# 4. gyakorlat

## Előrendezettség, Fejelemes ciklikus kétirányú láncolt lista

A bemenet tendenciájában monoton növő, de van néhány elem, amelyik kilóg a sorból.

Ekkor a tömbös rendezés ugyan olyan jól működik, de az egy irányba fűzött listás rendezés lassú.

Két irányba fűzött lista esetén viszont jó.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| L→ | NULL | adat | → |  | ← | adat | → |  | ← | adat | → |  | ← | adat | NULL |

Ha nincs körbefűzve, az első és utolsó elemmel másképp kell bánni.

Ha körbe van láncolva, a szélső elemek kifűzése nem gond. De üres listába még mindig nem tudunk elágazás nélkül befűzni.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| L→ | ← | adat | → |  | ← | adat | → |  | ← | adat | → |  | ← | adat | → |

Ha még egy fejelemet is bevezetünk, akkor már minden rendben. Megspórolunk vagy 4 elágazást. Így minden új elemet mindig két elem közé tudunk befűzni.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| L→ | ← | NULL | → |  | ← | adat | → |  | ← | adat | → |  | ← | adat | → |

Fejelemes ciklikus kétirányú láncolt lista.

Természetesen ekkor kétszer annyi pointert kell átállítani minden ki-/befűzésnél.

## Kétirányú ciklikus fejelemes láncolt lista alapműveletei

Egy "láncszem":

|  |  |  |
| --- | --- | --- |
| ←bal | adat | jobb→ |

### Létrehozás

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

### Kifűzés

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

Korábban kifűzött elem újra kifűzése esetén nem történik semmi. Ez jól is van így.

### Destruktor

Kifűzheti a láncszemet, de​allokálás előtt:

|  |
| --- |
|  |
|  |

### Befűzés, átfűzésre optimalizálva

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

Mindig az elemet fűzöm az adatszerkezethez először, és csak utána fűzöm az adatszerkezet az elemhez. (Hogy minél rövidebb legyen az inkonzisztencia. Ha elszáll a program, csak az egy láncszem veszik el.)

Mi történik, ha -val hívjuk meg? Elromlik az adatszerkezet.  
Ennek elkerülése miatt az egész fölé kell írni egy elágazást:

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

### Befűzés, most jobbra, már hibakezeléssel

|  |  |  |
| --- | --- | --- |
|  | | |
|  |  | |
|  | | |
|  | | H  I  B  A |
|  | |
|  | |
|  | |
|  | |
|  | |

## Beszúró rendezés a fenti eljáráscsomaggal

(utolsó elem az elejére visszamutat)

Insert sort:   
Kettévágjuk a listát rendezett és rendezetlen szakaszra.  
 rendezett szakasz vége  
 rendezetlen szakasz eleje

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

Ilyen feladat lesz a ZH-n.

## Maximumkiválasztásos rendezés

Lényegében ugyan az, mint a minimum​kiválasztásos. Mindkettő szörnyen nem hatékony.

Jobbról balra keresés jobb, ha előrendezett a lista, de nem nyerünk vele túl sokat.

A ciklus addig megy, amíg már csak egy eleme van a rendezetlen szakaszon.

A lista ciklikussága miatt minden szélsőséges esetre jól működik.

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

Hatékonyság: Ciklusfeltétel kiértékelések száma + elágazás kiértékelések száma = összehasonlítások száma.

A ciklusfeltétel -szer fog kiértékelődni.

A belső összehasonlítás -ször fog végrehajtódni.

A másik -szer fog.

Összesen:

Tehát a beszúró rendezés a jobb.