ECS 122A Lecture 7 Jasper 1	_ec
Answering range queries	
A nice trick from doing DP	
Setting: An array A [1n]	
Prepare a "data structure" to answe	r
range que-res (i,j) >> \(\frac{1}{k-i}\)	
Naive: Every query spend O(j-i)	time.
Ketter:	
Observe $\frac{1}{k}$ $A(k) = \frac{1}{k}$ $A(k) - \frac{1}{k}$	(-({ A[E] C=r
So, compute prefix sum in O(n) T[:]= {\frac{1}{k=1}} A[k]	time
T [i+1] = TCi] + ACi+1]	
Very simple, useful trick to have	Ve.

Longest Common Subsequence
Given a seg S[1n], (just an array)
· ·
a subsequence formed by removing some Symbols.
Symbols.
J
e.g. S=abcdefg
J
Subsequences: adf adfg Soou, bce
$\alpha d f q$
So ou,
bce
2 ⁿ many subsequences.
LCS:
LCS: Given seg A[1m], B[1n]
Find the longest subseque appearing in both
A and B.
Exhanstive Search takes exp(m) exp(n) time
wan : complexity
) CC is "CETH - hard" (on eclines Traves no
Want: O(mn) time - Fre-grained complexity LCS is "SETH-hard" Conjectures there's no (m' 99 n. 99) alg.
A 1
Civen runtine, can afford mu size table.
Let's try the gropping to from hofine, use
Let's try the approach from before, use save problem on smaller input for table meaning.
Solve property of solver in party of the solver in the sol

T[i,j] = LCS for A[1..i]
B[1..j] Obs: MUST LCS either uses ALT, BLIT well need to check if ACT = BIJT. either cloesn't use or Btil TLi+1, j+1] = max(ifAii)=Rtj) thun I+T(i,j) else 0, TLi,j+1, T(i+1,j)in w/2d loop, like in #1, => path Exercise: Do longest common substring 0-1 Knapsack Setting: Knapsack has capacity C (integer) A list of n items each w/
- Reward ri
- Weight wi integer Find a subset of items with total weight

and maximize total reward.

10-1" be cause either we take an item or not. No multiple copies of the same item.

One of different formulations cothers include integer reward instead of integer weight/capacity)

require différent algs!

Haven't talked about greedy, but can we? No!

E.g. greedy for ri/wi fails

Item 1, $r_1 = 1$, $w_1 = 10^{-5}$ Item 2, $r_2 = 10$, $w_2 = 10$ Capacity = 10.

Can solve w/ DP.

First try

To [i] = 0-1 knapseck on items 1.. i? Can't possibly work, because of the capacity constraint.

New try, let's make the DP table 2-D!

keep track of info we need V m a table (Subproblem index

T[i, C] = 0-1 knap sack on items 1...i
under compactly C

Recurrence observation:

Opt soln must either

Use item i ~ rit [[i-1, C-wi] Notuse item in T[i-1, C]

T[i, C] = max (re+T[i-1, C-wi],

How to fill table?

for i = 1 t. n c = 0 to C O(Cn) time

How much space? O(Ca) space for Space optimization

Only need to keep the row Tti-1,.] to compute Tti, J. O(C) Space.