5. A gráf, mint adatstruktúra

5.1. Gráfelméleti bevezető

Definíció: Az irányított gráf (digráf)

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

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

Hurok az (a,a) él.

Definíció: Az irányítatlan gráf

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

$E=\{(u,v)\,rendezettlen\ pár \mid u\in V, v\in V\}\subset V\times V$.

Hurok nem megengedett.

Definíció: Az u csúcsból kiinduló és a v csúcsba mutató él digráfban

Az (u,v) él.

Definíció: Az u csúcsból kiinduló és a v csúcsba mutató él irányítatlan gráfban

Az (u,v) él.

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.

Definíció: Csúcs fokszáma irányítatlan gráfban

A csúcs fokszáma a belőle kiinduló élek száma.

Definíció: Csúcs fokszáma digráfban

Kimenő fokszám (kifok): a csúcsból kimenő élek száma

Bemenő fokszám (befok): a csúcsba bemenő élek száma

Csúcs fokszáma: kifok+befok

Definíció: Ionizált csúcs

Csúcs, melynek fokszáma zérus.

Definíció: Az u csúcsot az u' csúccsal összekötő k hosszúságú út

Csúcsok véges sorozata: $v_0, v_1, \ldots, v_k$,

ahol $u=v_0$, $u'=v_k$ és $(v_{i-1}, v_i)\in E,\ i=1,\ldots,k$.

Definíció: Egyszerű út

Út, melyben a benne szereplő csúcsok páronként különbözőek.

ahol $u = v_0$, $u' = v_k$ és $(v_{i-1}, v_i) \in E$, $i = 1, \ldots, k$.

Definíció: Út része
Legyen $v_0, v_1, \ldots, v_k$ út. Az út része $v_i, v_{i+1}, \ldots, v_j$, ahol $0 \le i \le j \le k$.

Definíció: Az u' csúcs elérhető az u csúcsból
Azt mondjuk, hogy az u' csúcs elérhető az u csúcsból (jelölésben $u \xrightarrow[P]{} u'$), ha van olyan út, amely az *u* csúcsot az *u'* csúccsal összeköti.

Definíció: Kör digráfban
Út, melyre $v_0 = v_k$ és az út tartalmaz legalább egy élt.

Definíció: Egyszerű kör
Kör, melynek csúcsai mind különbözőek.

Definíció: Hurok
1 hosszúságú kör.

Definíció: Egyszerű gráf
Hurok nélküli digráf

Definíció: Kör gráfban
Egyszerű kör és $k \ge 3$, $v_0 = v_k$.

Definíció: Körmentes gráf
Gráf, amely nem tartalmaz kört..

Definíció: Összefüggő gráf
Ha bármely két csúcsa összeköthető úttal.

Definíció: Összefüggő komponens
Csúcsok alkotta ekvivalencia-osztály, ahol az ekvivalencia reláció a csúcsok közötti elérhetőség.

Definíció: Digráf erősen összefüggő
Tetszőleges két csúcs esetén mindegyik elérhető a másikból.

Definíció: Izomorf gráfok
A $G = (V, E)$ és a $G' = (V', E')$ gráfok izomorfak, ha létezik olyan $f : V \to V'$ bijekció, hogy $(u, v) \in E \Leftrightarrow (f(u), f(v)) \in E'$.

Definíció: A $G = (V, E)$ gráf részgráfja
$G' = (V', E')$ gráf, melyre $V' \subset V$ és $E' \subset E$.

Definíció: A G gráf V' által meghatározott részgráfja
$G' = (V', E')$, ahol $E' = \{(u, v) \in E,\ u, v \in V'\}$.

Definíció: A $G=(V,E)$ gráfhoz tartozó digráf

Az a $G'=(V',E')$ digráf, melyre $(u,v)\in E' \Leftrightarrow (u,v)\in E$, azaz az éleket két irányított éllel helyettesítjük).

Definíció: A $G=(V,E)$ digráfhoz tartozó irányítatlan gráf

Az a $G'=(V',E')$ gráf, melyre $(u,v)\in E' \Leftrightarrow u\neq v,(u,v)\in E$ azaz elhagyjuk a hurkokat és az irányítást.

Definíció: Teljes gráf

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

Definíció: Páros gráf

Irányítatlan gráf, melynél V felbontható V_1, V_2 diszjunkt unióra úgy, hogy $(u,v)\in E$ esetén vagy $u\in V_1$ és $v\in V_2$, vagy pedig $u\in V_2$ és $v\in V_1$. (Azaz V_1-ben és V_2-ben nincs belső él.)

Definíció: Erdő

Körmentes, irányítatlan gráf.

Definíció: (Nyílt) fagráf

Összefüggő, körmentes, irányítatlan gráf.

Tétel: A nyílt fák tulajdonságai

Legyen $G=(V,E)$ 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.

Bizonyítás

1.⇒2. bizonyítása

G fa ⇒ G összefüggő. ⇒ G bármely csúcspárja között van út. Be kell látni, hogy csak egy van. Ha több lenne, akkor kettőből már kör alakítható ki, ami ellentmondás.

2.⇒3. bizonyítása

G bármely két csúcsa egyértelműen köthető össze úttal. ⇒ G összefüggő. ⇒
⇒ Tetszőleges (u,v) élt választva az él az u és v csúcsokat köti össze egyelemű útként.
Ő az egyetlen út u és v között.
Ha elhagyom, akkor nem lesz ott út, tehát a gráf nem lesz összefüggő.

3.⇒4. bizonyítása

G (3) miatt összefüggő, tehát ezt nem kell bizonyítani.
Másrészt ebből adódóan automatikusan $|E| \geq |V| - 1$.
Teljes indukcióval látjuk be, hogy $|E| \leq |V| - 1$.
Legyen $n=|V|$. Ha n=1 vagy 2, akkor ez igaz, mert a gráfnak n-1 éle van.
Legyen most $n \geq 3$ és minden kevesebb csúcsú gráfra teljesüljön (3).
Hagyjuk el tetszőleges élét. Ezáltal k darab összefüggő komponens keletkezik, ahol $k \geq 2$. Minden komponens (3) tulajdonságú.
Az élek száma legfeljebb $n-k \leq n-2$.
Az elvett élt is hozzávéve az élek száma legfeljebb n-1.

$4. \Rightarrow 5.$ bizonyítása
Indirekt módon bizonyítunk.
Tegyük fel, hogy van kör. Erre a körre, mint részgráfra igaz, hogy éleinek és csúcsainak száma megegyezik. Legyen ez k.
Ha $k<|V|$, akkor van még csúcs a körön kívül, mely szomszédos a kör valamely csúcsával G összefüggősége miatt. Vegyük hozzá a körhöz ezt a csúcsot és az élt. Az így kapott részgráfban is a csúcsok száma és az élek száma azonos (k+1). Újabb és újabb csúcsok és élek hozzávételével az összes csúcspontot felhasználjuk. Ekkor G-re azt kapjuk, hogy $|E| \geq |V|$, ami ellentmondás

$5. \Rightarrow 6.$ bizonyítása
Legyen G összefüggő komponenseinek száma k.
Minden komponens fa és (1) $\Rightarrow$ (5). Ezért G komponenseiben $|V| - k$ él van.
$|E| = |V| - 1$ miatt k=1 és így G fa. Ekkor viszont bármely két G-beli csúcs összeköthető egyszerű úttal. Hozzávéve egy új élt a két csúcs között, az az úttal együtt kört alkot

$6. \Rightarrow 1.$ bizonyítása
Azt kell belátni, hogy G összefüggő.
Legyen u,v két tetszőleges csúcs. Ha szomszédosak, akkor van közöttük út.
Ha nem szomszédosak, akkor vegyük fel az u,v élt.
Ekkor kör keletkezik (6) miatt, A kör élei az (u,v) él kivételével G-hez tartoznak.és így utat alkotnak u és v között. Tehát G összefüggő, tehát fa.

■

5.2. Gyökeres fák

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

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

Definíció: y valódi megelőzője x-nek
ha megelőzője x-nek, de $y \neq x$.

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

Definíció: x valódi rákövetkezője y-nak
ha megelőzője y-nak, de y ≠ x.

Definíció: x-ben gyökerező részfa
Az x és a rákövetkezőiből álló részgráf (fa).

Definíció: x szülője y
ha az $r \xrightarrow[P]{} x$ úton (y,x) az utolsó él. (A gyökérnek nincs szülője T-ben.)

Definíció: x az y gyereke
ha y az x szülője.

Definíció: Testvérek
azok a csúcsok, amelyeknek ugyanaz a csúcs a szülője.

Definíció: Külső csúcs vagy levél
az a csúcs, amelynek nincs gyereke.

Definíció: Belső csúcs
az a csúcs, amely nem levél.

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

Definíció: x szintje
az $r \xrightarrow[P]{} x$ út hossza.

Definíció: T magassága
a T-beli csúcsok szintjei közül a legnagyobb.

Definíció: Rendezett gyökeres fa
minden csúcs gyerekei rendezettek. (Azaz van első, második,…, k-adik)

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

Definíció: Null fa
Üres bináris fa.

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

Definíció: Súlyozott fa
A csúcsok gyerekeit különböző pozitív, egész számmal indexeljük. (1,2,3,…)

Definíció: csúcs i-dik gyereke hiányzó
nincs i indexű gyereke.

Definíció: k-adrendű fa
Súlyozott fa, ahol minden csúcsnál a k-nál nagyobb indexű gyerekek hiányoznak.

(A bináris fa másodrendű.)

Definíció: k-adrendű teljes fa
k-adrendű fa, melyben a levelek ugyanazon szintűek és az összes belső csúcs fokszáma k.

A h magasságú teljes *k*-adrendű fának k^h számú levele van.
Ha a levelek száma *n*, akkor a teljes k-adrendű fa magassága $\log_k n$.
A *h* magasságú teljes *k*-adrendű fa belső csúcsainak a száma: $1+k+k^2+\ldots+k^{h-1}=\sum_{i=0}^{h-1}k^i=\frac{k^h-1}{k-1}$.
Teljes bináris fa belső csúcsainak száma: 2^h-1

A gyökeres fa adatstruktúra. A fa minden csúcsa egy objektum. Az objektumok tartalmaznak kulcs mezőt és mutatókat. A **T** fa attribútuma: **gyökér[T]** egy mutató, mely a fa gyökerére mutat. Ha **gyökér[T]=NIL**, akkor a fa üres.

Bináris fa esetén az x csúcs ábrázolható az alábbi sémával:

Szülőmutató	Kulcs
Bal gyerek mutató	Jobb gyerek mutató

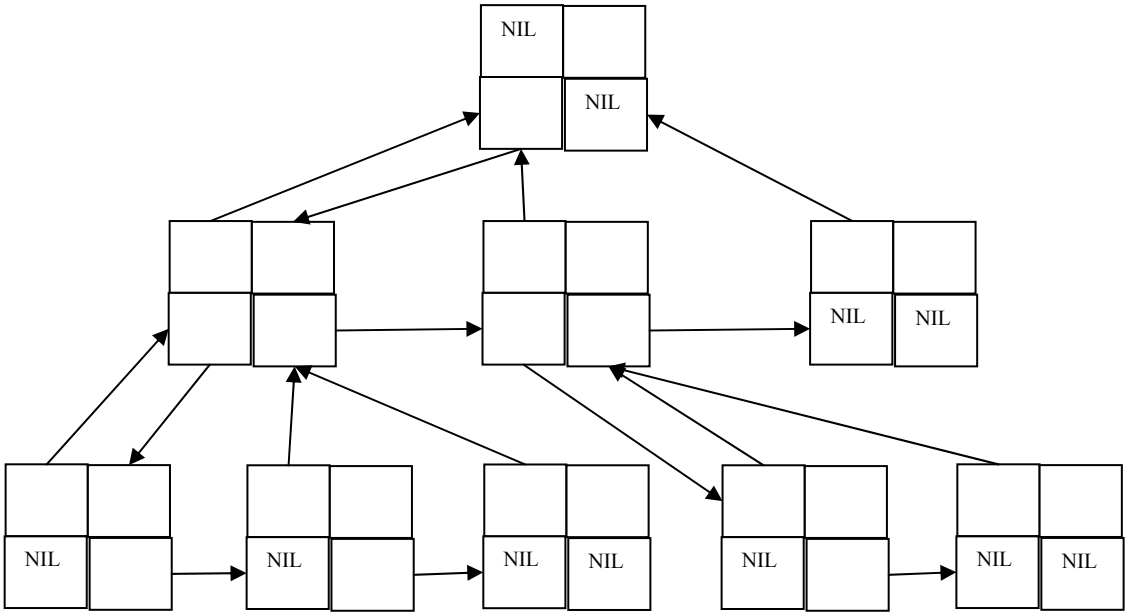
Csúcsattribútumok: **szülő[x], kulcs[x], bal[x], jobb[x]**
Ha **szülő[x]=NIL**, akkor **x** gyökér. Ha **bal[x]=NIL**, vagy **jobb[x]=NIL**, akkor amelyik **NIL**, az a gyerek nincs. Ha mindkettő **NIL**, akkor **x** levél.

A bináris fa mintájára k-adrendű fa esetén használható, bár memória pazarló az x csúcs ábrázolására az alábbi séma:

Szülő mutató			Kulcs
1. gyerek mutató	2. gyerek mutató	…	k. gyerek mutató

Helyette javasolható az úgynevezett bal gyerek – jobb testvér séma, melynek memóriaigénye: $O(n)$.

Szülőmutató	Kulcs
Bal gyerek mutató	Jobb testvér mutató



5.2.1. Bináris kupac, kupacrendezés

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,

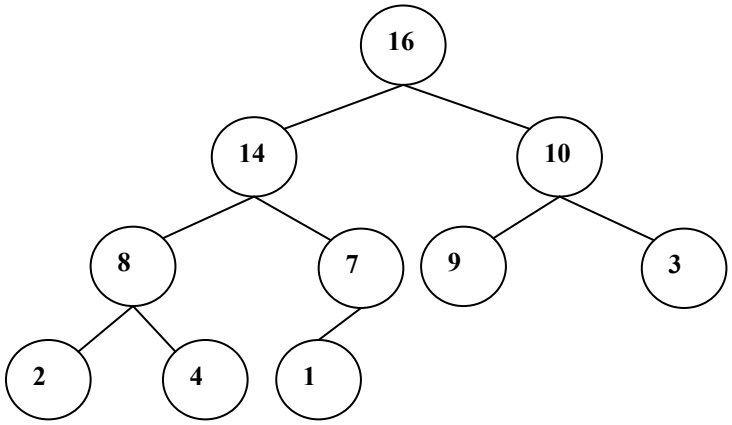
A következőkben a maximumkupacról fogunk beszélni. Kupac attribútumok (feltételezve, hogy a kupac reprezentálása egy egyindexes A tömbbel történik, amelyben a kupacot tároljuk és amelynek indexelése 1-gyel kezdődik) a következők:

hossz[A]	a tömb fizikai maximális mérete (elemszám a tömbben).
kupac_méret[A	a kupacelemek (csúcsok) száma
gyökér[T])	A_1

Az i indexű elem esetén az attribútumok: **Szülő(i), Bal(i), Jobb(i)**

Kupac magasság = a fa magassága = $h = \Theta(\log n)$

Példa kupacra és tömbös realizációjára:



16	14	10	8	7	9	3	2	4	1
1	2	3	4	5	6	7	8	9	10

	5.2.1.1. algoritmus **Szülő**	**5.2.1.2. algoritmus** **Bal gyerek**	**5.2.1.3. algoritmus** **Jobb gyerek**
	// $T(n)=\Theta(1)$	// $T(n)=\Theta(1)$	// $T(n)=\Theta(1)$
1	SZÜLŐ(i, x)	BAL(i, x)	JOBB(i, x)
2	// Input paraméter: i a vizsgált index	// Input paraméter: i a vizsgált index	// Input paraméter: i a vizsgált index
3	// Output paraméter: x – a szülő indexe	// Output paraméter: x – a bal gyerek indexe	// Output paraméter: x – a jobb gyerek indexe
4			
5	$x \leftarrow \left\lfloor \frac{i}{2} \right\rfloor$	$x \leftarrow 2i$	$x \leftarrow 2i+1$
6	**RETURN(** x⌋ **)**	**RETURN(** x⌋ **)**	**RETURN(** x⌋ **)**

	5.2.1.4. algoritmus **Kupacol** (Eljárás a kupac tulajdonság fenntartására)
	// $T(n)=\Theta(\log n)$
1	KUPACOL (A, i)
2	// Input paraméter: A - a kupacot tároló tömb
3	// i - a vizsgálandó elem indexe
4	// Output paraméter: A - a kupacot tároló tömb
5	// Akkor használjuk, ha az i baloldali és jobboldali részfái ugyan kupacok, de i-ben sérülhet a kupac tulajdonság
6	BAL(i, b)
7	JOBB(i, j)
8	**IF** b ≤ **Kupac_méret[**A**] és** $A_b > A_i$
9	**THEN** legnagyobb ← b
10	**ELSE** legnagyobb ← i
11	**IF** j ≤ **Kupac_méret[**A**] és** $A_j > A_{legnagyobb}$

12	**THEN** legnagyobb ← j
13	**IF** legnagyobb ≠i
14	**THEN** **csere** $A_i \leftrightarrow A_{legnagyobb}$
15	**KUPACOL(** A, legnagyobb **)**
16	**RETURN** (A)

5.2.1.5. algoritmus
Kupacot épít
(Eljárás, mely tetszőleges adatokból kupacot épít)

// $T(n) = \Theta(n)$

1	KUPACOT_ÉPÍT(A)
2	// Input paraméter: A - a kiinduló kulcsok tömbje
4	// Output paraméter: A – az elkészült kupac tömbje
5	
6	**Kupac_méret[**A**]** ←**hossz[**A**]**
7	**FOR** $i \leftarrow \left\lfloor \frac{hossz[A]}{2} \right\rfloor$ **DOWNTO 1 DO**
8	**KUPACOL(** A, i **)**
9	**RETURN** (A)

5.2.1.6. algoritmus
Kupacrendezés
(Eljárás, mely helyben rendez)

// $T(n) = \Theta(n \log n)$

1	KUPACRENDEZÉS(A)
2	// Input paraméter: A - a rendezendő kupac tömbje
3	// Output paraméter: A – a rendezett kulcsok tömbje
4	
5	KUPACOT_ÉPÍT**(** A **)**
6	**FOR** i ← **hossz[**A**]** **DOWNTO** 2 **DO**
7	**csere** $A_1 \leftrightarrow A_i$
8	**DEC (Kupac_méret[**A**])**
9	KUPACOL**(** A, 1 **)**
10	**RETURN** (A)

5.2.1.7. algoritmus
Kupacba beszúrás

// $T(n) = \Theta(\log n)$

1	KUPACBA_BESZÚR(A, kulcs)
2	// Input paraméter: A - a kupac tömbje
3	// kulcs – a beszárandó kulcs
4	// Output paraméter: A – a kupac a beszúrt kulccsal
5	
6	**INC(Kupac_méret[** A **])**
7	i ← **Kupac_méret [**A**])**
8	SZÜLŐ(i, x)

9	**WHILE** i>1 és A_x<kulcs
10	$A_i \leftarrow A_x$
11	i ← x
12	SZÜLŐ(i, x)
13	A_i ←kulcs
14	**RETURN** (A)

5.2.1.8. algoritmus
Kupacban maximális kulcs lekérdezése

// $T(n) = \Theta(1)$

1	KUPACBAN_MAX(A, max, hiba)
2	// Input paraméter: A - a kupac tömbje
3	// Output paraméter: max – a maximális kulcs
4	// hiba – jelzi a művelet sikerességét
5	
6	**IF** **Kupac_méret**[A]<1
7	**THEN** hiba ← „kupac alulcsordulás”
8	**RETURN** (hiba)
9	max ← A_1
10	hiba ← „sikeres művelet”
11	**RETURN** (max, hiba)

5.2.1.9. algoritmus
Kupacból a maximális kulcs lekérdezése eltávolítással

// $T(n) = \Theta(\log n)$

1	KUPACBÓL_KIVESZ_MAX (A, max, hiba)
2	// Input paraméter: A - a kupac tömbje
3	// Output paraméter: max – a maximális kulcs
4	// hiba – jelzi a művelet sikerességét
5	
6	**IF** **Kupac_méret**[A]<1
7	**THEN** hiba ← „kupac alulcsordulás”
8	**RETURN** (hiba)
9	max ← A_1
10	$A_1 \leftarrow A_{\textbf{Kupac\_méret}[A]}$
11	DEC(Kupac_méret[A])
12	KUPACOL(A, 1)
13	hiba ← „sikeres művelet”
14	**RETURN** (max, hiba)

5.2.2. Az elsőbbségi sor

Definíció: Az elsőbbségi sor (priority queue)

Az **elsőbbségi sor** (priority queue) olyan S halmaz, amelynek minden eleméhez egy kulcs értéket rendelünk. A kulcs egy elsőbbségi értéket, prioritási értéket szimbolizál. Szolgáltatást igénybevevők sorbanállási sorában a sorrakerülő elem az lesz, amelynek a prioritása a legnagyobb (maximum prioritásos sor), vagy amelyiké a legkisebb (minimum prioritásos sor).

BESZÚR(S,x): egy elemet hozzáadunk **S**-hez. **S** $\leftarrow$**S** $\cup$ **{x}**
MAXIMUM(S): **S** legnagyobb kulcsú elemének meghatározása
KIVESZ_MAX(S): megadja és törli a maximális kulcsú elemet.

Az elsőbbségi sor célszerű realizációja a kupac.

5.2.3. A Huffman kód, a mohó algoritmus

A Huffman kód egy adattömörítési célokat szolgáló eljárás.

A probléma megfogalmazása: Legyen adott egy adatfile, amelyben ismert az egyes adatelemek (pl.: byte-ok) gyakorisága. A feladat olyan kódot találni az adatelemekre, amely révén a kódolt file hossza rövidebb lesz (a lehető legrövidebb), mint az eredeti hossz volt.

Definíció: Kódszó
Kódszónak nevezzük az üzenet ábécé betűjéhez hozzárendelt jelek nemüres sorozatát, amelyben a kódábécé betűi szerepelhetnek.

Definíció: Kód
Kódnak nevezzük az üzenet ábécé betűit és a hozzájuk rendelt kódszavak táblázatát. (Kódtábla, kódszótár).

Definíció: Kódolás
Kódolásnak nevezzük az üzenet betűinek a helyettesítését a betűnek megfeleltetett kódszóval. A kódolás eredménye a kódolt üzenet.

Az egyszerűség kedvéért a kódoláshoz használjunk két jelet, a 0 és az 1 jeleket. Az adatelemek kódja lehet fix hosszúságú vagy lehet változó hosszúságú. A kódot egy kódfával ábrázolhatjuk, amely egy bináris fa, amelynek levelei a kódszavak. A kódszó a levélhez vezető útból olvasható ki a gyökértől indulva balra lépéskor 0, jobbra lépéskor 1 hozzáírásával a kódszóhoz.

Definíció: Dekódolás
Dekódolásnak nevezzük a kódolt üzenet kódszavakra bontását.

A kódnak **dekódolható**nak kell lennie, ha azt akarjuk, hogy a dekódoláskor az eredeti üzenetet kapjuk vissza.

Definíció: Dekódolható kód
Dekódolhatónak nevezzük a kódot, ha minden kódolt üzenet csak egyféleképpen bontható kódszavakra.

Definíció: Prefix kód
Prefix kódnak nevezzük a kódot, ha egyik kódszó sem eleje semelyik másik kódszónak. (Semelyik kódszót sem lehet valamely másikból kódjelek hozzáírásával megkapni.)

A prefix kód dekódolható kód.

Definíció: Dekódolható kódok ekvivalenciája
Két dekódolható kódot **ekvivalens**nek nevezünk, ha a megfelelő kódszavaik

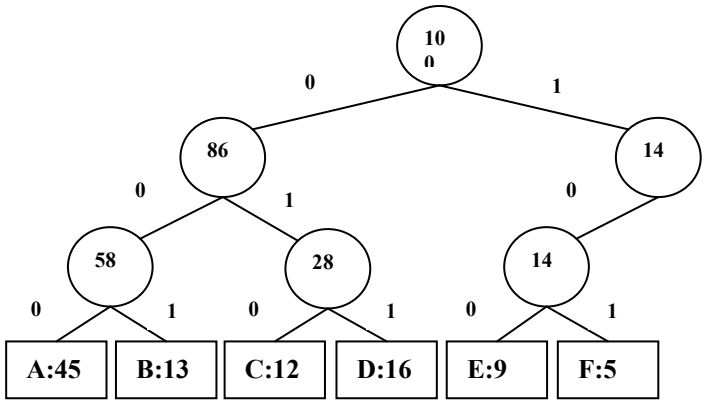
hossza megegyezik.

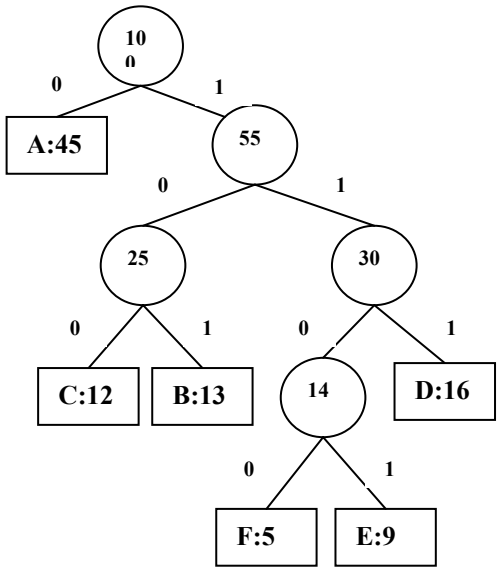
Bizonyítható, hogy minden dekódolható kódhoz létezik vele ekvivalens prefix kód.

Példa:

Betű *c*	Gyakoriság *f(c)*	Fix hossz		Változó hossz	
		Kódszó	Hossz	Kódszó	Hossz
A	45	000	3	0	1
B	13	001	3	101	3
C	12	010	3	100	3
D	16	011	3	111	3
E	9	100	3	1101	4
F	5	101	3	1100	4

Megadjuk a két kód kódfáját. A téglalapok a fa levelei, a kódolandó ábécé betűi a gyakoriságokkal. A belső csúcsok körök, bennük a csúcs gyerekeinek gyakoriságösszege szerepel.





Definíció: A fa költsége

A fa költségének nevezzük a: $B(T)=\sum_{c\in C} f(c)\cdot d(c)$ számot, ahol *c* a *C* ábécé betűje, *f(c)* a betű gyakorisága és *d(c)* a betű mélysége a fában (kódszóhossz).

A fa költségét leosztva a file-t alkotó betűk számával az egy betűre jutó átlagköltséget kapjuk, ami úgy is értelmezhető, mint az átlagos kódszóhossz. Ha ez kisebb, mint egy, akkor tömörítésről beszélünk.

Definíció: Optimális kód

Optimális a kód, ha a fa költsége minimális.

Az optimális kód bináris fája mindig teljes. A Huffman-kód optimális kód.

A Huffman-kód szerkesztése:

Jelölje a kódolandó ábécé betűinek halmazát *C* és legyen a betűk száma *n*. A Huffman kód szerkesztési algoritmusának pszeudokódja kiindul egy prioritási sorból, amelyben a betűk a gyakoriságaik szerint vannak kulcsolva. Ezután *n*-1 lépésben mindig a két éppen akkor legkisebb gyakoriságú elemet összevonva felépít egy optimális kódfát. Az optimum nem egyértelmű, mert az optimális fában bármely csúcs bal és jobb gyerekét felcserélve újra optimális fát kapunk.

Egy algoritmustervezési startégia a **mohó algoritmus** tervezése. Ezt a heurisztikát általában optimalizálásra használják. A mohó stratégia elve szerint az adott pillanatban mindig az ott legjobbnak tűnő lehetőséget választjuk a részprobléma megoldására, azaz a pillanatnyi lokális optimumot választjuk. Ez a választás függhet az előző választásoktól, de nem függ a későbbiektől. A mohó stratégia nem mindig vezet globális optimumra.

A Huffman kód szerkesztési algoritmusa mohó algoritmus, mivel minden lépésben úgy von össze, hogy a költség a legkisebb mértékben növekedjen. A teljes fa költsége megegyezik az összevonási lépések költségének összegével.

5.2.3.1. algoritmus
Huffman kód készítése

	//	$T(n)=\Theta(n\log n)$
1	HUFFMAN(C, T)	
2	// Input paraméter: C - a kódolandó abécé a gyakoriságokkal	
3	// Output paraméter: T – a kódfa	
4	//	
5	// Jelölje $\lvert C \rvert$ az ábécé betűinek a számát	
6	// Létrehozunk egy Q minimum prioritási sort az ábécé betűiből a gyakoriságot használva prioritásnak. A sor elemei fák lesznek, kezdetben minden fát egyetlen betű alkot.	
7	Q ← C	
8	**FOR** i ← 1 **TO** $\lvert C \rvert - 1$ **DO**	
9	PONTOT_LÉTESÍT(z)	//Egy új z csomópontot hoz létre, melyet később a Q sorba be fogunk szúrni. Egy új fa gyökere lesz.
10	x ← Bal[z] ← KIVESZ_MIN(Q)	//Kivesszük a Q sorból legkisebb elemet és az új z csomópont bal gyerekének nyílvánítjuk valamint elnevezzük x-nek átmenetileg
11	y ← Jobb[z] ← KIVESZ_MIN(Q)	//Kivesszük a Q sorból a következő legkisebb elemet és az új z csomópont jobb gyerekének nyílvánítjuk valamint elnevezzük y-nak átmenetileg
12	f[z] ← f[x]+f[y] gyakoriság	// Az új csomópontra meghatározzuk a értéket (prioritás)
13	BESZÚR(Q,z)	// Az új z csomópontot beszúrjuk a Q sorba
14	KIVESZ_MIN(Q, T)	// A Q sor egyetlen elemét, mint eredményt Kivesszük a sorból