# Feladatok

## 16 Alap absztrakt adattípusok

1. Implementálja a dinamikus tömb adattípust statikus tömbök segítségével (számokra).

2. Implementálja a halmaz adattípust
   - statikus tömbök használatával,
   - dinamikus tömbök használatával,
   - dinamikus tömbök használatával, de hatékonyan (számokra).

3. Implementálja a multihalmaz (vagy zsák) adatszerkezetet minél kisebb memóriaigénnyel boolean értékek tárolására.

4. Implementáljon egy forgóvilla adattípust. A frissen létrehozott forgóvilla mindig zárt állapotban van. Zárt állapotú forgóvilla pénzbedobásra nyitottá válik. Nyílt forgóvilla áthaladási kísérlet hatására zárul. Az adattípusnak a következő műveletei vannak:

   - Pénzbedobás
   - Áthaladási kísérlet
   - Aktuális állapot lekérdezése
   - Eddigi sikeres áthaladások számának lekérdezése

5. Implementálja a multihalmaz (vagy zsák) adatszerkezetet egész számokra. Ezt az adatszerkezetet felhasználva írja ki egy szám prím felbontását (pl: $120 = 2^3 3^1 5^1$).

6. A dinamikus tömb kiegészítése a következő funkciókkal:
   - adat beszúrása a lista elejére
   - adat beszúrása a lista végére
   - adat beszúrása adott pozícióra
   - adat törlése adott pozícióból
   - adat keresése
   - adat törlése
   - lista hosszának lekérdezése

7. Implementálja a sor adattípust tömb használatával (pontosan akkora tömbbel, ahány elem a sorban van).

8. Implementálja a verem adattípust tömb használatával (pontosan akkora tömbbel, ahány elem a veremben van).

9. Implementálja a sor adattípust tömb használatával úgy, hogy a tömb mérete duplázódik, ha már nincs elég hely benne. Ezen kívül tartsa nyilván, hogy melyik indexen van az első hasznos elem és melyiken az utolsó. Gondoljon úgy a tömbre, mint aminek a két vége össze van kötve és az utolsó tömbelem „után" a 0. következik. Ezzel *körkörös buffer* megvalósítással a láncolt listás megoldással vetekedő, lineáris idejű implementációt hoz létre.

# 17 Láncolt listák

1. Implementálja az egyszeresen láncolt lista adattípust (számokra) a következő műveletekkel
    a. adat beszúrása a lista elejére
    b. adat beszúrása a lista végére
    c. adat beszúrása adott pozícióra
    d. adat törlése adott pozícióból
    e. adat keresése
    f. adat törlése
    g. lista hosszának lekérdezése
    h. egy lista hozzáfűzése az aktuális lista végéhez
    i. adott listaelem elkérése (nem csak adat, hanem egész elem)

2. Egészítse ki az egyszeresen láncolt lista adattípust egy olyan függvénnyel, ami visszaadja az adott lista megfordítottját.
    a. Oly módon, hogy egy teljesen új lista jön létre, ami az eredeti listában szereplő értékeket forított sorrendben tartalamzza.
    b. Oly módon, hogy az aktuális listát huzalozza úgy át, hogy önmaga megfordítottja legyen.

3. Írjon olyan láncolt lista típust, ami rendezetten tárolja, növekvő sorrendben az elemeket. Definiálja az értelmes műveleteket (létrehoz, beszúr, keres, töröl).
   Gondolja végig, hogy hatékonyabb-e egy tömböt felépíteni majd azt rendezni, vagy egy rendezett láncolt listát építeni (pl, ha az a feladat, hogy egy fájlból felolvasott számokat rendezetten kell egyszer kiírni).

4. Írjon olyan függvényt egy láncolt lista típusra, ami két pozíció paramétert kap és a két adott helyen lévő elemet felcseréli a listában.

5. Implementálja a láncolt lista adattípust tömb segítségével. A tömb elemei a lista elemei, de nem feltétlenül a lista szerinti sorrendben. A következő mezőben nem egy referencia van a következő elemre, hanem a tömbbeli pozíciója a következő elemnek.
   Gondolja meg, hogy milyen esetekben előnyösebb egy ilyen implementáció a hagyományos referencia alapúhoz képest?

# 18 Gráfok

1. Implementáljon gráf adattípust szomszédsági mátrix segítségével. A mellékelt bemeneti fájlból tölts fel egy mátrixot ami diákok azt a viszonyát hivatott reprezentálni, hogy űltek-e egy padsorban. *(Irányított vagy irányítatlan gráf szükséges?)*
   A gráf feltöltése után a program *-ig kérjen be neveket. Ha a megadott név szerepel a gráfban, akkor írja ki a vele egy padsorban valaha ült diákok neveit.
   A bemeneti fájl formátuma:
   első sorban egy szám $n$, ami a diákok számát adja meg
   utána soronként az $n$ diák nevei
   utána egy sorban egy $m$ szám, ami a valaha volt összes padsor kombinációk számát jelöli
   utána soronként ,-vel elválasztva az adott padsorkombinációban lévő diákok nevei

2. Implementáljon gráf adattípust éllisták segítségével. A mellékelt bemeneti fájlból tölts fel egy mátrixot ami diákok azt a viszonyát hivatott reprezentálni, hogy űltek-e egy padsorban. *(Irányított vagy irányítatlan gráf szükséges?)*
   A gráf feltöltése után a program *-ig kérjen be neveket. Ha a megadott név szerepel a gráfban, akkor írja ki a vele egy padsorban valaha ült diákok neveit.

3. Implementáljon gráf adattípust szomszédsági mátrix segítségével. A mellékelt bemeneti fájlból tölts fel egy mátrixot ami diákok azt a viszonyát hivatott reprezentálni, hogy ki kiről másolt valaha dolgozatírás közben. *(Irányított vagy irányítatlan gráf szükséges?)*
   A gráf feltöltése után a program *-ig kérjen be neveket. Ha a megadott név szerepel a gráfban, akkor írja ki azok neveit akik az adott embertől másoltak, illetve azok neveit akiktől ő másolt.
4. Implementáljon gráf adattípust éllisták segítségével. A mellékelt bemeneti fájlból tölts fel egy mátrixot ami diákok azt a viszonyát hivatott reprezentálni, hogy hogy ki kiről másolt valaha dolgozatírás közben. *(Irányított vagy irányítatlan gráf szükséges?)*
   A gráf feltöltése után a program *-ig kérjen be neveket. Ha a megadott név szerepel a gráfban, akkor írja ki a vele egy padsorban valaha ült diákok neveit.

5. Írjon programot ami egy fájlban tárolt szomszédsági mátrix alapján felépít egy irányítatlan gráfot. A fájl formátuma az első sorban a csomópontok száma található $n$. Utána egy $n$ soros fájlrészben soronként $n$ hosszúságban 0-1 sorozat jelöli a szomszédsági mátrixot.
   a. A fenti fájl alapján építsen fel szomszédsági mátrixszal tárolt gráfot.
   b. A fenti fájl alapján építsen fel éllistásan tárolt gráfot.

6. Írjon programot ami egy fájlban tárol éllista alapján felépít egy irányított gráfot. A fájl formátuma az első sorban a csomópontok száma található $n$. Utána egy sorban kapcsolatok száma $m$, majd $m$ sorban soronként két szám szóközzel elválaszta, ami mutatja, hogy melyik két csomópontot kell összekötni.
   a. A fenti fájl alapján építsen fel szomszédsági mátrixszal tárolt gráfot.
   b. A fenti fájl alapján építsen fel éllistásan tárolt gráfot.

7. Labirintus reprezentációja gráffal (gondoljuk végig, hogy szomszédsági mátrixszal vagy éllistásan érdemes ebben a feladatban dolgozni). A szöveges fájl az első sorában két számot tartalmaz

szóközzel elválasztva, ami a labirintus méretét jelzi $n, m$. Utána $n$ darab sor található, mindegyikben $m$ hosszú karakterláncok vannak. A ‘ ’-el jelölt részek járhatóak, az ’x’-el jelölt részen fal van. A fájl végén ismét két szám van egy sorban, ami az induló pozíció koordinátáit jelöli (bal felső sarok a 0,0).
Írjon programot ami a bemeneti fájlt beolvasva a labirintust egy gráffal reprezentálja, majd a megkeres egy kivezető utat a labirintusból. A fájl végén két szám van egy sorban, ami az induló pozíció koordinátáit jelöli (bal felső sarok a 0,0). Használjon vermet és mélységi bejárást a megoldáshoz.

8. Válasszon egy tetszőleges gráf reprezentációt (él listás vagy szomszédsági mátrixos) és egészítse ki a típust egy metódussal, amely eldönti, hogy bármely két csúcspont elérhető-e egymásból, azaz, hogy a gráf összefüggő-e.

9. Válasszon egy tetszőleges gráf reprezentációt (él listás vagy szomszédsági mátrixos) és egészítse ki a típust egy metódussal, amely megszámolja, hogy a gráf hány összefüggő komponensből áll.

10. Egy fa adatszerezetben a csomópontokban számokat tárolunk. Írjon egy programot ami mélységi bejárást végez egy ilyen fán.
    a. Stack segítségével.
    b. Rekurzió felhasználásával.

11. Egy fa adatszerezetben a csomópontokban számokat tárolunk. Írjon egy programot ami szélességi bejárást végez egy ilyen fán.
    a. Sor segítségével.
    b. Rekurzió felhasználásával.

12. Egy bináris fa értékei szóközzel elválasztva szerepelnek egy fájlban, abban a sorrendben ahogyan a szélességi bejárás írná ki őket. A fájl alapján építsen fel egy bináris fát.

13. Készítsen egy programot, ami fájlból beolvas egy tömböt (a fájl első sorában szóközzel elválasztva vannak számok), majd a mediánhoz képest felépít egy bináris keresőfát. (A medián az a szám, aminél ugyanannyi kisebb és nagyobb elem van a tömbben.)

14. Készítsen egy általános gráfot (nem fa, tehát vannak benne körök), válasszon ki egy tetszőleges csomópontot, majd írja ki az összes olyan csomópontot a képernyőre amelyek tetszőleges lépésben elérhetőek a kiválasztott pontból.

15. Valamelyik előzőleg implementált gráf adattípust egészítse ki olyam metódussal ami eldönti, hogy a gráf összefüggő-e vagy sem.

16. Valamelyik előzőleg implementált gráf adattípust egészítse ki olyam metódussal ami megszámolja, hogy hány összefüggő komponens van a gráfban.

17. Készítsen egy általános gráfot (nem fa, tehát vannak benne körök). Válasszon ki két csomópontot és ezek között határozza meg a legrövidebb távolságot. Szomszédsági mátrixos vagy éllistás gráfot használjon.
    a. Használjon egy módosított mélységi bejárást a feladathoz.
    b. Használjon egy módosított szélességi bejárást a feladathoz.

    Melyik megoldással volt egyszerűbb a megoldás?

# 19 Vermek és sorok

1. Írjon egy programot ami a képernyőről ételnevekkel megtölt egy jégvermet. Minden lépésben a felhasználó eldöntheti, hogy betenni szeretne a verembe [b], kivenni szeretne [k], vagy kilép a programból [*].
   A program müködése imitálja a valódi jégvermet, ha a sonka után savanyú káposztát teszünk be, akkor először kelljen kivenni a savanyú káposztát, hogy a sonkát kivehessük.

2. Írjon programot, amivel városházán követni tudják, hogy kik várakoznak ingatlan adó befizetésre. A beengedéskor a portás felveszi az adatokat, fizetés után már nem találkozik az emberekkel (mert a hátsó ajtón távoznak), de a kasszából kap értesítést, hogy valaki fizetett.
   A program *-ig kérjen be parancsot (érkezés, fizets, listázás). Érkezés esetén kérje be az éppen érkező ember nevét. Fizetési jelzés esetén a legrégebben érkezett embert felejtse el, hiszen elmenet a hivatalból. Listázás esetén írja ki a képernyőre a hivatalban lévő emberek neveit érkezési sorrendben.

3. Írj egy programot ami *-ig kér be neveket a console-ról, majd ezeket kiírja fordított sorrendben a képernyőre. Használjon erre igazán alkalmas adatszerkezetet, majd hasonlítsa össze az első kurzusban adott megoldással.

4. Írjon egy programot ami bemenetként egy zárójelekből álló karaktersorozatot kap (olvassuk fájlból), majd erről eldönti, hogy helyesen van-e zárójelezve a sorozat vagy sem.
   Pl.: (({})[()()](())) helyes.

5. Írjon programot ami egy karakterláncról eldönti, hogy palindrom-e (kis és nagybetűk, szóközök nem számítanak). Használjon erre igazán alkalmas adatszerkezetet, majd hasonlítsa össze az első kurzusban adott megoldással.

6. Fix $B$ kapacitású sor esetén, ha $B$-nél több elem szeretne beállni a sorba, akkor az új elem nem kerül be a sorba.
   Módosítsa a sor adattípust oly módon, hogy az Enque függvénynek legyen egy igaz/hamis visszatérési értéke a sorbaállás sikerességétől függően.
   Ilyen fix méretű sort nagyon hatékonyan lehet statikus tömb segítségével implementálni, oly módon, hogy a tömb mellett nyilvántartjuk, hogy melyik az első hasznos elem, melyik az utolsó hasznos elem és a kettő között úgy tekintünk a tömbre, mintha az eleje és a vége össze lenne kötve. Írjunk egy ilyen implementációt a fix méretű sorra. Ezt hívják körkörös buffernek is.

# 20 Asszociatív tömbök

1. Egy fájl első sorában van egy szám $K$, a második sorában számok szóközzel elválasztva. Hatékonyan keresse meg az összes olyan $x + y$ párt amelynek összege éppen $K$. Ha jól dolgozott, a teljes algoritmus komplexitása $\Theta(n)$, ahol $n$ a fájl második sorában lévő számok száma.

2. Implementáljon karakterláncokat tartalmazó multihalmaz adatszerkezetet asszociatív tömbökre építve.

3. Permutációs bingó. A játék során a játékvezető véletlenül sorsol egész számokat a $[0, 1000]$ tartományban. Az a játékos nyer, aki először észreveszi, hogy van olyan számhármas a húzott számok sorozatában, ami minden lehetséges kombinációban előfordult.
   Pl.: $(2,19,4,1,100,1,4,19,1,4,1,19,100,192,100,4,19,2,1,19,4)$ egy nyerő sorozat a $\{1,4,19\}$ számhármasra.
   A győzelemhez az kell, hogy a mi algoritmusunk hamarabb ismerje fel, hogy nyertűnk, mint bárki másé. Írjon olyan algoritmust ami menden egyes új szám sorsolása után konstans időben eldönti, hogy BINGO van-e vagy sem.

4. A 14.8-as feladat, irodaház címekkel, folytatása. Egy nagy adatbázisból kell folamatosan keresésre válaszolni, amit mindig az adatbázis fájl olvasásával oldunk meg. Írjunk olyan adattípust, ami ezt elfedi egy cache réteggel a hatékony válaszadás miatt. Mindig az 5 utolsó kérés legyen a cache-ben.