Sujets d'oral technique

Version 1.18-SNAPSHOT

2018-08-22

Table of Contents

Table 1. History

Date	Author	Detail
2018-08-22	NeVraX	HTML Asciidoc to Github
2018-08-11	NeVraX	Anonymisation
2018-07-25	NeVraX182	Images relocation
2018-07-25	bcouetil	First (real) commit

1. Introduction

Lors d'un entretien d'embauche, de stage ou d'alternance, l'idéal est de déterminer, en un temps très limité, si le candidat :

1. **S'intègrera**
    - Il a un bon contact / état d'esprit / projet professionnel / posture, il correspond à la culture nvx
2. **A une tête bien pleine**
    - Il a des diplômes et compétences informatique directement utilisables sur les projets
3. **A une tête bien faite**
    - Est capable de résoudre les problèmes techniques quotidiens, sera autonome à moyen terme pour produire un code de qualité suffisante

Les points 1 et 2 sont assurés par l'entretien RH, et confirmé par un entretien opérationnel (non technique la plupart du temps).

L'on peut s'arrêter là, en considérant que le dernier point sera jugé en fin de période d'essai ou de stage. Mais il est possible d'éviter cette période incertaine, en tout cas réduire les chances de non transformation de l'essai.

C'est l'objet et l'ambition de cet oral technique : déterminer, en projection, l'autonomie technique du candidat à 1 an.

Il va sans dire que nvx Rennes, dans notre ambition de forte croissante à long terme, n'a pas vocation à trier pour ne prendre que les meilleurs ingénieurs : une productivité moyenne est suffisante. Les algorithmes sélectionnés doivent donc rester simples !

Exemple d'algorithme trop complexe : zig zag conversion

```java
public String convert(String s, int numRows) {
    if (numRows == 1)
        return s;

    StringBuilder sb = new StringBuilder();
    // step
    int step = 2 * numRows - 2;

    for (int i = 0; i < numRows; i++) {
        //first & last rows
        if (i == 0 || i == numRows - 1) {
            for (int j = i; j < s.length(); j = j + step) {
                sb.append(s.charAt(j));
            }
        //middle rows
        } else {
            int j = i;
            boolean flag = true;
            int step1 = 2 * (numRows - 1 - i);
            int step2 = step - step1;

            while (j < s.length()) {
                sb.append(s.charAt(j));
                if (flag)
                    j = j + step1;
                else
                    j = j + step2;
                flag = !flag;
            }
        }
    }
    return sb.toString();
}
```

2. Algorithme 1 : le palindrome

2.1. Définition

Un palindrome est un mot que l'on peut lire dans les deux sens. "kayak", "Laval", "radar", sont des palindromes.

2.2. Exercice

Ecrivez une fonction (réutilisable ailleurs dans un programme), permettant de vérifier si un mot donné est un palindrome ou non.

Vous pouvez utiliser la fonction **[String.charAt(int)]** qui récupère un charactère à une position donnée dans une chaîne.

```
"kayak".charAt(0) == "kayak".charAt(4);//true
```

Vous pouvez également utiliser la fonction **[String.substring(int,int)]** qui retourne une sous-partie d'une chaine en prenant l'index initial et la longueur.

```
String ay = "kayak".substring(1, 2)
```

2.3. Solutions

Algorithme classique

```
public static boolean isPalindrome(String word) {
    for (int i = 0; i <= sub.length() / 2; i++) {
        if (word.charAt(i) != word.charAt(sub.length() - 1 - i)) {
            return false;
        }
    }
    return true;
}
```

Algorithme récursif

```
public static boolean isPalindrome(String word) {
    if(word.length() == 0 || word.length() == 1) {
        return true;
    }
    if(word.charAt(0) == word.charAt(word.length()-1)) {
        return isPalindrome(word.substring(1, word.length()-1));
    }
}
```

Algorithme le plus concis

```
public static boolean isPalindrome(String word) {
    return new StringBuilder(word).reverse().equals(word);
}
```

2.4. Notation

Pour chaque point :

- 1/1 : en autonomie
- 0.5/1 : avec un peu d'aide ou partiel
- 0/1 : non réussi ou trop aidé

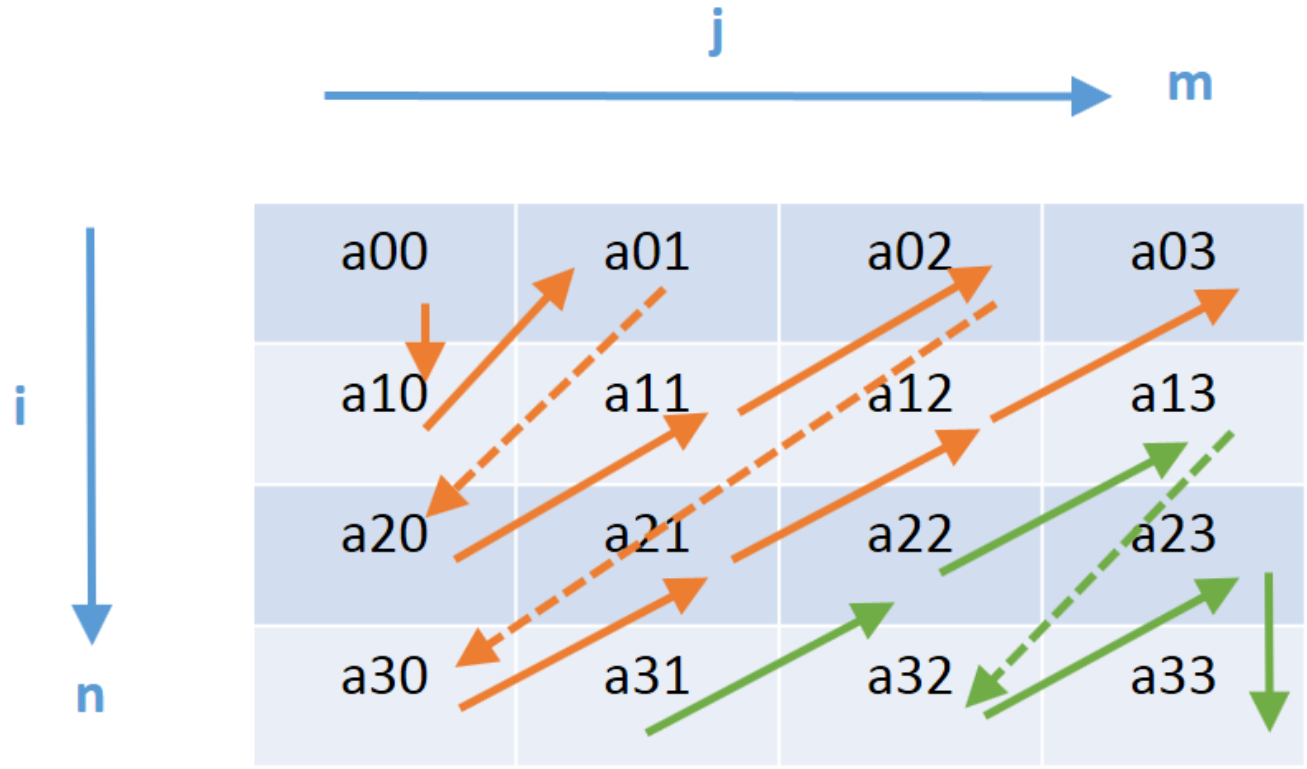
Table 2. Tableau de notations

Point	Note
Ecriture lisible	
Communication orale claire	
Signature : boolean isPalindrome(String mot)	
Boucle for() + if	
Parcours s'arrêtant à la moitié du mot (perf)	
Conditions aux limites vérifiées	
1 solution terminée parmi les 3	
…en moins de 15 minutes	
2 solutions terminées parmi les 3	
…en moins de 20 minutes	

3. Algorithme 2 : traversée de tableau en biais

3.1. Définition et exercice

Traversez un tableau de n lignes et m colonnes en biais et afficher le contenu des cases.



Affichage pour cet exemple

```
a00, a10, a01, a20, a11, a02, a30, a21, a12, a03, a31, a22, a13, a32, a23, a33
```

3.2. Solutions

Algorithme en 2 blocs

```
String[][] table = { { "a00", "a01", "a02", "a03" }, { "a10", "a11", "a12", "a13" },
        { "a20", "a21", "a22", "a23" }, { "a30", "a31", "a32", "a33" } };
int n = table.length;
int m = table[0].length;

int iAlternate;
for (int i = 0; i < n; i++) {
    iAlternate = i;
    for (int j = 0; j <= i; j++) {
        if (j < n) {
            System.out.print(table[iAlternate][j] + " ");
            iAlternate--;
        }
    }
}

int jAlternate;
for (int j = 1; j < m; j++) {
    jAlternate = j;
    for (int i = n - 1; i > 0; i--) {
        if (jAlternate < m)
            System.out.print(table[i][jAlternate] + " ");
        jAlternate++;
    }
}
// a00 a10 a01 a20 a11 a02 a30 a21 a12 a03 a31 a22 a13 a32 a23 a33
```

3.3. Notation

Pour chaque point :

- 1/1 : en autonomie
- 0.5/1 : avec un peu d'aide ou partiel
- 0/1 : non réussi ou trop aidé

Table 3. Tableau de notations

Point	Note
Ecriture lisible	
Communication orale claire	
Concept maitrisé (présentation a main levée)	
2 parties/boucles	
Boucle 1 : 2 for() 1 if	
Boucle 1 : conditions aux limites	
Moins de 15 minutes	
Boucle 1 : 2 for() 1 if	
Boucle 1 : conditions aux limites	
Moins de 20 minutes	

4. Algorithme 3 : arbre de recherche binaire

4.1. Définition

Un arbre de recherche binaire est un arbre ou la valeur de chaque noeud ("Node") est :

- plus grande ou égale à la valeur de tous les sous-nœuds de gauche
- plus petite ou égale à la valeur de tous les sous-nœuds de droite

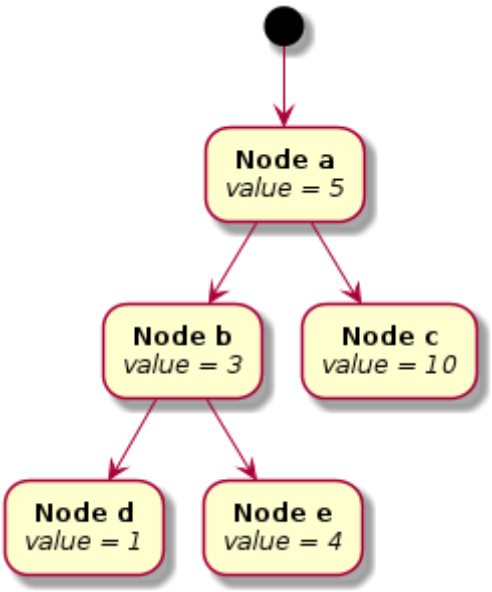


Figure 1. Arbre binaire

4.2. Exercice

Ecrire une fonction **[contains(node, value)]** qui vérifie si un certain arbre contient une certaine valeur. Avant de commencer, pour s'assurer de la compréhension de l'exercice, indiquer le résultat de **[contains(n2, 3)]** avec l'exemple suivant :

- n1 (Value: 2, Left: null, Right: null)
- n2 (Value: 4, Left: n1, Right: n3)
- n3 (Value: 6, Left: null, Right: null)

Node.java

```
class Node {
    public int value;
    public Node left;
    public Node right;

    public Node(int value, Node left, Node right) {
        this.value = value;
        this.left = left;
        this.right = right;
    }
}
```

4.3. Solutions

Algorithme basique

```
public static boolean contains(Node root, int value) {
    if (root == null) return false;
    if (root.value == value)
        return true;
    if (root.value > value)
        return contains(root.left, value);
    if (root.value < value)
        return contains(root.right, value);
    return false;
}
```

4.4. Notation

Pour chaque point :

- 1/1 : en autonomie
- 0.5/1 : avec un peu d'aide ou partiel
- 0/1 : non réussi

Table 4. Tableau de notations

Point	Note
Ecriture lisible	
Communication orale claire	
Exemple	
Signature de fonction	
Condition null	
Condition true	
Condition false	
Condition récursive droite	
Condition récursive gauche	
Moins de 10 minutes	

5. Algorithme 4 : deviner un chiffre

5.1. Exercice

Il s'agit d'un jeu de devinette.

Je m'appelle Ben. Vous me demandez de choisir un nombre entre 1 et n. J'indique pour chaque essai si c'est "plus" ou "moins". Il s'agit de créer une fonction de recherche qui simule votre comportement de joueur. Cette fonction s'arrête donc quand elle a trouvé le nombre que j'avais en tête depuis le début.

Dans cette fonction à créer, il faut appeler une fonction existante qui simule mon comportement :

```
int askBen(int num)
```

Elle me demande si un certain nombre est celui que j'avais en tête et retourne 3 valeurs possibles :

- -1 : Mon nombre est plus petit
- 1 : Mon nombre est plus grand
- 0 : Bravo ! vous avez deviné mon nombre

5.1.1. Exemple

Si n = 10, et que je choisis 6 (sans le dire), votre fonction retourne 6.

5.2. Solutions

Algorithme basique

```
public int guessNumber(int n) {
    int low=1;
    int high=n;

    while(low <= high){
        int mid = low+((high-low)/2);
        int result = askBen(mid);
        if(result==0){
            return mid;
        }else if(result==1){
            low = mid+1;
        }else{
            high=mid-1;
        }
    }
    return -1;
}
```

5.3. Notation

Pour chaque point :

- 1/1 : en autonomie
- 0.5/1 : avec un peu d'aide ou partiel
- 0/1 : non réussi

Table 5. Tableau de notations

Point	Note
Ecriture lisible	
Communication orale claire	
Exemple	
Signature de fonction	
Algo valide	
Cas aux limites	
Performance	
En moins de 20 minutes	
En Moins de 10 minutes	
Solution récursive	