Workspace - Greedy Problems [8/29]

1 - UVa10382 - Watering Grass ✅

Example problem in Competitive Programming 3. Interval covering, but note the way we compute the intervals is nontrivial. Morespecifically, the interval is less than the radius but corresponds to the rectangle we want to cover.

Compute interval: pos - sqrt((w/2)^2 - rad^2), pos + sqrt((w/2)^2 + rad^2)

Note that rad >= w/2 is a must; if rad < w/2 we do not include the sprinkler.

https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=1323

2 - UVa1193: Radar installation.

Reduce to a familiar instance of interval covering problem. (Nope… Not so easy. Requires more thinking!)

Given a radar covering distance d we can construct a circle. For each point (x_i, y_i) compute sqrt(d^2-y_i^2) to get a segment s_i parallel to the x-axis. This is our covering distance. Construct array of intervals (x_i-s_i, x_i+s_i) K. What we need to do is to use the minimum amount of intervals of length d to any point in every interval in K. This can be accomplished by a traditional left-right scan.

https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=3634

3 - UVa11264 - Coin Collector ✅

O(n^2) solution: Take from left to right. For each coin scan through the rest of the array, find if our current sum + the value of coin would >= any other coin. If it would, then we do not include it in our solution.

O(n) version: Because of monotonicity guaranteed by the problem input we can just look at if sum + current >= next coin for 0…n-2 (if so, do not include, as then we can just withdraw the next coin instead of the current sum). For n-1 th coin we see if sum +C[i] <= C[i]*2; if so, we include (because otherwise we can withdraw the n-1th coin twice or more).

https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=2231

4 - UVa12321 - Gas stations ✅

Treat this as interval-covering. The gas stations are the intervals and we want the minimum amount of intervals covering the road. Do interval-covering and then compare the result array with the array (sorted) containing all gas stations; the ones that are not in the solution can be removed. **(Postscript: I overcomplicated this problem. Turns out we just need to take the size of the interval cover and subtract it from the size of the input.)**

Just pay attention that the final step can be linear.

https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=3743

5 - UVa12405 - Scarecrow.

This is similar to UVa1193; except our intervals are now in lengths of 3, and we attempt to cover disjoint intervals which are constructed from the 1D field with obstacles serving as dividing points.

https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=3836

6 - UVa311 - Packets.

Needs case-by-case analysis.

https://uva.onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=247

7 - UVa668 - Parliament.✅

Note how each group needs to be of DISTINCT sizes.Therefore

https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=609

8 - UVa10152 - ShellSort.

It's a variant of pancake flipping. Needs more consideration.

https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=1093

9 - UVa10602 - Editor Nottoobad. ✅

It's actually greedy. Since deleting backwards one by one does not cost us anything (and only the common subsequence at the start really matters, because of the way we copy words), we can just sort lexicographically and then assign each word to their position.

https://uva.onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=1543

10 - UVa11389 - The Bus Driver Problem. ✅

一个极其让人感到无语的问题. Pair each smallest morning route with the largest evening route. "even it out"-type of problem.

https://uva.onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=2384

11 - UVa 10026 - Shoemaker's Problem

todo

12 - UVa 10785 - The Mad Numerologist

todo

13 - UVa 11369 - Shopaholic

todo

14 - UVa 12485 - Perfect Choir

todo

15 - UVa 11269 - Setting Problems

todo

16 - UVa 11103 - WFF'N Proof

todo

17 - UVa12482 Short story competition

Needs work.

https://uva.onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=3926

18 - UVa 11100 - The Trip, 2007

Todo

19 - UVa 12210 - A Match Making Problem

Todo

20 - UVa10656 ✅

这题他妈的有病

Postscript: Requires O(m) space where m is size of positive integers. Can't make do with O(1) mainly because of the \n output instead of " \n".

https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=1597

21 - UVa11240 - Antimonotonicity

Needs dynamic programming. LIS with a twist.

https://uva.onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=2181

22 - UVa108 - Maximum Sum.

Consider using 2D Kadane.

https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=44

23 - UVa10670 Work Reduction.✅

Postscript: 1) The problem is ambiguous on whether if for odd numbers the reduction is floored or the number of papers after the reduction is floored. The answer is that the reduction always reduces the ceiled part and the amount of work after the reduction is floored. 2) the amount of work reduced can't be negative actually means that we always cannot reduce more work than to the required M amount of work. Hence when M=7 and N=9, we cannot reduce by half because that would bring us down to N less than M. The algorithm we use here calculates the per-unit-work cost for halving the work at each iteration while N>M, and compares it to the 1-unit cost; if halving is more economic then we half, otherwise if the resultant by halving is too small or if the 1-unit cost is more economic, we subtract by a single unit and continue the iteration. This gives us an O(N-M) time algorithm, but we can do even better. Since we are halving a strictly decreasing N at each iteration, our halving per-unit-cost will be monotonically increasing; hence when we start using the 1-unit cost, we will remain using the one-unit cost without a choice.

https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=1611

24 - UVa 10718 - Bit Mask

todo

25 - UVa 10714 - Ants

todo

26 - UVa 10700 - Camel Trading

todo

27 - UVa 10672 - Marbles on a tree

todo

28 - UVa 11157 - Dynamic Frog

todo

29 - UVa 11335 - Discrete Pursuit

todo