Keresési módszerek összehasonlító elemzése

Készítette: Herbák Marcell – QAGSVA

# **Bevezetés**

Ön hogyan szokott menni a boltba vásárolni? Egyből a szükséges cikkekhez megy, vagy esetleg végigmegy a sorok között, hátha talál valami érdekeset? Esetleg csak benéz a polcok közé, hátha ott van, amit keres? Ha ez nem tűnt fel eddig Önnek, semmi gond, hiszen mind a vásárlás, mint a keresési módszerek a mindennapjaink része!

Bár mi emberek természetesnek vesszük, ha valamit keresünk akkor valamilyen úton, különböző jelzők és tulajdonságok alapján próbáljuk megtalálni a keresett dolgot, mégsem mindegy, hogy mi is, hogy bánunk az idővel, tehát mennyi idő egy bevásárlás, mennyire fáradunk el benne, ismerjük-e a bolt elrendezését, és hogy egyáltalán mit keresünk. Ha egyszer csak megállunk a boltban és elgondolkozunk, több kifejlet is lehet:

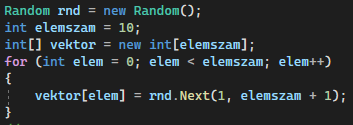
1. Elfogyott a tejföl!
2. Még nem találtunk tejfölt!

Akár tejfölt, akár bármit megtalálni a számítógépen is lehetséges (azért tejföl remélhetőleg nincs benne), az úgynevezett keresési módszerekkel, kereső algoritmusokkal. Ezek az algoritmusok segítenek egy halmazban/gyűjteményben megtalálni egy/több adott tulajdonsággal rendelkező elemet, és ennek a létezéséről jelentést tehet (benne van/nincs benne), visszaadhatja a pozícióját a gyűjteményen belül stb. Tehát keresni nem csak a boltban, és számítógépen lehet, hanem adatbázisokban, interneten, tömbökben, mátrixokban, matematikai terekben.

# Keresőalgoritmusok

Visszaidézve, nem mindegy, hogy a boltba életünkben először megyünk vagy már ismerjük a belső terének elrendezését, és optimálisabban tudunk keresni. Az algoritmusoknál szintén lehet ilyen módon különbséget tenni, azonban azt le kell szögezni, hogy a számítógép „sosem” fogja tudni, mi található a „boltban”, hiszen a legoptimálisabb algoritmus is jóformán „tippel”. Tudományosan értve tehát a számítógép különböző, felhasználó által betáplált lépéseken megy végig, így keresve az adott tulajdonságú elemet. Erről részletesebben a későbbiekben lesz szó!

## Rendezetlen sorozatokon

 A könnyebb szemléltetés érdekében az algoritmusokat számsorozatokon lesznek végrehajtva, hiszen a szám maga a „T” tulajdonság, a számokat tartalmazó „vektor” tömbben pedig ezt keressük! Ezek véletlenszerű pozitív egész számok lesznek, a tömbök jobb esetben nem ismétlődnek, és a létrehozás sorrendjét nem módosítjuk, tehát amilyen sorrendbe a tömbhöz hozzá lett adva a szám, úgy marad (pozíciója nem változik).

**A tömb feltöltése**

# Lineáris keresés

A képen szöveg látható

Automatikusan generált leírásA legegyszerűbb keresőalgoritmus, leánykori nevén szekvenciális keresőalgoritmus. Addig fut, amíg nem találja meg a T tulajdonságú elemet, vagy vége nem lesz a tömbnek.

Ebben az esetben az algoritmus visszaadja a pozícióját az adott tulajdonságú elemnek, ha pedig nem található benne, visszaadja a „-1” nem lehetséges értéket, tehát az elem nem található meg a tömbben.

A képen szöveg látható

Automatikusan generált leírásAz „i” változó egyben azonban a lépésszámot is jelenti, amennyiben az elem megtalálható benne. **Egészítsük ki a megírt keresést, hogy lépésszámot is nézze!**

A képen szöveg, eszköz, mérőeszköz, vezérlőpult látható

Automatikusan generált leírásA képen szöveg látható

Automatikusan generált leírásEzt a tulajdonságát figyelembe véve, a lépésszám meghatározható. Legjobb esetben a vektor elején van, akkor a lépésszám 1 lesz, legrosszabb esetben a végén, akkor a lépésszám megegyezik a tömb hossza + eggyel, mivel az utolsó elemnél még lefut egyszer a while! **Nézzük meg, hogyha lefut a program, mit kapunk? A tulajdonságom a „6” lesz, egy 10 elemű tömbön!**

Mint látható, a tömbben nincs benne a „6”, ezért a pozíciója -1, a lépésszám pedig 11.

**Most nézzük meg, hogy átlagosan a 6-os tulajdonságra mi jellemző!**

|  |  |  |  |
| --- | --- | --- | --- |
| Tulajdonság | Elemszám | **Pozíció** | **Lépésszám** |
| 6 | 10 | -1 | 11 |
| 6 | 10 | 4 | 4 |
| 6 | 10 | 6 | 6 |
| 6 | 10 | 1 | 1 |
| 6 | 10 | 10 | 10 |
| **Átlag:** | | **5,25** | **6,4** |

|  |  |  |
| --- | --- | --- |
| **Lefutás** | **Pozícióátlag** | **Lépésátlag** |
| **10** | 5,2 | 3,75 |
| **100** | 4,56 | 6,43 |
| **1000** | 4,486 | 6,968 |
| **10000** | 4,656 | 6,8598 |

A többszörös elemzés után a pozícióátlag és lépésátlag normalizálódik, egyre kevesebb a különbség a mérések között, 10x lefutás után még nagyok a kilengések, míg mindig növelve csökken a különbség, azaz meghatározható a 10 elemű tömbben a 6-os átlagpozíciója és a keresésének átlaglépésszáma rendezetlen vektoron.

**Közelítsük meg más oldalról! A tulajdonságom ugyanúgy a „6” lesz, de növeljük az elemszámot! Az elemszám legyen a 100!**

A képen szöveg, eszköz, mérőeszköz látható

Automatikusan generált leírásLehet érzékelni a lineáris keresés hátrányát, hiszen minél hosszabb a tömb, annál tovább kell keresni benne, azaz a futási idő nő!

|  |  |  |
| --- | --- | --- |
| **Lefutás** | **Pozícióátlag** | **Lépésátlag** |
| **10** | **22,3** | **30,2** |
| **100** | **39** | **61,32** |
| **1000** | **41,123** | **61,421** |
| **10000** | **42,56** | **64,05** |

A trend azonban itt is megfigyelhető, hogy többszörös lefutás után az átlagok normalizálódnak!

A képen szöveg, eszköz, mérőeszköz, közel látható

Automatikusan generált leírásA képen szöveg látható

Automatikusan generált leírásMegvizsgáltuk, egy rendezetlen sorozatban az átlagpozíciót és lépésátlagot. **Vizsgáljuk meg a futási időt is akkor! Egészítsük ki a futtatást!**

|  |  |
| --- | --- |
| **Lefutás** | **Időátlag (ms)** |
| **10** | 7,9 |
| **100** | 7,715 |
| **1000** | 7,6295 |
| **10000** | 7,7022 |

A futási idő normalizálódik, minél többször futtatjuk a keresést. **Most nézzük meg, hogy nő a futási idő, ha nő az elemszám!**