Third hands-on report

Francesco Scarfato

December 18, 2024

1 Holiday planning

The target solution has to run in $\theta(nD^2)$ with n itineraries and D days. To solve the problem, it is used a standard dynamic programming approach.

- Sub-problems: fixed an arbitrary ordering of the itineraries, we have to solve the problem for any prefix and value $\leq V$.
- Combine: assuming to know the optimal selection for $i-1$ with i index of the itinerary, we want to compute $M(i,k)$ that is the maximum number of attractions, considering the first i itineraries and k days available. At this point, analyzing the i-th itinerary, there are different options:
  - not include any item of the i-th itinerary;
  - include k items from the current itinerary;
  - include 1 item from the previous itinerary and $k-1$ items form the current one;
  - include 2 item from the previous itinerary and $k-2$ items form the current one;
  - ...
- Recurrence:

$$M(i,k) = \begin{cases} 0 & \text{if } i = 0 \wedge k = 0 \\ \max \begin{cases} M(i-1,k) \\ \max_{j=0}^{k}\{M(i-1,k-j) + \sum_{z=0}^{j} A[z]\} \end{cases} & \text{otherwise} \end{cases}$$

The factor $\sum_{z=0}^{j} A[z]$ is the prefix sums and has to be computed for each itinerary. This is computed before to start with the dynamic programming approach.

2 Design a course

The target solution has to run in $\theta(n \log n)$. The problem can be reduced to a longest increasing subsequence research.

Having two dimensions, beauty x and complexity y, the first thing to do is to sort by the first coordinate.

Now the problem is to find the size of the longest increasing subsequence on y. An array `lis` is built to track the smallest possible ending values for any increasing subsequence. Considering each item in the original array A, we binary search its position in the array `lis` and we can have two possible situations:

- If the position is equal to the size of `lis`, it means that the pair has the coordinates larger than every element in `lis`, so it is pushed in the array `lis`;
- Otherwise the pair replaces the first element that is greater or equal than it.

The sorting costs $\theta(n \log n)$ like the binary search. So, the overall complexity is $\theta(n \log n)$.