Recherche par dichotomie

Le problème

- Un ensemble d'éléments E
- ≤ : Un ordre total sur E
- = : Une fonction permettant de tester l'égalité entre deux éléments
- T un tableau (indicé de 1 à N) d'éléments triés dans l'ordre croissant
- Un élément e de notre ensemble E

- Question : l'élément e est-il présent dans le tableau T ?

Comprendre le problème

- Vous pouvez répéter la question ?

- Ecrire une fonction qui répond Vrai si et seulement si l’élément e est présent dans le tableau T.

Spécifications

- **Précondition :** Traduisons que le tableau T est trié entre les indices 1 et N dans l'ordre (noté ≤) croissant.
- **Trié (T, d, f) :** Si T est un tableau de valeur et d et f deux indices (tel que d ≤ f) de ce tableau, T *est trié entre les deux indices* d *et* f si et seulement si ∀ x, y deux indices, d ≤ x ≤ y ≤ f implique T[x] ≤ T[y].
- La précondition se traduit alors par **Trié (T,1,N)**.

- S'agissant d'une précondition ce prédicat est considéré comme vrai initialement pour toute exécution de l'algorithme

Spécifications

- **Post-condition :** Elle doit traduire de façon formelle, le lien entre la ou les données et le résultat attendu.

  Ce lien sera indépendant du processus de calcul.

  Nous allons poser un prédicat qui sera équivalent à la réponse que devra donner notre fonction.

- Soit T est un tableau de valeurs, d et f deux indices entiers ($d \leq f$) de ce tableau et e un élément

- **PredTrouv(T,d,f,e) :** $\exists$ i, $d \leq i \leq f$ tel que T[i] = e.

- Notre post condition devient donc **PredTrouv(T,1,N,e)**

Idées et justifications

- Plaçons nous à une étape intermédiaire et considérons que si e est présent dans le tableau c'est entre les deux indices d et f avec d ≤ f.
- Choisir un indice Ind compris entre d et f (au sens large).

1	**…**	**d**	**…**	**Ind**	**…**	**…**	**f**	**…**	**N**
Val1	…	Vald	…	ValInd	…	…	Valf	…	ValN

- Nous devons étudier 3 cas

Idées et justifications Cas 1 : T[Ind] = e

- Nous avons trouvé l'élément recherché la postcondition est satisfaite avec l'indice i = Ind. Il faut répondre Vrai.

1	**…**	**d**	**…**	**Ind**	**…**	**…**	**f**	**…**	**N**
Val1	…	Vald	…	e	…	…	Valf	…	ValN

Idées et justifications Cas 2 : T[Ind] < e

- Nous n'avons pas trouvé l'élément recherché. Par contre puisque le tableau est trié dans l'ordre croissant pour tout j, d≤j≤Ind, nous pouvons affirmé T[j] ≤ T[Ind]. Or puisque T[Ind] < e nous avons T[j] < e et donc T[j] ≠ e.
- Donc e est présent dans le tableau entre les indices d et f si et seulement si e est présent dans le tableau entre les indices Ind+1 et f

1	**…**	**d**	**…**	**Ind**	**…**	**…**	**f**	**…**	**N**
Val1	…	Vald	…	ValInd	…	…	Valf	…	ValN

Idées et justifications Cas 3 : e < T[Ind]

- Nous n'avons pas trouvé l'élément recherché. Par contre puisque le tableau est trié dans l'ordre croissant pour tout j, ind≤j≤f, nous pouvons affirmé T[Ind] ≤ T[j]. Or puisque e < T[Ind] nous avons : e < T[j] et donc T[j] ≠ e.
- Donc e est présent dans le tableau entre les indices d et f si et seulement si e est présent dans le tableau entre les indices d et Ind-1.

1	**…**	**d**	**…**	**Ind**	**…**	**…**	**f**	**…**	**N**
Val1	…	Vald	…	ValInd	…	…	Valf	…	ValN

Idées et justifications Choisir Ind

- Nous devons choisir la méthode de calcul de Ind. Pour ce faire nous devons avoir une réflexion sur le pire des cas.
- le pire des cas ne peut pas être lorsque T[Ind]=e car nous avons la réponse tout de suite
- Lorsque T[Ind]≠e il faut poursuivre la recherche dans l'un des deux sous tableaux d'indices d…Ind-1 ou Ind+1…f.

  En choisissant Ind = d + (f-d)/2, on a deux sous tableaux de taille sensiblement égale.

1	**…**	**d**	**…**	**Ind**	**…**	**…**	**f**	**…**	**N**
Val1	…	Vald	…	ValInd	…	…	Valf	…	ValN

Idées et justifications

- Quand s’arrêter ?
  - Quand on a trouvé la valeur e dans le tableau
  - Quand il n’est plus possible de choisir une valeur pour Ind donc quand f < d.

- Comment initialiser ? d = 1 et f = N sont des valeurs correctes qui permettent de chercher e dans tout le tableau.

Fonction RechDichoVersionItérative

- Données :
  - T un tableau d’éléments indicé de 1 à N // T est trié dans l’ordre croissant
  - e un élément
- Résultat : Booléen;
  - // RechDicho réponds vrai si et seulement si e est un élément du tableau T
- Variables : d,f,Ind trois indices de T

Fonction RechDichoVersionItérative

- DébutCode
  - d ←1;f←N; Trouvé ← Faux
  - Tantque (non trouvé) et (d ≤ f) faire
    - Ind ← d + ((f-d) div 2); Trouvé ← (T[Ind] = e);
    - Si non trouvé alors
      - Si T[Ind] < e alors d ← Ind +1
      - Sinon f ← Ind – 1
      - FinSi
    - FinSi
  - FinTq
  - Renvoyer(Trouvé)
- Fincode

Preuve : Partie 1 L'algorithme se termine

- **Argument 1 :** Si trouvé devient vrai alors l'algorithme se termine

- **Argument 2 :** Si trouvé reste faux, à chaque passage dans la boucle la valeur de f-d décroit strictement.

  En fait il est facile de montrer qu'après i passages dans la boucle si trouvé est faux alors f-d < N-i (car d ≤ Ind ≤ f).

  Puisque on sort de la boucle dès que d > f, l'algorithme sort de la boucle « tant que » en au plus N tours.

Preuve : Partie 2 le résultat donné est valide

- Cas 1 : La fonction répond Vrai.
  - Si la fonction répond Vrai c'est que l'algorithme est sorti de la boucle avec Trouvé = Vrai.
  - Si trouvé = Vrai alors T[Ind] = e.
  - Or Ind = d +((f-d) div 2) donc d ≤ Ind ≤ f.
  - Or, f ≤ N, d ≥ 1 en permanence et d ≤ f en début de boucle, donc 1 ≤ d ≤ f ≤ N lorsque l'on calcule Ind
  - En conséquence, Ind est un indice valide du tableau.
  - Donc, le prédicat PredTrouv(T, 1, N, e) est lui aussi vérifié puisque T[Ind] = e.

Preuve : Partie 2 le résultat donné est valide

- Cas 2 : La fonction répond Faux.
  - Il nous faut démontrer que la fonction répond faux à bon escient.
    Ce qui peut se formuler de plusieurs façons.
  - Pres(T,d,f,e) : e est présent dans T si et seulement si ∃ i, d ≤ i ≤ f tel que T[i] = e.
  - PasAvantd(T,d,e) : ∀ i, 0 < i < d ⇒ T[i] ≠ e
  - PasAprèsf(T,f,e) : ∀ i, f < i < N+1 ⇒ T[i] ≠ e
  - **Remarque :** PasAvantd(T,d,e) et PasAprèsf(T,f,e) avec d > f ≡ Pres(T,1,N,e) = faux

Fonction RechDichoVersionItérative + Repères

```
DébutCode
    d ← 1; f ← N; Trouvé ← Faux /*Pos1*/
    Tantque (non trouvé) et (d ≤ f) faire /*Pos2*/
        Ind ← d + ((f-d) div 2); Trouvé ← (T[Ind] = e);
        Si non trouvé alors
            Si T[Ind] < e alors /*Pos4*/ d ← Ind +1 /*Pos5*/
            Sinon /*Pos6*/ f ← Ind – 1 /*Pos7*/
            FinSi
        FinSi /*Pos8*/
    FinTq
    /*Pos9*/ Renvoyer(Trouvé)
Fincode
```

Preuve : PasAvantd(T,d,e)

- En Pos1, d=1 donc PasAvantd(T,d,e) est vrai car il n'y a pas de cases d'indice inférieur à 1
- Si après i tours de boucle PasAvantd(T,d,e) est vrai en Pos2 nous devons distinguer 3 cas :
  - Trouvé est vrai : le cas est déjà étudié
  - Trouvé est faux et T[Ind] < e puisque T et e restent inchangés, que T est trié (voir prédicat) on a ∀ j, 1 ≤ j ≤ Ind T[j] < e, ce qui implique que T[j] ≠ e.
    Donc en Pos4 on a PasAvantd(T,Ind+1,e) et en Pos5 PasAvantd(T,d,e).
  - Trouvé est faux et T[Ind] > e puisque d, T et e restent inchangés PasAvantd(T,d,e) est vrai en Pos6 et Pos7.

Preuve : PasAvantd(T,d,e)

- En Pos8, PasAvantd(T,d,e) est vrai car il est vrai pour chaque branche du Si.

- Si le test du « tant que » est satisfait on revient en Pos2 et PasAvantd(T,d,e) reste vrai après i+1 tours de boucle car on n’a pas changé les valeurs de T, d et e depuis Pos8.

- Dans le cas contraire (le test du tant que est faux) on arrive en Pos9 et PasAvantd(T,d,e) est vrai car on n’a pas changé les valeurs de T, d et e depuis Pos8.

Preuve : PasAprèsf(T,f,e)

- La preuve est laissé à votre charge car elle est similaire à la précédente.

Preuve : Bon résultat

- En Pos9, si La fonction répond Faux, on a PasAvantd(T,d,e) et PasAprèsf(T,f,e).

- Si on est sorti de la boucle avec Trouvé = Faux, on a f < d.

- Donc, PasAvantd(T,d,e) et PasAprèsf(T,f,e) avec d > f ≡ Pres(T,1,N,e) = faux : Il ne reste donc plus aucune cases du tableau T où nous serions susceptibles de trouver notre élément e.

- Donc, le prédicat PredTrouv(T,1,N,e) est faux et la fonction renvoie elle aussi faux comme résultat.

Complexité :

- Le pire des cas est celui où l'élément e n'est pas présent dans le tableau T.

- Alors on remarque qu'à chaque tour de boucle le nombre de cases entre d et f est divisé par 2 et on sort de la boucle dès que le nombre de cases est égal à 0.

- Ainsi si i est le nombre de tours de boucle, on peut affirmer que le nombre de case entre d et f sera
  - NbCase(0) = N /* après 0 tour de boucle */
  - NbCase(i+1) ≤ NbCase(i) div 2

Complexité :

- C’est bien gentil mais je n’ai pas d’intuition
  - NbCase(j) ≤ NbCase(j-1) div 2^1
  - NbCase(j) ≤ NbCase(j-1) div 2^1 ≤ NbCase(j-2) div 2^2
  - NbCase(j) ≤ NbCase(j-t) div 2^t (A démontrer par récurence)

  - Donc après t remplacements (donc t tours de boucle) le nombre de cas a été divisé par 2^t.
    Nous cherchons donc le plus petit k tel que N div 2^k = 0.
    C’est-à-dire quel est le plus petit k tel que N < 2^k.
    Donc, est le plus petit k tel que k > $\log_2(N)$.
    Donc, k = $\lfloor \log_2(N) \rfloor + 1$

- D’où une complexité en $\Theta(\log_2(N))$ dans le pire des cas.

Oui mais moi j’aime la récursivité

Fonction RecDicTete \\ Les spécifications sont identiques

Données

    Tableau T indicé de 1 à N

    Un élément e

Résultat Booléen

DédutCode

    Renvoyer(RecDicRec(T,1,N,e))

FinCode

Oui mais moi j’aime la récursivité

Fonction RecDicRec \\ Les idées sont identiques

Données

- Tableau T indicé de 1 à N
- d,f deux indices de T
- Un élément e

Résultat Booléen

Variable Ind Indice de T

Oui mais moi j'aime la récursivité

Fonction RecDicRec \\ La suite partie code

DédutCode

```
    Si d > f alors renvoyer(Faux) \\Ligne1
    Sinon Ind ← d+((f-d)div2) \\Ligne2
        Si T[Ind] = e alors renvoyer (Vrai) \\Ligne3
        Sinon
            Si T[Ind] < e alors renvoyer (RecDicRec(T, Ind+1, f,e)) \\Ligne4
            Sinon renvoyer (RecDicRec(T,d, Ind-1,e)) \\Ligne5
            Finsi
        Finsi
    Finsi
```

FinCode

Idées sous-jacentes au programme

- Si Le tableau ne contient aucune cases entre d et f, alors e n’est pas dans le tableau : C’est la ligne1
- Si le tableau contient des cases entre d et f alors on choisit un indice Ind entre d et f : C’est la Ligne2.

  Il nous reste 3 sous-cas à étudier
  - T[Ind] = e : Répondre Vrai, c’est la ligne3
  - T[Ind] < e : Si e est présent dans le tableau T entre les indices d et f ssi il est présent entre les indices Ind+1 et f. C’est la ligne4
  - T[Ind] > e : Si e est présent dans le tableau T entre les indices d et f ssi il est présent entre les indices d et Ind-1. C’est la ligne5

Preuves : 1 - ça se termine

- Si le tableau est vide ou si on a trouvé e, fin (Ligne1, Ligne3)

- Dans les autres cas il y a un appel récursif avec un nombre de cases strictement inférieur (c'est une récurrence sur le nombre de cases). En conséquence, la profondeur des appels récursifs est bornée par N.

- Donc, la fonction se termine

Preuves : 2 – Bon résultat

- Ecrivons proprement le prédicat :
- P(Nb) : pour tout tableau T et pour tout couple indice de ce tableau (d,f) tel que f-d = Nb-1, RecDicRec(T,d,f,e) renvoie un résultat équivalent au prédicat Pres(T,d,f,e).
- P(0) est satisfait car RecDicRec(T,d,f,e) renvoie Faux puisque f<d. De même Pres(T,d,f,e) est faux puisque il n'existe pas d'indice i tel que d≤i≤f.
- Supposons qu'il existe un entier naturel u tel que P(0), P(1), …, P(u) soient vrais. Démontrons que P(u+1) est vrai

Preuves : 2 – Bon résultat suite

- Supposons qu'il existe un entier naturel u tel que P(0), P(1), …, P(u) soient vrai. Démontrons que P(u+1) est vrai.
- Puisque u est un entier naturel, u+1 est strictement positif

  De plus, f-d = u+1-1 = u, donc d≤f.

  On calcule Ind qui vérifie d≤Ind≤f. Trois sous-cas :
  - T[Ind] = e : la fonction répond vraie, de même que le prédicat Pres(T,d,f,e)
  - T[Ind] < e : Puisque T est trié, Pres(T,d,f,e) ≡ Pres(T,Ind+1,f,e) et RecDicRec(T,d,f,e) renvoie le résultat de l'appel de RecDicRec(T,Ind+1,f,e).

    Or f-d = u et Ind = d + (u div 2).

    Donc f-(Ind+1) < u, c'est-à-dire f-(Ind+1) ≤ u-1 : il y a au plus u cases entre Ind+1 et f et par hypothèse de récurrence, RecDicRec(T,Ind+1,f,e) ≡ Pres(T,Ind+1,f,e))

    Donc, RecDicRec(T,d,f,e) ≡ Pres(T,d,f,e)).
  - T[Ind] > e : cas similaire au cas précédent. La preuve est à votre charge.

Complexité de RecDicRec

- Soit t le nombre de cases. Pire des cas inchangé : e n'est pas dans T.
- Soit CoûtRDR le coût de notre fonction en fonction du nombre de cases.
- CoûtRDR(0) = 5
- CoûtRDR(t) = 1*15 + CoûtRDR(t div 2^1) = 2*15 + CoûtRDR(t div 2^2)
  = 3*15 + CoûtRDR(t div 2^3) = 4*15 + CoûtRDR(t div 2^4)
  = v*15 + CoûtRDR(t div 2^v) = 15 log(t) + CoûtRDR(0)
  = 15 log(t) + 5

Fin du cours