SHOPEE CODE LEAGUE 2021

Guidelines for Approaching the Programming Problems

Competition Details

This competition was launched on **20 March 2021.**

Please refer to the problem statements here.

Shopee Farm

Task:

Help the character to get the maximum amount of health.

Solution:
We can use DP to solve this problem. Let DP[i][j] be the maximum health that the character can get at the end of day i, and j = (0/1), 0 means the character is at the left of the farm and 1 means the character is at the right of the farm.

So the idea is if currently you are at the left side of the farm, then it's either you will go all the way to the right of the farm, or you will only take up to M fruits at your right. It's the same logic when you are at the right side of the farm, either you go all the way to the left of the farm and stay there, or only take up to M fruits.

Sample of code when the character is at the left of the farm:

```
if (at the left of the farm):
    for i = 0; i <= m; i++:
        picked = health_picked_at_day_and_cell(idx, i)
        if (at_right_of_farm()):
            ans = max(ans, max(go_to_next_day(), picked + solve(idx+1, 1 - where)))
        else:
            ans = max(ans, max(go_to_next_day(), picked + solve(idx+1, where)))

return DP[idx][where] = ans


========

def at_right_of_farm():
    return i == m

def go_to_next_day():
    return solve(idx + 1, where)

def health_picked_at_day_and_cell(idx, j):
    return pref[idx][j-1]
```

You can do precomputation on the number of fruits taken from the left / right (prefix sum / suffix sum) to make the DP computation faster

Space complexity: O(2*N) = O(N), Time Complexity: O(NM)

Shoffee

Task:
Given an array of positive integers of length N and a positive integer K, calculate how many consecutive subsequences that have an average value >= K.

Solution:
Our target is to count how many pairs $< j, i >$ $(1 \leq j < i \leq N)$ in the array that meet the following inequality

$$\sum_{p=i}^{j} a_p / (i - j + 1) \geq K$$

with prefix sum, it can be represented as

$$(sum_i - sum_{j-1}) / (i - j + 1) \geq K$$

let's move $(i - j + 1)$ to the right,

$$sum_i - sum_{j-1} \geq K * (i - j + 1)$$

then our target becomes

$$sum_i - K * i \geq sum_{j-1} - K * (j - 1) \text{ where } i > j$$

let $b_i = sum_i - K * i$, then we have

$$b_i \geq b_{j-1} \text{ where } i > j$$

which is quite clear, all you need is to count the number of non reverse ordered pairs in $O(NlogN)$ time, segment tree or fenwick tree could be useful data structures for this problem.

Order Delivery

Task:

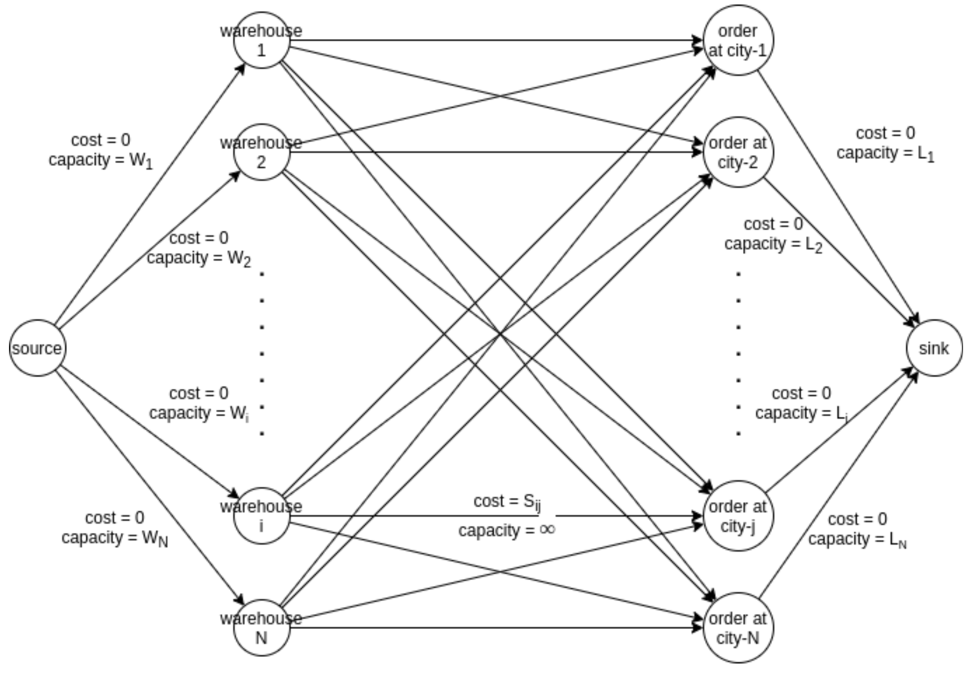
Because of the free shipping discounts, Shopee needs to pay the delivery fee of all the orders. Your task is to help Shopee to minimize the delivery fee in this 13.13 campaign.

It is obvious that the orders from the same city can be combined into one order. Suppose there are 3 orders to city-1. Order 1 contains 10 items, order 2 contains 20 items, and order 3 contains 30 items. Those 3 orders can be combined into 1 order containing 60 items and need to be delivered to city-1. By doing this, the number of orders can be reduced to maximum 20 orders because there are at most 20 cities. Define L_i as the number of items in all orders in city-i.

Since the number of cities is small, Floyd-Warshall algorithm can be performed to calculate the shortest-path of all pairs of cities. By knowing the shortest-path for all pairs of cities, the delivery cost for all warehouses and orders pairs can be calculated. Define C_x as the shipping cost of warehouse at city-x per kilometer, $distance_{xy}$ as the distance of city-x and city-y in kilometer, and S_{xy} as the cost of shipping one item from warehouse at city-x to city-y.

$$S_{xy} = distance_{xy} * C_x$$

To calculate the minimum cost, we can use the minimum-cost-maximum-flow algorithm. What we need to do is just construct a bipartite graph and perform a minimum-cost-maximum-flow algorithm to that bipartite graph. The picture below shows the generated bipartite graph.



source
warehouse 1
warehouse 2
warehouse i
warehouse N
order at city-1
order at city-2
order at city-j
order at city-N
sink
cost = 0 capacity = W_1
cost = 0 capacity = W_2
cost = 0 capacity = W_i
cost = 0 capacity = W_N
cost = S_{ij}
capacity = ∞
cost = 0 capacity = L_1
cost = 0 capacity = L_2
cost = 0 capacity = L_j
cost = 0 capacity = L_N

Divider

Task:
Given N integers, partition them into K groups such that the total cost is minimum. The cost of a partition is formalized as:

$$cost(l, r) = (\sum_{i=l}^{r} A_i)(r - l + 1)$$

Naive solution

Let:
$dp(r, k)$ be the minimum total cost if we want to partition r elements into k different groups.

$$dp(r, k) = 0; r = 0 \land k = 0$$
$$dp(r, k) = \infty; (r > 0 \land k = 0) \lor (r = 0 \land k > 0)$$
$$dp(r, k) = min_{l=1}^{r} dp(l - 1, k - 1) + cost(l, r); r > 0, k > 0$$

This solution runs in $O(K N^2)$ which is too slow to solve this problem within a proper time limitation.

Optimal solution

Let:
$opt(r, k)$ be the best l to be chosen for solving $dp(r, k)$

The following inequality holds:
$$opt(r, k) \leq opt(r + 1, k)$$

We can solve this problem using divide and conquer technique in which the time complexity of this solution becomes $O(K N logN)$.

Shopee Planet

Task:

Find the maximum number of Shopee coins that you can collect at the end of day D − 1.

Observations:

- We are given a complete graph of 12 nodes, node i corresponds to the pentagon face i.
- For all pairs of nodes (u, v), if the shortest path is of length w, there are always other paths of length w+1, w+2, etc (property of soccer ball graph).
- At any time, we only need to care about the last 6 days.
- The history of visited nodes can be stored as an array of 6 integers from 0 to 12, where zero represents a hexagon face and non-zero represents a pentagon face.
- Given a history, we can determine the possible moves at the next step. For example, if the history is 001020, then the next step can be 1, 2, 3, 4, 6, 8 or 0.
- The number of valid histories is quite small (much less than 13^6)

Solution:

Let f(d, history) be the maximum number of coins that we can collect from day d to day D-1 where history is the list of the 6 most recent visited nodes. Note that history is an array of 6 integers from 0 to 12 where 0 denotes an hexagon face while 1 to 12 denotes an pentagon face; d is an integer from 0 to D-1.

We can use DP to solve this problem as follows:

$$d \in \{0, 1, ..., D-1\}$$

$$history = (x_6, x_5, x_4, x_3, x_2, x_1) \in (\{0, 1, ..., 12\})^6$$

$$f(d, history) = 0, \text{ if } d = D$$

$$f(d, history) = max \begin{cases} f(d+1, append(history, 0)) \\ f(d+1, append(history, i)) + C[i][lastVisited(i, history)] \\ \text{for all node } i \text{ reachable given } history \ (i \in \{1, ..., 12\}) \end{cases} \quad \text{, otherwise}$$

$$append((x_6, x_5, x_4, x_3, x_2, x_1), i) = (x_5, x_4, x_3, x_2, x_1, i)$$

$$lastVisited(i, (x_6, x_5, x_4, x_3, x_2, x_1)) = j,$$
$$\text{where } j \text{ is the minimum index that } x_j = i \text{ or } 7 \text{ if not exists}$$

The answer is f(d’[i], (0, 0, 0, 0, 0, i)) + C[i][7] where d’[i] is the number of steps from the starting point to node i.

When implementing this algorithm, history can be stored using an array of 6 integers. To optimize running time and memory usage, the optimal way is to use a bitset of 24 bits.

Time complexity: O(D * H * 12) where H is the number of valid histories. Note that, on the soccer ball graph, two pentagon faces are never adjacent. Therefore, in each history, two non-zero elements cannot be adjacent (e.g. 100203 is valid, but 012003 is not valid). This property reduces the number of valid histories by a lot. For any valid history (x6, x5, x4, x3, x2, x1), each pair (x6, x5), (x4, x3), (x2, x1) has 27 different values. Therefore, the number of valid histories must be less than 27^3 < 20000. In reality, the actual number is much smaller.