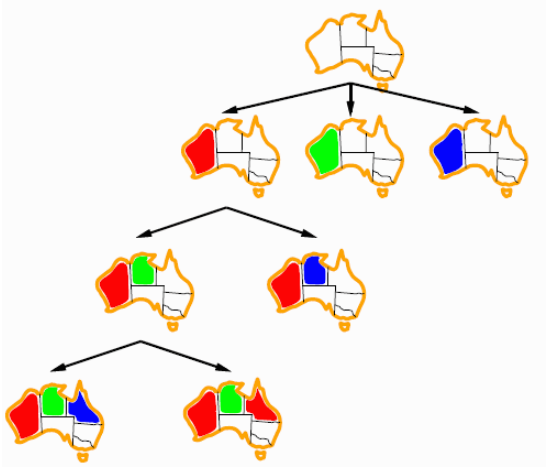
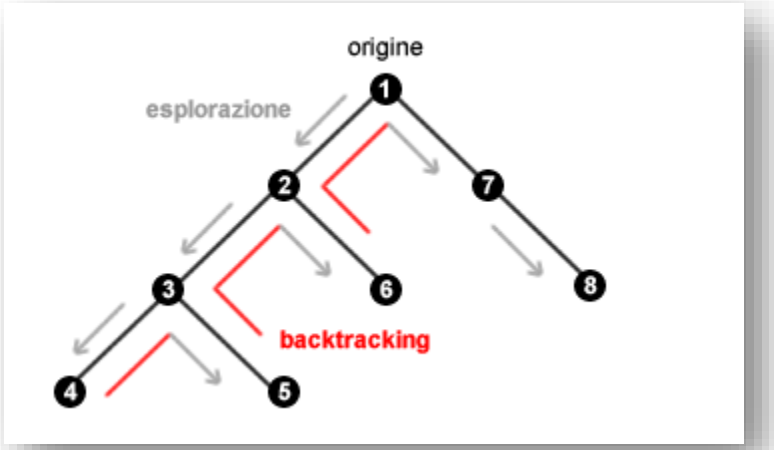
UNIVERSITÀ
DI PARMA

backtracking

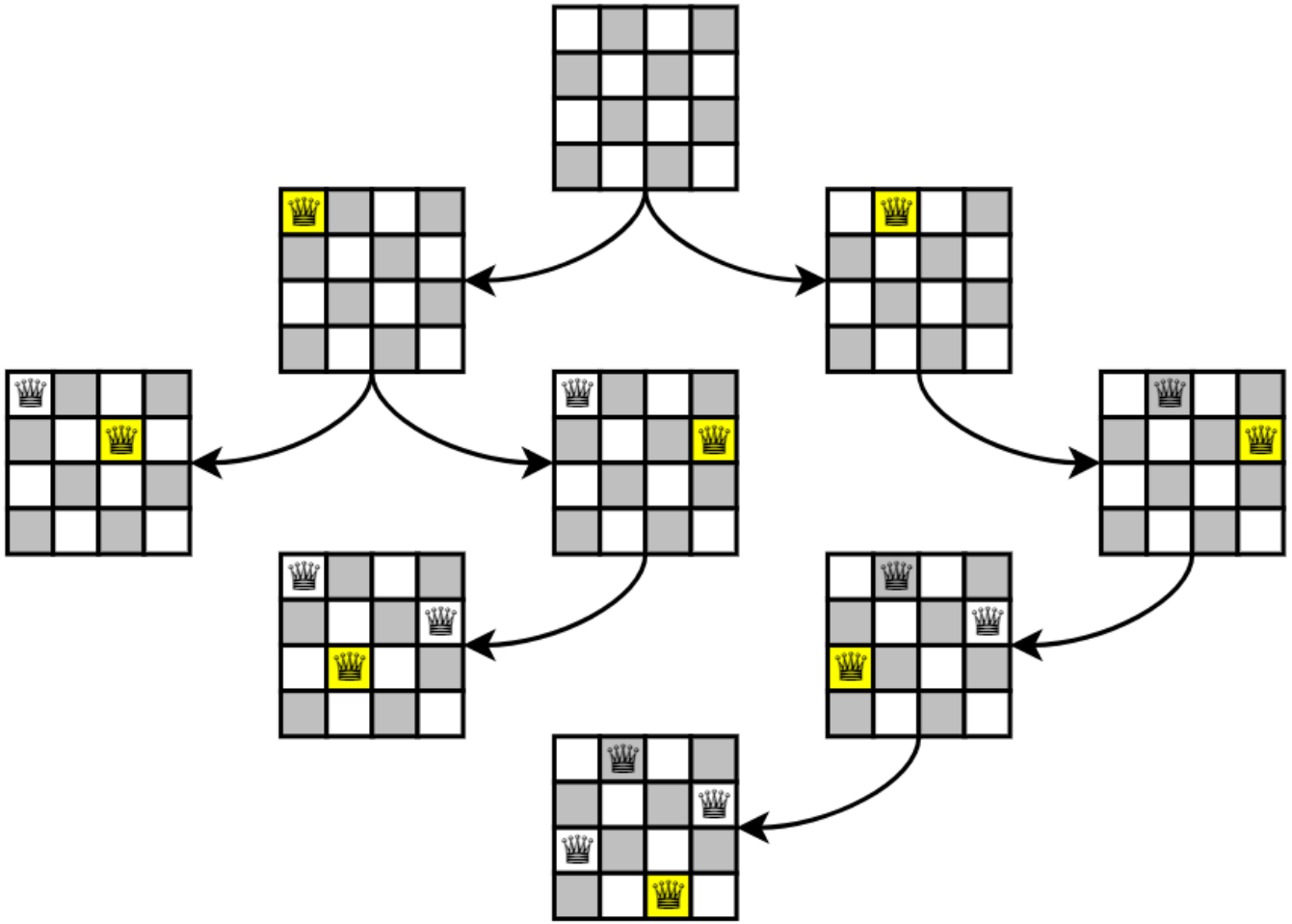
informatica e laboratorio di programmazione



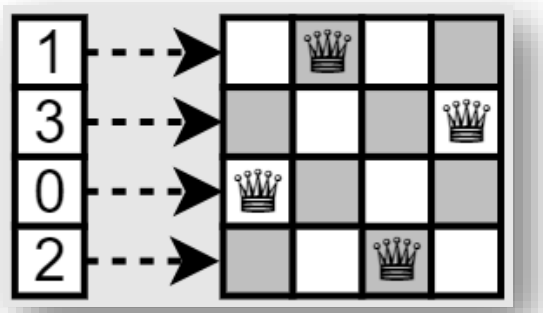
- ***backtracking*** (monitoraggio a ritroso) tecnica di ricerca soluzioni per problemi in cui devono essere soddisfatti ***vincoli***
- ***enumera*** tutte le ***possibili soluzioni*** e ***scarta*** quelle che non soddisfano i vincoli



n regine - backtracking



```
def stampa_scacchiera(scacchiera: list):
    ''' visualizza la scacchiera '''
    for r in range(len(scacchiera)):
        for c in range(len(scacchiera)):
            if c == scacchiera[r]:
                print('|Q', end='')      #presente regine
            else:
                print('| ', end='')      #casella vuota
        print('|')                       #fine riga
```



*scacchiera è una lista di **int**: per ogni riga della scacchiera, memorizza la posizione della regina sulla colonna*

```
def sotto_attacco(scacchiera: list, x: int, y: int) -> bool:
    ''' true se la regina in colonna x e riga y
    è sotto attacco da altre regine '''
    # controllo le righe precedenti
    for r in range(1, y + 1):  # fino alla riga attuale
        # direzioni di controllo: ↖ ↑ ↗
        if scacchiera[y - r] in (x - r, x, x + r):
            return True
    return False
```

n regine - ricorsione

```
def posiziona_regine(scacchiera: list, r=0) -> bool:
    '''
    cerca di posizionare una regina nella scacchiera
    in riga r (le precedenti righe contengono regine
    '''
    if r == len(scacchiera):
        return True  # successo! tutte le righe contengono regine
    # prova inserimento regina in tutte le colonne
    for c in range(len(scacchiera)):
        if not sotto_attacco(scacchiera, c, r):
            scacchiera[r] = c  # possibile inserire regina in colonna c
            # passaggio alla riha successiva
            if posiziona_regine(scacchiera, r + 1):
                return True

            scacchiera[r] = None  # non è possibile inserire, backtrack
    return False
```

backtracking

esercizi

- ***11.1 puzzle di Cindy***
  - piano di gioco: ***2n+1 celle allineate***
  - si parte con ***n pedine rosse tutte a sinistra***, ***n pedine verdi tutte a destra***, ed una cella libera in mezzo
  - le pedine ***rosse*** si possono spostare solo a ***destra***, quelle ***verdi*** solo a ***sinistra*** (senza poter tornare indietro)
  - ad ogni mossa, una qualsiasi pedina può:
    - ***avanzare*** di una posizione, se davanti ha una cella libera
    - oppure ***scavalcare*** esattamente una pedina dell'altro colore, se c'è una cella libera subito dopo
  - l'applicazione deve trovare ***automaticamente*** le mosse per ***invertire*** la posizione di tutte le pedine