

Arbre n-aire

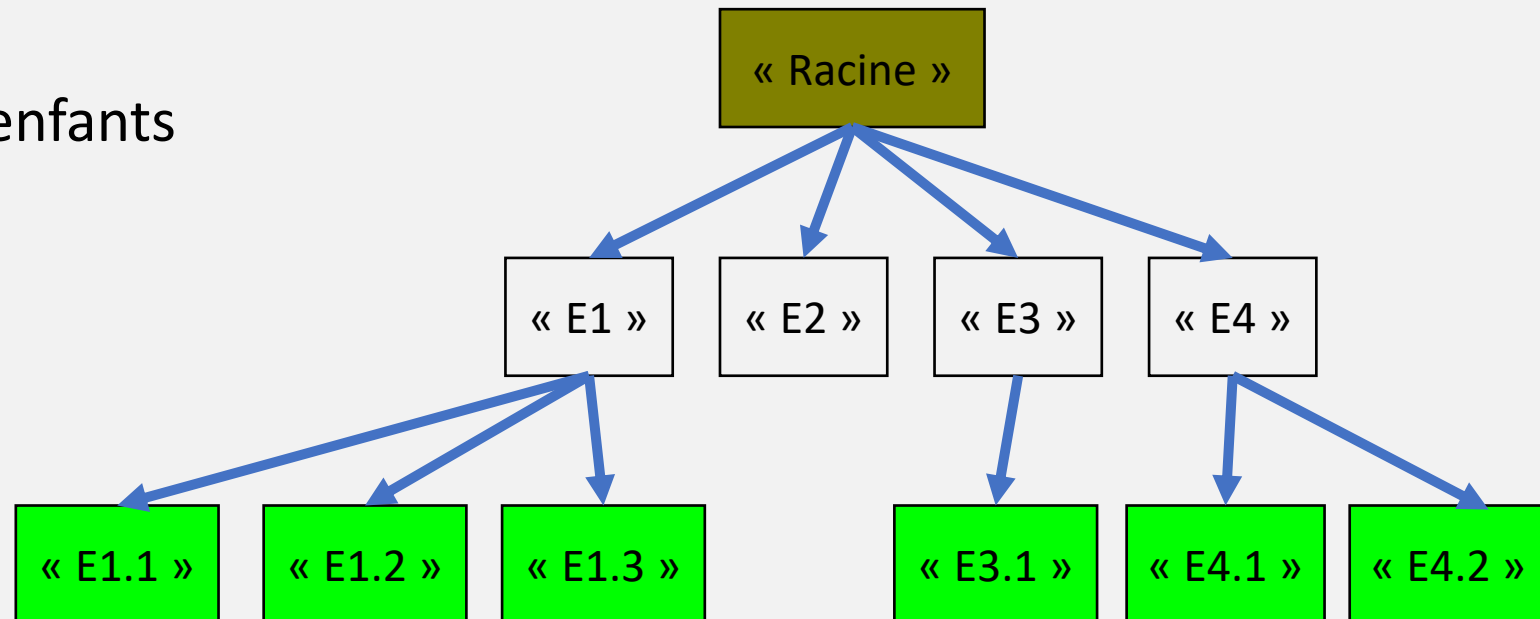


Objectifs

- Rappel de la définition et structure
- Application à l'auto-complétion

Arbre – Définition - Rappel

- Un arbre n-aire est un ensemble de nœuds liés par une relation de parenté
- Un nœud porte une valeur et ses liens vers ses enfants →
- Chaque nœud a un parent
- Chaque arbre a un nœud particulier qui n'a pas de parent.
 - Il est appelé **racine**
- Certains nœuds n'ont pas d'enfants
 - Ils sont appelés **feuilles**



Exercice 1 - Structure

- Écrivez la classe « **NoeudArbre** » qui permet de représenter les liens vers les différents enfants et qui permet de stocker une valeur d'un type paramétré
- Écrivez la classe « **Arbre** » qui contient simplement le nœud racine

Exercice 1 – Correction partielle

```
public class NoeudArbre<TypeDonnee> {  
    public NoeudArbre(TypeDonnee p_valeur) {  
        this.Enfants = new List<NoeudArbre<TypeDonnee>>();  
        this.Valeur = p_valeur;  
    }  
  
    public List<NoeudArbre<TypeDonnee>> Enfants { get; set; }  
    public TypeDonnee Valeur { get; set; }  
}
```

```
public class Arbre<TypeDonnee> {  
    public NoeudArbre<TypeDonnee> Racine { get; protected set; }  
  
}
```

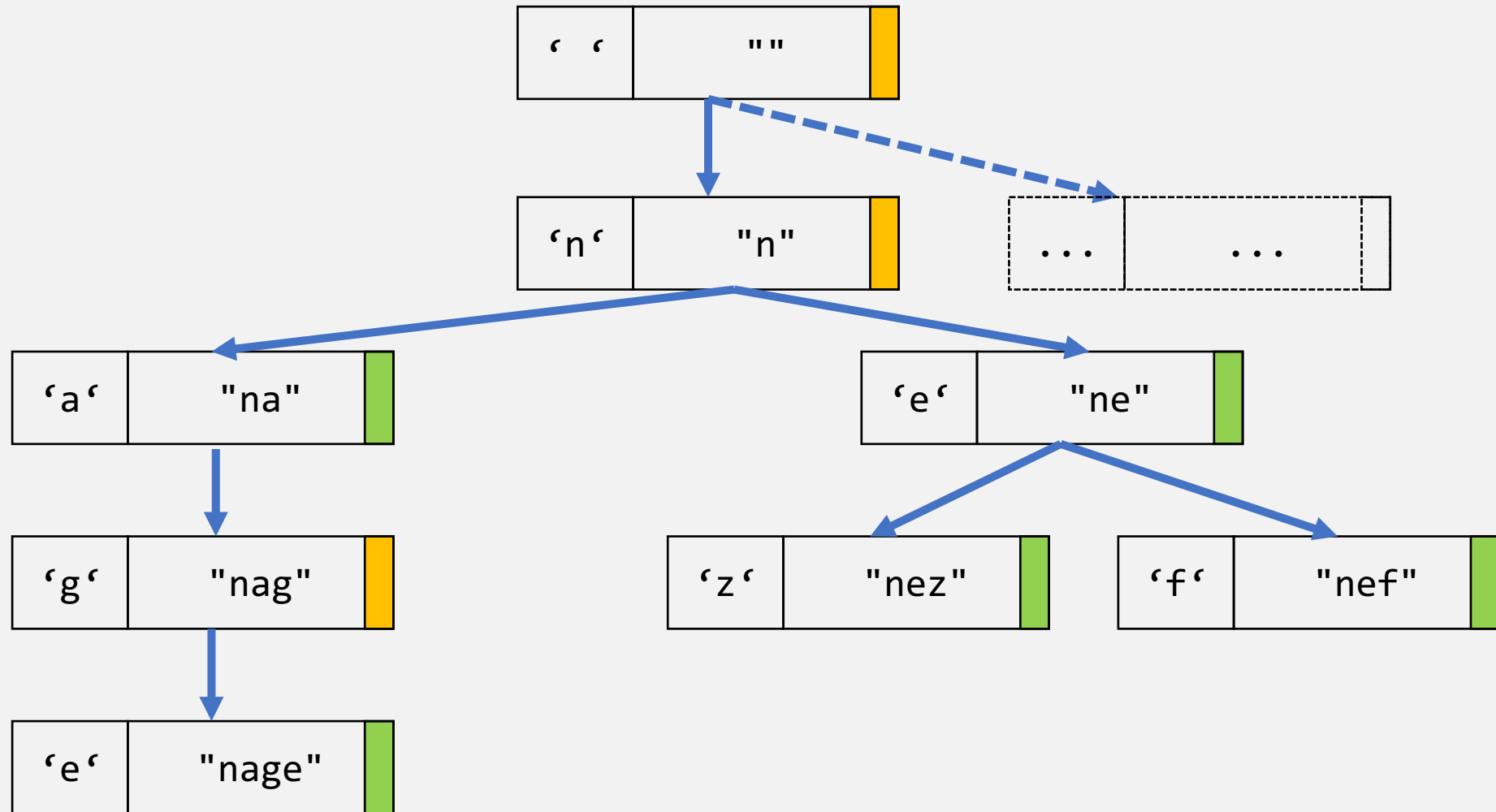
Application – Auto-complétion (Trie)



- Pour réaliser un algorithme d'auto-complétion, nous allons utiliser un arbre où :
 - Chaque nœud porte :
 - La valeur du préfixe qui correspond à la concaténation des lettres depuis la racine
 - La valeur de la lettre ajoutée au préfixe
 - Un indicateur permettant d'identifier si la valeur du préfixe est un mot valide
 - La liste des nœuds enfants
 - La racine est la chaîne vide, n'est pas un mot valide et la valeur de la lettre préfixe ne sera pas importante ici. Nous allons l'initialiser à « _ » (espace)

Application – auto-complétion : exemple

Mots :

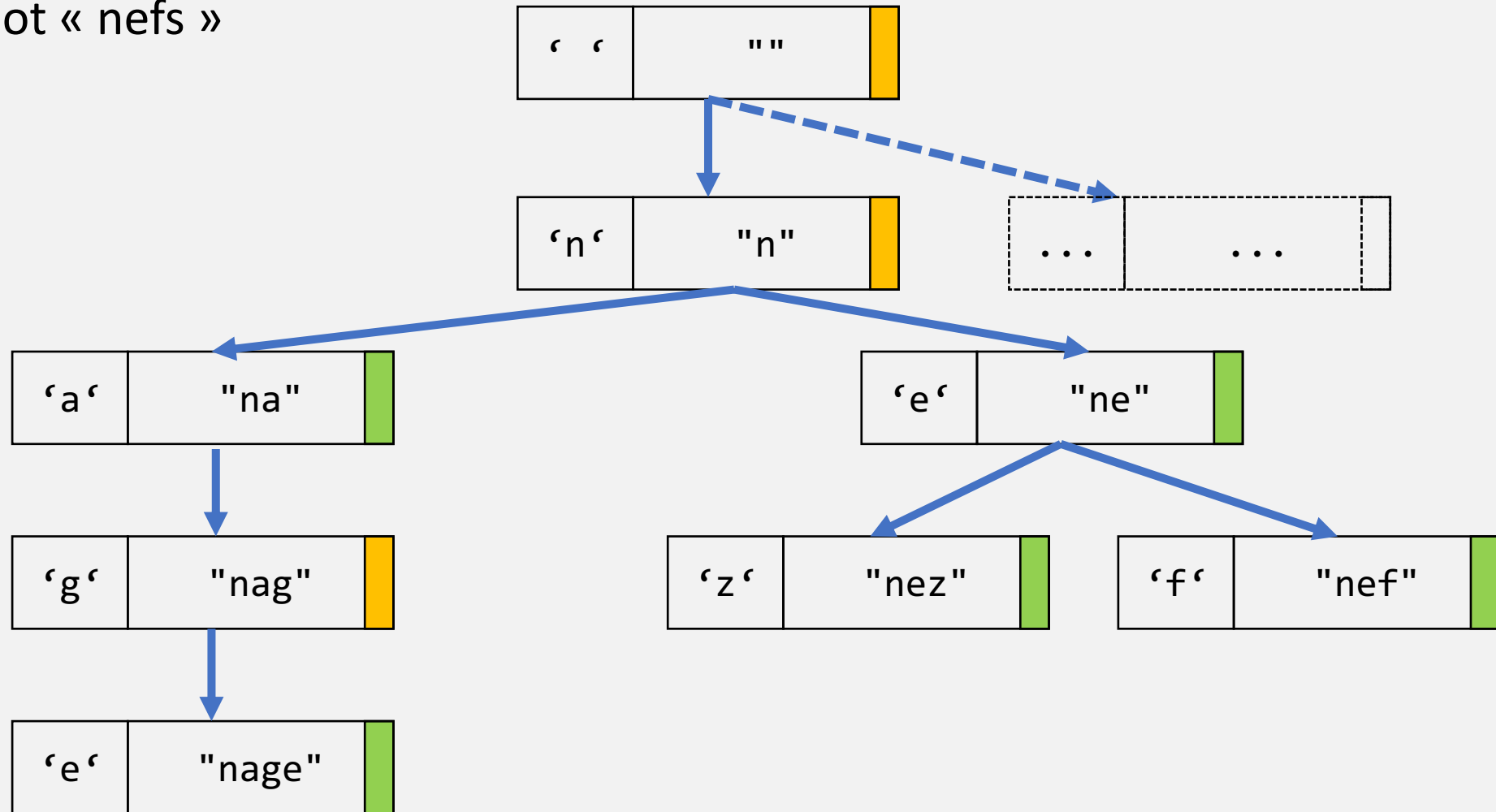
- na
- nage
- ne
- nez
- nef



 Mot invalide
 Mot valide

Application – auto-complétion : exemple

Ajout du mot « nefs »



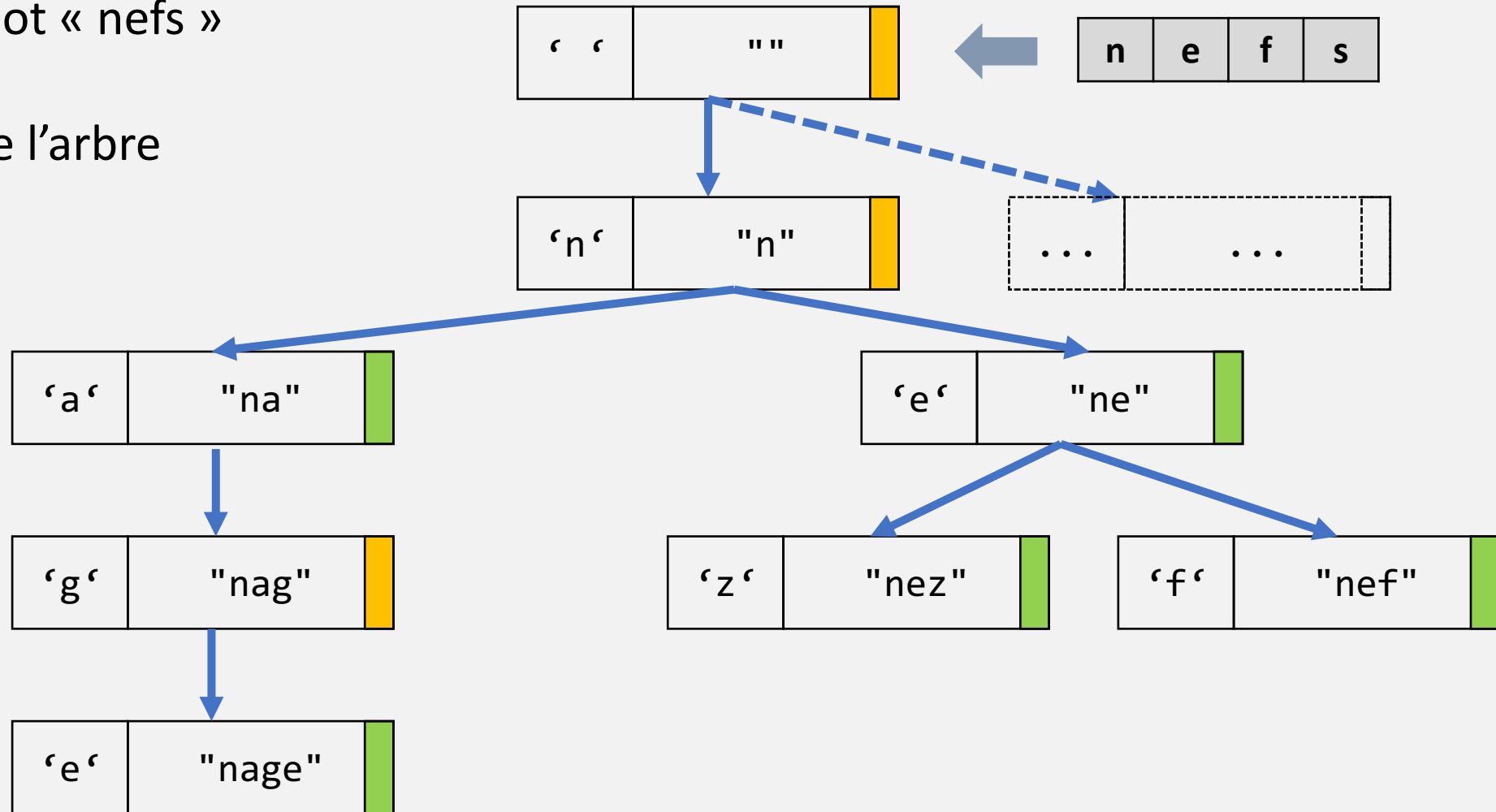
Mot invalide

Mot valide

Application – auto-complétion : exemple

Ajout du mot « nefs »

Parcours de l'arbre



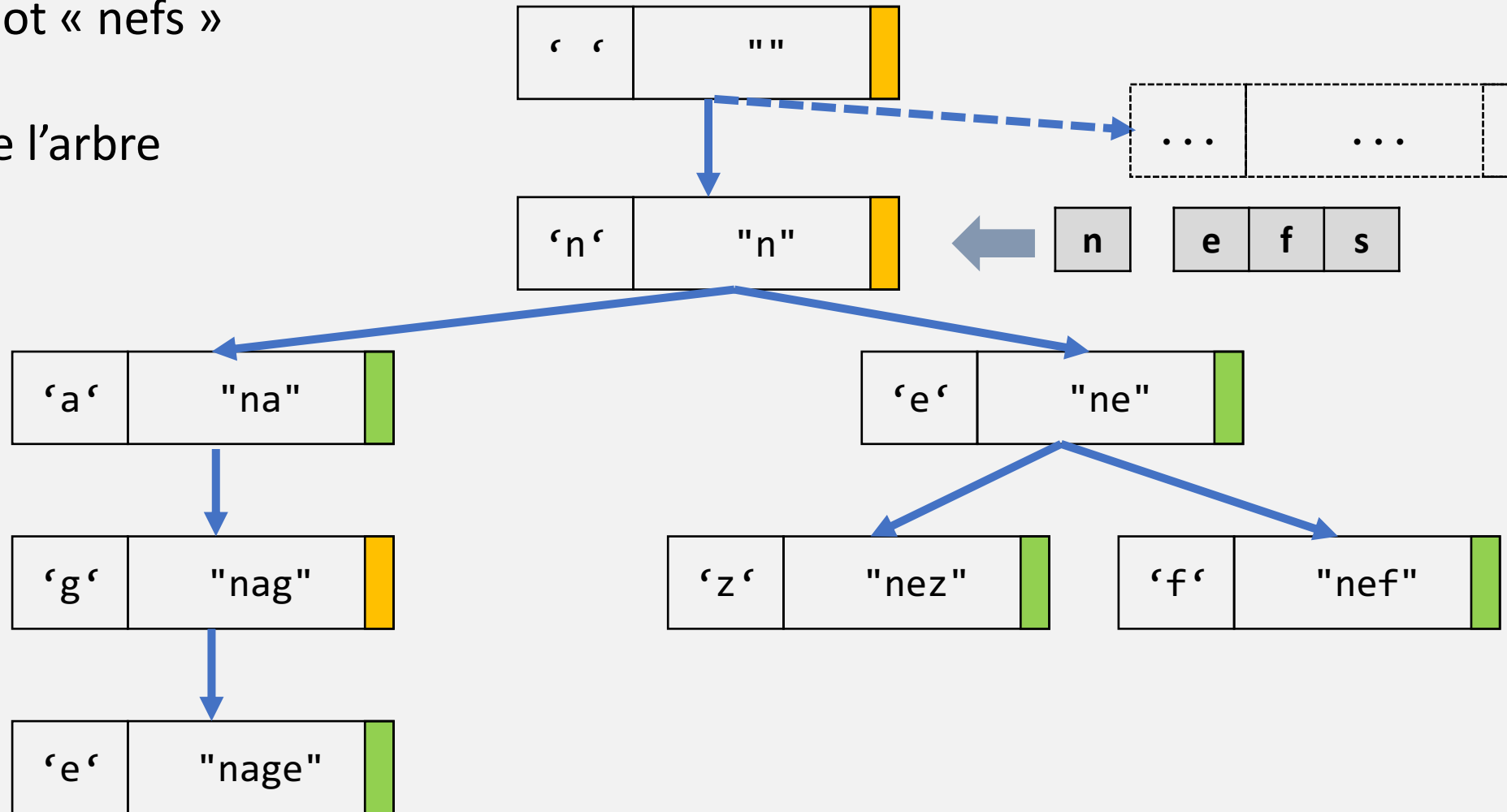
Mot invalide



Mot valide

Application – auto-complétion : exemple

Ajout du mot « nefs »

Parcours de l'arbre

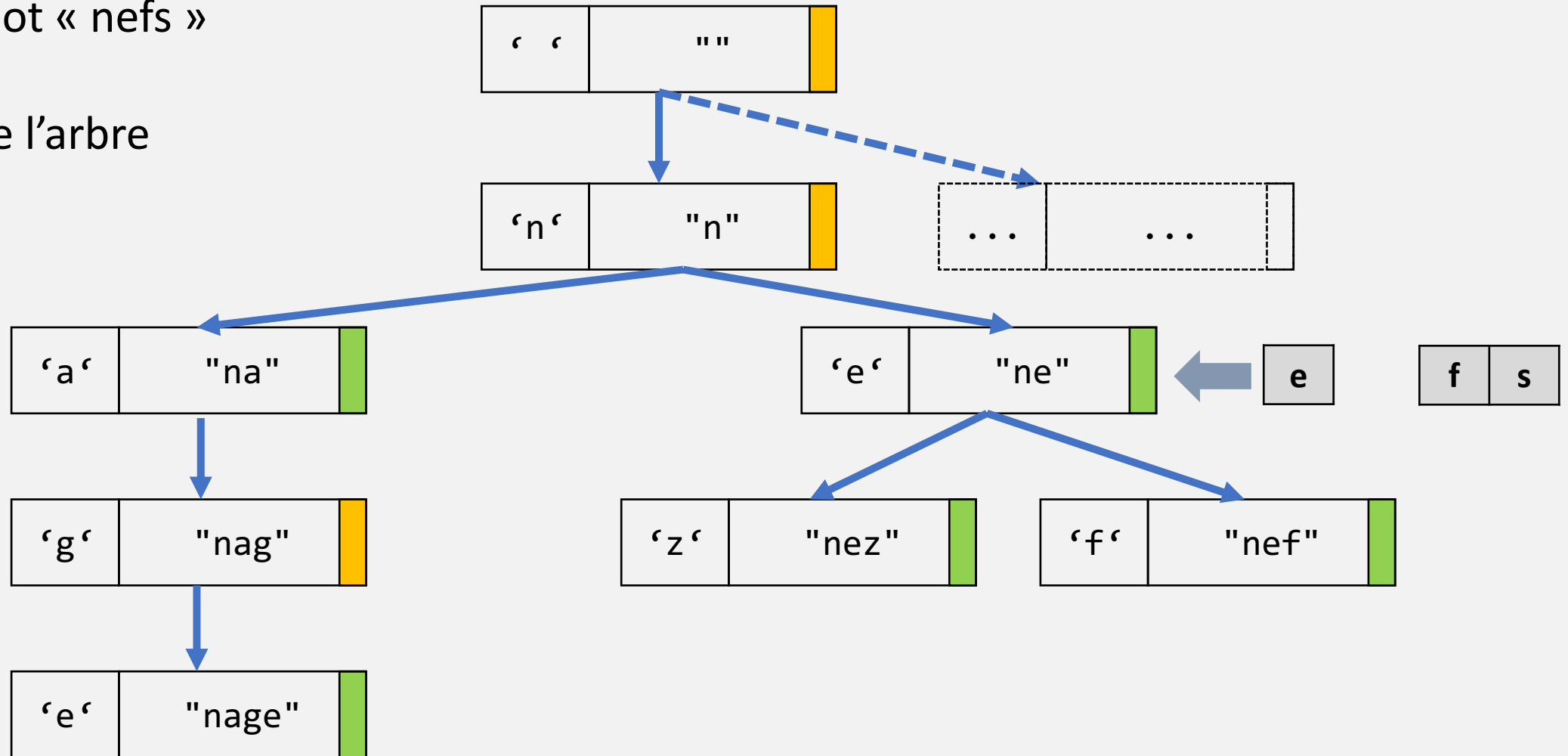


 Mot invalide
 Mot valide

Application – auto-complétion : exemple

Ajout du mot « nefs »

Parcours de l'arbre



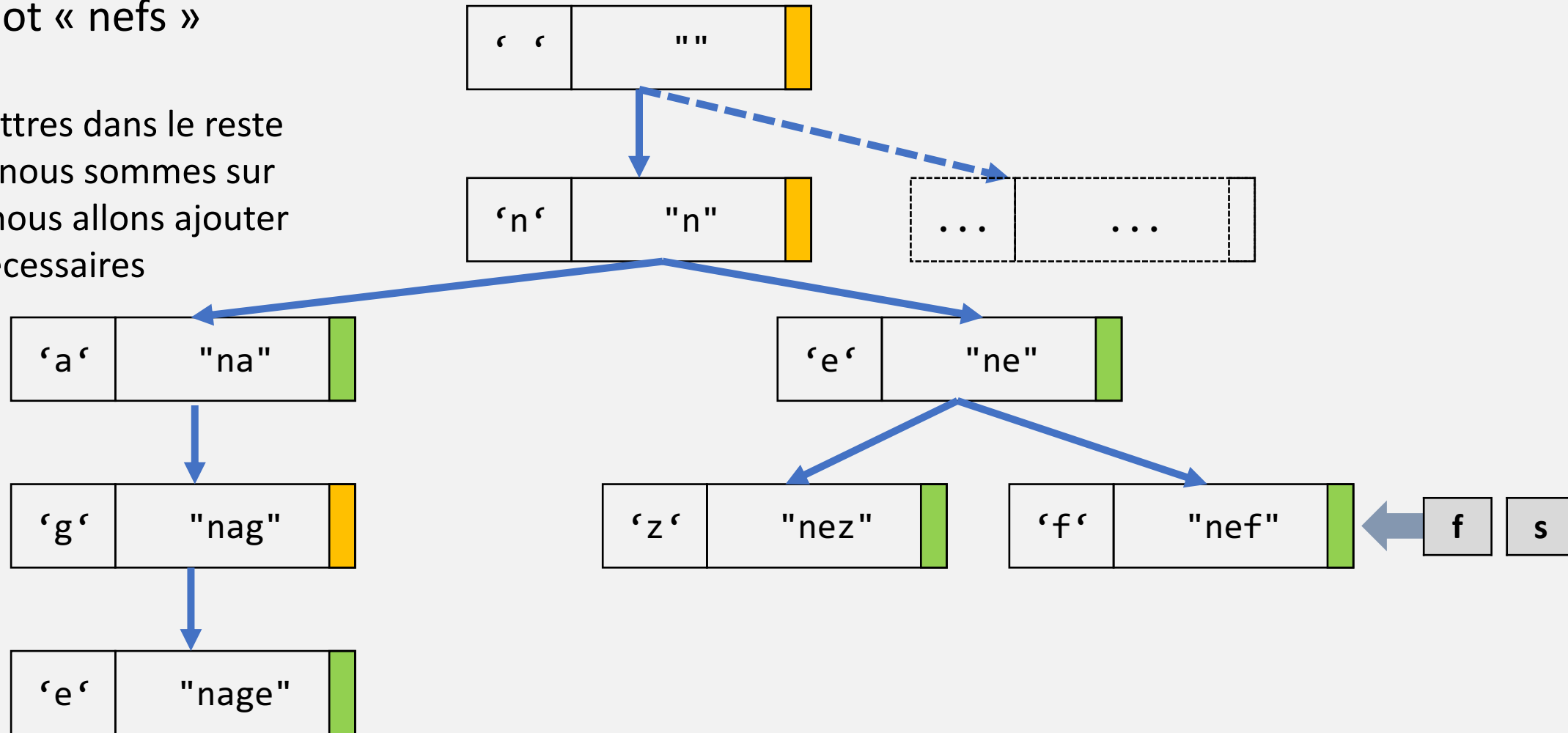
Mot invalide



Mot valide

Application – auto-complétion : exemple

Ajout du mot « nef »

Il reste des lettres dans le reste du mot mais nous sommes sur une feuille : nous allons ajouter les nœuds nécessaires

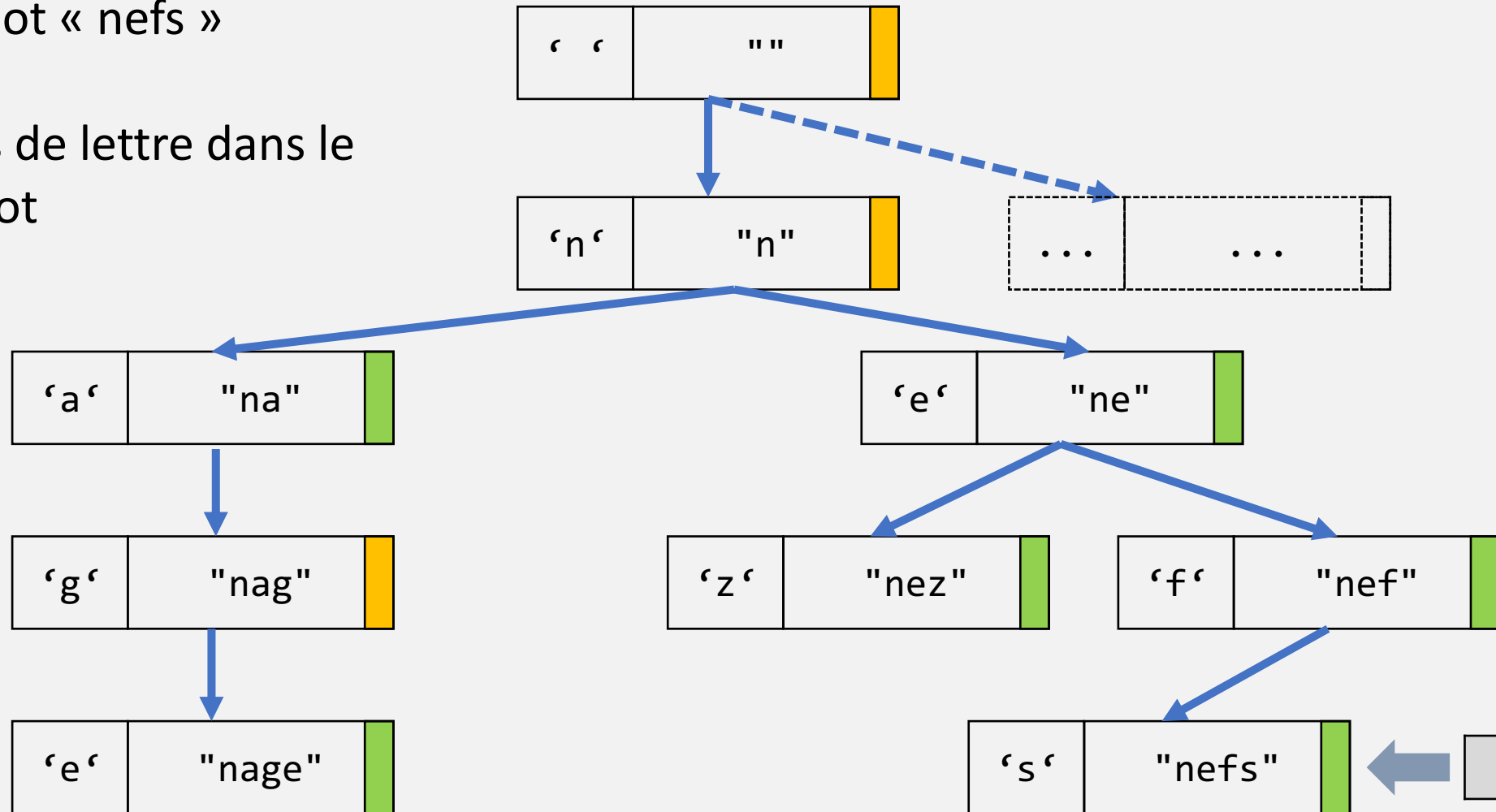




 Mot invalide
 Mot valide

Application – auto-complétion : exemple

Ajout du mot « nefs »

Il n'y a plus de lettre dans le reste du mot



 Mot invalide
 Mot valide

Exercice 2 – Structure arbre complétion

- Sur papier :
 - Déterminez la structure « **DonneeNoeudTrie** » qui permet de représenter l'ensemble des informations que nous avons besoin par nœud
 - À partir des structures précédentes, déterminez la classe « **ArbreAutoCompletion** » qui permet de représenter de tels arbres
 - Déterminez l'algorithme d'ajout d'un mot à partir de la racine
- Sur machine :
 - Implantez les classe « **DonneeNoeudTrie** » et « **ArbreAutoCompletion** »
 - Implantez l'algorithme d'ajout d'un mot dans la méthode « **AjouterMot** » dans une version publique qui ne prend que le mot à ajouter en paramètre et une privé qui prend tous les paramètres nécessaires à votre algorithme
 - À partir d'une fonction dans la classe « **Program** », essayez d'insérer des mots dedans pour tester votre méthode
 - Faites une fonction qui charge les mots du dictionnaire contenu dans le fichier donné avec le cours

Exercice 2 – Correction partielle

```
public class DonneeNoeudTrie {  
    public bool EstMotValide { get; set; }  
    public char Lettre { get; set; }  
    public string Prefixe { get; set; }  
  
    public DonneeNoeudTrie(char p_lettre, string p_prefixe, bool p_estMotValide) {  
        this.EstMotValide = p_estMotValide;  
        this.Lettre = p_lettre;  
        this.Prefixe = p_prefixe;  
    }  
}
```

```
public class ArbreAutoCompletion : Arbre<DonneeNoeudTrie> {  
    public ArbreAutoCompletion() {  
        this.Racine = new NoeudArbre<DonneeNoeudTrie>(new DonneeNoeudTrie(' ', "", false));  
    }  
}
```

Exercice 2 – Correction partielle

```
aucun AjouterMot_rec(chaine p_restMot, Noeud p_noeud) {
    caractère premiereLettre = p_restMot[0];
    chaine autresLettres = ChaineSansPremiereLettre(p_restMot);

    Noeud noeudDepart = TrouverNoeudEnfantPour(p_noeud, premiereLettre);

    SI (noeudDepart == null) {
        noeudDepart = créer Noeud(premiereLettre, p_noeud.Prefixe + premiereLettre,
                                   p_restMot.Taille == 1);

        // OU
        noeudDepart = créer Noeud(premiereLettre, p_noeud.Prefixe + premiereLettre,
                                   autresLettres.Taille == 0);

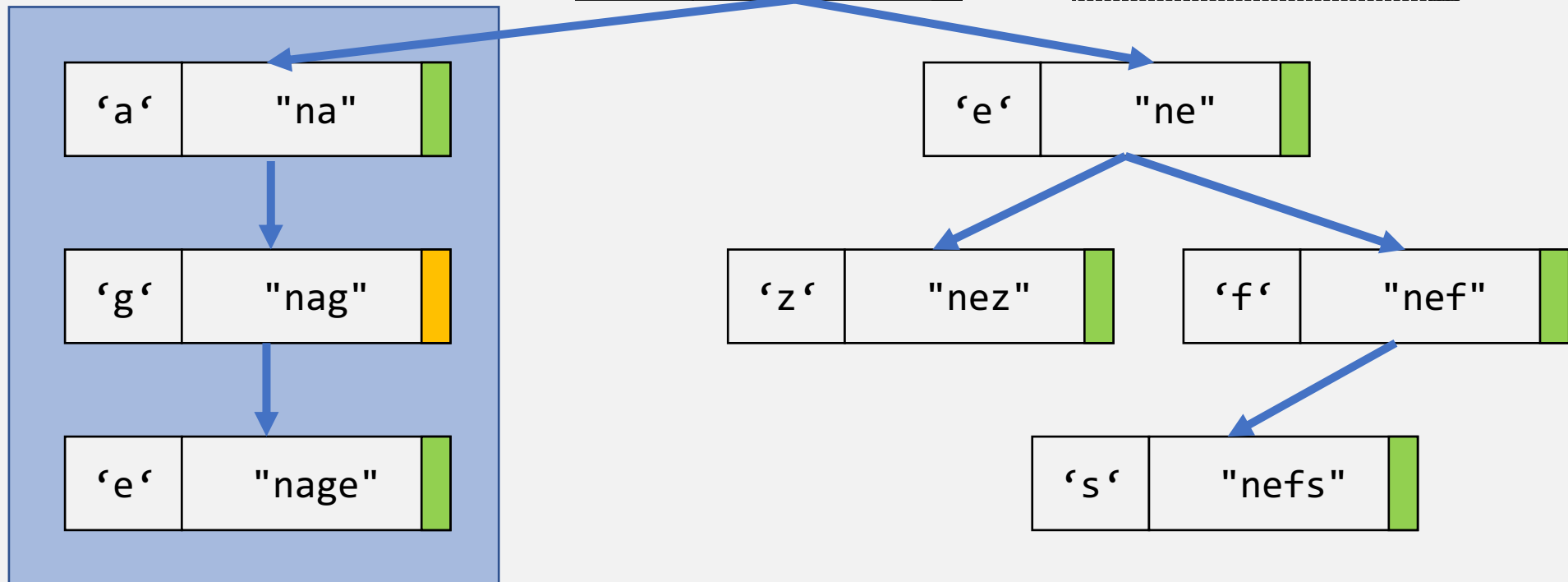
        p_noeud.Enfants.Ajouter(noeudDepart);
    }

    SI (p_restMot.Taille > 1) {
        AjouterMot_rec(autresLettres, noeudDepart);
    } SINON {
        noeudDepart.EstMotValide = vrai;
    }
}
```


Application – auto-complétion : exemple

Demande d'auto-complétion
avec "na" :

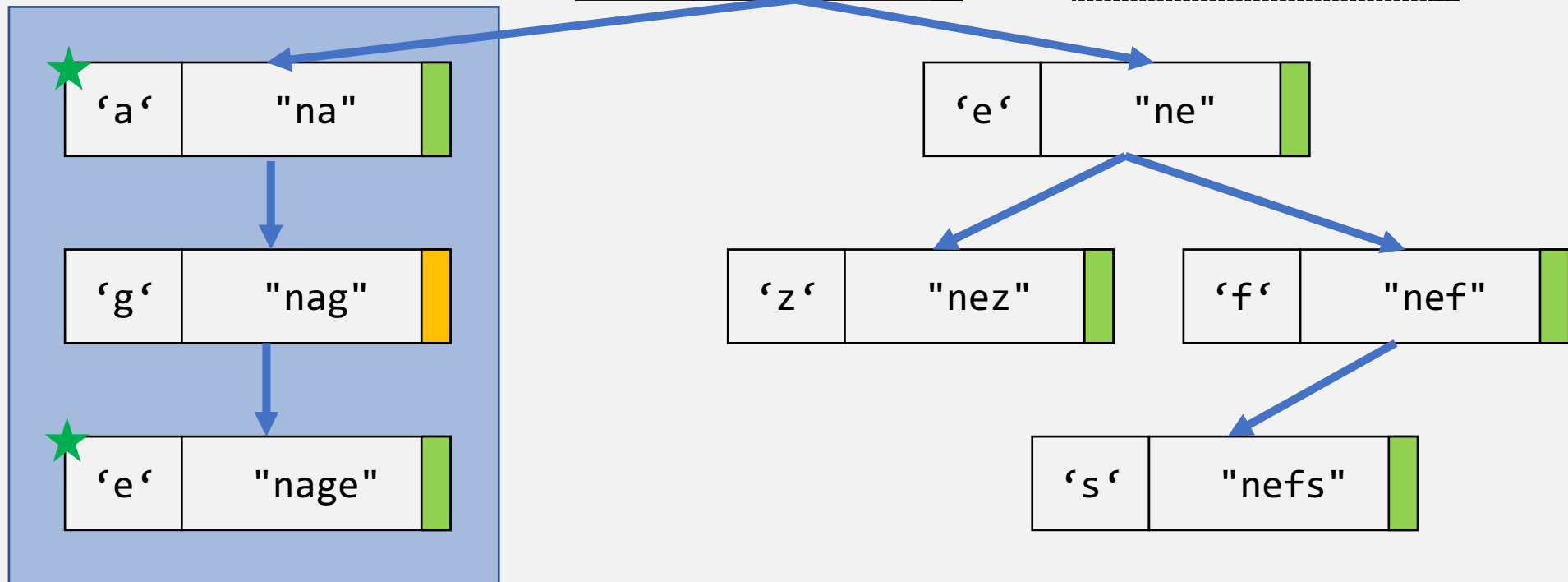
- Recherche du nœud "na »
(parcours ressemblant à l'ajout)



Application – auto-complétion : exemple

Demande d'auto-complétion
avec "na" :

Nous avons trouvé un nœud
"na". Il ne reste plus qu'à parcourir
ses fils pour extraire les mots valides



Exercice 3 – Compléter préfixe

- Sur papier :
 - Écrire l'algorithme « **CompleterPrefixe** » permettant, à partir de l'arbre et d'un mot, de trouver tous les mots possibles à partir de ce préfixe
- Sur machine :
 - Implantez cet algorithme au travers :
 - D'une méthode publique qui ne prend que le préfixe à compléter en paramètre et qui retourne la liste des mots possibles
 - Des méthodes privées qui prennent tous les paramètres nécessaires à votre algorithme. Conseils, faites au moins deux méthodes :
 - Une pour trouver le nœud (s'il existe) qui correspond au préfixe
 - Une autre pour collecter tous les mots valides à partir d'un nœud

Exercice 3 – Correction partielle

```
Liste<chaine> CompleterPrefixe(chaine p_prefixe) {  
    Noeud noeudPrefixe = RechercherNoeudPrefixe(p_prefixe, Racine);  
  
    SI (noeudPrefixe != null) {  
        renvoyer CollecterMots(noeudPrefixe);  
    }  
  
    renvoyer créer Liste<chaine>();  
}
```


Exercice 3 – Correction partielle

```
Liste<chaine> CollecterMots(Noeud p_noeud) {  
    Liste<chaine> resultat = créer Liste<chaine>();  
  
    SI (p_noeud.EstMotValide) {  
        resultat.Ajouter(p_noeud.ValeurPrefixe);  
    }  
  
    POUR CHAQUE Noeud enfant DE p_noeud.Enfants {  
        resultat = resultat.concat(CollecterMots(enfant));  
    }  
  
    renvoyer resultat;  
}
```