02-CODE

Problem-1 (Δώρο Χριστουγέννων)

(βλ. Longest Increasing Subsequence)

Initial array : Arr[ ]
Example: A[ ] = {7, 3, 5, 12, 2, 7, 3, 4}

Prerequisites :

1. DP[ ] array to store for each A[i] the LIS length until that element (including the element)
   Example: DP[ ] = {1, 1, 2, 3, 1, 3, 2, 3}
2. Array of stacks to store in each stack ST[i] the elements that are last in subsequence of length i
   Example: ST[ ] =
   1 | → 7 → 3 → 2
   2 | → 5 → 3
   3 | → 12 → 7 → 4
   (note: the top elements of each stack will always be sorted)
3. DPreverse[ ] array to store for each A[i] the LIS length upwards (from that element until the A.end())
   Example: DPreverse[ ] = {2, 3, 2, 1, 3, 1, 2, 1}

Steps :

1. For each element A[i] (starting from the end) :
   a. Pop A[i] from stack DP[i] of array ST[ ]
   b. Increment A[i] by K → curr = A[i] + K
   c. Do binary search for `curr` in the top elements of ST[ ] array
   d. If it lands on the index i of ST[ ] take the number i-1 (that means that there is a subsequence of length i-1 in which we could add the element `curr` at the end of it and continue the subsequence
   e. Finally, add the number `i+1` with the `DPreverse[i]` and compare with the current maximum. `ans = max(ans, i+1+DPreverse[i])`

Γενικότερη ιδέα : Έχοντας κάνει preprocessing για τα επιμέρους LIS (για κάθε A[i] βρίσκουμε τα LIS από εκεί και κάτω και τα LIS από εκεί και πάνω), για κάθε στοιχείο αυξάνοντάς το κατά K, ψάξε αν μπορείς να το κάνεις να συνεχίζει κάποια από τις προηγούμενες υπακολουθίες (με binary search).

Problem-2 (Αγορές στο φαράγγι)

Δεδομένα εισόδου **Παρατηρήσεις**

N : αριθμός πλήρων σετ εξοπλισμού (A + B + C)

M : αριθμός προσφορών

xy A P :
x → αριθμός έμπορα {1, 2, 3}
y → τύπος αντικειμένου {A, B, C}
A → πλήθος αντικειμένων στη προσφορά
P → κόστος προσφοράς

- Αν κάποιος έμπορος δεν έχει και των τριών ειδών αντικείμενα (A,B,C) τότε δεν είναι χρήσιμος.
- Ο μέγιστος αριθμός πλήρων σετ κάθε έμπορα ορίζεται από το ελάχιστο πλήθος κάποιου τύπου αντικειμένου

Γενική ιδέα

Μέσω συνδυαστικής βρίσκουμε πόσα σετ θα πάρουμε από τον πωλητή 1 (έστω α) και πόσα από τον πωλητή 2 (έστω β). Τότε μένει μοναδική λύση για το πωλητή 3 και οι συνολικοί συνδυασμοί είναι N^2. Από πριν ωστόσο έχουμε βρεί για το πωλητή X ποιό είναι το βέλτιστο κόστος αν αγοράσουμε α σετ από εκείνον. Η λογική πίσω από αυτό είναι ότι αρκεί να βρούμε τα επιμέρους κόστη για κάθε ένα αντικείμενο ξεχωριστά για ποσότητα α και να τα αθροίσουμε → DP_A[i, a] : όπου i οι πρώτες i προσφορές που αφορούν το προιόν A και a η ποσότητα που ψάχνουμε.

Σχέσεις

1. $DP\_X[i, n] = min(dp[i-1, n], dp[i-1, n-a_i] + p_i)$

   DP_X → for object X from {A,B,C} the best cost for first i offers and capacity n
2. $cost(x, k) = DP\_A[M, k] + DP\_B[M, k] + DP\_C[M, k]$

   cost(x, k) → for vendor x calculates in O(1) the best value for k sets
3. $ans = \min_{0 \le a, b \le N} (cost(1, a) + cost(2, b) + cost(3, N - a - b))$

Complexity

$O(N^2 + NM)$