اخطار : محتویات فایل‌ها تایید شده نیستند و مفاهیم و روابط ممکن است اشتباه باشند

1- درخت هافمن

1-1- فشرده سازی

در علم کامپیوتر و فناوری اطلاعات،مفهومی با عنوان فشرده سازی اطلاعات (data compression) وجود دارد که به مفهوم کدگذاری اطلاعات با استفاده از تعداد بیت های کمتر نسبت به نسخه اصلی است. متدهای فشرده سازی به دو دسته تقسیم می شوند:

- فشرده سازی با اتلاف (Lossy)
- فشرده سازی بدون اتلاف (Loss Less)

1-1-1- کد گذاری با اتلاف :

کدگذاری با اتلاف حجم بیت ها را با تشخیص اطلاعات غیر ضروری و حذف آنها،کاهش می دهد؛به طور کلی پروسه کاهش سایز یک داده به عنوان عمل فشرده سازی شناخته می شود.

حذف جزئیات غیر ضروری می تواند حجم اشغال شده به وسیله داده را کاهش دهد.این نوع فشرده سازی از نحوه دریافت اطلاعات توسط افراد الهام می گیرد.برای مثال یکی از چیز هایی که هرروزه با آنها در ارتباطیم،عکسها با فورمتِJPEG هستند.در این نوع فشرده سازی ،جزئیاتی از عکس که برای چشم ما نامحسوس ترند حذف می شود.در نتیجه از وضوح عکس کاسته می شود؛از طرفی این موضوع کمک بزرگی به کاهش سایز عکس میکند.

بسیاری از Data Type های مطرح و پر کاربرد این روزها،از نوع Lossy هستند.از فورمت های ویدیوئی و عکس ها گرفته تا موسیقی (نظیر ام پی تری).

1-1-2- کدگذاری بدون اتلاف :

در نظریه اطلاعات،همواره حجمی از بیت های بدست آمده حاوی Redundancy است.Redundancyعبارت است از تفاضل تعداد کل بیت های استفاده شده باری انتقال یک پیام (Data) و تعداد بیت های پیغام که از ابتدا مطلوب کاربر بوده است.

فشرده سازی ،در عمل Redundancy را که در اصل اطلاعات ناخواسته محسوب می شود،کاهش داده یا از بین می برد.نکته قابل توجه اینکه برخلاف حالت قبل ،این روش فشرده سازی قابل بازگشت بوده به طوری که اطلاعات اولیه قابل بازیابی به صورت کامل هستند.

در دنیای واقعی،مثال های زیادی از این روش در دسترس است؛برای مثال در فشرده سازی یک عکس،اگر تمام نواحی تصویر که رنگ یکسانی دارند،یعنی پیکسل های تصویر در یک ناحیه ،بدون تغییر بماند،بجای عبارت هایی نظیر :

"Red Pixel,RedPixel,Red…"

خواهیم نوشت: "278 Red Pixels".

روش ها و مثال های دیگری از این دست وجود دارند.یکی از آنها روش کدگذاری هافمن است.

2-1- ایده هافمن

در کامپیوترها،ما تنها با اعداد در مبنای 2 سروکار داریم؛مشخص است که برای استفاده از یک عدد،کافی است آن را به مبنای 2 ببریم.اما در مورد حروف چطور؟کلمات و یا دستورات چگونه شکل می گیرند؟

برای حل این مشکل،از کد گذاری استفاده می کنیم (Encoding).به این شکل که به هر حرف؛ یک عدد نسبت می دهیم.برای اینکار ،2 روش وجود دارد.

- کدگذاری با اندازه ثابت(fix length encoding) : در روش اول که کد گذاری با طول ثابت نام دارد،تعداد بیتهای مشخصی را در نظر گرفته و هر کاراکتری را به یک عدد نسبت میدهیم.یعنی طول هر 2 کاراکتر دلخواه همان مقدار ثابت (مثلا 8 بیت) است.واضح است که تعداد کاراکترهایی که این روش پوشش میدهد، $2^8 = 256$ است. در کدگذاری ASCII[1]،طول بیت های استفاده شده برای هر کاراکتر مشخص و ثابت است.

نکته خیلی مهمی در اینجا وجود دارد که باید به آن توجه کرد: یک متن را در نظر بگیرید.در یک فایل که حاوی تعداد زیادی کلمه است،احتمال وجود حرف های صدادار و حرف های خاصِ پر کاربرد،خیلی بیشتر از تعداد حرف های دیگر است.مثلا در زبان فارسی ، حروفی مانند 'ا' ، 'ی' ، 'ه' و... پرکاربردتر از حرفهایی مانند 'ظ' است.[2]

با در نظر گرفتن این موضوع،به روش دوم و ایده کدگذاری هافمن می رسیم.روش دوم،کدگذاری با طول متغیر است.

تعریف 1.فرکانس : عبارت است از تعداد تکرار یک کاراکتر مشخص در یک رشته از اعداد (پاراگراف،متن و...).

ایده فشرده سازی هافمن،با توجه به قانون پارتو،کدگذاری متغیر است.یعنی،به کاراکتر هایی که بیشترین فرکانس را دارند،کدهایی با کمترین طول ممکن نسبت دهیم و به کاراکترهای کم تکرار تر،طول بیشتر.

تعریف 2.خاصیت پیشوندی : در کدهایی که برای هر کاراکتر انتخاب می کنیم،باید دقت کنیم یک کد ،پیشوند کد دیگر نباشد.مثلا نمی توانیم برای کاراکتری ،کد 100 را انتخاب کنیم و برای کاراکتر دیگری کد 1000.

---

[1] American Standard Code for Information Interchange

[2] Pareto Principle. که همچنین به نام های **قانون 80-20**،**قانون افراد اندک اساسی** و **اصل ثُنُکی فاکتور** شناخته می شود،بیان می کند که 80درصد رخداد ها از 20 درصد دلایل بوجود می آید.مثالهای زیادی از این قانون در اطراف ما وجود دارد.مثلا "80 درصد خرید مربوط به 20 درصد مشتریان است." ویا "80 درصد ثروت جهان در انحصار 20 درصد جمعیت آن است.".در مثالی که زده شد، میتوان گفت حدود 80 درصد متن از 20 درصد کاراکترهای موجود در زبان تشکیل شده است.

اخطار : محتویات فایل‌ها تایید شده نیستند و مفاهیم و روابط ممکن است اشتباه باشند

- اگر کد هایی که انتخاب میکنیم خاصیت پیوندی نداشته باشند در این صورت اگر کد یک کاراکتر 10001 و کد یک کاراکتر دیگر 1000 باشد وقتی به عدد 10001 برخورد میکنیم نمی توانیم تشخیص دهیم که این عدد حاصل 1+1000 است یا که کل آن یک کاراکتر است .

به جدول زیر دقت کنید:

Char	Fixed-length	Variable-length	Frequency
A	000	0	50
B	001	110	10
C	010	10	20
D	011	1110	10
E	100	11110	4
F	101	111110	4
G	110	1111110	1
H	111	1111111	1

جدول 1

کد های انتخاب شده برای کاراکترهای جدول 1،از نوع متغیر،دارای خاصیت پیشوندی هستند.حال،با دانستن ایده ی کدگذاری،نیاز به روشی برای بدست آوردن کدها داریم.

1-3- درخت هافمن

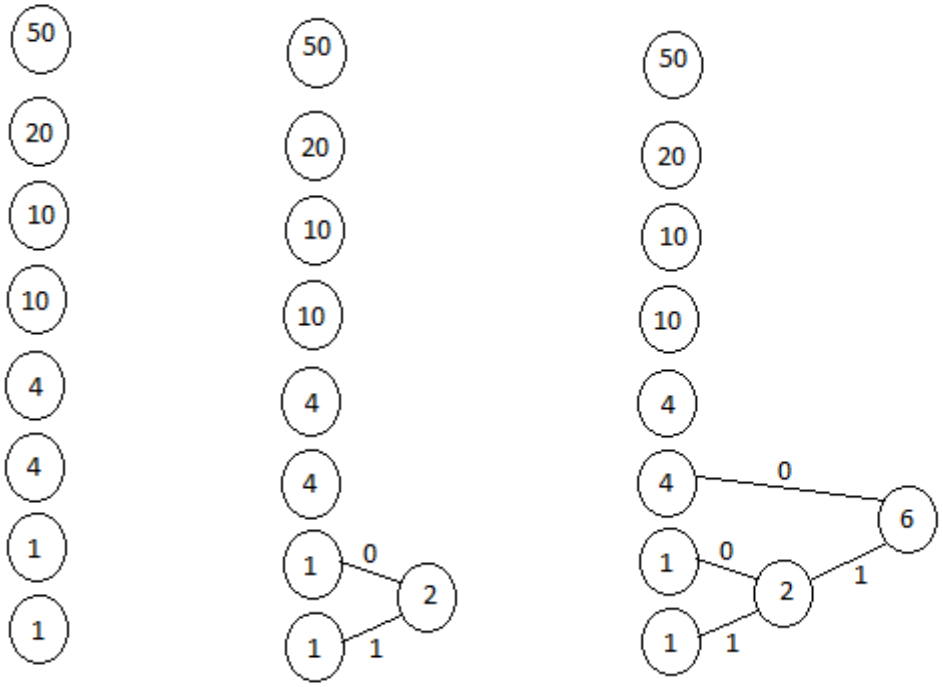
تعریف : درخت هافمن یک درخت دودویی است که برای کد گذاری با طول متغیر در فشرده سازی،به کار می رود.

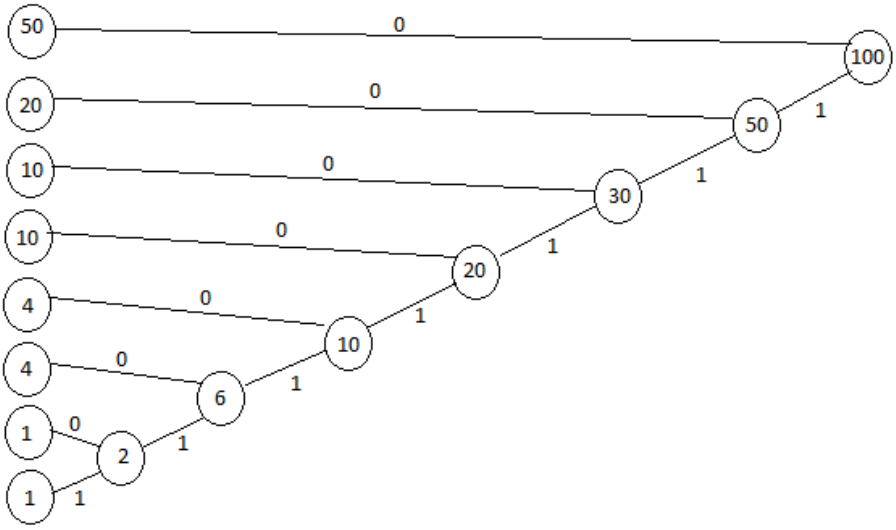
بر خلاف معمول که برای ساخت درخت از ریشه شروع می کردیم،در این جا ابتدا برگها را قرار می دهیم.برگهای درخت هافمن،در واقع همان کاراکترها و فرکانس تکرارشان است.بهتر است کاراکتر ها را به ترتیب نزولی یا صعودیِ فرکانسشان قرار دهیم. سپس،هر بار ،فرکانس دو کاراکتری را که جمعشان از جمع فرکانس هر دو تای دیگری کمتر بود،به یک گره ی جدید تبدیل میکنیم به صورتی که عدد گره ی مورد نظر،برابر مجموع دو فرزندش باشد.چون درخت دودویی است (یعنی هر گره حداکثر دو فرزند دارد) ،به هر فرزندِ هر گره یک عدد 0 یا 1 نسبت می دهیم.این کار را تا زمانی که به آخرین گره برسیم (ریشه درخت) متوقف نمیکنیم.به عبارت بهتر،فرکانس ریشه درخت،برابر تعداد کل کاراکترهایی است که در کل داریم. فرض کنید فایلی داریم که تکرار حروف در آن به شکل زیر است :

Char	**Freq**
A	**50**
B	**20**
C	**10**
D	**10**
E	**4**
F	**4**
G	**1**
H	**1**

\\

اخطار : محتویات فایل‌ها تایید شده نیستند و
مفاهیم و روابط ممکن است اشتباه باشند





شکل1

کدهایی که بر روی یال های درخت بدست آمده نوشته شده اند،همان کد های هافمن هستند.برای خواندن کدها،کافی است از ریشه شروع کرده،مسیری را انتخاب کنیم که به راسی که حاوی کاراکتر مورد نظر است ،برسد.بیت های خوانده شده از سمت چپ به راست همان کد هافمن کاراکتر است. جواب نهایی به شکل زیر است :

Char	**Code**
A	**0**
B	**10**
C	**110**
D	**1110**
E	**11110**
F	**111110**
G	**1111110**
H	**1111111**

1-3-1- نکات :

1.ENCODING به روش هافمن برای هر فایل با فایل های دیگر متفاوت است و به فرکانس هر کاراکتر در کل فایل بستگی دارد.

2.الگوریتم هافمن در جایی که توضیع فرکانس نسبتا یکنواخت باشد،خوب عمل نمی کند. (درخت به درخت دودویی متقارن نزدیک شده ودر نتیجه طول کاراکترها به یک مقدار ثابت میل می کند.)- شکل 5

در درخت هافمن،کاراکترها با تعداد تکرار بیشتر،در عمق کمتری قرار دارند.

3.برای محاسبه ی حجم یک فایل،کافیست فرکانس کاراکتر های آن را در طول هر کاراکتر ضرب کنیم و نتایج را بهم جمع؛مقدار به دست آمده،حجم فایل بر حسب BIT است. برای مثال، در جدول بالا به ازای کد های با طول ثابت،حجم فایل 300 = 100*3 است.در حالی که اگر کد هافمن را استفاده کنیم،حجم فایل از راه زیر به دست می آید :

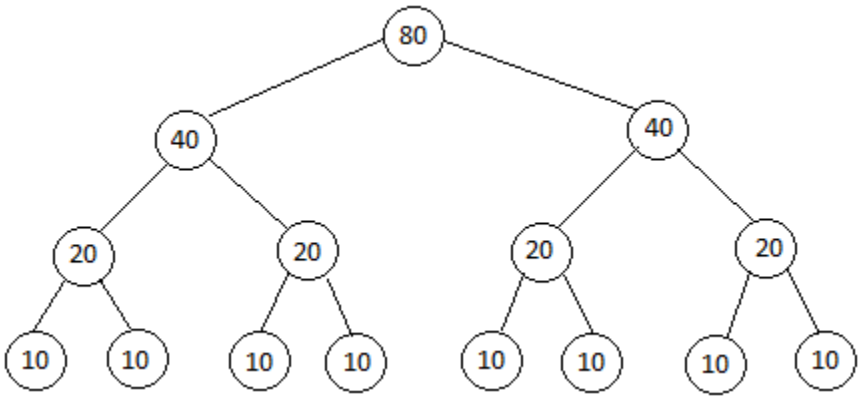
$$50 * 1 + 20 * 2 + 10 * 3 + 10 * 4 + 4 * 5 + 4 * 6 + 1 * 7 + 1 * 7 = 218\ bit$$

اخطار : محتویات فایل‌ها تایید شده نیستند و مفاهیم و روابط ممکن است اشتباه باشند

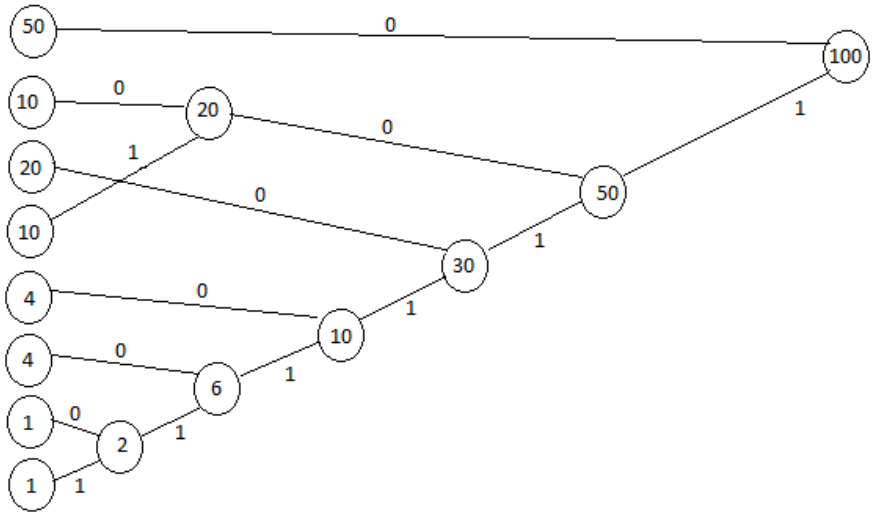
4.پیشتر گفته شد ترتیب چیدمان برگ ها بهتر است صعودی یا نزولی باشد .علت این امر فقط یک دست شدن درخت است.وگرنه از چیدمان های مختلف برگها ،ممکن است کد های مختلفی بدست آید اما در نهایت،حجم فایل هیچ گاه از 218 کمتر و یا بیشتر نمی شود. به عبارت دیگر،با اینکه درخت هافمن برای یک مسئله ی خاص لزوما یکتا نیست،اما در مسائل فشرده سازی ،مقدار حجم نهایی همواره یکتاست. – شکل 6

5.یکی از ایرادات درخت هافمن اینست که فایل هایی که با این روش فشرده شده اند باید یک جدول که در آن کد هر کاراکتر وجود دارد در کنار خود ذخیره کنند که این کار خود به حجم فایل می افزاید .

6. یک ایراد دیگر هافمن این است که باید متن دو بار خوانده شود یعنی یک بار باید فایل خوانده شود و از یک جدول که کد هر کاراکتر را ذخیره میکند فایل را بازنویسی کرد بعد میتوان از آن فایل استفاده کرد .



شکل2



شکل3