≡ Textbook Problem 13

You're in line to meet a fistulated cow, and while you wait, you decide to solve some DP problems.

(based on a true story!)

You're given two arrays of length n

Problem[i] = happiness you get from solving problem i

Time[i] = time it takes to solve problem i

If you choose to solve problem i, then you will be unable to solve problems $i+1 \ldots i+\text{Time}[i]$ (i.e. next problem you can solve is problem $i+\text{Time}[i]+1$)

Compute the maximum total happiness you can achieve from solving problems.

Let MaxHap(i) = the max happiness I can get from Problem[i..n].

$$\text{MaxHap}(i) = \begin{cases} 0 & \text{if } i > n \\ \max\{\text{MaxHap}(i+1), \text{Problem}[i] + \text{MaxHap}(i+\text{Time}[i]+1)\} & \text{if } i+\text{Time}[i]+1 \le n \\ \max\{\text{MaxHap}(i+1), \text{Problem}[i]\} & \text{else} \end{cases}$$

1D Array [1..n+1]

Decreasing i

O(n)

Return MaxHap(1)

↳ we have this for memoization... i+Time[i] could be >>n, so without this case the array bounds could be very large

"Knapsack"

You're in charge of scheduling ads for the Superbowl. You're given two arrays of length n.

Rev[i] = revenue gained from running ad i

Time[i] = amount of time running ad i takes.

Compute the max possible revenue attainable such that the total time of the ads is $\leq T$.

Let MaxRev(i, t) = max rev possible from Rev[i..n] with total time $\leq t$. Return MaxRev(1, T)

$$\text{MaxRev}(i,t) = \begin{cases} 0 & \text{if } i > n \\ \max\{\text{Rev}[i] + \text{MaxRev}(i+1, t-\text{Time}[i]),\ \text{MaxRev}(i+1, t)\} & \text{if Time}[i] \leq t \\ \text{MaxRev}(i+1, t) & \text{else} \end{cases}$$

skip

take

2D Array [1..n+1, 0..T]

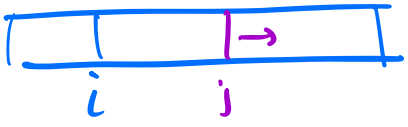
Decreasing i, increasing t

Runtime = $O(nT)$

From DP Review

2.

Let MinRt(i, ℓ) = min cost of partition A[i..n] into ℓ intervals.



Intuition: Look at block starting with A[i], try every $j \geq i$ to end the block.

$$MinRt(i,\ell) = \begin{cases} \min_{i \leq j \leq n} \{\max\{sum(i,j), MinRt(j+1, \ell-1)\}\} \\ \infty \text{ if } i > n \\ sum(i,n) \text{ if } \ell = 1 \text{ and } i \leq n \end{cases}$$

2D Array [1..n+1, 1..k] Runtime: $O(n^3 k)$

i decreasing, ℓ increasing.

Prefix Sum $\Rightarrow O(n^2 k)$

1.

Let Gerry(i) = max # of good districts that can be made from GoodVote[i..n]

Intuition: Same as 2! Look at block starting with i and try all ways to end that block

$$Gerry(i) = \begin{cases} -\infty & \text{if } i+k > n \text{ and } i \leq n \\ 0 & \text{if } i > n \\ \max_{i+k-1 \leq j \leq n} \{\text{GoodVote}[i..j] + Gerry(j+1)\} \end{cases}$$

$$\left(\max_{i+k \le j \le i+2k} \{ Good(i..j) + Gerry(j+1) \} \right) \text{ else}$$

Where Good = 1 if Vote[i..j] is good
0 else

↑ O(k)

Try O(k) js. O(n) is ⇒ O(nk²)

Keep prefix counter: # good votes in 1..i = PrefixGood(i)

Good(i,j) can be computed using PrefixGood(j) − PrefixGood(i) in O(1) time.

⇒ O(nk)

4b. Intuition: "building a tree" – balanced Parentheses make a tree!

For "building a tree" problems, subproblems usually look like this...
(eg. woodcutter)

Let Stupid(i,j) = max value attained from partitions in A[i..j]

$$Stupid(i,j) = \begin{cases} \max_{i \le k < j} \left(Stupid(i,k) \oplus Stupid(k+1,j) \right) \\ A[i] \quad \text{if} \quad i = j \end{cases}$$

I missed this... allows singletons in left subtree

Increasing j−i

2D Array: [1..n, 1..n]

O(n³)