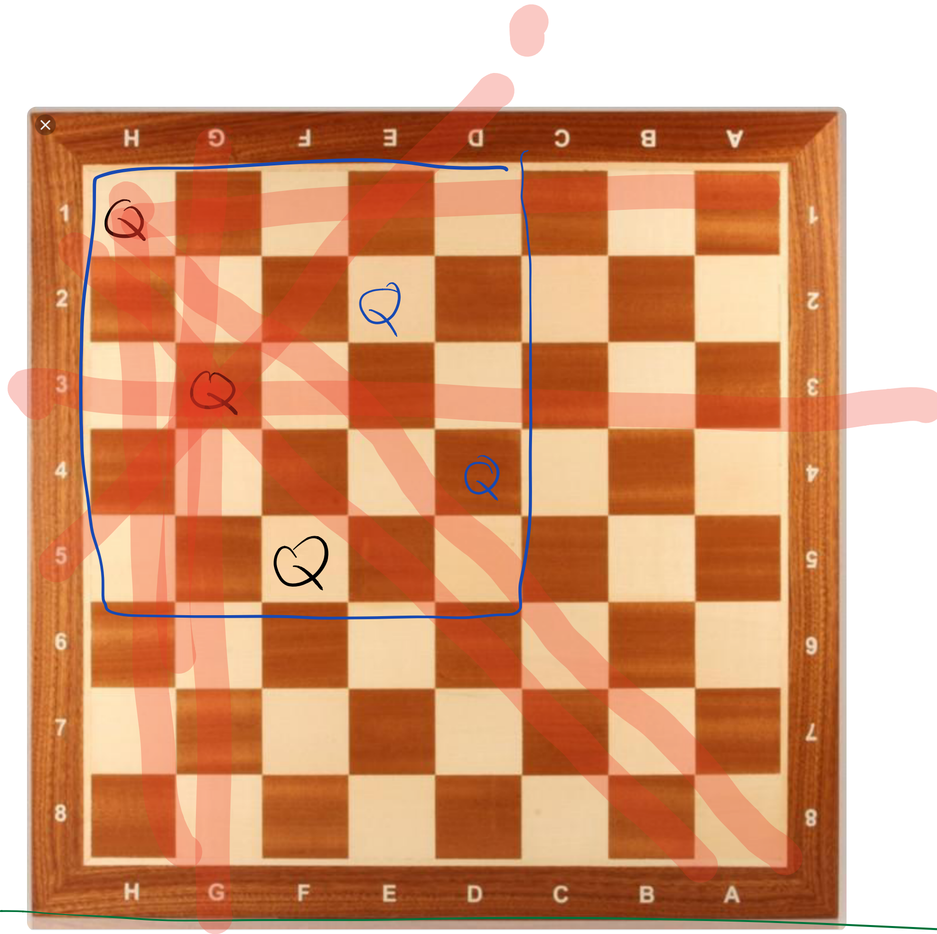
Today : Backtracking
- n queens
- subset sum
- text segmentation
- longest increasing subsequence

karatsuba



x, y represented as arrays of digits

x · y

$2019 \to [9, 1, 0, 2] = 9 \cdot 10^0 + 1 \cdot 10^1$

$\sum_{i=0}^{n} 10^i \, x[i]$ $\qquad + 0 \cdot 10^2 + 2 \cdot 10^3$

$x \qquad y \qquad$ n- digit #'s

$x^h \cdot 10^{n/2} + x^L$ $\qquad x^h \; x^L \; y^h \; y^L$ n/2 digit #'s

$y^h \cdot 10^{n/2} + y^L$

$$(x \cdot y) = (x^h \cdot 10^{n/2} + x^L)(y^h \cdot 10^{n/2} + y^L)$$

$$= \underline{x^h \cdot y^h} \cdot 10^n + \boxed{(x^h \cdot y^L + \underline{y^h \cdot x^L})} \cdot 10^{n/2}$$

$$+ \underline{x^L \cdot y^L}$$

$T(n) = 4T(n/2) \qquad \Theta(n^2)$

$$x^h \cdot y^h + x^L \cdot y^L - (x^h - x^L)(y^h - y^L)$$

$$= \cancel{x^h \cdot y^h} + \cancel{x^L \cdot y^L} + (\cancel{x^h y^h} + x^h y^L + x^L y^h + \cancel{x^L y^L})$$

$$= x^h \cdot y^L + x^L \cdot y^h$$

$$T(n) = 3T(n/2) + O(n)$$

Karatsuba (x, y, n)

// $x, y \to n$ -digit #'s

if n == 1

return $x[0] \cdot y[0]$

split x into x^h, x^L

split y into y^h, y^l

compute $a = x^h y^h$ ← recursive calls

$b = x^l y^l$ ←

$c = (x^h - x^l)(y^h - y^l)$ ←

return $10^n \cdot a + 10^{n/2} \cdot (a+b-c) + b$

$$T(n) = 3T(n/2) + \Theta(n)$$

$\log_2 n$

n n

$n/2$ $n/2$ $n/2$ $3n/2$

$9n/4$

$n/4$ . . .

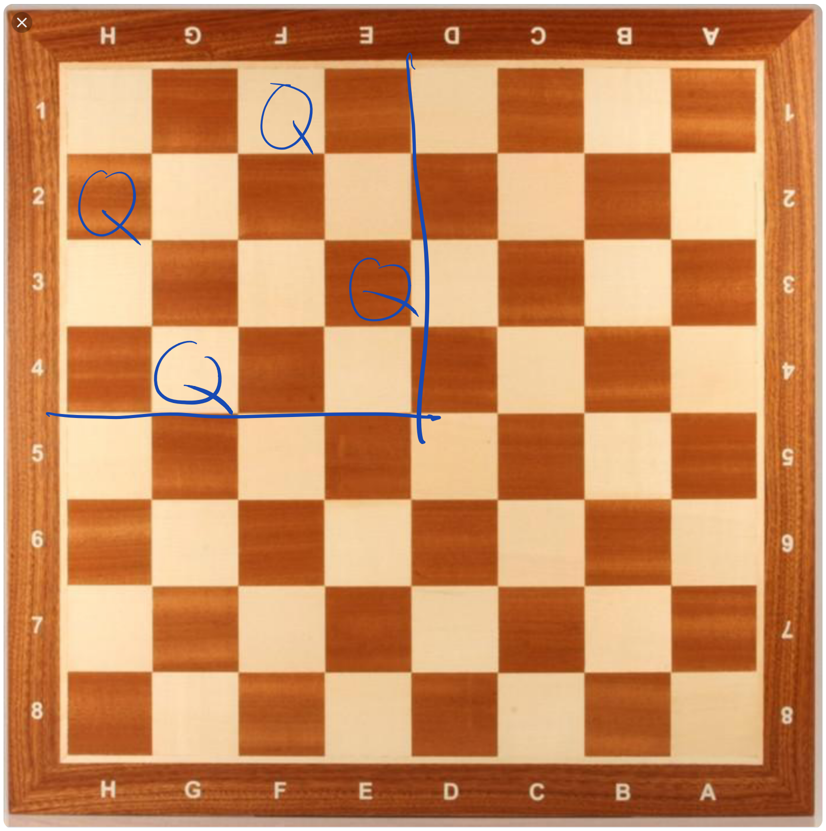
$$aT(n/b) + f(n)$$

↓ (master theorem)

$n^{\log_2 3}$ $+ n^{\log_b a}$

non-Karatsuba $4T(n/2) + n$

$\to n^{\log_2 4} = n^2$



Subset sum

Given list of integers $[x_1, \ldots, x_n]$

Find a subset that sums to t (exactly)

$[1, 1, 5, 5, 10, 25]$

$38 = \emptyset$ $\qquad$ $22 = 10+5+5+1+1$

target = 26

take 1 $\qquad$ make(25, [1, 5, 5, 10, 25])

take 1 $\qquad$ make(24, [5, 5, 10, 25])

---

leave 1 $\qquad$ make(26, [1, 5, 5, 10, 25])

$x_1, \ldots, x_n$ $\quad t$

take x_1 $\qquad$ leave x_1

$x_2, \ldots, x_n$ $\quad t - x_1$ $\qquad$ $x_2, \ldots, x_n$ $\quad t$

```
subset_sum( list, target)
base case { ① if target > 0 and list empty: return False
            if target == 0 : return True
            if target < 0 : return False
  if subset_sum( list[2..n], target - list[1])
      return True
  if subset_sum( list[2..n], target)
      return True
  return False
```

---

decompose problem into steps

- make partial solution
- try choices for next step
  - recursively solve remainder
  - backtrack if "stuck"