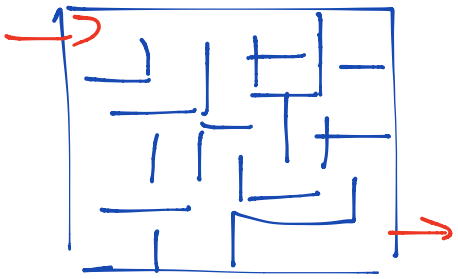
Grande famille d'algo. BACKTRACKING

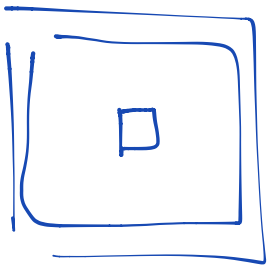
idée: recherche exhaustive dans un espace de recherche (souvent très grand).

- recherche systématique.
- exhaustive : on explore tout.
- éviter la redondance.

) chaque configuration sera examinée 1 et 1 seule fois.

Recherche d'un chemin dans un labyrinthe





Algorithme "brute force".

Là on cherche partout.

Schéma général:

On va chercher des solutions de la forme $(a_1, a_2, \dots, a_n)$

La séquence de choix, une seq. de coups, ...

Exemples:

- Sudoku: un choix = 1 chiffre ds une case.
- lister tous les sous-ensemble d'un ens: $a_i = \top/\bot$ indiquant si l'élt i est dans le ss-ens. ou non.
- chemin dans 1 graphe: un "a_i" sera une transition/arc/arête.

A chaque étape de l'algo, on fait:

- on cherche comment compléter la solution partielle dont on dispose à l'instant.
- on les essaie tous (ou jusqu'à trouver 1 solution).

Fondamentalement, on a un arbre que l'on va parcourir.

une solution partielle = la config. courante de mon pb.

```
BT (S) :
    Si S est une solution à mon pb de départ → bingo!
    Sinon :
        - voir les possibilités L pour continuer.
        - Pour tout p ∈ L :
              BT(S ∪ {p})
```

Lister tous les sous-ensembles d'un ens. de départ.

input : un tableau d'entiers (tous différents).
↳ taille n, nom : E. indices 0 ... n-1

la solution partielle en construction

```
TLS (E, S, k)        → n° choix.
    Si (k==n) Alors afficher le ss-ens. S → "complet"
                    ↳ pour chaque élément de E, j'ai fait un choix.
    Sinon :
        j'ajoute E[k] à S.
        TLS (E, S, k+1)
        j'enlève E[k] de S
        TLS (E, S, k+1)
```

Appel initial avec
TLS (E, ∅, 0)