D'arbre en arbre.

1 Simulation "à la main"

1. Que fait le code ci-contre selon la valeur de n ? On supposera que l'appel initial est toujours fait avec le prefixe étant `""`. On précisera en particulier le nombre de lignes de texte affichées.
2. Même question si l'on supprime le (mot) `else` ?
3. Comment faire pour afficher l'ensemble des mots d'au plus 5 lettres (quelconques, qu'ils existent ou non) dans l'ordre de celui du dictionnaire ? dans l'ordre inverse ?

```
void mystere(int n, String prefixe){
  if(n>0){
    mystere(n-1, prefixe + "a");
    mystere(n-1, prefixe + "b");
    mystere(n-1, prefixe + "c");
  }
  else
    System.out.println(prefixe);
}
```

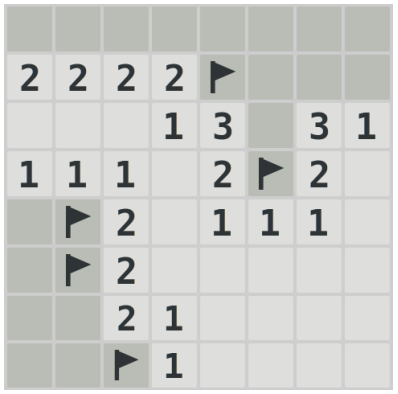
2 Propagation du démineur

Dans cette section, on s'intéresse au jeu du démineur. Il s'agit d'un jeu où l'on cherche des mines dans une grille. On peut cliquer droit sur une case pour la marquer d'une hypothétique mine, et gauche pour la révéler.
Lorsqu'on révèle une case, le jeu se comporte ainsi :

— Si la case contient une mine, on a perdu.

— Si elle est voisine d'une ou plusieurs mines, elle affiche le nombre de mines parmi les 8 cases voisines.

— Enfin, si elle n'est voisine d'aucune mine, la valeur 0 apparait (ou une case vide dans le schéma ci-contre). Alors, toutes les cases voisines sont automatiquement révélées, dévoilant leurs valeur. Si on dévoile une autre valeur 0, on révèle a leur tour toutes les cases voisines de cette nouvelle case, et ainsi de suite. On peut ainsi dévoiler d'un coup une grande partie du plateau.



Exemple de démineur

On s'intéresse à la méthode pour traiter le clic sur une case avec un 0, que l'on va appeler `revele`. On supposera l'existence d'une classe `Demineur` avec les méthodes `nbMinesVoisines(x,y)` qui retourne le nombre de mines dans les cases voisines, `affiche(x,y)` qui affiche la valeur d'une case et la marque comme révélée, `estRevelee(x,y)` qui vous indique si une case est déjà révélée ou non, et `estCaseValide(x,y)` qui indique si une case appartient bien au démineur.

Demineur
+estCaseValide(x, y) : bool +affiche(x, y) +nbMinesVoisines(x, y) : int +revele(x, y) +estRevelee(x, y) : bool

4. Comment faire pour que la fonction `revele` révèle toute la zone de 0 quand on révèle initialement un 0 ? Quelle méthode utiliseriez vous en récursif ? En itératif ?
5. Quel est le risque de boucle infinie ? Quelle stratégie adopter pour éviter cette boucle infinie ?
6. Écrivez le code de la méthode permettant de révéler toute une zone. On supposera que le cas où le joueur perd est traité en amont. [1]

1. Notez que la stratégie précédente vous permet de simplifier le parcours des cases voisines.

3 Lettres manquantes

Cet exercice s'appuie sur l'exercice *Lettres Manquantes* de la BattleDev de mars 2019 [2].

Étant donné un ensemble de mots, il faut trouver une séquence de lettres qui apparaissaient dans chacun des mots dans le même ordre. Par exemple, dans les mots `algorithme` et `artificiel`, la plus longue séquence possible est `arte` (ou de façon équivalente `arie`).

Dans l'énoncé de l'exercice, on pouvait lire :

> Indication : vous pouvez procéder par énumération exhaustive (force brute).

Prenons les deux mots `egal` et `aigle`. L'objectif est d'énumérer toutes les sous-suites de lettres de `egal` et de tester si elles sont dans cet ordre dans `aigle`.

7. Combien y a-t-il de sous-suites de lettres de `egal` ? Pour définir une sous-suite de lettres dans le mot égal, quels choix faut-il faire ? Représentez sous la forme d'un arbre binaire les sous-suites de lettres du mot égal. [3]
8. Comment générer récursivement toutes les sous-suites de lettres d'un mot ?
9. Comment utiliser cette énumération de tous les sous-mots pour répondre à la question de la BattleDev ?

4 Le jeu de la tablette de chocolat

On s'intéresse à un jeu à deux joueurs qui se joue avec une tablette de chocolat. La tablette est constituée de carreaux de chocolats formant un rectangle. Sur cette tablette, le carreau le plus en bas à gauche (de coordonnées $(0,0)$) a été piégé. Le but du jeu est de forcer l'adversaire à manger ce carreau.

À son tour, chaque joueur choisit un carreau de chocolat restant, et mange tous les carreaux en haut et à droite de ce carreau. Ainsi, s'il choisit un carreau de coordonnées (x, y), il mange tous les carreaux de coordonnées (x', y') avec $x' \geq x$ (à droite) et $y' \geq y$ (en haut).

Le joueur qui mange le carreau piégé (sans doute car c'est le seul carreau restant) perd la partie.

10. Sur une tablette de chocolat à une seule ligne ($1 \times n$ carreaux), comment le premier joueur s'assure-t-il la victoire ?
11. Fabienne et Eric jouent une partie sur une tablette de 2×2 carreaux. Fabienne commence. Sachant que tous deux sont de fins stratèges, qui va manger le carreau piégé ? Dessinez l'arbre des possibilités pour décrire la stratégie.
12. Même question sur une tablette de 2×3 carreaux.
13. Comment écririez vous un algorithme qui trouve le gagnant ? Peut-on optimiser un peu l'algorithme ?
14. Quelle est la stratégie optimale sur des tablettes de deux carreaux de large ?
15. Essayez-vous enfin à une tablette de 3×3 carreaux, voire plus grandes...

2. voir `https://www.isograd.com/FR/solutionconcours.php?contest_id=46&que_str_id=&reg_typ_id=2`
3. Attention, ce n'est pas du tout la même stratégie que l'exercice 1