Best Time to Buy and fell stock IV

Pick the right model first: |

1. If k is large: when k >= [n/2], it's equivalent to unlimited Tip: just sum every positive price difference (sem max (0, prices [i]-prices (i-13/)).
Why: you can buy sell on every rise without exceeding k.

2. Otherwise (kis small): use DP with k layers. The classic U(n. KIBP: Therik in terms of t completed transactions (cell actions) up to day i. A) "Hold (Not-hold state per transaction count: Maintain arrays of Mize 12+1: hold [t] = max profit after bouring for the t-th transaction (i.e. currently holding a stock; you've completed t-1 rells).

cash [t] = max profit after relling for the t-the time (i.e. most holding; you've completed to sells). Initiolize: · cosh 607=0, hold 607=- ~ (can to hold before any buy). For t=1: hold [t]=-o, cash [t] =-o initially (or a very small number) You each pricep: for t=1... k: hold (t)=max (hold [t] cosh [t-1]-p) cosh [t]= max (cash [t], hold [t]+p) Answer: max_{t=0.k} cash(t) (remally cash(k) is enough since cosh is

mondecreaming in t)

bly it works: each layer trepresents you've working on the t-th transaction; buying user profit from cook [t-1], selling closes hold[t].

B) Best -so-far tricle per layer:

Let of Ct7Ci7 - max profit up to day i with at most t transactions.

hamition when you sweep is

ap [t][i] = max (ap [t][i-1], prices [i]+ best) Best = max (lest, ap [t-1][i]-prices [i]). Here best tracks the best "Deep point with t-1 transaction above" as you go. I pace - opt: keep only two 1D arrays for ap [t-1] (k] and ap (t)(k). [Complexety: I Time: O(n·k) (early-return to gready when k >= n /2). I pace: O(k) with the hold/lash form, or O(n)— O(1) per layer with solling arrays.
Best = max (lest, apct-1][i]-prices[i]). Here best tracks the best "lovey point with t-1 transactions alone" as you go. space-opt: keep only two 1D arrays for apct-1] (k) and apct)(k). [Complexity: 1 · Terme: O(n·k) (early-return to greedy when k >= n/2). · space: O(k) with the hold/cash form, or O(n)-10(1) per layer with
Space- got: keep only two 1D aways for apt-13(k) and aptilks. [Complexity: 1 · Time: O(n·k) (early-return to greedy when k >= n /2). · Space: O(k) with the hold/lash form, or O(n)-10(1) per layer with
Space- got: keep only two 1D aways for apt-13(k) and aptilks. [Complexity: 1 · Time: O(n·k) (early-return to greedy when k >= n /2). · Space: O(k) with the hold/lash form, or O(n)-10(1) per layer with
Space- got: keep only two 1D aways for apt-13(k) and aptilks. [Complexity: 1 · Time: O(n·k) (early-return to greedy when k >= n /2). · Space: O(k) with the hold/lash form, or O(n)-10(1) per layer with
(Complexity.) Teme: O(n.k) (early-return to greedy when $k \ge n/2$). Space: O(k) with the hold/lash form, or O(n) - O(1) per layer with
· Teme: O(n·k) (early-return to greedy when k >= n/2). · Space: O(k) with the hold/cosh form, or O(n) - O(1) per layer with
· Teme: O(n·k) (early-return to greedy when k >= n/2). · Space: O(k) with the hold/cosh form, or O(n) - O(1) per layer with
Time: O(n·k) (early-return to greaty when k >= n /2). Space: O(k) with the hold/lash form, or O(n)-10(1) per layer with solling arrays.
- Space: O(b) with the hold/lash form, or O(u) - O(u) per layer with solling arrays.
Alling amays.
Scling amays.