Eötvös Loránd Tudományegyetem

Informatikai Kar

Algoritmusok és Alkalmazásaik Tanszék

Rendezési algoritmusok szemléltetése

Veszprémi Anna
mestertanár

Márföldi Péter Bence
programtervező informatikus BSc

Budapest, 2015

Tartalomjegyzék

1. fejezet

Bevezetés

Az bizonyos, hogy minden informatikus - beleértve a leendőeket is - tanulmányaik kezdetén találkoztak a rendezési algoritmusokkal. Nagyszerű terület arra, hogy megérthessük a műveletigény kérdését, azt hogy mi számít igazán sok adatnak, vagy éppen, hogy mit értünk egy algoritmus stabilitásán.

1.1. A feladat és annak értelmezése

1.2. Definíciók és jelölések

1.3. Alkalmazott technológiák

A Következőkben röviden összefoglaljuk a Java[1] és a JavaFX[2] jellegzetességeit.

1.3.1. A Java-ról röviden

A Java egy általános célú, objektumorientált programozási nyelv, melyet 2009-ig a *Sun Microsystems* fejlesztett, ezt követően pedig az *Oracle*. A szakdolgozatban használt 1.8-as verziót már az *Oracle* adta ki 2014-ben. A Java nyelv a szintaxisát a C és C++ nyelvektől örökölte, azonban utóbbitól eltérően egyszerű objektummodellel rendelkezik.

A Java platformra készült programok túlnyomó többsége asztali alkalmazás. Manapság egyre több helyen találkozhatunk a Java nyelven írt programokkal, például mobil eszközökön, banki rendszereknél vagy akár egy szórakoztató elektronikai eszközön. Nagy előnye, hogy sok nyelvvel ellentétben platformfüggetlen, azaz egy adott platformról egy program minimális változtatással átültethető egy másik platformra.

A Java legfontosabb része a *Java virtuális gép (JVM)*. A *JVM*-et sokféle berendezés és szoftvercsomag tartalmazza, így a nyelv egyaránt platformként és kö-

zépszintként is működik. Összefoglalva a Java program három fontos szerepet tölt be:

- programozási nyelv
- köztes réteg (middleware)
- platform

1.3.2. JavaFX

Olyan szoftverplatform, amelynek célja, hogy gazdag internetes alkalmazást lehessen készíteni és futtatni eszközök széles skáláján. Eredetileg a *Swing* könyvtárat váltotta volna fel, azonban jelenleg mindkettő része a *Jave SE*-nek.

A 2.0-ás verzióig a fejlesztők egy külön nyelvet használtak, amelyet *JavaFX Script*-nek neveznek. Azonban mivel ez szintén Java bájtkódot generál a későbbiekben megadatott a lehetőség, hogy a programozók Java kódot használjanak helyette. A JavaFX egyik legnagyobb előnye, hogy egy egyszerű *XML* struktúrában leírhatók a program grafikus felületének összetevői, melyhez ezt követően elegendő az egyes interakciókhoz tartozó funkciókat implementálni.

Az elterjedtebb operációs rendszerek mindegyikét támogatja. Ahogyan előnye, úgy hátránya is a *Swing*-hez képest az, hogy jelenleg is folyik a fejlesztése, ezért olykor csak hosszas utánajárást követően sikerül megoldást találni egy-egy problémára.

2. fejezet

Felhasználói dokumentáció

2.1. A vizsgált algoritmusok

2.1.1. Buborékrendezés

A legrégebbi és a legegyszerűbb rendezési algoritmus. Mindemellett a legtöbb esetben a leglassabb is. Már az 1965-ös évben megjelent egy teljes körű elemzése[3].

A rendezés minden egyes elemet összehasonlít a rákövetkező elemmel, és ha szükséges megcseréli őket. Mindezt addig, amíg nincs egy olyan menet, amelyben egyetlen elem sem cserél helyet. Ez azt eredményezi, hogy lépésenként a maximális elem "buborék" szerűen a lista végére kerül, ezzel egyidejűleg a kisebb elemek "lesüllyednek" a tömb elejére. Az algoritmus javítható azzal, hogy nem vizsgáljuk meg minden menetben a tömb összes elemét, hanem amennyiben egy maximális elem elérte a helyét visszavezetjük a problémát az eggyel "rövidebb" rendezési feladatra[4].

2.1.2. Beszúró rendezés

A nevéből egyszerűen kikövetkeztethető az eljárás: beszúrja az elemeket a megfelelő helyükre a végleges tömbbe. Lényege, hogy a soron következő elemet egy ideiglenes változóba mentjük, és a rendezett tömb elemeit jobbra csúsztatjuk, mindaddig amíg a kiválasztott érték nem kerül a helyére. Kezdetben a tömb első elemét tekintjük rendezettnek. A legtöbb esetben akár kétszer hatékonyabb a Buborékrendezéshez képest.[4] Továbbá kis (néhány száz) elemszámú bemenetre az egyik leghatékonyabb algoritmus.

2.1.3. Shell rendezés

Donald Shell nevéhez fűződik, a legtöbb esetben a leggyorsabb négyzetes idejű algoritmus. Többször vizsgálja a tömböt, és minden alkalommal egy részén beszúró

rendezést hajt végre. Arra, hogy mekkora méretű résztömböt vizsgáljon az egyes lépésekben az algoritmus több javaslat is található. Az algoritmus nevét is adó Donald Shell $\lfloor N/2^k \rfloor$ (k>=1) lépésközt javasol. Azonban így az algoritmus időkomplexitása legrosszabb esetben $\Theta(N^2)$. Az algoritmus sebessége nagyban függ a lépésköz megválasztásától.

2.1.4. Gyorsrendezés

Helyben rendező, oszd meg és uralkodj[5] elven működő rekurzív algoritmus. A következő négy lépésre bontható fel az algoritmus:

- Ha csak egy vagy nulla elemű az elemzett rész, akkor ne tegyünk semmit.
- Válasszunk egy vezérelemet (legjobb oldalibb elem).
- A rendezendő részt vágjuk ketté, az egyik oldalán a vezérelemtől kisebb, míg a másikon a nagyobb elemek kerüljenek.
- Rekurzívan ismételjük meg az előbbi lépéseket résztömbön.

A rendezés műveletigényét befolyásolja, hogy hogyan választjuk meg a vezérelemet. Például a legnagyobb műveletigényt ($\mathcal{O}(n^2)$) eredményezi, ha mindig a legjobboldalibb elemet választjuk vezérelemnek, és a tömb elemei csökkenő sorrendben vannak. Éppen ezért a gyakorlatban javasolt ezen elem véletlenszerű megválasztása.

2.1.5. Versenyrendezés

A maximum-kiválasztó rendezések közé tartozik, minden egyes menetben kiválasztja a legnagyobb elemet, kiírja és végül eltávolítja. A maximum kiválasztásnak a gyakorlati hátterét a sportesemények lebonyolítási rendje adja, azaz meghatározza az elemek között a "nyertest". A módszert n=2^k inputhossz esetén érdemes alkalmazni, mivel ettől értérő bemenetre sokkal kedvezőbb eredményt lehet elérni a kupacrendezéssel. Az algoritmus által használt adatszerkezet egy teljes bináris fa.

3. fejezet

Fejlesztői dokumentáció

Irodalomjegyzék

[1] *Java (programming language)*, Wikipedia the free encyclopedia. [ONLINE] [Hivatkozva: 2015.04.21] `http://en.wikipedia.org/wiki/Java_(programming_language)/`

[2] *JavaFX*, Wikipedia the free encyclopedia. [ONLINE] [Hivatkozva: 2015.04.21] `http://en.wikipedia.org/wiki/JavaFX/`

[3] Demuth, H.: *Electronic Data Sorting*, PhD thesis, Stanford University, 1956, [184]

[4] Dr. Fekete István: *Algoritmusok és adatszerkezetek I. jegyzet*, [ONLINE] [Hivatkozva: 2015.04.20] `http://people.inf.elte.hu/fekete/algoritmusok_bsc/alg_1_jegyzet/`

[5] Umesh V. Vazirani: *Alrogithms* [ONLINE] [Hivatkozva: 2015.04.23] `https://www.cs.berkeley.edu/~vazirani/algorithms/chap2.pdf`