Backtracking + Meet in the Middle + Probleme

BUZATU GIULIAN & NIȚĂ ALEXANDROS

Probleme clasice de Backtracking

Să rezolvăm problema: https://infoarena.ro/problema/permutari

Probleme clasice de Backtracking

Să rezolvăm problema: https://infoarena.ro/problema/permutari

Soluție: Reținem elementele curente într-un vector. Completăm $v[i]$ în funcție de numerele alese până acum. Când ajungem la n elemente în vector ne oprim.

Implementare: https://infoarena.ro/job_detail/2486169?action=view-source

Probleme clasice de Backtracking

Să rezolvăm problema: https://infoarena.ro/problema/combinari

Probleme clasice de Backtracking

Să rezolvăm problema: https://infoarena.ro/problema/combinari

Soluție: Vom reține combinarea curentă într-un vector $comb$. Completăm $comb[i]$ cu fiecare element de la $comb[i-1]+1$ la n și până ajungem la k elemente în vector.

Implementare: https://infoarena.ro/job_detail/3199656?action=view-source

Probleme clasice de Backtracking

Să rezolvăm problema: https://infoarena.ro/problema/submultimi

Probleme clasice de Backtracking

Să rezolvăm problema: https://infoarena.ro/problema/submultimi

Soluție: Reținem submulțimea curentă într-un vector $submul$. Completăm poziția $submul[i]$ cu fiecare element de la $submul[i-1]+1$ la n și afișăm fiecare configurație obținută astfel.

Implementare: https://infoarena.ro/job_detail/3199658?action=view-source

Probleme clasice de Backtracking

Observații:

1. Problema este echivalentă cu afișarea tuturor combinărilor de lungime de la 1 până la n.

2. Problema se poate rezolva și generând toate numele de la 1 la $2^n - 1$ în care fiecare bit reprezintă apartenența elementului de pe poziția respectivă la mulțime.

Implementare: https://infoarena.ro/job_detail/3199671?action=view-source

Probleme clasice de Backtracking

Să rezolvăm problema: https://infoarena.ro/problema/damesah

Probleme clasice de Backtracking

Să rezolvăm problema: https://infoarena.ro/problema/damesah

Soluție: Să considerăm vectorul $dame$, unde $dame[i]$ reprezintă coloana pe care se află regina pusă pe linia i. Astfel, fiecare regină este determinată de perechea $(x, dame[x])$, i.e. de linia și coloana pe care se află. Dacă două regine se află pe aceeași diagonală, observăm că ele determină un pătrat. Astfel, condiția pentru care două regine $(x, dame[x])$ și $(y, dame[y])$ să nu fie puse pe aceeași diagpnală este $|x - y| \neq |dame[x] - dame[y]|$.

Implementare: https://infoarena.ro/job_detail/3199659?action=view-source

Problemă

Să rezolvăm problema: https://infoarena.ro/problema/flip

Problemă

Să rezolvăm problema: https://infoarena.ro/problema/flip

Hint: Putem să folosim o problemă clasică?

Problemă

Să rezolvăm problema: https://infoarena.ro/problema/flip

Soluție: Vom genera toate submulțimile formate din coloane. Elementele unei submulțimi sunt coloanele pe care le înmulțim cu -1. Facem suma pe fiecare linie și dacă suma este negativă înmulțim linia respectivă cu -1, i.e. schimbăm semnul sumei. Afișăm maximul obținut din suma acestor sume de pe coloane.

Implementare: https://infoarena.ro/job_detail/3199675?action=view-source

Backtracking în plan

Să rezolvăm problema: https://pbinfo.ro/probleme/342/soarece

Backtracking în plan

Să rezolvăm problema: https://pbinfo.ro/probleme/342/soarece

Soluție: Pornim din (is,js) cu distanța 1, după care facem backtracking și pentru fiecare celulă în care ajungem vedem dacă ne putem deplasa în vecinii ei, crescând distanța.

Implementare: https://github.com/Giulian617/Hai-la-olimpiada-2023-2024/blob/main/11-12/resources/soarece.cpp

Problemă

Să rezolvăm problema: https://cses.fi/problemset/task/1628/

Problemă

Să rezolvăm problema: https://cses.fi/problemset/task/1628/

Hint: Putem împărți cumva vectorul pentru a calcula mai rapid de 2^{40}?

Problemă

Să rezolvăm problema: https://cses.fi/problemset/task/1628/

Soluție: Numărul de elemente este prea mare pentru a putea face backtracking. Totuși, putem să împărțim vectorul inițial în două părți separate, după care calculăm toate posibilitățile pe ambele părți. Pe fiecare parte vom avea maxim 20 de elemente, deci în total 2^{20} de rezultate pe fiecare parte, deci pentru a le combina vom face 2^{21} de operații. Pentru fiecare suma din prima jumătate, vom verifica câte elemente s-ar potrivi cu ea din cea de-a doua jumătate, pentru a face asta vom folosi căutarea binară.
Implementare: https://github.com/Giulian617/Hai-la-olimpiada-2023-2024/blob/main/11-12/resources/MITM.cpp

Problemă

Să rezolvăm problema: https://infoarena.ro/problema/lanterna

Problemă

Să rezolvăm problema: https://infoarena.ro/problema/lanterna

Hint 1: Pentru un tip de lanternă fixat, putem folosi algoritmul lui Dijkstra modificat pentru a afla răspunsul.

Problemă

Să rezolvăm problema: https://infoarena.ro/problema/lanterna

Hint 1: Pentru un tip de lanternă fixat, putem folosi algoritmul lui Dijkstra modificat pentru a afla răspunsul.
Hint 2: Pentru a găsi tipul optim de lanternă folosim căutare binară.

Problemă

Să rezolvăm problema: https://infoarena.ro/problema/lanterna

Soluție: Pentru a afla timpul minim de traversare, vom face Dijkstra cu o lanternă cu capacitatea maximă. Pentru a determina tipul minim al lanternei, vom face căutare binară. Deși folosim algoritmul lui Dijkstra, nu ne mai interesează doar timpul minim în care putem ajunge într-un nod, ci și numărul de watti rămași. Deci, se modifică și modul în care facem tranzițiile de la un nod la celălalt.

Implementare: https://infoarena.ro/job_detail/3171437?action=view-source

Temă

- https://infoarena.ro/problema/avd
- https://pbinfo.ro/probleme/937/hercule
- https://pbinfo.ro/probleme/1518/sudoku
- https://codeforces.com/contest/888/problem/E

Probleme suplimentare

- https://infoarena.ro/problema/badea
- https://infoarena.ro/problema/joc19
- https://infoarena.ro/problema/immortal
- https://codeforces.com/contest/1006/problem/F

Lectură suplimentară

- https://usaco.guide/gold/meet-in-the-middle
- https://codeforces.com/blog/entry/95571