AA - examen 2024

Exercitiu 1

O problemă de decizie care este în clasa **NP**, dar **nu este NP-hard**, ar fi **Problema Parantezelor Echilibrate** (Balanced Parentheses Problem).

Problema Parantezelor Echilibrate

Enunțul problemei:

Este o secvență de paranteze (deschise și închise), trebuie să verifici dacă fiecare paranteză deschisă are o paranteză închisă corespunzătoare și dacă parantezele sunt corect închise într-un mod echilibrat.

De exemplu:

- Secvența `()()` este echilibrată.
- Secvența `(()())` este echilibrată.
- Secvența `(()` nu este echilibrată.

Justificare

1. **Este în NP**:
    - O soluție pentru această problemă este pur și simplu o secvență de paranteze care este echilibrată.
    - Verificarea dacă o secvență de paranteze este corect echilibrată se poate face într-un timp polinomial. Practic, putem parcurge secvența și să ținem evidența deschiderii și închiderii parantezelor.
    - Prin urmare, verificarea unei posibile soluții se poate face în timp liniar, ceea ce înseamnă că este într-o complexitate de timp **polinomială**, deci este în NP.
2. **Nu este NP-hard**:
    - Problema parantezelor echilibrate nu este NP-hard pentru că ea poate fi rezolvată într-un timp polinomial.
    - De fapt, este o problemă foarte simplă și poate fi rezolvată într-un algoritm de timp liniar (O(n)), deci nu face parte din clasa NP-hard.

- NP-hard se referă la probleme care sunt **la fel de grele** ca orice problemă din NP, dar această problemă nu este la fel de greu de rezolvat, având o soluție eficientă în timp polinomial.

Concluzie

Problema parantezelor echilibrate este un exemplu de problemă care este în NP, deoarece soluțiile pot fi verificate rapid, dar nu este NP-hard, deoarece există un algoritm eficient pentru a o rezolva.

Exercitiu 2

Un exemplu de problemă care se află în clasa **NP** dar **nu în clasa NP-completă (NPC)**, presupunând că **P ≠ NP**, este **Problema de Căutare a Căii Minime într-un Graf cu Costuri Pozitive** (Shortest Path Problem).

Enunțul problemei:

Având un graf orientat sau neorientat, cu noduri și muchii, fiecare muchie având un cost pozitiv, trebuie să determini **cea mai scurtă cale** între două noduri date. Problema este formulată astfel:

- **Input:** Un graf $G=(V,E)$, unde V este mulțimea de noduri, E este mulțimea de muchii, și costurile asociate fiecărei muchii.
- **Output:** Cea mai scurtă cale de la un nod sursă la un nod destinație.

Justificare

1. **Este în NP**:
   - Această problemă poate fi **verificată** rapid în timp polinomial. Dacă îți dau o soluție posibilă (un drum între sursă și destinație), poți verifica într-un timp polinomial dacă drumul respectă condițiile cerute și dacă este, într-adevăr, cel mai scurt (adunând costurile pe drum și verificând dacă este mai mic decât alte soluții posibile).
   - Verificarea unei soluții posibile se poate face eficient într-un timp polinomial, deci problema se află în NP.
2. **Nu este NP-completă (NPC)**:

- Problema căutării celei mai scurte căi nu este NP-completă, deoarece **poate fi rezolvată în timp polinomial**. Există algoritmi eficienți pentru a rezolva această problemă, cum ar fi algoritmul lui Dijkstra (pentru grafuri cu costuri pozitive), care funcționează în timp polinomial, de obicei O(VlogV+E)O(V \log V + E)O(VlogV+E) utilizând structuri de date eficiente.
- Dacă ar fi fost o problemă NP-completă, nu am fi avut algoritmi polinomiali pentru a o rezolva. În schimb, faptul că există un algoritm polinomial pentru rezolvare înseamnă că această problemă nu face parte din clasa NP-completă.

Exercitiul 3

Da, există probleme care sunt **NP-hard** dar **nu sunt NP-complete**. Diferența cheie este că **NP-hard** se referă la probleme care sunt cel puțin la fel de dificile ca orice problemă din NP, dar nu sunt neapărat în NP. O problemă NP-hard nu trebuie să aibă o soluție care să poată fi **verificată** în timp polinomial, ceea ce înseamnă că nu face neapărat parte din NP.

Exemplu de problemă NP-hard, dar care nu este NP-completă:

Un exemplu clasic de problemă NP-hard care nu este NP-completă este **Problema Halting (Halting Problem)**.

Problema Halting

Enunțul: Dând un program PPP și o intrare xxx, trebuie să determinăm dacă programul PPP se oprește (adică se oprește după un număr finit de pași) când este rulat pe intrarea xxx.

- Aceasta este o problemă **decizională** (sau de decizie) de interes în teoria computabilității.
- **Halting Problem** nu este în NP, pentru că nu există un algoritm care să poată verifica rapid într-un timp polinomial dacă un program se oprește sau nu, pentru orice program și orice intrare. Acest lucru este demonstrat de **teorema lui Turing**.

Justificare:

1. **Este NP-hard**: Problema Halting este **NP-hard** pentru că orice problemă din NP poate fi redusă la aceasta (de fapt, mai general, oricare problemă din clasa **recursivă** poate fi redusă la Halting Problem). Practic, oricare altă problemă din NP poate fi "encapsulată" într-un program care, atunci când este rulat, simulează acea problemă, iar determinarea dacă programul se oprește poate fi văzută ca un mod de a rezolva acea problemă.
2. **Nu este în NP**: Această problemă nu este în NP deoarece nu există o metodă de verificare rapidă (în timp polinomial) a soluției. Așadar, nu este o problemă pentru care soluțiile pot fi verificate în mod eficient într-un timp polinomial. De fapt, problema este **nelimitată** în complexitate, iar Turing a demonstrat că nu există un algoritm care să rezolve problema pentru toate cazurile.

Alte exemple de probleme NP-hard care nu sunt NP-complete:

- **Problema 3-SAT pentru grafuri infinite**: Dacă avem grafuri infinite (sau foarte mari), anumite variațiuni ale problemei 3-SAT devin NP-hard, dar nu pot fi considerate în NP pentru că nu există un algoritm care să verifice soluția într-un timp polinomial pentru grafuri infinite sau foarte mari.
- **Problema de satisfiabilitate pentru limbaje formale non-deterministe**: Există unele variante de satisfiabilitate pentru limbaje care sunt NP-hard, dar nu pot fi clasificate în NP, deoarece nu se potrivesc cu definițiile problemei NP.

Concluzie:

O problemă poate fi **NP-hard** fără a fi **NP-completă** dacă:

- Este cel puțin la fel de dificilă ca orice problemă din NP (adică poate fi redusă la ea orice problemă din NP),
- Dar nu este în NP, adică nu există un algoritm care să verifice soluțiile într-un timp polinomial.

Exemplul clasic de problemă care este **NP-hard** dar nu **NP-completă** este **Problema Halting**.

Exercitiul 4