دانشگاه کردستان
University of Kurdistan

طراحی و تحلیل الگوریتم ها

مدرس: سعدون عزیزی

s.azizi@uok.ac.ir

گروه مهندسی کامپیوتر

نیم سال دوم ۹۷-۹۶

تکنیک‌های طراحی الگوریتم‌ها

- تقسیم و حل
- برنامه‌ریزی پویا
- **الگوریتم‌های حریصانه**
- الگوریتم‌های عقبگرد
- الگوریتم‌های شاخه و حد

کدهای هافمن (Huffman Codes)

- دنباله‌ای از کاراکترها (مثلاً یک فایل متنی) را در نظر بگیرید که در آن هر کاراکتر دارای یک فراوانی است.
- می‌خواهیم چنین فایلی را به صورت یک کد دودویی نمایش دهیم که در آن هر کاراکتر با یک رشته دودویی یکتا نمایش داده می‌شود.
- کد با طول ثابت: در این روش، طول کد تمام کاراکترها ثابت و یکسان است.
- کد با طول متغیر: می‌توان به کاراکترهای پرکاربرد (فراوانی بالا) کدهای کوتاه و به کاراکترهای کم کاربرد (فراوانی پایین) کدهای بلند نسبت داد.

کدهای هافمن: مثال

- فرض کنید یک فایل داده شامل ۱۰۰،۰۰۰ کاراکتر روی الفبای {a,b,c,d,e,f} داریم که فروانی کاراکترهای درون فایل به صورت زیر می‌باشد:

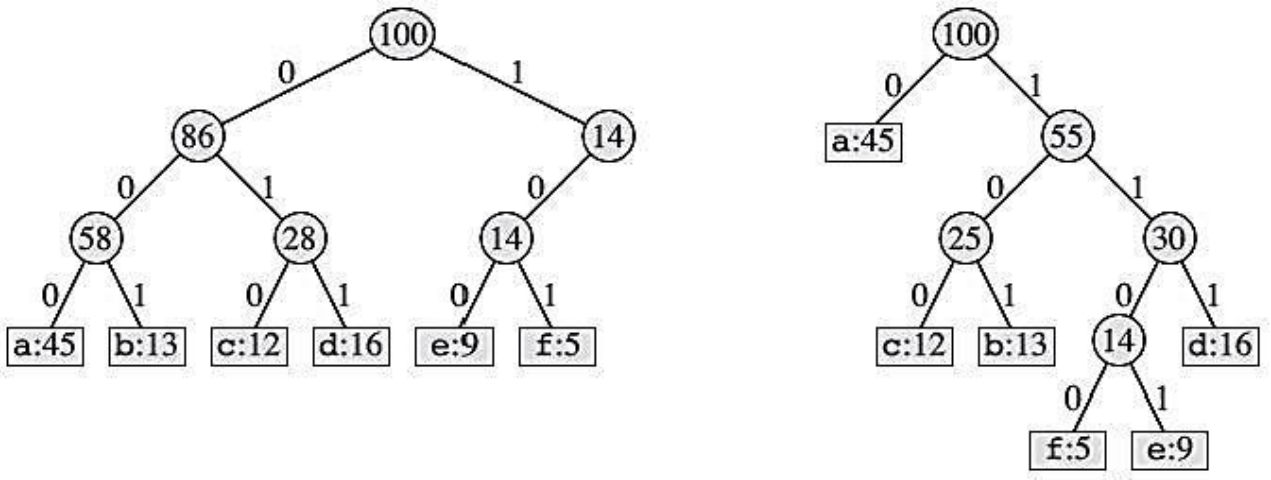
	a	b	c	d	e	f
فراوانی (تقسیم بر هزار)	۴۵	۱۳	۱۲	۱۶	۹	۵

- دو روش کد کردن این فایل به صورت زیر می‌باشند:

	a	b	c	d	e	f
کد کلمه با طول ثابت	۰۰۰	۰۰۱	۰۱۰	۰۱۱	۱۰۰	۱۰۱
کد کلمه با طول متغیر	۰	۱۰۱	۱۰۰	۱۱۱	۱۱۰۱	۱۱۰۰

کدهای هافمن: مثال

- درخت‌های مربوط به روش‌های کدگذاری مختلف (0 یعنی «برو به فرزند چپ» و 1 یعنی «برو به فرزند راست»)



- تعداد بیت‌های مورد نیاز برای کد کردن کل فایل
  - ۳۰۰ هزار بیت برای کدگذاری با طول ثابت
  - ۲۲۴ هزار برای کدگذاری با طول متغیر (۲۵٪ صرفه جویی در حافظه)

کدهای هافمن: کدهای پیشوندی

- برای نمایش یک فایل به صورت کد با طول متغیر، از کدهای پیشوندی استفاده می‌شود.
- کدهای پیشوندی (prefix codes): هیچ کد کلمه‌ای پیشوند کد کلمه دیگر نیست.
- به عنوان مثال، اگر 0 کد کلمه کاراکتر a باشد آنگاه کد کلمه هیچ کاراکتر دیگری با 0 شروع نمی‌شود (به اسلایدهای قبل نگاه کنید).
- کدهای پیشوندی از این رو مناسب هستند که رمزگشایی فایل را آسان می‌کنند. در مثال قبل، رشته 001011101 به صورت یکتا به شکل 0.0.101.1101 تجزیه می‌شود که رمزگشایی شده‌ی آن به صورت aabe است.

کدهای هافمن: هزینه درخت

- فرض کنید که یک فایل متنی روی الفبای C داده شده است که در آن فراوانی تمام کاراکترها مثبت باشد.

- با داشتن یک درخت T مربوط به یک کد پیشوندی، می‌توان تعداد بیت‌های مورد نیاز برای کد کردن یک فایل را به صورت زیر محاسبه کرد:

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

که در آن c.freq نشان‌دهنده فراوانی کاراکتر c در فایل است و $d_T(c)$ نشان‌دهنده عمق برگ c در درخت (طول کد کلمه کاراکتر c).

کدهای هافمن: الگوریتم حریصانه

- هافمن یک الگوریتم حریصانه طراحی کرده است که یک کد پیشوندی بهینه با نام کد هافمن تولید می‌کند: در هر گام دو شیء با کمترین فراوانی را پیدا کن و آنها را با یکدیگر ادغام کن. این کار را تا زمانی ادامه بده که فقط یک شیء باقی بماند (ریشه درخت).
- از یک صف اولویت کمینه Q برای شناسایی دو شیء با کمترین فراوانی و ادغام آنها استفاده می‌شود
- نتیجه ادغام دو شیء یک شیء خواهد بود که فراوانی آن از مجموع فراوانی دو شیء ادغام شده بدست می‌آید

کدهای هافمن: الگوریتم حریصانه

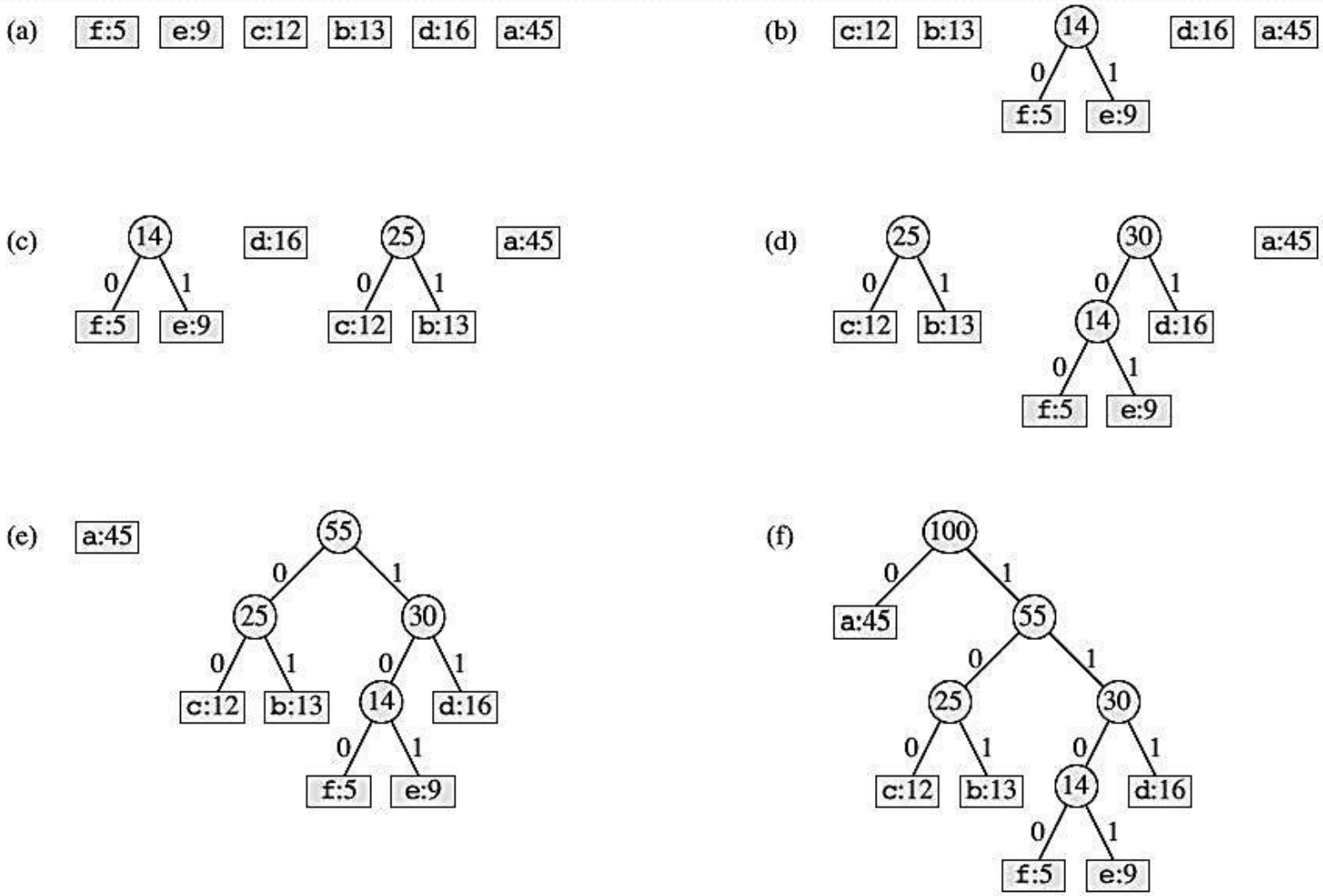
```
HUFFMAN(C)
n = |C|
Q = C
for i = 1 to n − 1
    allocate a new node z
    z.left = x = EXTRACT-MIN(Q)
    z.right = y = EXTRACT-MIN(Q)
    z.freq = x.freq + y.freq
    INSERT(Q, z)
return EXTRACT-MIN(Q)    // return the root of the tree
```

کدهای هافمن: تحلیل زمان اجرا

- زمان لازم برای ساختن هرم کمینه دودویی: O(n)
- هر یک از اعمال هرم‌ها (EXTRACT-MIN و INSERT) به زمان O(log *n*) نیاز دارد
- حلقه for دقیقاً *n-1* بار اجرا می‌شود
- بنابراین، کل زمان اجرای الگوریتم برابر است با O(*n* log *n*)

n: اندازه الفبا

کدهای هافمن: ساختن



کدهای هافمن: درستی الگوریتم

- **قضیه:** الگوریتم HUFFMAN یک کد پیشوندی بهینه تولید می‌کند.
- اثبات با استفاده از برهان خلف (برای جزئیات اثبات به کتاب CLRS مراجعه کنید)