*Összehasonlítás a rendezések között*

*1. Beadandó feladat Speciális algoritmusok tantárgyból*

Kalmár Dániel

Tomcsik Bence Tibor

**Feladat**

Készítsünk olyan programot, amely összehasonlítást végez a maximumkiválasztó rendezés, a beszúró rendezés, valamint a gyorsrendezés műveletigénye között! A rendezések inputja az számok véletlen (azaz, egyenlő valószínűségű) permutációi legyenek, ahol az és a generált mennyiség megadható. A hatékonyság mérhető időben, illetve a meghatározó művelet(ek) számában.

**Kérdések**

1. Tudnánk-e ellenőrizni azt a tapasztalati tényt, hogy a beszúró rendezés kis *n*-re gyorsabb, mint a gyorsrendezés? Mennyi lehet ez az *n* korlát? (Az irodalom 30 körüli értékre teszi.)
2. A maximumkiválasztó rendezés gyorsabb-e, mint a beszúró rendezés, azáltal, hogy átlagosan kevesebb elemmozgatást végez?

**Feladat megoldásához használt algoritmusok leírása**

1. **Maximumkiválasztó rendezés:**

Kezdetben az egész tömbön kiválasztjuk a maximális elemet és ezt kicseréljük a tömb legutolsó elemével. Ezt követően vesszük az egyel kisebb résztömböt ami nem tartalmazza az utolsó elemet és ezen is megkeressük a maximális elemet amit kicserélünk a résztömb legutolsó elemével. Ezt addig folytatjuk míg el nem értük a kételemű résztömböt és el nem végeztük az utolsó párra a cserét.

Hátránya, hogy nagyon lassú ezért csak olyan esetekben érdemes használni, amikor a cserék költsége magas, ugyanis ez a rendezés a cserék számát minimalizálja.

1. **Beszúró rendezés:**

Vegyük a tömb első elemét. Ez egy egyelemű rendezett részsorozat. Vegyük fel a résztömb első elemét és hasonlítsuk össze a már rendezett részsorozattal. Ha szükséges cseréljük ki. Ezt folytatva mindig felveszünk egy elemet a résztömbből és beszúrjuk a rendezett részsorozatunkban a megfelelő helyre, a többi elemet pedig eltoljuk a megfelelő helyre.

Hátránya, hogy hasonlóan a maximumkiválasztó rendezéshez, ez is lassú, ellenben rendkívül hatékony ha kis elemszámú sorozatra alkalmazzuk, vagy ha a sorozat közel rendezett.

1. **Gyorsrendezés:**

A rendezés lényege, hogy tetszés szerint kiválasztunk egy elemet, kivesszük a sorozatból, majd a sorozat többi tagját a következőképpen helyezzük el a kivett elem két oldalán:

- egyik oldalra a kivett elemnél nem nagyobbakat  
- másik oldalra a kivett elemnél nagyobbakat.   
Az így létrejött sorozat mindkét oldalán ismét végrehajtjuk a rendezést, amíg teljesen rendezett sorozatot nem kapunk.

1. **3-as felosztású gyorsrendezés:**

Az gyorsrendezéshez képest nem 2 részre bontjuk a sorozatot hanem 3 részre. A három csoport a főelemnél kisebb, vele egyenlő, és annál nagyobb elemek lesznek. Így a középső csoport már a megfelelő helyen van és "rendezett" (mivel egyenlők az elemek), így az kihagyható a rekurzióból. Ez a rendezés akkor hatékonyabb a sima gyorsrendezésnél, ha sok egyenlő elem van.

1. **Optimalizált gyorsrendezés:**

Kis elemszámú sorozat esetén beszúró rendezést használ, egyébként gyorsrendezést.

**Forráskód**

A feladatot C# programozási nyelven oldottuk meg .NET 4.0 verzióval.

**Program használata**

A programot a specalg-sorts.exe fájlal indíthatjuk el.

**3 funkció** használata közül választhatunk a program futtatásakor:

1. *"Lets sort!":*

Amely az általunk kiválasztott rendezéseket hajtja végre az általunk megadott paraméterekkel. Eredményét a Log tab alatt láthatjuk.

1. *"Lets race!":*

Amely a beszúró- és a gyorsrendezést hasonlítja össze 1-50 közötti elemszámon. Eredményét a Log tab alatt, valamint a Race chart tab alatt láthatjuk.

1. *"Lets race 2!":*

Amely az összes rendezés eredményét mutatja be 1-100 közötti elemszámon. A Log és a Race chart all tab alatt látható az eredménye.

Az összehasonlítások alapja a valós idő amely mellett a belső iterációk, tömb hozzáférések és cserék száma is mérve vannak.

**Eredmények**

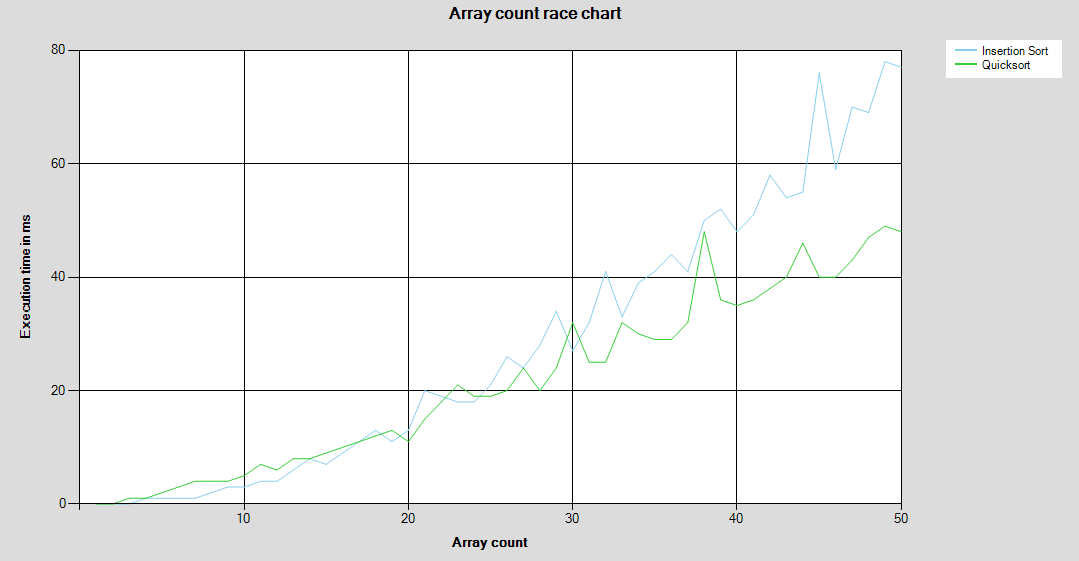
**Válaszok a kérdésekre:**

1. Annak ellenére, hogy az irodalom 30 körüli értékre teszi ezt az elemszámot, több futásnál is látható volt, hogy ez az érték 20 és 30 között változott. Azonban az esetek többségében valóban 30 körülire tehető amikor már egyértelműen jobban teljesít kivétel nélkül. Kivétel az olyan esetek, amikor a sorozat nagyon redundáns, ugyanis ekkor a beszúró rendezés nyer minden esetben.
2. Nem lesz gyorsabb a maximumkiválasztó rendezés a beszúró rendezésnél. Ez bizonyítható a "Lets sort!" funkció használatával. Minden esetben gyorsabb lesz a beszúró rendezés, mert a sokkal jobban alkalmazkodik a sorozat elemeihez, míg a maximumkiválasztó rendezés egyáltalán nem alkalmazkodik az adatokhoz.

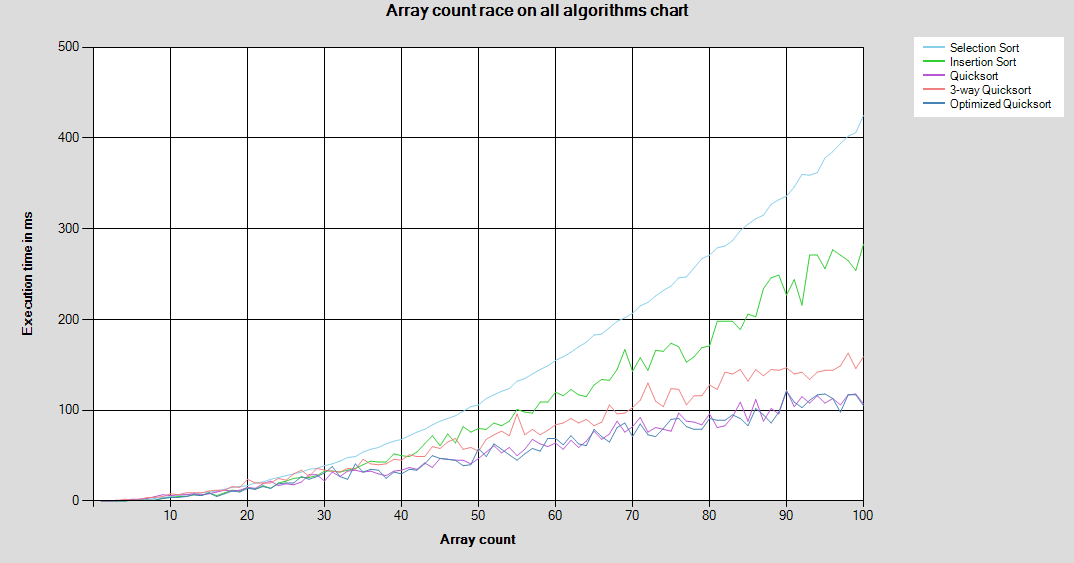
Egyetlen előnye az lehetne a beszúró rendezéssel szemben, ha arra akarunk optimalizálni, hogy minél kevesebb elemmozgatást végezzünk ha annak nagy a költsége.

**Grafikonok, futási eredmények**

Beszúró- és gyorsrendezés egy futási eredménye 1-50 közötti elemszámú sorozaton:



Az 5 rendezés futási eredménye 1-100 közötti elemszámú sorozaton:



**Megjegyzés:**

Az rendezéseket többször (a race funkciók esetében 10000-szer, a sort funkció esetében felhasználó által választott mennyiségszer) kellett lefuttatni, különben kis elemszám esetén nem mérhető a rendezések közötti teljesítmény különbség.