1. Algoritmusok bonyolultságának mérése, bonyolultságok típusai. Bonyolultsági függvények, ordó jelölés. Keresési és rendezési algoritmusok, gráfalgoritmusok bonyolultsága.

Algoritmusok bonyolultságának mérése

- **Időbonyolultság:** Az algoritmus futási idejének mértéke a bemeneti adatok méretének függvényében.
  - Általában az algoritmus által végrehajtott alapvető műveletek számával mérjük.
  - Különböző esetekre szokás mérni:
    - Legjobb eset: A lehető legkedvezőbb futási idő.
    - Legrosszabb eset: A lehető legrosszabb futási idő.
    - Átlagos eset: Az átlagos futási idő a bemenetek valamilyen eloszlása esetén.
- **Tárbonyolultság:** Az algoritmus által felhasznált memória mértéke a bemeneti adatok méretének függvényében.

Bonyolultságok típusai

- **Konstans időbonyolultság (O(1)):** Az algoritmus futási ideje független a bemenet méretétől.
- **Logaritmikus időbonyolultság (O(log n)):** Az algoritmus futási ideje logaritmikusan növekszik a bemenet méretével.
- **Lineáris időbonyolultság (O(n)):** Az algoritmus futási ideje arányosan nő a bemenet méretével.
- **Polinomiális időbonyolultság (O(n^k)):** Az algoritmus futási ideje a bemenet méretének valamilyen hatványával nő.
- **Exponenciális időbonyolultság (O(2^n)):** Az algoritmus futási ideje exponenciálisan nő a bemenet méretével.
- **Faktoriális időbonyolultság (O(n!)):** Az algoritmus futási ideje faktoriálisan nő a bemenet méretével.

Bonyolultsági függvények, ordó jelölés

- **Bonyolultsági függvény:** Az algoritmus futási idejét vagy tárigényét leíró matematikai függvény.
- **Ordó jelölés (Big-O notation):** Az algoritmus futási idejének vagy tárigényének felső határát jelöli, a bemenet méretének függvényében.
  - **Példák:**
    - $O(1)$: Konstans időbonyolultság.
    - $O(n)$: Lineáris időbonyolultság.
    - $O(n \log n)$: Lineáris-logaritmikus időbonyolultság.
    - $O(n^2)$: Négyzetes időbonyolultság.
    - $O(2^n)$: Exponenciális időbonyolultság.

Keresési algoritmusok bonyolultsága

- **Lineáris keresés:**
  - Legrosszabb esetben $O(n)$
  - Az algoritmus sorban ellenőrzi a bemenet összes elemét.
- **Bináris keresés:**
  - Legrosszabb esetben $O(\log n)$
  - Az algoritmus rendezetten keres a bemenet elemei között, mindig felezi a keresési tartományt.

Rendezési algoritmusok bonyolultsága

- **Buborék rendezés (Bubble Sort):**
  - Időbonyolultság: $O(n^2)$
  - Az algoritmus többször végigmegy a listán, és mindig kicseréli a szomszédos elemeket, ha azok rossz sorrendben vannak.
- **Beszúrásos rendezés (Insertion Sort):**
  - Időbonyolultság: $O(n^2)$
  - Az algoritmus minden elemet a helyére tesz a már rendezett részlistában.
- **Összefésüléses rendezés (Merge Sort):**
  - Időbonyolultság: $O(n \log n)$
  - Az algoritmus felosztja a listát kisebb részekre, majd azokat rendezi és összefésüli.

- **Gyors rendezés (Quick Sort):**
  - Átlagos időbonyolultság: $O(n \log n)$
  - Legrosszabb esetben: $O(n^2)$
  - Az algoritmus kiválaszt egy pivot elemet, és a listát kisebb és nagyobb elemekre osztja, majd rekurzívan rendezi azokat.

Gráfalgoritmusok bonyolultsága

- **Szélességi keresés (Breadth-First Search, BFS):**
  - Időbonyolultság: $O(V+E)$
  - Az algoritmus rétegszerűen bejárja a gráf csúcsait.
- **Mélységi keresés (Depth-First Search, DFS):**
  - Időbonyolultság: $O(V+E)$
  - Az algoritmus mélységben bejárja a gráf csúcsait.
- **Dijkstra algoritmus:**
  - Időbonyolultság: $O(V^2)$ (Prioritási sor nélkül)
  - Prioritási sorral: $O((V+E)\log V)$
  - Az algoritmus a legkisebb súlyú utat találja meg egy forráscsúcsból a többi csúcsba.
- **Floyd-Warshall algoritmus:**
  - Időbonyolultság: $O(V^3)$
  - Az algoritmus a legrövidebb utat találja meg minden csúcs között a gráfban.