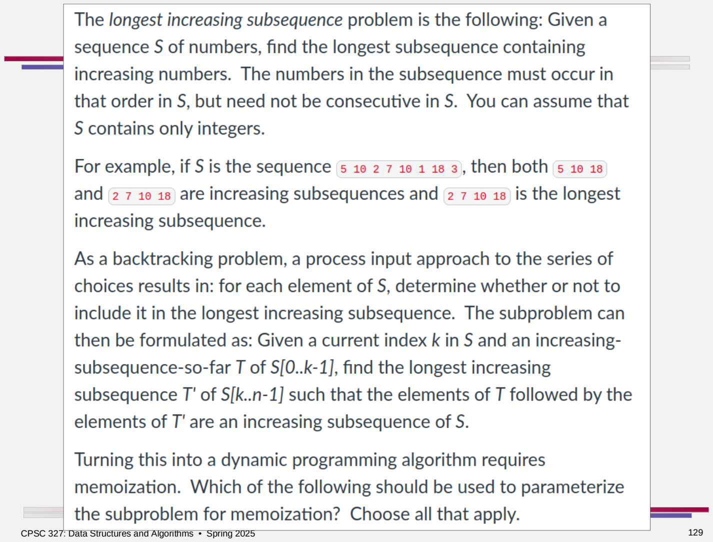
Dynamic Programming

The idea of dynamic programming –

- formulate the problem as a backtracking problem
  - series of choices approach
    - solution is constructed by making a series of decisions
  - case analysis recursive structure
    - you consider the next possibilities for the current decision, then ask friends to solve the problem given the consequences of each choice
  - subproblem solution is just the subproblem solution, not a complete solution
- identify how to parameterize the subproblems so that subproblem solutions can be stored instead of recomputed – *memoization*
- compute subproblem solutions by iterating through the subproblem states rather than doing a depth-first search of the solution space

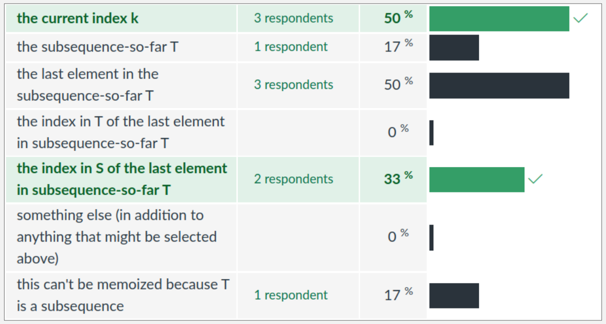


The *longest increasing subsequence* problem is the following: Given a sequence *S* of numbers, find the longest subsequence containing increasing numbers. The numbers in the subsequence must occur in that order in *S*, but need not be consecutive in *S*. You can assume that *S* contains only integers.

For example, if *S* is the sequence `5 10 2 7 10 1 18 3`, then both `5 10 18` and `2 7 10 18` are increasing subsequences and `2 7 10 18` is the longest increasing subsequence.

As a backtracking problem, a process input approach to the series of choices results in: for each element of *S*, determine whether or not to include it in the longest increasing subsequence. The subproblem can then be formulated as: Given a current index *k* in *S* and an increasing-subsequence-so-far *T* of *S[0..k-1]*, find the longest increasing subsequence *T'* of *S[k..n-1]* such that the elements of *T* followed by the elements of *T'* are an increasing subsequence of *S*.

Turning this into a dynamic programming algorithm requires memoization. Which of the following should be used to parameterize the subproblem for memoization? Choose all that apply.



the current index k	3 respondents	**50 %**	✓
the subsequence-so-far T	1 respondent	17 %	
the last element in the subsequence-so-far T	3 respondents	50 %	
the index in T of the last element in subsequence-so-far T		0 %	
the index in S of the last element in subsequence-so-far T	2 respondents	**33 %**	✓
something else (in addition to anything that might be selected above)		0 %	
this can't be memoized because T is a subsequence	1 respondent	17 %	