Probleme OJI

NIȚĂ ALEXANDROS

Dragoni OJI 2015

Să rezolvăm problema: https://www.infoarena.ro/problema/dragoni.

Dragoni OJI 2015

Să rezolvăm problema: https://www.infoarena.ro/problema/dragoni.

Hint cerința 1: Problema e una de parcurgere.
Hint cerința 2: Putem aplica algoritmul lui Dijkstra pentru a determina distanțe minime.

Dragoni OJI 2015

Să rezolvăm problema: https://www.infoarena.ro/problema/dragoni.

Rezolvare: Prima cerință ne cere să parcurgem graful cu condiția că costul muchiei este mai mic sau egal cu capacitatea dragonului de pe poziția 1. Pentru cerința 2 vom implementa Algoritmul lui Dijkstra, doar ajungând într-un nou nod, vom considera soluția folosindu-ne și de dragonul din acel nod (vom lua dragonul doar dacă are o capacitate mai mare pentru a ne asigura că avem soluție). Astlfe, soluția va fi memorată într-o matrice dist, unde $\text{dist}[nod][dragon]$ este distanța minimă de la nodul 1 la nodul nod folosind dragonul $dragon$. Soluția finală este $\min_{1 \le i \le n} dist[n][i]$, dacă structurile sunt indexate de la 1.

Implementare: https://www.infoarena.ro/job_detail/3210967.

Tairos OJI 2019

Să rezolvăm problema: https://www.infoarena.ro/problema/tairos.

Tairos OJI 2019

Să rezolvăm problema: https://www.infoarena.ro/problema/tairos.

Hint: Putem adăuga arbori doar prin frunze, deci ar fi bine să reținem cumva numărul acestora (e o problemă de numărare).

Tairos OJI 2019

Să rezolvăm problema: https://www.infoarena.ro/problema/tairos.

Rezolvare: Vom folosi o parcurgere BFS pentru a determina numărul de noduri aflat la fiecare nivel, adâncimea arborelui și numărul de frunze de pe fiecare nivel. De asemenea, vom reține numărul de frunze aflate la fiecare nivel în arborele creat. Apoi, putem determina numărul de noduri de pe un nivel $i + j$ din arborele construit, unde i este adâncimea unui nivel în arbore, iar j este adâncimea unui nivel în arborele inițial, adunând la nodurile deja prezente acolo produsul $frunzePeNivel[i] * Count[j]$ ($frunzePeNivel[i]$ = numărul de frunze pe nivelul i în noul arbore, $Count[j]$ = numărul de noduri de la nivelul j în arborele inițial).

Implementare: https://www.infoarena.ro/job_detail/3211002.

Galeti OJI 2018

Să rezolvăm problema: https://www.infoarena.ro/problema/galeti2.

Galeti OJI 2018

Să rezolvăm problema: https://www.infoarena.ro/problema/galeti2.

Hint 1: Putem obține un interval în care se află efortul.

Galeti OJI 2018

Să rezolvăm problema: https://www.infoarena.ro/problema/galeti2.

Hint 1: Putem obține un interval în care se află efortul.
Hint 2: Avem două tipuri de turnări:

1. Turnăm ultimele $n - 1$ găleți în găleata 2, iar apoi din găleata 2 turnăm în găleata 1.
2. Turnăm primele $n - 1$ găleți în găleata 1, iar apoi din găleata 2 turnăm în găleata 1.

Galeti OJI 2018

Să rezolvăm problema: https://www.infoarena.ro/problema/galeti2.

Rezolvare: Fie e_i efortul necesar pentru a vărsa i găleți. Putem obține inegalitățile $n - 1 \leq e_i \leq \frac{n(n-1)}{2}$ (turnăm fiecare găleata de la 2 la n în prima găleata, obținând minimul sau turnăm din găleata i în găleata i – 1, de la i = n). Dacă mai adăugăm o găleata la cele n și aplicăm cele două strategii prezentate la Hint-ul 2, obținem că dacă $e_i \leq 2m - 3$, atunci am aplicam prima opțiune, iar daca inegalitatea este inversata, atunci am aplicat a doua.

Implementare: https://www.infoarena.ro/job_detail/3211017.

Superhedgy OJI 2022

Să rezolvăm problema: https://kilonova.ro/problems/135.

Superhedgy OJI 2022

Să rezolvăm problema: https://kilonova.ro/problems/135.

Hint: Încercați să găsiți niște relații pentru valorile minime obținute dacă ajungem până la poziția i deasupra pământului și dedesuptul său.

Superhedgy OJI 2022

Să rezolvăm problema: https://kilonova.ro/problems/135.

Rezolvare: Fie $solAbove[i]$ distanța minimă necesară pentru a trece de primele i clădiri și a fi deasupra pământului iar $solBelow[j]$ analogul său. Obținem recurențele

$$solAbove[i] = \min\{solAbove[i] + vertAbove, solBelow[i] + Elevator + vertAbove\}$$
$$solBelow[i] = \min\{solBelow[i] + vertBelow, solAbove[i] + Elevator + vertBelow\}$$

unde $vertAbove$ și $vertBelow$ sunt distanțele dintre înălțimile a două clădiri adiacente aflate deasupra pământului, respectiv dedesubtul pământului.

Implementare: https://kilonova.ro/submissions/243949.

Ateleport OJI 2020

Să rezolvăm problema: https://kilonova.ro/problems/17.

Ateleport OJI 2020

Să rezolvăm problema: https://kilonova.ro/problems/17.

Hint: Când am găsit soluția pentru un nod, ce ar trebui să știm pentru a determina posibile soluții pentru nodurile vecine?

Ateleport OJI 2020

Să rezolvăm problema: https://kilonova.ro/problems/17.

Rezolvare: Vom implementa Algoritmul lui Dijkstra modificat. Trebuie să reținem valoarea minimă cu care ajungem într-un nod, numărul de teleportări necesare pentru a ajunge în nodul respectiv (pentru a ști dacă pot să mai folosesc teleportări), precum și numărul de muchii traversate în ultima teleportare (pentru a putea decide dacă continui teleportarea). Vom modifica Algoritmul lui Dijkstra pentru a ține cont de aceste noi parametri.

Implementare: https://kilonova.ro/submissions/242468.