Elemi dinamikus halmazok

1. **Definíció.** *Dinamikus halmaz Az olyan halmazt, amely az őt felhasználó algoritmus során változik (bővül, szűkül, módosul) dinamikus halmaznak nevezzük.*

2. Definíció

Lekérdező műveletek	
KERES (S, k, x)	adott k kulcsú elem x mutatóját adja vissza, vagy NIL, ha nincs.
MINIMUM (S, x)	A legkisebb kulcsú elem mutatóját adja vissza
MAXIMUM (S, x)	A legnagyobb kulcsú elem mutatóját adja vissza
KÖVETKEZŐ (S, x, y)	az x elem kulcsa utáni kulcsú elem mutatóját adja vissza,NIL, ha x után nincs elem
ELŐZŐ (S, x, y)	az x elem kulcsa előtti kulcsú elem mutatóját adja vissza,NIL, ha x előtt nincs elem

Módosító műveletek	
BESZÚR (S, x)	az S bővítése az x mutatójú elemmel
TÖRÖL (S, x)	az x mutatójú elemet eltávolítja S-ből

3. Definíció. *(A sorozat adatstruktúra) Sorozatnak nevezzük az objektumok (elemek) olyan tárolási módját (adatstruktúráját), amikor az elemek a műveletek által kijelölt lineáris sorrendben követik egymást. Tipikus műveletek: keresés, beszúrás, törlés.*

4. Definíció. *(Tömb) A* tömb *azonos felépítésű (típusú) egymást fizikailag követő memóriarekeszeket jelent. Egy rekeszben egy elemet,* adatrekordot *helyezünk el. Az egyes tömbelemek helyét az indexük határozza meg. A tömb* attributumai*:*

Attributum	Leírás
fej[A]	A tömb első elemének indexe. NIL, ha a tömbnek nincs eleme.
vége[A]	A tömb utolsó elemének indexe. NIL, ha a tömbnek nincs eleme.
hossz[A]	A tömbelemek száma. Zérus, ha a tömbnek nincs eleme.
tömbméret[A]	annak a memóriaterületnek a nagysága tömbelem egységben mérve, ahová a tömböt elhelyezhetjük. A tömb ezen terület elején kezdődik.

5. Definíció. *(Láncolt lista adatstruktúra) A láncolt lista (linked list) olyan dinamikus halmaz, melyben az objektumok, elemek lineáris sorrendben követik egymást. A lista minden eleme mutatót tartalmaz a következő elemre. Műveletei: keresés, beszúrás, törlés.*

6. **Definíció.** *(A verem adatstruktúra) A verem (stack) olyan dinamikus* halmaz*, amelyben előre meghatározott az az elem, melyet a TÖRÖL eljárással eltávolítunk. Ez az elem mindig az időben a legutoljára a struktúrába elhelyezett elem lesz. Műveletei: beszúrás (push), törlés (pop). Az ilyen törlési eljárást Utolsóként érkezett – Elsőként távozik (Last In – First Out, LIFO) eljárásnak nevezzük.*

7. Definíció. *(A sor adatstruktúra) A sor (queue) olyan dinamikus halmaz, amelyben előre meghatározott az az elem, melyet a TÖRÖL eljárással eltávolítunk és az az elem is amelyet a BESZÚR eljárással a halmazba beteszünk. Törlésre mindig az elemek közül a legrégebben beszúrt kerül. A beszúrt elem lesz a legfrissebb elem. Műveletek: beszúrás, törlés. Az ilyen törlést Elsőként érkezik – Elsőként távozik (First In – First Out, FIFO) eljárásnak nevezzük.*

8. Definíció (*A telítettségi arány)*

Az $\alpha = \frac{n}{m}$ *számot a hasító tábla telítettségi arányának nevezzük, ahol* m *a tábla réseinek a száma,* n *pedig a táblába beszúrt kulcsok száma.*

9. Tétel. (*A láncolt hasító tábla időigénye)*

Ha α *a kitöltési arány, akkor a láncolt hasító táblában*

$$C_n = \Theta\left(1 + \alpha\right)$$

és

$$C_n^{'} = \Theta\left(1 + \alpha\right).$$

ahol a megvizsgált kulcsok átlagos számát jelöli C_n a sikeres keresés esetén és $C_n^{'}$ a sikertelen keresések esetén.

10. Definíció. *Klaszternek* nevezzük a kipróbálási sorozat mentén az egymást követő kitöltött rések összességét. Ezen halmaz elemeinek a száma a klaszter mérete.

11. Definíció. *(A kiválasztási probléma) Legyen adott egy A halmaz (n különböző szám), és egy i index $1 \leq i \leq n$. Meghatározandó az A halmaz azon x eleme, melyre nézve pontosan $i-1$ darab tőle kisebb elem van az A halmazban.*

12. Definíció. *(Medián) Mediánnak nevezzük az adatsor azon elemét, amely a rendezett sorban a középső helyet foglalja el. Ha páratlan számú elem van az adatsorban, akkor $n = 2k-1$ és így a medián indexe a rendezés után k. Ha páros számú elem van az adatsorban, akkor $n = 2k$, és ekkor két középső elem van a k és a $k+1$ indexű a rendezés után. (Alsó medián, felső medián.)*

13. Definíció. *A rendezési reláció*

A ϱ relációt rendezési relációnak nevezzük az A halmazon, ha

1. *reflexív*, azaz $a \varrho a$ teljesül minden $a \in A$ esetén;
2. $a \varrho b$ és $b \varrho a$ akkor és csak akkor áll fenn, ha $a = b$
3. *tranzitív*, azaz $a \varrho b$ és $b \varrho c$ maga után vonja az $a \varrho c$ teljesülését;
4. *antiszimmetrikus*, azaz vagy az $a \varrho b$, vagy a $b \varrho a$ fennáll minden $a, b \in A$ esetén.

14. Definíció. *(Az oszd meg és uralkodj elv) Az oszd meg és uralkodj elv egy algoritmus tervezési stratégia A problémát olyan kisebb méretű, azonos részproblémákra osztjuk föl, amelyek rekurzívan megoldhatók. Ezután egyesítjük a megoldásokat.*

15. Definíció. *(A rendezési algoritmusokkal kapcsolatban felmerülő igények)*

a.	*Helyben rendezés*, azaz a rendezés eredménye az eredeti helyén jelenjen meg, legfeljebb konstans méretű többletmemória felhasználása révén.
b.	*Gyorsaság.* A rendezési idő legyen minél rövidebb.
c.	*Adaptivitás.* Az algoritmus használja ki a kulcsok között már meglévő rendezettséget.
d.	*Stabilitás.* A rendezés őrizze meg az azonos kulcsú rekordok esetén a rekordok egymáshoz képesti eredeti sorrendjét. (Például telefonszámlák készítésekor az azonos kulcsú előfizetői hívások időrendi sorrendje maradjon meg.)
e.	Az algoritmus csak a kulcsokat rendezze a rekordokra mutató pointerekkel, vagy az összes rekordot mozgassa.
f.	*Belső rendezés* legyen (csak a belső memóriát vegye igénybe a rendezéshez), vagy *külső rendezés* legyen (háttértárakat is igénybe vehet).
g.	*Összehasonlítás*on alapuljon a rendezés, vagy azt ne vegye igénybe az algoritmus. (Ez utóbbi esetben a kulcsokra további megszorításokat kell tenni.)
h.	Optimális legyen a rendezési algoritmus, vagy sem. (Nem biztos, hogy az adatok az optimális algoritmus által megkívánt módon vannak megadva.)
i.	Az összes rendezendő adatnak rendelkezésre kell-e állnia a rendezés teljes folyamata alatt, vagy sem.
j.	A rendezésnek csak a befejeztével van eredmény, vagy menet közben is a már rendezett rész tovább nem változik.

16. Tétel. *A Batcher-féle összefésülés tétele*

A Batcher összefésülés során

$$c_{2i-1} = \min\{u_i, v_i\}$$

és

$$c_{2i} = \max\{u_i, v_i\},$$

ahol $1 \leq i \leq \frac{n+m}{2}$.

17. Tétel. *A Stirling formula*

Igaz az alábbi összefüggés az n!-ra:

$$\frac{n^n}{e^n} < n! < \frac{(n+1)^{n+1}}{e^n}, \qquad n = 3, 4, 5, \ldots$$

18. Tétel. *Alsó korlát összehasonlító rendezésre*

Bármely n *elemet rendező döntési fa magassága* $T(n) = \Omega(n \cdot \log n)$

Gráfelméleti fogalmak, jelölések

1. Definíció. *Legyen V egy véges halmaz, E pedig V-beli rendezetlen elempárok véges rendszere. Ekkor a $G = (V, E)$ párt gráfnak nevezzük.*

2. Definíció. *A $G = (V, E)$ rendezett párt irányított gráfnak (digráfnak) nevezzük. A rendezett pár elemeire tett kikötések: V véges halmaz, a G-beli csúcsok halmaza. E bináris reláció a V halmazon, az élek halmaza.*

E={ (u, v) rendezett pár | $u \in V, v \in V$} $\subset V \times V$. Hurkok megengedettek.

3. Definíció. *A $G = (V, E)$ rendezett párt irányítatlan gráfnak nevezzük. A rendezett pár elemeire tett kikötések: V véges halmaz, a G-beli csúcsok halmaza. E bináris reláció a V halmazon, az élek halmaza.*

E={ (u, v) rendezetlen pár | $u \in V, v \in V$} $\subset V \times V$. Hurok nem megengedett.

4. Definíció. *Az a gráf (irányított vagy irányítatlan), amelynek minden éléhez egy számot (súlyt) rendelünk hozzá, hálózatnak (súlyozott gráfnak) nevezzük.*

5. Definíció. *A gráf egymáshoz csatlakozó éleinek olyan sorozatát, amely egyetlen ponton sem megy át egynél többször, útnak nevezzük.*

6. Definíció. *Az u csúcsból kiinduló és a v csúcsba mutató él digráfban az (u,v) él.*

7. Definíció. *Az u csúcsból kiinduló és a v csúcsba mutató él irányítatlan gráfban az (u,v) él*

8. Definíció. (*Az u csúcs szomszédja) Legyen (u,v) él egy G = (V,E) gráfban. Ekkor a v csúcsot az u csúcs szomszédjának nevezzük. A szomszédság reláció irányítatlan gráfban szimmetrikus, digráfban nem.*

9. Definíció. *(Csúcs fokszáma irányítatlan gráfban) A csúcs fokszáma a belőle kiinduló élek száma.*

10. Definíció. *(Csúcs fokszáma digráfban)*

Kimenő fokszám (kifok): *a csúcsból kimenő élek száma*
Bemenő fokszám (befok): *a csúcsba bemenő élek száma*
Csúcs fokszáma: *kifok+befok*

11. Definíció. *(Izolált csúcs) Csúcs, melynek fokszáma zérus.*

12. Definíció. *(Az u csúcsot az u' csúccsal összekötő k hosszúságú út)*
Csúcsok véges sorozata: $v_0, v_1, \dots, v_k$, *ahol* $v_0 = u, v_k = u'$ *és* $(v_i v_{i+1}) \in E$ $(i = 0,1, \dots, k-1)$

13. Definíció. *(Egyszerű út) Olyan út, melyben a benne szereplő csúcsok páronként különbözőek.*

14. Definíció. *(Az u' csúcs elérhető az u csúcsból) Azt mondjuk, hogy az u' csúcs elérhető az u csúcsból, ha van olyan út, amely az u csúcsot az u' csúccsal összeköti.*

15. Definíció. *(Körmentes gráf) Olyan gráf, amely nem tartalmaz kört.*

16. Definíció. (*Összefüggő gráf) Ha bármely két csúcsa összeköthető úttal.*

17. Definíció. *(Gráf részgráfja) A G=(E,V) gráf részgráfja a G'=(E',V') melyre* $V' \subseteq V$ *és* $E' \subseteq E$.

18. Definíció. *(Teljes gráf) Irányítatlan gráf, melyben bármely két csúcs szomszédos. (Minden lehetséges él benne van.)*

19. Definíció. *(Erdő) Körmentes, irányítatlan gráf.*

20. Definíció. *(Nyílt fagráf) Összefüggő, körmentes, irányítatlan gráf.*

21. Tétel: *A nyílt fák tulajdonságai Legyen G=(E,V) irányítatlan gráf. Az alábbiak ekvivalensek.*

1. *G nyílt fa*
2. *G bármely két csúcsához egyértelműen létezik egy őket összekötő egyszerű út.*
3. *G összefüggő, de tetszőleges élének elhagyása után a visszamaradó gráf már nem összefüggő*
4. *G összefüggő és* $|E| = |V| - 1$
5. *G körmentes és* $|E| = |V| - 1$
6. *G körmentes, de akár egyetlen éllel is bővítve E-t a kapott gráf már tartalmaz kört.*

22. Definíció. *(Gyökeres fa) T fagráf, amely egyik csúcsának kitüntetett a szerepe a többihez képest. Ez a csúcs a gyökér vagy gyökércsúcs (r).*

23. Definíció. *(Az x csúcs megelőzője) A gyökérből x-be vezető úton fekvő bármely csúcs. (x is a saját megelőzője.)*

24. Definíció. *(x az y rákövetkezője) ha y x-nek megelőzője. (x is a saját rákövetkezője.)*

25. Definíció. *(x szülője y) ha az r-ből y-ba vezető úton (y,x) az utolsó él. (A gyökérnek nincs szülője T-ben.)*

26. Definíció. *(x az y gyereke) ha y az x szülője.*

27. Definíció. *(Testvérek) Azok a csúcsok, amelyeknek ugyanaz a csúcs a szülője.*

28. Definíció. *(Külső csúcs vagy levél) Az a csúcs, amelynek nincs gyereke.*

29. Definíció. *(Belső csúcs) Az a csúcs, amely nem levél.*

30. Definíció. *(x fokszáma gyökeres fában) az x gyerekeinek száma. (A szülő nem számít bele a fokszámba!)*

31. Definíció. *(x szintje) Az r-ből x-be vezető út hossza.*

32. Definíció. *(T magassága) a T-beli csúcsok szintjei közül a legnagyobb.*

33. Definíció. *(Bináris fa) Rendezett fa, melyben minden csúcs fokszáma legfeljebb kettő. Beszélhetünk bal gyerekről és jobb gyerekről.*

34. Definíció. *(Teljes bináris fa) Bináris fa, melyben a csúcsok fokszáma kettő, kivéve a leveleket, melyeké 0, valamint az összes levél azonos szinten helyezkedik el.*

35. Definíció. *(k-adrendű teljes fa) k-adrendű fa, melyben a levelek ugyanazon szintűek és az összes belső csúcs fokszáma k.*

36. Definíció. *(A bináris kereső fa)*
A bináris kereső fa egy bináris fa, amely rendelkezik az alábbi bináris kereső fa tulajdonsággal:

Legyen x a bináris kereső fa tetszőleges csúcsa.
Ha y az x baloldali részfájában van, akkor $kulcs[y] \leq kulcs[x]$.
Ha y az x jobboldali részfájában van, akkor $kulcs[x] \leq kulcs[y]$.

A jellemző műveletek bináris kereső fában:
KERES, MINIMUM, MAXIMUM, ELŐZŐ, KÖVETKEZŐ, BESZÚR, TÖRÖL.

37. Definíció. *A piros-fekete fa*

A piros-fekete fa olyan bináris keresőfa, melynek minden csúcsa egy extra bit információt tartalmaz (a csúcs színét, amely piros, vagy fekete) és rendelkezik az alábbi Piros-fekete fa tulajdonságokkal:

- *Minden csúcs színe piros, vagy fekete.*
- *A gyökér színe fekete.*
- *Minden levél (NIL) színe fekete, a levél nem tartalmaz kulcsot.*
- *Minden piros csúcsnak mindkét fia fekete.*
- *Bármely két, azonos csúcsból induló, levélig vezető úton ugyanannyi fekete csúcs van.*

38. Definíció: *A bináris kupac (heap)*

A bináris kupac (heap) egy bináris gyökeres fa, amely minden szintjén kitöltött, kivéve esetleg az utolsó szintet, ahol balról jobbra haladva vannak a levelek kihagyás nélkül (a szint balra tömörített) továbbá teljesül a kupac tulajdonság, mely szerint a gyökércsúcsot kivéve minden x mutatójú csúcsra fenn kell álljon, hogy Kulcs(Szülő(x)) ≥ Kulcs(x), ha maximumkupacról beszélünk, illetve Kulcs(Szülő(x)) ≤ Kulcs(x) minimumkupac esetén.

39. Definíció. *Egy irányítatlan gráf feszítőfája a gráfnak az a részgráfja, amely fagráf és tartalmazza a gráf összes cúcspontját.*

40. Definíció. *A fa súlya a*

$$w(T) = \sum_{(u,v)\in T} w(u,v)$$

számérték.

41. Definíció. *Minimális feszítőfáról beszélünk, ha* $w(T)$ *értéke minimális az összes* T *feszítőfára nézve. A minimális feszítőfa nem feltétlenül egyértelmű.*

Algoritmusok

1. *Keresés tömbben:*

```
1  KERESÉS_TÖMBBEN (A, k, x )
2  // Input paraméter: A - a tömb
3  //                  k – a keresett kulcs
4  // Output paraméter: x - a k kulcsú elem pointere (indexe), ha van ilyen elem
                          vagy NIL, ha nincs
5  // Lineárisan keresi a k kulcsot.
6  //
7  x ← fej[A]
8  IF  hossz[A] ≠ 0
9      THEN    WHILE    x ≤ vége[A] és kulcs[A_x] ≠ k DO
10                      INC(x)
11             IF       x> vége[A]
12                      THEN x ← NIL
13 RETURN (x)
```

2. *Láncolt listában keresés:*

```
1  LISTÁBAN_KERES ( L, k, x )
2  // Input paraméter:  L – a lista
3  //                   k – a keresett kulcs
4  // Output paraméter: x  a kulcselem pointere.
                         NIL, ha a kulcs nincs a listában.
5  //
6  x ← fej[L]
7  WHILE   x ≠ NIL és kulcs[x] ≠ k DO
8              x ← köv[x]
9  RETURN (x)
```

3. *Láncolt listába beszúrás:*

```
1  LISTÁBA_BESZÚR ( L, x )
2  // Input paraméter:   L – a lista
   //                    x  a beszúrandó elemre mutató mutató
3  // Az új elemet a lista elejére teszi
4  //
5  köv[x] ← fej[L]
6  elő[x]  ← NIL
7  IF    fej[L] ≠ NIL
8        THEN elő[fej[L]] ← x
9  fej[L] ← x
10 RETURN
```

4. *Minimumkeresés tömbben:*

```
1  MINIMUMKERESÉS( A, min )
2  // Input paraméter: A – a tömb
3  // Output paraméter: min -  a minimum értéke
4  //
5  min ← A1
6  FOR      i←2 TO hossz[A] DO
7                IF     min > Ai
8                       THEN min ← Ai
9  RETURN ( min )
```

5. *Feloszt eljárás:*

	4.1.5.1. algoritmus **Résztömb felosztása előre adott érték körül**
	// $T(n)=\Theta(n)$
1	FELOSZT(A,p,r,x, q)
2	// Input paraméter: A – a tömb
3	// p - a felosztandó rész kezdőindexe
4	// r - a felosztandó rész végindexe
5	// x - az előre megadott érték, amely a felosztást szabályozza
6	// Output paraméter: A – a megváltozott tömb
7	// q – a felosztás határa $A_{p...q}$; $A_{q+1...r}$
8	//
9	i $\leftarrow$ p – 1
10	j $\leftarrow$ r + 1
11	**WHILE IGAZ DO**
12	**REPEAT** j $\leftarrow$ j – 1
13	**UNTIL** $A_j \leq x$
14	**REPEAT** i $\leftarrow$ i + 1
15	**UNTIL** $A_i \geq x$
16	**IF** i < j
17	**THEN** Csere $A_i \leftrightarrow A_j$
18	**ELSE** q $\leftarrow$ j
19	**RETURN** (A, q)

6. *Beszúró rendezés:*

1	BESZÚRÓ_RENDEZÉS (A)
2	// Input paraméter: A - a rendezendő tömb
3	// Output paraméter: A - a rendezett tömb
4	//
5	**FOR** $j \leftarrow 2$ **TO** ***hossz***$[A]$ **DO**
6	$kulcs \leftarrow A_j$
7	// Beszúrás az $A_{1..j-1}$ rendezett sorozatba
8	$i \leftarrow j-1$
9	**WHILE** $i > 0$ és $A_i > kulcs$ **DO**
10	$A_{i+1} \leftarrow A_i$
11	DEC(i)
12	$A_{i+1} \leftarrow kulcs$
13	**RETURN** (A)

+

7. *Bináris fába beszúrás:*

1	**FÁBA_BESZÚR**(T, z)
2	$y \leftarrow NIL$
3	$x \leftarrow gyökér[T]$
4	**WHILE** $x \neq NIL$ **DO**
5	$y \leftarrow x$
6	**IF** $kulcs[z] < kulcs[x]$
7	**THEN** $x \leftarrow bal[x]$
8	**ELSE** $x \leftarrow jobb[x]$
9	$szülő[z] \leftarrow y$
10	**IF** $y = NIL$
11	**THEN** $gyökér[T] \leftarrow z$
12	**ELSE IF** $kulcs[z] < kulcs[y]$
13	**THEN** $bal[y] \leftarrow z$
14	**ELSE** $jobb[y] \leftarrow z$

8. *Bináris fából törlés:*

FÁBÓL_TÖRÖL(T, z)	
Három esetre bontjuk a törlést:	
z-nek nincs gyereke:	egyszerűen kivágjuk
z-nek egy gyereke van:	kivágjuk úgy, hogy a szülője és a gyereke között kapcsolatot hozunk létre.
z-nek két gyereke van:	kivágjuk z azon legközelebbi rákövetkezőjét, amelynek már nincs baloldali gyereke és ezt a rákövetkezőt z helyére illesztjük.

9. *Inorder fabejárás:*

```
1  INORDER_FA_BEJÁRÁS(x)
2  IF x ≠ NIL
3      THEN INORDER_FA_BEJÁRÁS(bal[x])
4           Print(kulcs[x])
5           INORDER_FA_BEJÁRÁS(jobb[x])
```