Lecture 2021-03-09

Tuesday, 9 March, 2021 10:48

Last week: for QuickSort we decided to use median as pivot, and reduced to Selection

↑ more general than median!

Imagine (recursively) looking for the median:

$$\left[\quad \le A[p] \;\middle|\; A[p] \;\middle|\; \ge A[p] \right]$$

pick a pivot & partition

if $p > \frac{n}{2}$

recurse left

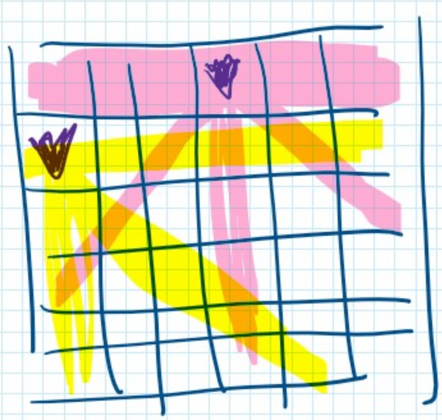
$A[1..p-1]$ "median" of $A[1..n]$ is not median of $A[1..p-1]$.

Immediately realize that "recursing" requires thinking of the more general Selection problem.

→ Sometimes, to do recursion, sometimes first generalize the problem ←

—— × ——

Recursive Backtracking



$n \times n$ chessboard

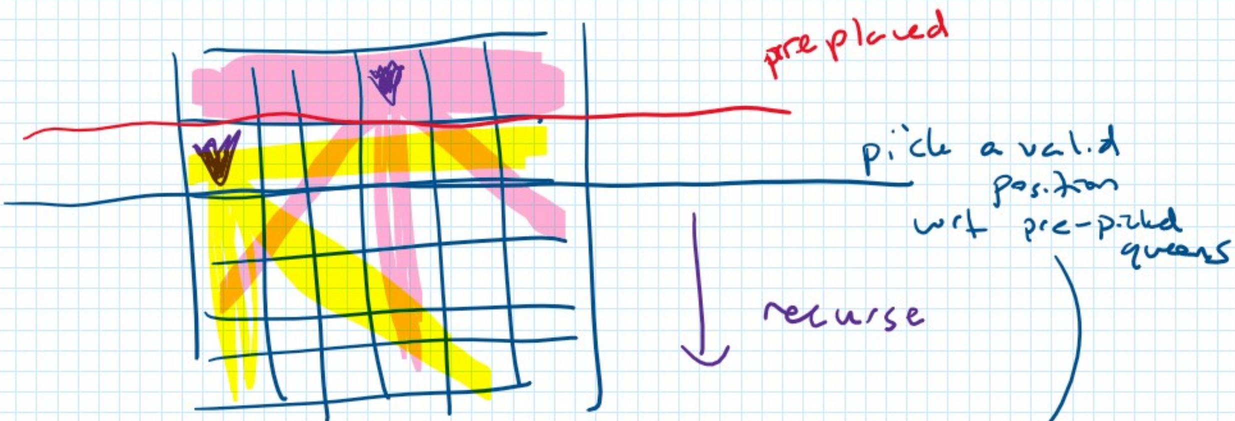
place n "queens" on the board s.t none of them are attacking each other (or report impossible)

Generalize:

Generalize:

Given an n x n chessboard & placements of r queens on the first r rows,

Place n−r queens on the remaining (n−r) rows so that no queens attack each other.



here, which (valid) position to pick?

recursive backtracking:

try all possible choices at current step
→ recurse

figure out based on result of recursive call which choice was correct.

(doing DFS through dependency tree of the recursive function)

PlaceQueens(Q[1..r])
    if r = n
        return Q
    else
        for j ← 1 to n
            brute force check if column j is valid for row r+1

for $j \leftarrow 1$ to n:

brute force check if column j is valid for row $r+1$

if so,

recurse w/ $Q[1..r]$ appended w/ j.

if recursive call did not return fail

return output.

return fail.

---

learn   to live

學生活

student   job

name of a café / bookstore on Green st?

notwithstanding

Preprocessing problem (to comp linguistics / AI / etc.): given a string & a dictionary can we split the string into valid seq of words in the dictionary?

BOTHEARTHANDSATURNSPIN

(assume black box access to IsWord(w))

BOTHEARTHANDSATURNSPIN

↑ is a word   recurse   yes!

↑ hear is a word   recurse   no!

$$\text{IsSplittable?}(w[1..n]) = \begin{cases} \text{True} & \text{if } w = \varepsilon \\ \bigvee_{i=1}^{n} \left(\text{isWord}(w[1..i]) \wedge \text{IsSplittable}(w[i+1..n])\right) & \text{o.w.} \end{cases}$$

outputs <u>True</u> if
w can be split into seq
of valid words
<u>False</u> o.w.

for loop!

equivalently:
fix input $w[1..n]$

$$\text{IsSplittable?}(i) = \begin{cases} \text{True} & \text{if } i > n \quad // \; w[n+1..n] = \varepsilon \\ \bigvee_{j=i+1}^{n} \left(\text{isWord}(w[i..j]) \wedge \text{IsSplittable}(j+1)\right) & \text{o.w.} \end{cases}$$

output True if
$w[i..n]$ can be split
False o.w.

(version of the)
what problem are you <u>really</u> solving recursively?
→ what are the recursive subproblems?
what do you need to remember about past decisions?
(optional) encode subproblem more efficiently?

— Longest Increasing Subsequence

Input sequence $A[1..n]$ of numbers
(array)

looking for the length of the
longest inc subseq of A.

String
Sing is a subsequence

a sequence $B[1..n]$ is increasing.
$B[i] > B[i-1]$ for all $i \geq 2$.

Q: what is a (local) decision we can try?
is $A[i]$ in a LIS?

↳ recursive backtracking says
try both ... and

3 1 4 1 5 9 2 6

↪ recursive backtracking says try both ways!

3 1 4 1 5 9 2 6

recurse here?

Q: what is the actual problem I'm solving recursively?

3 1 4 1 5 9 2 6

↑ include this?

recurse here..

still LIS?

really: LIS of #s > 3.

Q: what to remember? k for solving LIS for #s > k.

$$
\text{LISgreater}(A[\cancel{1..n}]^{\,i}, k) = \begin{cases} 0 & \text{if } \cancel{A \text{ is empty}}\ i > n \\ \max\begin{cases} 1 + \text{LISgreater}(A[\cancel{2}^{\,i+1}..n], A[\cancel{1}^{\,i}]) \\ \text{LISgreater}(A[\cancel{2}_{\,i+1}..n], k) \end{cases} & \text{if } \cancel{A[1]}\ A[i] > k \\ \text{LISgreater}(A[\cancel{2}_{\,i+1}..n], k) & \text{o.w.} \end{cases}
$$

return length of LIS of $A[i..n]$ consisting of #s > k.

Q: can we encode more efficiently? Yes!

How to use LISgreater to solve original problem?

LIS(A[1..n]):
    return LISgreater(1, −∞)