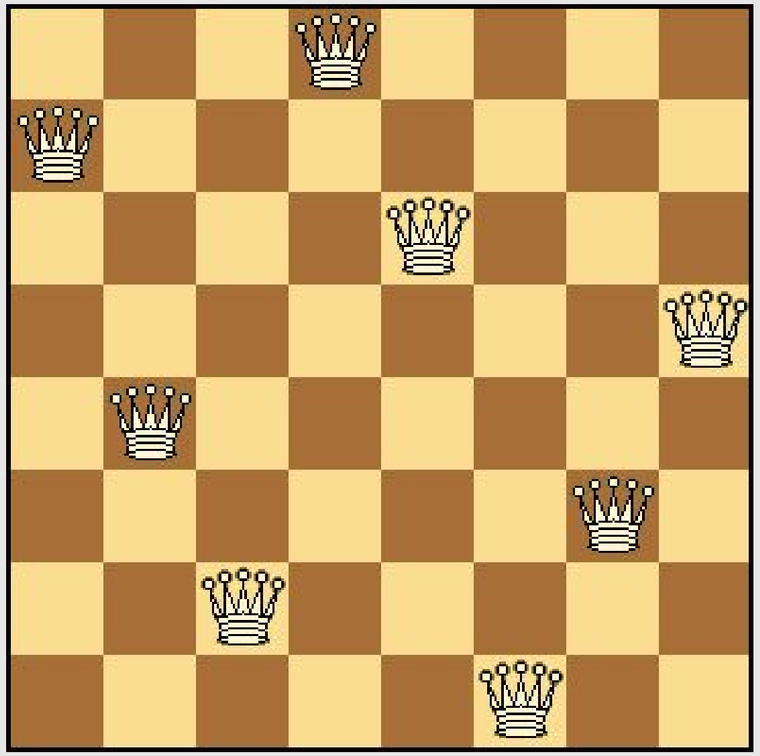
Le problème des huit reines

S2.02-EXPLORATION ALGORITHMIQUE

Plan

1. Présentation du problème
2. Solutions au problème
3. Analyse des solutions



EXEMPLE DE SOLUTION POUR UNE GRILLE DE 8*8

Présentation du problème

- placer *n* dames sur un échiquier de *n***n*
- chaque reine ne doit pas pouvoir manger les autres

Objectifs

- représenter la situation à l'aide d'un graphe
- créer un algorithme trouvant une solution au problème.
- créer un algorithme trouvant le nombre de solutions possibles pour un échiquier de taille n*n.

Représentation du problème

```
[1, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0]
[0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0, 0]
```

REPRÉSENTATION DE L'ÉCHIQUIER

1.L'échiquier

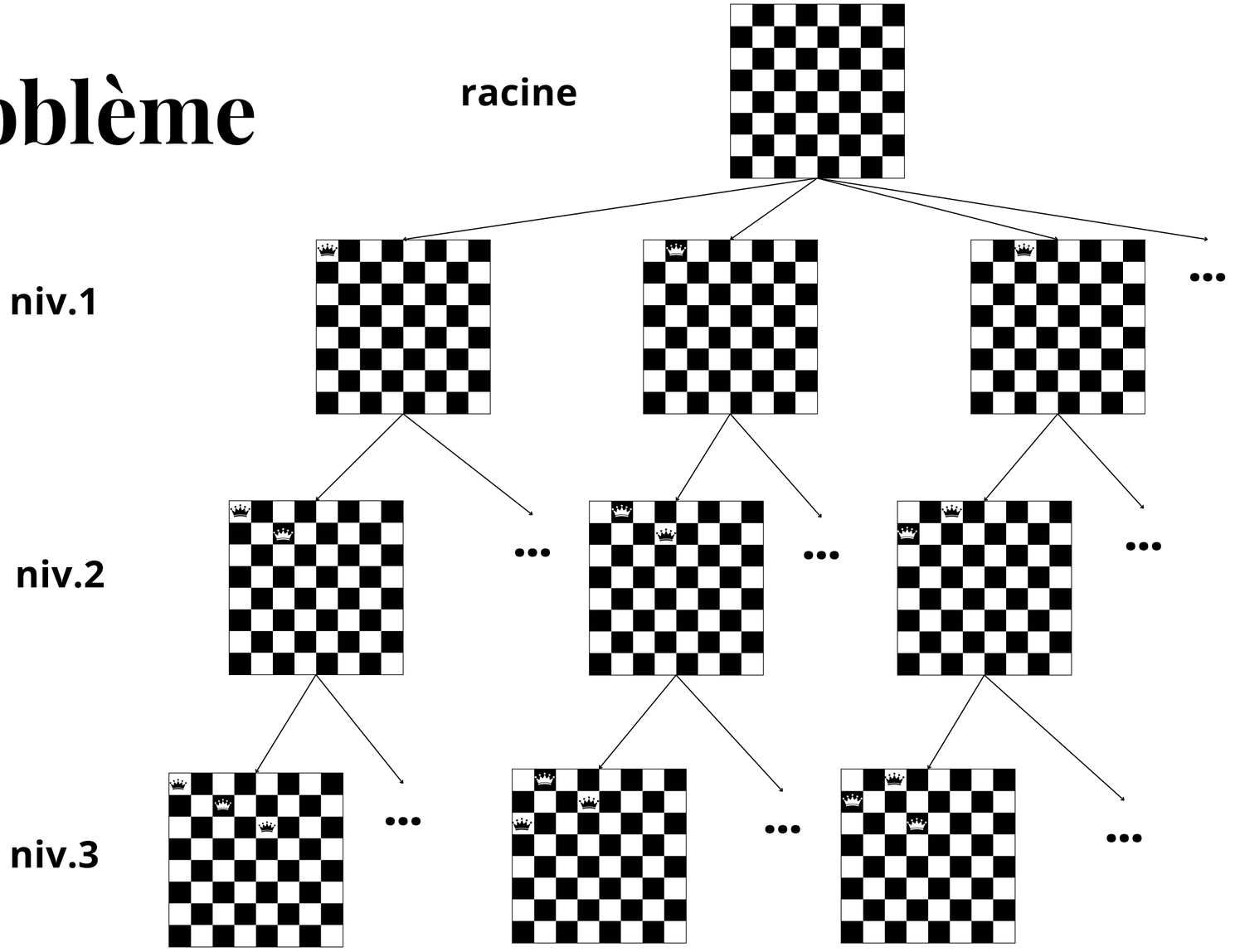
Classe Echiquier (self, grille=None)

- variables
  - self.n
  - self.grille
  - self.layout
- méthodes
  - affichergrille(self)
  - casePossible(self, l, c)

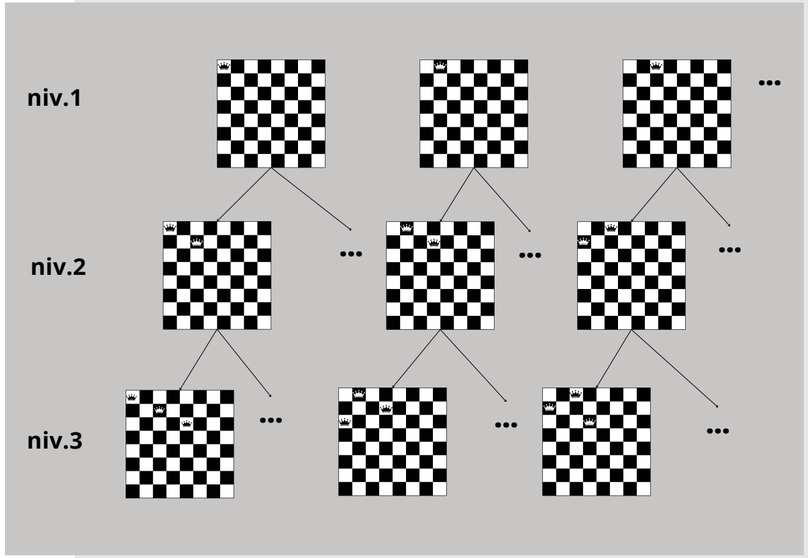
Représentation du problème

2.Le graphe

- Chaque sommet est un échiquier ayant une ou plusieurs reines
- Tous les enfants du sommet sont les échiquiers où l'on peut ajouter une reine
- Un niveau n de l'arbre contient donc tous les échiquiers ayant des reines sur les n premières lignes



Représentation du problème



2.Le graphe

Classe Sommet (self, e : Echiquier)

- self.echiquier
- self.enfants
- trouverEnfants(self, ligne)

Classe Arbre(self, profondeur)

- self.profondeur
- self.niveaux
- remplir(self, display=False)

Solutions au problème

le backtracking

```python
def backtrackingGrille(e : cl.Echiquier, l, lReine1 = None):
    """algorithme de backtracking"""
    if(l == lReine1):
        return backtrackingGrille(e, l+1, lReine1)
    if (l >= e.n):
        return True
    else:
        for c in range(e.n):
            if(e.casePossible(l, c)):
                e.grille[l][c] = 1

                if(backtrackingGrille(e, l+1, lReine1)):
                    return True
                else:
                    e.grille[l][c] = 0
        return False
```

Solutions au problème

1.le backtracking

Chaque fois qu'une solution est trouvée, on vérifie si elle est unique en calculant un hachage de sa configuration. Si la solution est unique on l'ajoute à l’ensemble de solutions.

2.le graphe

On trouve toutes les solutions au dernier niveau de l’arbre. On peut ensuite les filtrer pour trouver celle ayant la reine imposée.

3.Le bruteforce

On choisit la position imposée de la 1ere reine. Ensuite le programme va essayer tous les cas possibles et si il n’y a aucun conflit alors le nombre de solutions augmente de 1.

Analyse des solutions

1.le backtracking

- **Avantages** : le plus efficace pour trouver une solution
- **Inconvénients** : temps d'exécution long pour trouver toutes les possibilités

2.le graphe

- **Avantages** : besoin de créer le graphe une seule fois
- **Inconvénients** : concept compliqué à comprendre, long s'il n'y a besoin que d'une solution

3.Le bruteforce

- **Avantages** : principe simple, fonctionne toujours
- **Inconvénients** : temps d'exécution très long