# 8. gyakorlat

## Gyorsrendezés (Quick​Sort)

Ugyan az az alapelve, mint az összefésülő rendezésnek. Rokonrendezések.

Divide and Conquer elv. Megosztjuk, majd a részmegoldásokat összerakjuk.

konstans, nemnegatív egész szám.

### Vegyes Merge​Sort

|  |  |
| --- | --- |
|  | |
|  |  |
|  | |
|  | Ha elég kicsi a tömb, áttérünk beszúró rendezésre |

Egyirányú láncolt listák optimális rendező programja.

### Vegyes Quick​Sort

|  |  |  |
| --- | --- | --- |
|  | | |
|  |  | |
|  | | |
| A két fél összerakása üres program. Már eleve ott van. | | Ha elég rövid a tömb, áttérünk egyszerűbb rendezésre |

Tömbök optimális rendező programja. Tapasztalat ezt mutatja.

Mekkora legyen a konstans? között valahol. Sok mindentől függ.

Előrendezett bemenet esetén magas, random esetén alacsony. Hardvertől, Processzor CACHE-től is függ.

Könyvtári csomag beállíthatja -t az érzékelt hardvertől függően. De ez nem része a tananyagnak.

Ha , akkor egyelemű a tömb.

A tömb szétvágásán () sok múlik.

* Szélső elem levágása előrendezett vagy monoton szakaszokat tartalmazó bemenetre nem működik jól.
* Középen szétvágás hasonlóképp, bár erre nehezebb példát mondani.
* Véletlenszerű helyen szétvágás jó.

### Divide előtt nézzünk egy példát

A bemenet véletlenszerű szétvágása:

-öt választja.

Kivesszük ideiglenes változóba, , helyére betesszük a -at.

Balról , jobbról elindul középre. Most a megy, mert épp lyukra mutat. Lép kettőt, -ig. Majd is lép egyet, mert biztosan kisebb, mint .

-at berakjuk a lyukba, most megy jobbra.

3-mat berakjuk a lyukba. Most és ugyan oda mutat:

Ide beírjuk az -öt:

Most lehet szétvágni. (?)

### Divide

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | |
|  | | | |  | | | |
|  | | | | | | | |
|  | | | | | | | |
|  |  | | | | | | |
|  | | | |  | | |
|  |  | | |  |  | |
|  | | | |  | | |
|  | | vagy\* | |  | | vagy\* |
|  | | | | | | | |

\* szép strukturált lesz a program, csak egy helyen lép ki a ciklusból. áttekinthetetlen a program, de gyorsabb.

Annak, hogy a random mindig a minimumra/maximumra mutat és lelassítja a programunkat, a valószínűsége nagyon kicsi.

## Kupacrendezés, piramisrendezés (Heap​Sort)

Intelligens maximumrendező.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  | |  | |
|  | | | |
|  | | |  |
|  | | |
|  |  | |
|  | | |

### Kupac (Heap)

Bináris fa: Minden elemnek van 1 vagy 2 leszármazottja és 1 őse. Gyökérnek nincs őse.

Vektor elemeit bináris fába szintfolytonosan elhelyezzük.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | | | | | | | |
| 3 | | | | 5 | | | |
| 2 | | 4 | | 8 | | 9 | |
| 5 | 4 | 3 |  |  |  |  |  |

### Heapify

Úgy alakítja át a piramist, hogy a szülő mindig ≥ lesz, mint a gyerekei.

(Nem látom a táblát. Írom, amit hallok…)

Alulról felfelé. Alsó levélszint egyelemű piramisok.

Fel egy szinttel, jó helyen van-e?

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | | | | | | | |
| 5 | | | | 9 | | | |
| 4 | | 4 | | 8 | | 5 | |
| 2 | 3 | 3 |  |  |  |  |  |

Csere a nagyobb gyerekkel.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | | | | | | | |
| 5 | | | | 9 | | | |
| 4 | | 4 | | 8 | | 5 | |
| 2 | 3 | 3 |  |  |  |  |  |

Lesüllyesztjük a megfelelő ágon.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | | | | | | | |
| 5 | | | | 8 | | | |
| 4 | | 4 | | 1 | | 5 | |
| 2 | 3 | 3 |  |  |  |  |  |

.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | | | | | | | |
| 5 | | | | 5 | | | |
| 4 | | 4 | | 1 | | 3 | |
| 2 | 3 | 9 |  |  |  |  |  |

.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | | | | | | | |
| 5 | | | | 5 | | | |
| 4 | | 4 | | 1 | | 3 | |
| 2 | 8 | 9 |  |  |  |  |  |

HF befejezni.