Greedy Algorithms

Code Theory

نظریه کد

- کارایی از نظر حجم داده کد شده

- آشکارسازی و تصحیح خطا

کدگذاری هافمن - فشرده سازی

Huffman Codes

CLRS Chapter 16.3

Encoding کدگذاری

Decoding کدگشایی

Cryptography

رمزنگاری

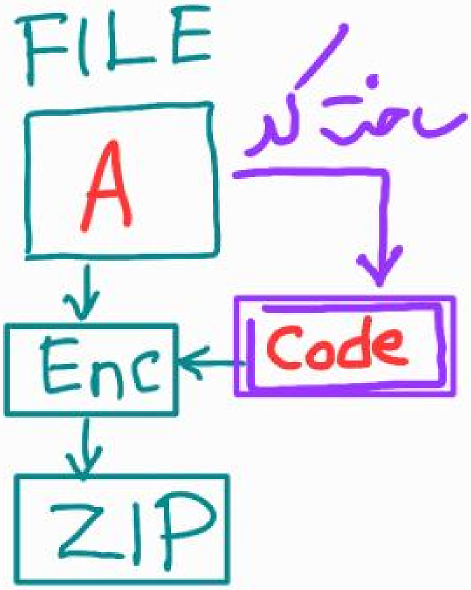
Encryption رمزگذاری

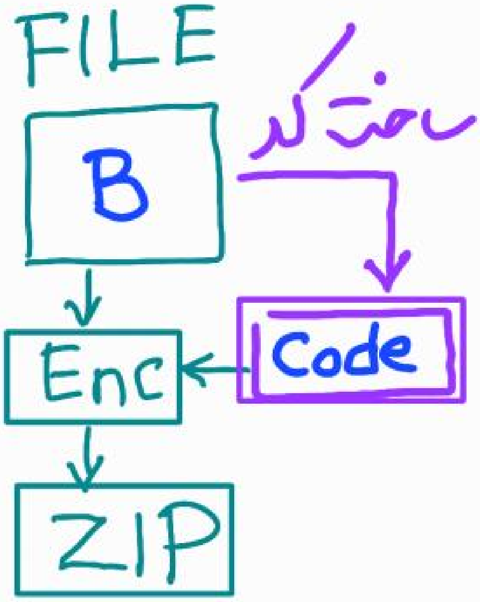
Decryption رمزگشایی

Huffman codes

کدگذاری هافمن
یک کدگذاری وابسته
به داده است

- Huffman codes compress data very effectively: savings of 20% to 90% are typical.
- Consider the data to be a sequence of characters. (string)
- Huffman's greedy algorithm uses
  الگوریتم ساخت جدول کد هافمن
  - a table giving *how often each character occurs* (i.e., its frequency)
    input
    جدول فراوانی هر کاراکتر در متن
  - build up an optimal way of representing each character as a binary string

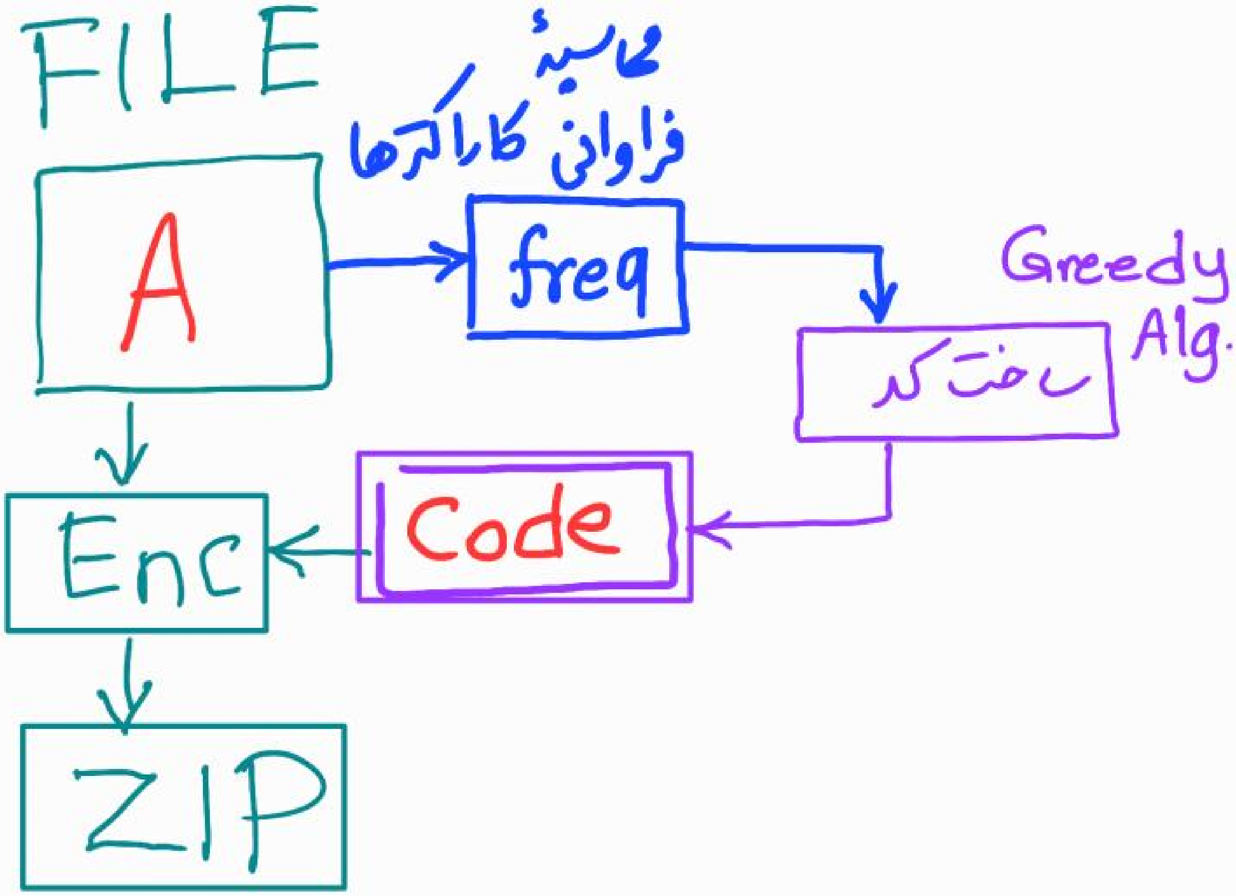




کد بطور اختصاصی بهترین فشرده سازی برای همان داده‌ای بدست می دهد که به ازای آن ساخته شده است.

- به عبارت دیگر کدگذاری هافمن یک کدگذاری **جهانی** نیست

- از آنجا که کد متناظر با هر فایل به همان فایل اختصاص دارد لازم است برای کدگشایی، گیرنده فایل فشرده سازی شده، **جدول کد** متناظر با آن فایل را نیز داشته باشد. **Code Table**



FILE
محاسبهٔ فراوانی کاراکترها
A
freq
Greedy Alg.
ساخت کد
Code
Enc
ZIP

یک کدگذاری اصطلاحاً کدگذاری طول-ثابت Fixed-Length نامیده می‌شود اگر طول تمام کدکلمه‌ها برابر باشد

کد "طول-متغیر" کدگذاری است که طول کدکلمه‌های آن متفاوت است.

- Suppose we have a 100,000-character data file that we wish to store compactly. = 100 KB
- And suppose that only 6 different characters appear, and the character a occurs 45,000 times.

$n=6$ تعداد بیت‌ها $= \lceil \log_2 n \rceil$

$\{a,b,c,d,e,f\}$ = مجموعه کاراکترهای تشکیل دهنده

طول کدکلمه = بهر کد

	a	b	c	d	e	f
Frequency (in thousands)	45	13	12	16	9	5
Fixed-length codeword	000	001	010	011	100	101
Variable-length codeword	0	101	100	111	1101	1100

طول کدکلمه = 3

طول کد متغیر

A 65 1000001

' ' 32 100000

جدول کد جهانی شبیه به ASCII (7bit) یا UniCode (16bit)

- Fixed length code
- Variable length code
  - Prefix-free code

کدگذاری بدون ابهام ممکن است تنها اگر

کد پیشوند-آزاد

فرض کنید رشته aabe را توسط کد طول متغیر کدگذاری کرده ام

001011101

چون کد، طول متغیر است از روش قطعه قطعه کردن نمی توان استفاده کرد

	a	b	c	d	e	f
Frequency (in thousands)	45	13	12	16	9	5
Fixed-length codeword	000	001	010	011	100	101
Variable-length codeword	0	101	100	111	1101	1100

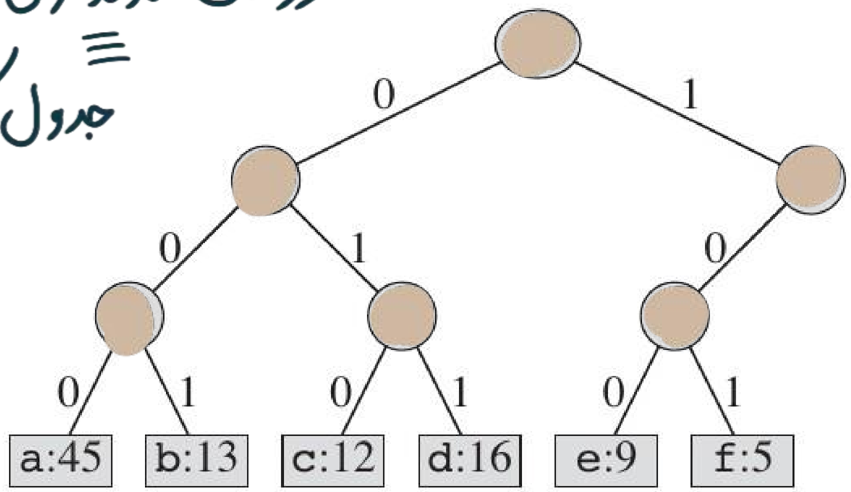
$$(45 \cdot 1 + 13 \cdot 3 + 12 \cdot 3 + 16 \cdot 3 + 9 \cdot 4 + 5 \cdot 4) \cdot 1{,}000 = 224{,}000 \text{ bits}$$

فضای کدهای پیشوند-آزاد چگونه بررسی می شود؟

Prefix-free code

فضای تمام
کدهای پیشوند-آزاد
درتناظر است با
فضای تمام
درختهای دودویی
با n گره
برگ که گره‌های برگ
درتناظر با کاراکترها قرار گرفته است

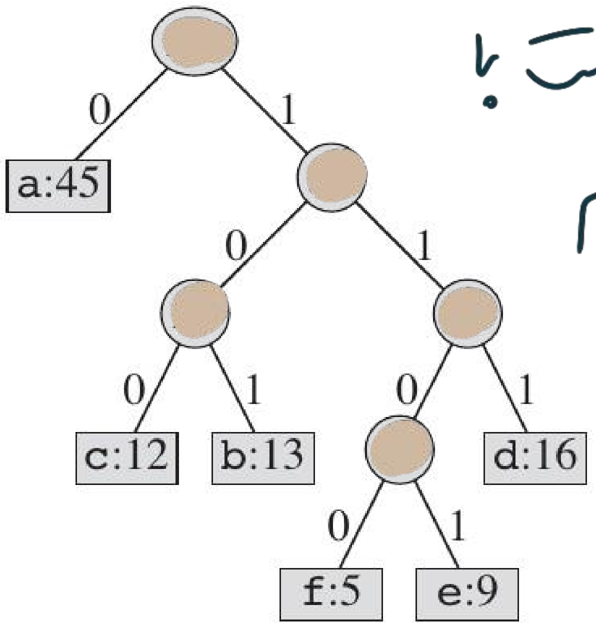
درخت کدگذاری
≡
جدول کد پیشوند آزاد



$n =$ (گره‌های پایانی) تعداد برگها

(a)

هر کاراکتر در تناظر با یک برگ



(b)

Optimal prefix code

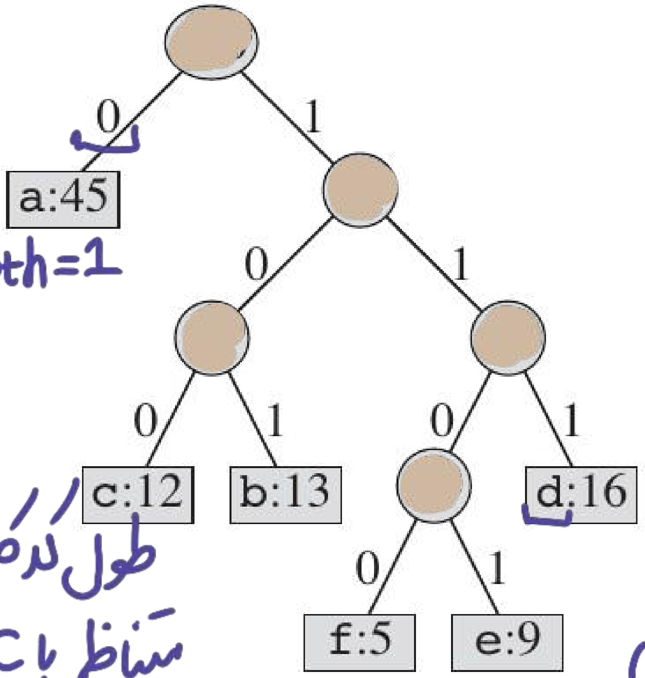
- Cost formulation

فرمول محاسبهٔ حجم فایل کدگذاری شده توسط
کد بازنمایی شده توسط درخت T
برحسب تعداد بیت

$$B(T) = \sum_{c \in C} c.freq \cdot d_T(c)$$

$$B(T) = \sum freq(c) \cdot \overbrace{depth_T(c)}$$

= طول کدکلمهٔ
متناظر با c



depth=1

depth =3

(T)

بدین ترتیب هدف جستجو در فضای درخت‌های دودویی با n گره برگ متناظر با کاراکترهای فایل است که

$$\min_{T \in \mathcal{T}} B(T)$$

$\mathcal{T}$: فضای تمام درخت‌های کدگذاری

$$\min \sum \underbrace{f(c)}_{\text{داده شده است}} \cdot \underbrace{d(c)}_{\text{متغیر}}$$

---

*) یک مسئله ساده‌تر. آرایه‌ای n تایی از اعداد مثبت داده شده

$A := \langle a_1, a_2, \dots, a_n \rangle$

بیابید جایگشتی از اعداد $\{b_1, b_2, \dots, b_n\}$ مثل

$X = \langle x_1, x_2, \dots, x_n \rangle$

به قسمی که

$$\min \sum_{i=1}^{n} a_i x_i$$

مثال. $A = \langle 3, 2, 7, 5 \rangle$

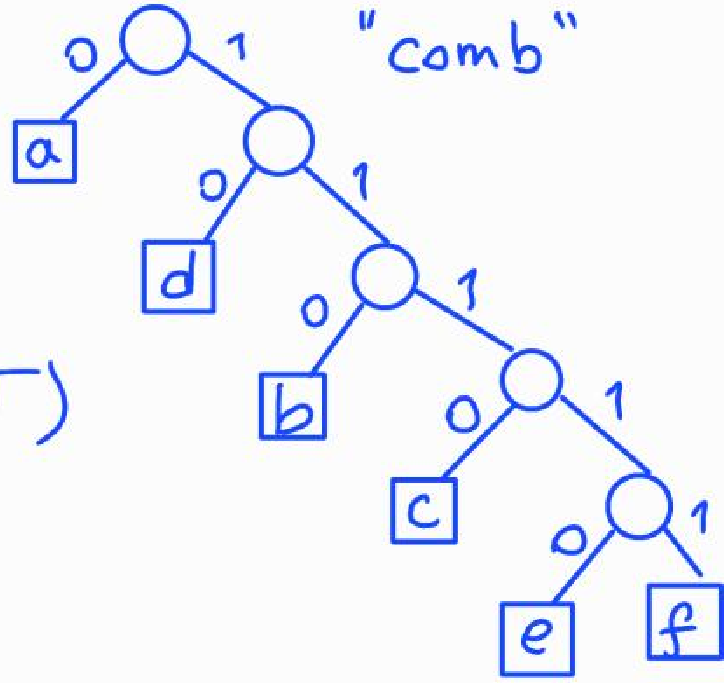
$B = \{6, 4, 9, 8\}$

یک الگوریتم حریصانه برای کدگذاری

$$\min \sum f(c)\cdot d(c)$$

یک ایده حریصانه این است که کاراکتری که بیشترین فراوانی را دارد متناظر با گره برگی قرار دهیم که در کمترین عمق است

c	a	b	c	d	e	f
f(c)	45	13	12	16	9	5



(T)

$$B(T) = 1\times 45 + 2\times 16 + 3\times 13 + 4\times 12 + 5\times 9 + 5\times 5$$

$$= 234$$

آیا درخت کدگذاری بهتری وجود ندارد؟

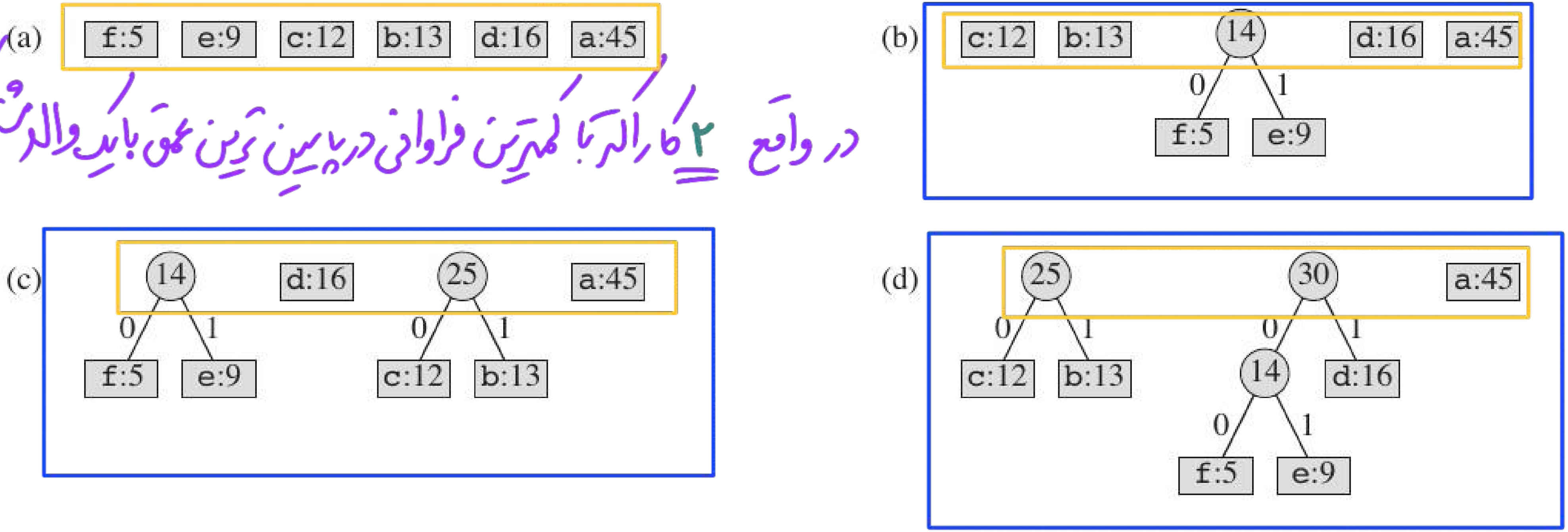
پاسخ: مثال قبلی 224 بیت بود.

$\Longleftarrow$ این الگوریتم حریصانه به جواب بهینه نمی‌رسد

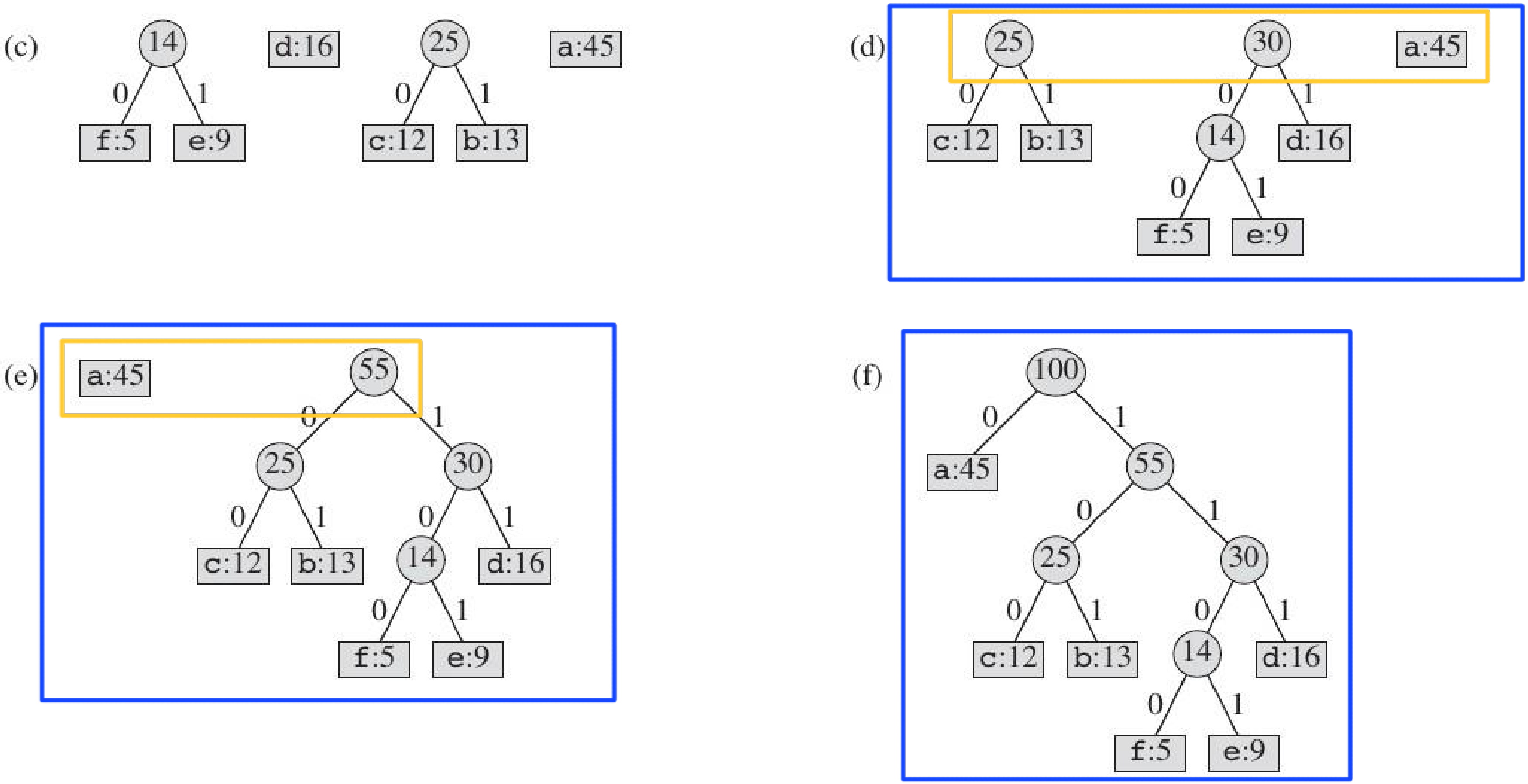
Huffman greedy algorithm

الگوریتم حریصانه ساخت درخت کدگذاری: "کاراکتر با کمترین فراوانی را در پایین ترین عمق درخت قرار دهید"

در واقع ۲ کاراکتر با کمترین فراوانی در پایین ترین عمق باید والد مشترک



Huffman greedy algorithm

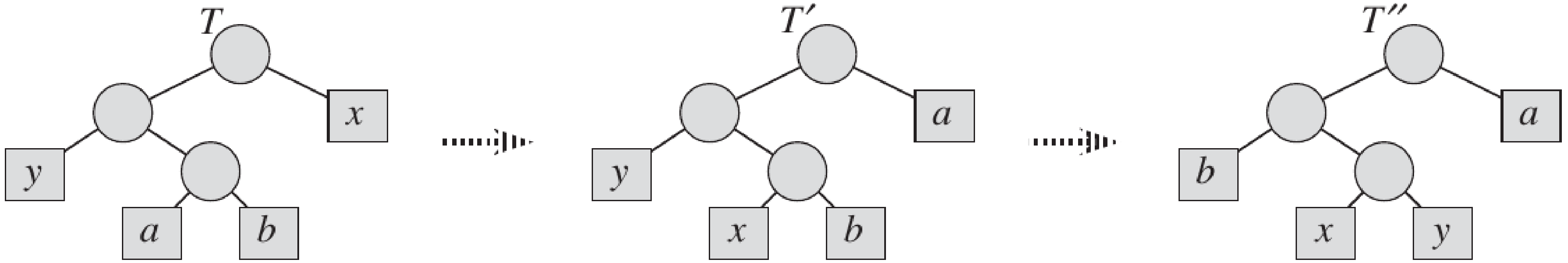


Huffman greedy algorithm

```
HUFFMAN(C)
n = |C|
Q = C        // Q یک صف اولویت بندی شامل کاراکترهای C که بر حسب
             فراوانی کاراکترها به ترتیب صعودی مرتب شده اند
for i = 1 to n − 1
    allocate a new node z
    z.left = x = EXTRACT-MIN(Q)
    z.right = y = EXTRACT-MIN(Q)
    z.freq = x.freq + y.freq
    INSERT(Q, z)
return EXTRACT-MIN(Q)        // return the root of the tree
```

Proof

به تناقض فرض کنید T یک درخت کدگذاری پیشوند آزاد است در راستای رسیدن به حجم فایل کدگذاری شده با آن کمینه است اما کاراکترهای y و x که کمترین فراوانی را دارند در پایین ترین عمق نیستند. حال ←



$$
\begin{aligned}
B(T) - B(T') &= \sum_{c \in C} c.freq \cdot d_T(c) - \sum_{c \in C} c.freq \cdot d_{T'}(c) \\
&= x.freq \cdot d_T(x) + a.freq \cdot d_T(a) - x.freq \cdot d_{T'}(x) - a.freq \cdot d_{T'}(a) \\
&= x.freq \cdot d_T(x) + a.freq \cdot d_T(a) - x.freq \cdot d_T(a) - a.freq \cdot d_T(x) \\
&= (a.freq - x.freq)(d_T(a) - d_T(x)) \\
&\geq 0\,,
\end{aligned}
$$

بطور مشابه با جابجا کردن b و y داریم

$B(T') \geq B(T'')$

$\Rightarrow B(T) \geq B(T'')$