1.3. tétel. *Létezik olyan $f : \mathcal{X} \to \mathcal{Y}^*$ prefix kód, amelyre*

$$\mathbf{E}|f(X)| < \frac{H(X)}{\log s} + 1.$$

1. BIZONYÍTÁS: **(Shannon–Fano-kód)** Feltehetjük, hogy $p(x_1) \geq p(x_2) \geq \cdots \geq p(x_{n-1}) \geq p(x_n) > 0$, mert különben az $\mathcal{X}$ elemeinek átindexelésével elérhetjük ezt. A bizonyítás technikája miatt ismét felhasználjuk az $y_i \mapsto i-1,\ i = 1,\ldots,s$ megfeleltetést. Legyenek a w_i számok a következők:

$$w_1 = 0, \qquad w_i = \sum_{l=1}^{i-1} p(x_l), \qquad i = 2,\ldots,n.$$

Kezdjük a w_i számokat felírni s-alapú számrendszerbeli tört alakban. A w_j-hez tartozó törtet olyan pontosságig írjuk le, ahol az először különbözik az összes többi w_i, $i \neq j$ szám hasonló alakbeli felírásától. Miután ily módon n darab véges hosszú törtet kapunk, az $f(x_j)$ legyen a w_j-hez tartozó tört az egészeket reprezentáló nulla nélkül. Könnyedén belátható, hogy az így kapott kód prefix. A konstrukció miatt x_j-hez létezik olyan x_k, hogy w_j és w_k végtelen tört alakjában az első $|f(x_j)|-1$ számjegy azonos. Nyilvánvaló, hogy vagy w_{j+1} vagy w_{j-1} tört alakja ilyen lesz, mivel ezek a w_j-hez legközelebbi számok. Az első esetben

$$p(x_j) = w_{j+1} - w_j < s^{-|f(x_j)|+1}, \tag{1.9}$$

a második esetben

$$p(x_{j-1}) = w_j - w_{j-1} < s^{-|f(x_j)|+1},$$

de ekkor $p(x_j) \leq p(x_{j-1})$ miatt (1.9) megint következik. Tehát mindkét esetben

$$-\log p(x_j) > (|f(x_j)|-1)\log s,$$

amiből mindkét oldalt $p(x_j)$-vel szorozva és minden j-re összegezve

$$-\sum_{j=1}^{n} p(x_j)\log p(x_j) > \left(\sum_{j=1}^{n} p(x_j)|f(x_j)| - 1\right)\log s$$

következik, így a tételt bebizonyítottuk. ■

2. BIZONYÍTÁS: Legyenek az l_i pozitív egész számok olyanok, hogy

$$-\log_s p(x_i) \leq l_i < -\log_s p(x_i) + 1, \qquad i = 1,\ldots,n \tag{1.10}$$

ahol $\log_s$ az s-alapú logaritmust jelöli. Az l_i számokat nyilván egyértelműen meg lehet így választani. A bal oldali egyenlőtlenségből következik, hogy

$$\sum_{i=1}^{n} s^{-l_i} \leq \sum_{i=1}^{n} s^{\log_s p(x_i)} = \sum_{i=1}^{n} p(x_i) = 1$$

tehát az l_i számok kielégítik az 1.2. lemma feltételét, és így létezik f prefix kód l_i hosszú kódszavakkal ($|f(x_i)| = l_i$). Ha a jobb oldali egyenlőtlenséget (1.10)-ben $p(x_i)$-vel megszorozzuk és minden i-re összegezzük, akkor azt kapjuk, hogy

$$\sum_{i=1}^{n} p(x_i) \cdot l_i < -\sum_{i=1}^{n} p(x_i) \log_s p(x_i) + \sum_{i=1}^{n} p(x_i) = \frac{H(X)}{\log s} + 1. \qquad \blacksquare$$

Az 1.1. definíció utáni megjegyzés b) pontjában említettük, hogy a betűnkénti kódolásnak van egy természetes általánosítása, a **blokk-kódolás**. Ezt formálisan egy $f : \mathcal{X}^m \to \mathcal{Y}^*$ leképezéssel definiálhatjuk, ahol tehát a forrásábécé betűiből alkotott rendezett m-eseket tekintjük forrásszimbólumoknak és ezeknek feleltetünk meg kódszavakat. A helyzet tulajdonképpen nem változik a betűnkénti kódolás esetéhez képest, hiszen egy új $\widehat{\mathcal{X}}$ forrásábécét definiálhatunk az $\widehat{\mathcal{X}} = \mathcal{X}^m$ jelöléssel, és az eddig elmondottak mind érvényben maradnak. Az egyértelmű dekódolhatóság definíciója ugyanaz marad, mint a betűnkénti kódolás esetében, az előbbieket szem előtt tartva. Legyen $\mathbf{X} = (X_1, \ldots, X_m)$ egy valószínűségi vektorváltozó, melynek koordinátái az $\mathcal{X}$-ből veszik értékeiket. Az entrópia 1.3. definíciójából jól látszik, hogy az csakis az eloszlástól függ. Mivel $\mathbf{X}$ is csak véges sok különböző értéket vehet fel, az 1.3. definíciót közvetlenül alkalmazhatjuk. Az $\mathbf{X}$ entrópiája tehát a

$$p(\mathbf{x}) = p(x_1, \ldots, x_m) = \mathbf{P}\{X_1 = x_1, \ldots, X_m = x_m\}, \qquad x_1, \ldots, x_m \in \mathcal{X},$$

jelölést bevezetve a következő:

$$\begin{aligned} H(\mathbf{X}) &= -\sum_{\mathbf{x} \in \mathcal{X}^m} p(\mathbf{x}) \log p(\mathbf{x}) = \\ &= -\sum_{x_1 \in \mathcal{X}} \cdots \sum_{x_m \in \mathcal{X}} p(x_1, \ldots, x_m) \log p(x_1, \ldots, x_m). \end{aligned}$$

Az $\mathbf{X} = (X_1, \ldots, X_m)$ entrópiájára a $H(\mathbf{X})$ jelölés mellett, ahol ez a célszerűbb, gyakran a $H(X_1, \ldots, X_m)$ jelölést fogjuk használni. Ha az $X_1, X_2, \ldots, X_m$ valószínűségi változók függetlenek, akkor $H(\mathbf{X})$ a koordináta valószínűségi változók entrópiáinak összege:

$$H(\mathbf{X}) = -\sum_{\mathbf{x} \in \mathcal{X}^m} p(\mathbf{x}) \log p(\mathbf{x}) =$$

$$
\begin{aligned}
&= -\sum_{x_1\in\mathcal{X}}\cdots\sum_{x_m\in\mathcal{X}} p_1(x_1)\cdots p_m(x_m)\log p_1(x_1)\cdots p_m(x_m) = \\
&= -\left(\sum_{x_1\in\mathcal{X}} p_1(x_1)\log p_1(x_1)+\cdots+\sum_{x_m\in\mathcal{X}} p_m(x_m)\log p_m(x_m)\right) = \\
&= \sum_{i=1}^{m} H(X_i), \qquad (1.11)
\end{aligned}
$$

ahol $p_i(x) = \mathbf{P}\{X_i = x\}, \quad i = 1,\ldots,m.$

Ha az $X_1,\ldots,X_m$ valószínűségi változók nemcsak függetlenek, hanem azonos eloszlásúak is, akkor (1.11)-ből

$$H(X_1,\ldots,X_m) = mH(X_1) \qquad (1.12)$$

következik.

A betűnkénti átlagos kódszóhossz definíciója értelemszerűen módosul a blokk-kódolás esetében: a kódszóhossz várható értékét el kell osztani az egy blokkot alkotó forrásbetűk számával, vagyis a betűnkénti átlagos kódszóhosszon a következő mennyiséget értjük:

$$\frac{1}{m}\mathbf{E}|f(\mathbf{X})| = \frac{1}{m}\sum_{\mathbf{x}\in\mathcal{X}^m} p(\mathbf{x})|f(\mathbf{x})|$$

Értelemszerűen az 1.2. tétel állítása blokk-kódolás esetén is igaz:

$$\mathbf{E}|f(\mathbf{X})| \geq \frac{H(\mathbf{X})}{\log s},$$

hiszen a tétel bizonyítása közvetlenül itt is alkalmazható.

Az 1.3. tétel következményeként megmutatjuk, hogy blokk-kódolás segítségével a betűnkénti átlagos kódszóhossz alsó korlátja tetszőlegesen közelíthető.

1.3. következmény. *Ha $X_1,\ldots,X_m$ független, X-szel azonos eloszlású valószínűségi változók, akkor létezik olyan $f:\ \mathcal{X}^m \to \mathcal{Y}^*$ prefix kód, hogy*

$$\frac{1}{m}\mathbf{E}|f(\mathbf{X})| < \frac{H(X)}{\log s} + \frac{1}{m},$$

tehát a betűnkénti átlagos kódszóhossz az m blokkhossz növelésével tetszőlegesen közelíti a $\frac{H(X)}{\log s}$ alsó korlátot.

BIZONYÍTÁS: Az 1.3. tételt felhasználva létezik olyan $f: \mathcal{X}^m \to \mathcal{Y}^*$ prefix kód, hogy

$$\sum_{\mathbf{x}\in\mathcal{X}^m} p(\mathbf{x})|f(\mathbf{x})| < \frac{H(\mathbf{X})}{\log s} + 1.$$

Ebből (1.12) miatt következik

$$\sum_{\mathbf{x}\in\mathcal{X}^m} p(\mathbf{x})|f(\mathbf{x})| < \frac{mH(X)}{\log s} + 1,$$

ahol mindkét oldalt m-mel osztva megkapjuk az állítást. ■

1.3. Optimális kódok, bináris Huffman-kód

Az 1.2. tétel alsó korlátot ad az egyértelműen dekódolható kódok átlagos kódszóhosszára, az 1.3. tétel pedig mutat egy olyan kódkonstrukciót, ahol ezt a korlátot jól megközelíthetjük. A jó kód konstrukciójának problémáját persze ennél általánosabban is felvethetjük: konstruáljuk meg az optimális, azaz legkisebb átlagos kódszóhosszú kódot, ha adott az X valószínűségi változó eloszlása. Először is gondoljuk át, hogy optimális kód valóban létezik. Ugyan véges kódábécé esetén is az egyértelműen dekódolható vagy prefix kódok halmaza végtelen, de a bizonyos átlagos kódszóhossznál (pl. $\frac{H(X)}{\log s}+1$) jobb kódok halmaza véges. Másodszor, vegyük észre, hogy az optimális kód nem feltétlenül egyértelmű; egyenlő valószínűségekhez tartozó kódszavakat felcserélhetünk, csakúgy, mint az egyenlő hosszú kódszavakat, anélkül, hogy az átlagos kódszóhosszat ezzel megváltoztatnánk.

Egy optimális kód konstruálását az 1.1. következmény értelmében egy optimális prefix kód konstruálására lehet visszavezetni. Ezért a következőkben kimondjuk az optimális prefix kódok néhány tulajdonságát. A továbbiakban az egyszerűség kedvéért a bináris, $s = 2$ esettel foglalkozunk; feltesszük, hogy $\mathcal{Y} = \{0,1\}$. Az általános, $s > 2$ eset bonyolultabb, és a dolog lényege így is jól látható.

1.4. tétel. *Ha az $f: \mathcal{X} \to \{0,1\}^*$ prefix kód optimális, és $\mathcal{X}$ elemei úgy vannak indexelve, hogy $p(x_1) \geq p(x_2) \geq \cdots \geq p(x_{n-1}) \geq p(x_n) > 0$, akkor feltehető, hogy f-re a következő három tulajdonság teljesül:*

a) $|f(x_1)| \leq |f(x_2)| \leq \cdots \leq |f(x_{n-1})| \leq |f(x_n)|$, *vagyis nagyobb valószínűségekhez kisebb kódszóhosszak tartoznak.*

b) $|f(x_{n-1})| = |f(x_n)|$, *vagyis a két legkisebb valószínűségű forrásbetűhöz tartozó kódszó egyenlő hosszú.*

c) Az $f(x_{n-1})$ és az $f(x_n)$ kódszavak csak az utolsó bitben különböznek.

BIZONYÍTÁS:

a) Tegyük fel, hogy $p(x_k) > p(x_j)$ és $|f(x_k)| > |f(x_j)|$. Ekkor x_j és x_k kódját felcserélve egy új f^* kódot vezethetünk be, amelyre

$$\begin{aligned}\sum_{i=1}^{n} p(x_i)|f(x_i)| - \sum_{i=1}^{n} p(x_i)|f^*(x_i)| = \\ = p(x_k)|f(x_k)| + p(x_j)|f(x_j)| - p(x_k)|f(x_j)| - p(x_j)|f(x_k)| = \\ = p(x_k)(|f(x_k)| - |f(x_j)|) - p(x_j)(|f(x_k)| - |f(x_j)|) = \\ = (p(x_k) - p(x_j))(|f(x_k)| - |f(x_j)|) > 0,\end{aligned}$$

tehát f nem lehet optimális.

b) Az állítás egyszerűen belátható, ha arra gondolunk, hogy $|f(x_{n-1})| < |f(x_n)|$ esetén az $|f(x_n)|$ utolsó bitjét levágva az optimálisnál kisebb átlagos kódszóhosszú kódot kapnánk, ami még mindig prefix tulajdonságú. Valóban, mivel az eredeti kódszó minden más kódszónál legalább eggyel hosszabb volt, a csonkítással kapott új kódszó az eredeti kód prefix tulajdonsága miatt nem azonos semelyik másikkal, és ugyanezen okok miatt nem is folytatása semmilyen más kódszónak.

c) Az előző gondolatmenetből világos, hogy ha létezik olyan $f(x_i)$ kódszó, hogy $f(x_i)$ és $f(x_n)$ csak az utolsó bitben különböznek, akkor az a) és b) miatt $|f(x_i)| = |f(x_{n-1})| = |f(x_n)|$. Így ha $i \neq n-1$, akkor x_i és x_{n-1} kódját felcserélve c) teljesül, és a kód optimális marad. ■

1.5. tétel. *Tegyük most fel, hogy az 1.4. tétel feltételei teljesülnek, és hogy a $\{p(x_1), p(x_2), \ldots, p(x_{n-1}) + p(x_n)\}$ valószínűségeloszláshoz ismerünk egy g optimális bináris prefix kódot. (Az x_{n-1} és x_n forrásbetűket összevonjuk egy $\bar{x}_{n-1}$ szimbólumba; $p(\bar{x}_{n-1}) = p(x_{n-1}) + p(x_n)$). Ekkor az eredeti $\{p(x_1), p(x_2), \ldots, p(x_{n-1}), p(x_n)\}$ eloszlás egy optimális f prefix kódját kapjuk, ha a $g(\bar{x}_{n-1})$ kódszót egy nullával, illetve egy egyessel kiegészítjük (a többi kódszót pedig változatlanul hagyjuk).*

BIZONYÍTÁS: A g és az f átlagos kódszóhosszát L_g-vel és L_f-fel jelölve azt kapjuk, hogy

$$L_f = L_g + p(x_{n-1}) + p(x_n).$$

Ha f nem lenne optimális, akkor a nála kisebb L_{f^*} átlagos kódszóhosszú f^* kódról az 1.4. tétel c) pontja szerint feltehetnénk, hogy $f^*(x_{n-1})$ és $f^*(x_n)$ nem rövidebbek semelyik más kódszónál, és csak az utolsó bitben különböznek egymástól. Ezt az utolsó bitet elhagyva a $\{p(x_1), p(x_2), \ldots, p(x_{n-1}) + p(x_n)\}$ eloszlásra egy g^* kódot kapnánk, amire

$$L_{g^*} = L_{f^*} - p(x_{n-1}) - p(x_n) < L_f - p(x_{n-1}) - p(x_n) = L_g$$

teljesülne, tehát az optimális g-nél jobb kódot kapnánk, ami lehetetlen. ■

Az előző tétel alapján már megadhatjuk az optimális prefix kód Huffman-féle konstrukcióját: A két legkisebb valószínűség összevonásával addig redukáljuk a problémát, amíg az triviális nem lesz, vagyis amíg összesen két valószínűségünk marad. Ezt $n-2$ lépésben érhetjük el. Ezután az összevonások megfordításával, mindig a megfelelő kódszó kétféle kiegészítésével újabb $n-2$ lépésben felépítjük az optimális kódot, a Huffman-kódot. A következő példában nyomon követhetjük az algoritmus egyes lépéseit.

1.4. példa. Legyen $n = 5$ és $p(x_1) = 0.4,\ p(x_2) = 0.3,\ p(x_3) = 0.15,\ p(x_4) = 0.1,\ p(x_5) = 0.05$, és keressük meg az ehhez tartozó Huffman-kódot:

Az egyes lépéseket az oszlopok számozása jelzi balról-jobbra, az összevonásokat a nyilak mentén lehet nyomon követni.

1.		**2.**		**3.**		**4.**
0.4		0.4		0.4		0.4
0.3		0.3		0.3	$\rightarrow$	0.6
0.15		0.15	$\rightarrow$	0.3	$\nearrow$	
0.1	$\rightarrow$	0.15	$\nearrow$			
0.05	$\nearrow$					

A 0.4 és 0.6 valószínűségek optimális prefix kódja nyilván a 0 és az 1 (vagy fordítva). Az összevonások megfordításával elvégezhetjük a Huffman-kód felépítését. A valószínűségek mellett szögletes zárójelben az egyes lépésekben kapott kódszavak állnak.

4.			**3.**			**2.**			**1.**	
0.4	[0]		0.4	[0]		0.4	[0]		0.4	[0]
0.6	[1]	$\rightarrow$	0.3	[10]		0.3	[10]		0.3	[10]
		$\searrow$	0.3	[11]	$\rightarrow$	0.15	[110]		0.15	[110]
					$\searrow$	0.15	[111]	$\rightarrow$	0.1	[1110]
								$\searrow$	0.05	[1111]

Tehát $f(x_1) = 0,\ f(x_2) = 10,\ f(x_3) = 110,\ f(x_4) = 1110,\ f(x_5) = 1111.$

Az előzőekben feltételeztük, hogy ismerjük a bemenet eloszlását. Ez azonban nincs mindig így, ekkor az eloszlást a relatív gyakoriságokkal kell becsülnünk.

A gyakorlati problémák során általában adott a $Z_1, \ldots, Z_N$ bemeneti sorozat (szöveg, fájl), amelyet szeretnénk optimálisan kódolni. Feladatunk a $\sum_{i=1}^{N} |f(Z_i)|$ minimalizálása, vagyis az optimális f kódolófüggvény megválasztása. Ez egyenértékű a kódszóhosszak átlagának, $\frac{1}{N}\sum_{i=1}^{N} |f(Z_i)|$-nek a minimalizálásával.

Belátjuk, hogy $\frac{1}{N}\sum_{i=1}^{N} |f(Z_i)| = \mathbf{E}_N|f(Z)|$, ha a várható értéket az empirikus eloszlás szerint vesszük, tehát a forrásbetűk valószínűségét a relatív gyakoriságukkal definiáljuk: $p_N(x_j) = \frac{1}{N}\sum_{i=1}^{N} I_{\{Z_i=x_j\}}$, ahol $I_{\{\,\}}$ az indikátor függvény. Nyilván

$$\begin{aligned}
\mathbf{E}_N|f(Z)| &= \sum_{j=1}^{n} p_N(x_j)|f(x_j)| = \\
&= \sum_{j=1}^{n} \frac{1}{N}\sum_{i=1}^{N} I_{\{Z_i=x_j\}}|f(x_j)| = \\
&= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{n} I_{\{Z_i=x_j\}}|f(x_j)| = \\
&= \frac{1}{N}\sum_{i=1}^{N} |f(Z_i)|
\end{aligned}$$

(A gyakorlatban magát a kódoló függvényt is le kell írni, át kell vinni a dekódolóhoz, és ez a „fejléc” így a fentinél nagyobb átlagos kódszóhosszat eredményez. Az aszimptotikus vizsgálat során azonban ettől a konstans költségtől eltekinthetünk.)

A Huffman-kódolás fenti algoritmusát csak két lépésben tudjuk végrehajtani. Először meghatározzuk a forrásbetűk relatív gyakoriságát, ami az előzőek értelmében megegyezik a valószínűségekkel, majd ennek felhasználásával elvégezzük a tényleges kódolást. Nem mindig engedhető meg azonban olyan nagy mértékű késleltetés, hogy csak az összes bemeneti adat megérkezése után kezdünk hozzá a kimenet előállításához, másrészt a kétmenetes beolvasás akkor is lassítja az algoritmust (bár kétségkívül optimális kódot eredményez), ha a bemenet már rendelkezésre áll. A gyakorlatban ezért sokszor érdemes „egymenetes” algoritmust használni. Így az optimalitás rovására időt takaríthatunk meg. Egy forrásbetűt az előző forrásbetűk előfordulásai alapján kódolunk, s ezzel együtt lépésenként változik maga a kód is. Tehát az aktuális forrásbetű kódolását egy, az előzőleg

feldolgozott forrásbetűkre optimális kóddal hajtjuk végre. Ezt az eljárást **adaptív Huffman-kódolás**nak nevezzük.

A Huffman-kódot bináris faként is ábrázolhatjuk. A leveleket a forrásszimbólumokkal címkézzük, az éleken pedig a kódábécé ($\{0,1\}$) elemei szerepelnek. A csúcsokban a relatív gyakoriságok állnak. Egy forrásszimbólumhoz tartozó kódszót úgy kapunk meg, hogy a fa gyökerétől a megfelelő levélig húzódó út élein szereplő kódbetűket sorban összeolvassuk (konkatenáljuk). A Huffman-kódolás szemléletesen úgy történik, hogy kiindulásként felvesszük a forrásszimbólumokhoz tartozó leveleket, a csúcsokba a forrásszimbólumok relatív gyakoriságait írjuk, majd lépésenként mindig a két legkisebb értéket tartalmazó csúcs fölé teszünk egy új csúcsot (szülőt), s ebbe a két régi érték összegét írjuk. Az eljárás végén kialakul az összefüggő fa, melynek a legutolsó lépésben megkapott csúcsa lesz a gyökér.

A bináris Huffman-fában a relatív gyakoriságok szerepelnek. Adaptív Huffman-kódolás esetén ezek helyett a gyakoriságokat használhatjuk (hiszen utóbbiakat a bemenet hosszával elosztva megkapjuk a relatív gyakoriságokat, és számunkra csak az értékek egymáshoz való aránya érdekes). Két összevont betű szülőjének súlyát a szokásos módon, a két gyakoriság összegeként kapjuk.

Amennyiben a fa testvér (vagyis közös szülővel rendelkező) pontpárjait a gyökértől a levelek felé haladva fel tudjuk sorolni súlyuk szerint nemnövekvő sorrendben, a fa rendelkezik a testvérpár tulajdonsággal (sibling pair property). Bebizonyítható, hogy egy Huffman-fa akkor és csak akkor optimális, ha rendelkezik a testvérpár tulajdonsággal.

1.5. példa. Az 1.3. ábrán látható fára teljesül a testvérpár tulajdonság. A párok a következők:

$$(28,18);\ (15,13);\ (11,7);\ (7,6);\ (6,5);\ (4,3);\ (3,2);\ (2,2);\ (2,1)$$

Ezek nemnövekvő rendezése:

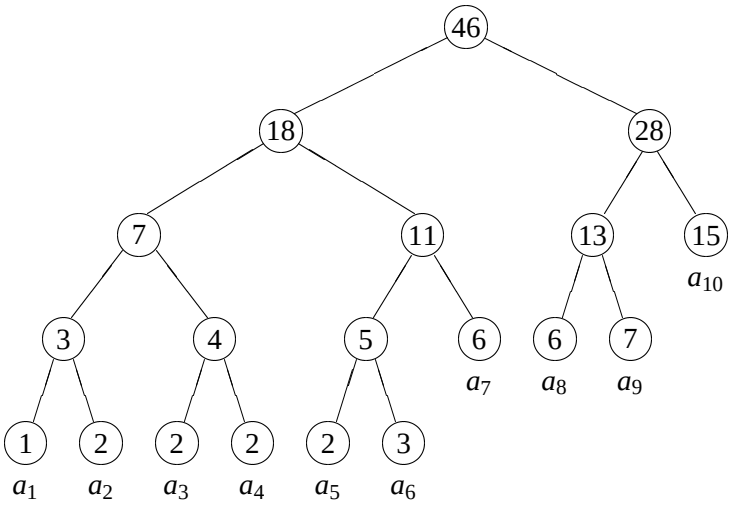
$$28 \geq 18 \geq 15 \geq 13 \geq 11 \geq 7 \geq 7 \geq 6 \geq 6 \geq$$

$$\geq 5 \geq 4 \geq 3 \geq 3 \geq 2 \geq 2 \geq 2 \geq 2 \geq 1$$

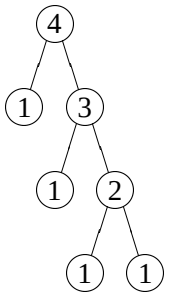
Az 1.4. ábra kódfájának párjai:

$$(3,1);\quad (2,1);\quad (1,1)$$

Ezeket nem tudjuk megfelelően sorba rendezni, így nem teljesül a fára a testvérpár tulajdonság.



1.3. ábra. Kódfa, amelyre teljesül a testvérpár tulajdonság.



1.4. ábra. Kódfa, amelyre nem teljesül a testvérpár tulajdonság.

Az adaptív algoritmus lépései a következők:
Ismerjük a forrásszimbólumokat: $\mathcal{X} = \{a_1, a_2, \ldots, a_k\}$. Kiindulásként felépítünk egy olyan Huffman-fát, amelyben — ha nincs információnk a betűk előfordulásának valószínűségéről — minden forrásszimbólum azonos, 1 gyakorisággal szerepel. Így egy kiegyensúlyozott bináris fát kapunk. (Ha ismerjük a kezdeti eloszlást, megtehetjük, hogy erre építjük fel az adaptív algoritmus kiindulási fáját.) Elküldjük a dekódolónak sorrendben (pl. a gyökértől a levelek felé, szintenként, balról jobbra) a lehetséges forrásszimbólumokat, amelyből az szintén felépíti a kiindulási Huffman-fát. (Ügyelni kell arra, hogy a kódoló és a dekódoló azonos algoritmust használjon.) A kódolás során beolvassuk a k-adik forrásszimbólumot. Ezt a

korábban beolvasott $k-1$ betű feldolgozásával kialakult, lokálisan optimális kódfával kódoljuk, majd eggyel növeljük a forrásszimbólum gyakoriságszámlálóját. Aktualizáljuk a fát, vagyis megvizsgáljuk, hogy fennáll-e még a testvérpár tulajdonság. Amennyiben nem, helyreállítjuk, így biztosítva az optimalitást. Ezeket a lépéseket a dekóder is elvégzi, tehát a következő kódbetű dekódolásakor ugyanazt a fát fogja használni, mint amelyet a kódoló használt annak előállításakor. A $(k+1)$-edik betű beolvasása után az előzőekhez hasonlóan folytatjuk.

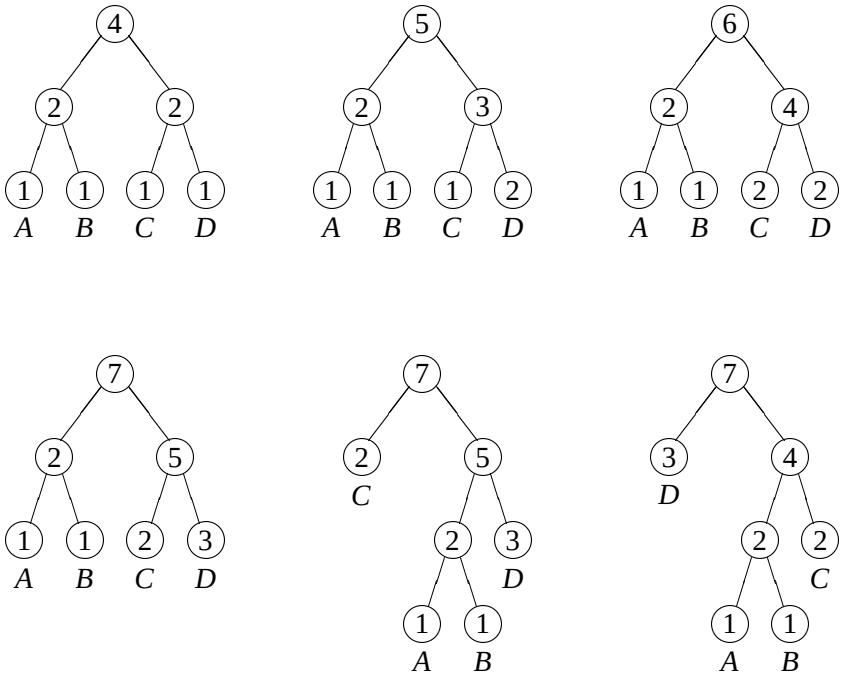
A következő két szabály használható a fa aktualizálása során a testvérpár tulajdonság helyreállítására:

1. Két, azonos súlyú csúcs a hozzá tartozó részfákkal együtt megcserélhető. Ez természetesen nem befolyásolja a súlyokat.

2. Két levél a súlyukkal együtt megcserélhető.

A szabályok kombinálásával átalakítható egy nem megfelelő kódfa egy olyanná, amely teljesíti a testvérpár tulajdonságot.

Ügyelni kell arra, hogy hosszú bemenet esetén előfordulhat a szimbólumok gyakoriságszámlálójának túlcsordulása. Ezt megfelelő technikával orvosolni kell.



1.5. ábra. Az 1.6. példa kódfájának alakulása.