trouve = ref false
3    and i = ref 0 in
4    while !i < (Array.length t) && not !trouve do
5      if t.(!i) = v then
6        trouve := true;
7      i := !i + 1
8    done;
9    !trouve
```

- le nombre d'opérations depend-il
  - de la taille de l'exemplaire ?
  - de la « forme » de l'exemplaire ?

Présence d'un élément dans un tableau (v3)

```
1  (* version recursive *)
2  let rec est_present_v3 t v =
3    let n = Array.length t in
4    if n = 0 then
5      false
6    else
7      let t' = (Array.sub t 1 (n-1)) in
8      t.(0) = v || est_present_v3 t' v
```

- comment évaluer les algorithmes récursifs ?
- calcul du nombre de comparaisons
- calcul de l'espace supplémentaire utilisé

Présence d'un élément dans un tableau (v4)

on peut faire mieux en espace que la v3 : au lieu de recopier le tableau, on utilise une sous-fonction qui va gérer les bornes du tableau

```
1  (* version recursive avec espace memoire constant *)
2  let est_present_v4 t v =
3    let n = Array.length t in
4    (* p = position dans le tableau *)
5    let rec aux p =
6      if p = n then
7        false
8      else
9        t.(p) = v || aux (p+1)
10   in
11   aux 0
```