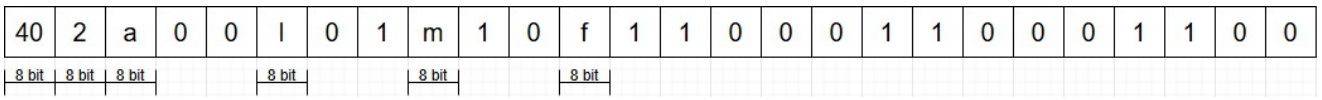
1. feladat (10 pont)

a.) Adott a következő naiv tömörítéssel tömörített állomány:



- Ahova 8 bit van írva az 8 bit, ahova nincs írva semmi az 1 bit.
- Az első 8 bit megadja a kódtáblázat méretét, ez legyen N.
- A következő 8 bit megadja egy kódszó méretét.
- A következő N bit a kódtáblázat.
- A többi bit a tömörített szöveg.

Add meg a kódtáblázatot, és a kitömörített szöveget.

b.) Tömörítsd a következő szöveget Huffman kóddal: almafalamalaba

- Add meg a karakterekhez tartozó kódszót.
- Add meg a tömörített állomány méretét.
- Add meg a tömörítés mértékét (tört alakban elég).

c.) LZW-vel tömörítsd a következő szöveget: almamaalmaaaa (nem ugyanaz, mint előbb!!!).

karakter / szó	kód
...	...

lépés	előző	beolvasott	konkatenált	ismert?	kimenet
...	...	...	...	...	...

2. feladat (7 pont)

Építs AVL fát a következő elemekből:

- 30, 20, 10, 40, 25, 27

Majd töröld az előző fából a következő elemeket:

- 10, 20, 40

Minden beszúrás / törlés után rajzold fel az új fát. Ha forgatni kell, akkor külön fában rajzold fel a forgatás utáni állapotot. Minden belső csúcsban (tehát ami nem levél) jelöld az adott csúcs súlyozottságát (++, +, --, -, =).

3. feladat (7 pont)

Adott az alábbi B+ fa:

{ [3 7] 10 [10 14 17] 20 [20 29] }

Ábrázold a fát grafikusan, majd

- szúrd be a következő elemeket: 19, 15, 16
- töröld a következő elemeket: 3, 17, 16

A levélben a pointereket az ábrázolásnál még rajzold be, beszúrás és törlés közben már nem kötelező.

A fát akkor kell újra rajzolni, ha új csúcs jön létre, vagy törlődik.

4. feladat (6 pont)

Adott az alábbi gráf mátrixos ábrázolással

i↓ j→	A	B	C	D	E	F	G
A	0	1	0	1	0	0	1
B	1	0	1	0	0	1	0
C	0	1	0	1	1	0	0
D	1	0	1	0	1	0	0
E	0	0	1	1	0	1	1
F	0	1	0	0	1	0	0
G	1	0	0	0	1	0	1

Rajzold fel a gráfot grafikusan, majd járd be BFS segítségével, és ezt ábrázold a tanult táblázatban.

d (távolság)	kifejtett csúcs	sor (queue)	π (szülő)
csúcsok			csúcsok
. . .			

5. feladat (8 pont)

Feladat leírása:

Adott egy éllistásan ábrázolt gráf (G), mely **súlyozott**, **irányított**, és **lehetnek benne hurokélek.**

Írd át a G gráfot mátrixos ábrázolásba úgy, hogy a hurokéleket törlöd a gráfból (nem veszed fel a mátrixba).

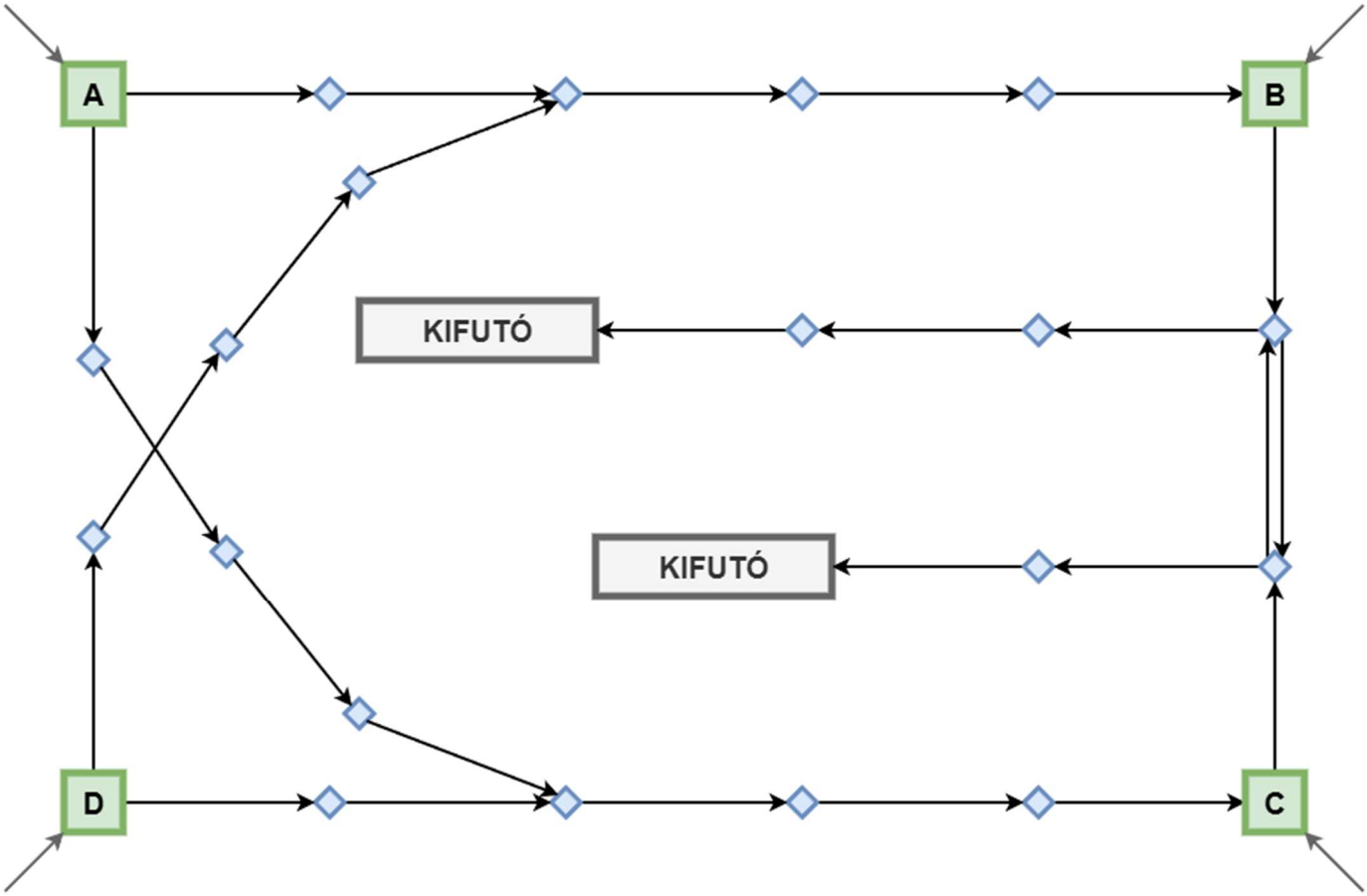
Bemenet: G[n], a gráf

Kimenet: G'[n, n], a gráf hurokélek nélkül

Segítség:

- G egy eleme egy (key, pointer) pár
- G-t be tudjuk járni pl.
    - for i:=1 .. n
        - key := G[i].key
        - ptr := G[i].next
        - ...
    - vagy
    - for key, ptr in G // foreach jellegű
        - ...
- G[i] a stacken van tárolva, ezért
    - key := G[i].key
    - ptr := G[i].next
- az előző pontban lévő ptr már a heapre mutat, ezért
    - key := ptr -> key vagy key := *(ptr.key)
    - ptr := ptr -> next vagy ptr := *(ptr.next)

6. feladat (4+4+4 pont)



Feladat leírása

Adott egy irányított gráf (G), amely egy légiforgalmi irányítási rendszer útvonalait ábrázolja.
A gráf csúcsai háromféle típusúak lehetnek:

1. Belépési pontok (zöld négyzetek) – ezeknél a pontoknál lép be a repülőgép a légtérbe.
2. Kifutópályák (szürke téglalapok) – ezek azok a pontok, ahol a repülőgép leszáll.
3. Útvonali pontok (kék rombuszok) – a repülőgépnek ezeken a pontokon kell végig haladnia az irányított élek mentén.

Minden repülőnek a belépési ponton kell belépnie a légtérbe, és a kifutópályán leszállnia.

Ha egy csúcsból több csúcsba is tovább lehet haladni, akkor a pilóta a diszpécsertől kap utasítást, hogy merre haladjon tovább.

Minden él azonos hosszúságú, tehát a repülőgép által bejárt távolságot az általa bejárt élek száma határozza meg.

A gráfban lehet kör, de a repülőgépek nem térhetnek vissza olyan csúcsba, amin már áthaladtak (és nem indulhat el olyan úton, amin kör keletkezése miatt zsákutcába érne).

Feltételezhetjük, hogy a gráf nem üres, és van legalább egy belépési pont és egy kifutópálya.

A fenti ábra csak egy példa, az algoritmusnak minden ilyen típusú gráfra működnie kell.

Feladat

Határozd meg azt az útvonalat, amelyen a repülőgép a lehető leghosszabb távolságot teszi meg, azaz:

- indul egy belépési pontból,
- érkezik egy kifutópályára,
- és közben az élek mentén halad végig a gráfban.

Ha több ilyen is van, akkor bármelyiket megadhatod.

Bemenet

- G[n]: a gráf, éllistásan ábrázolva (a gráfban minden csúcs benne van, legyen az belépési pont, kifutópálya vagy útvonali pont)
- B[m]: a belépési pontok, tömbösen ábrázolva (kizárólag a csúcsok vannak benne, NEM egy külön részgráf)

Kimenet

- (b, k, t): a belépési pont, kifutópálya és a távolság, amelyek esetén a repülő a leghosszabb utat teszi meg.

Segítség (nem feltétlen kell mind a feladat megoldásához)

- Valahogy számolni kell, hogy adott csúcsba adott belépési pontból hány lépés volt eljutni.
- Egy sorba (queue) nem csak egy elemet lehet belerakni, hanem például egy párt is.
    - q.add({a, b})
- Egy csúcs típusát például így vizsgálhatod meg:
    - típus(csúcs) = kifutó
    - típus(csúcs_A) = típus(csúcs_B)
- Ha szeretnéd megkapni azt a listát, ahova A csúcsból el lehet menni
    - lista := G.get(csúcs_A) vagy lista := G[csúcs_A]
    - ez maga a listára mutató pointerrel tér vissza (ami a heapen van tárolva)
- Referencia jelölésére használd az: &
- Az eredménnyel például így lehet visszatérni: return (b, k, t)
- Ha szeretnél írni olyan segédfüggvényt, ami több mint egy értékkel tér vissza, akkor a return-nél használhatod a fenti példát
    - Ha pedig ki szeretnéd bontani, akkor
        - b, k, t = függvény(...)
        - ezt akkor is használhatod, ha például a sorból szeretnél kivenni egy párt
            - a, b = q.pop()
    - Ez gyakorlatilag annak az egyszerűsített változata, mintha egy saját struktúrával térnél vissza, amiben több elem van