دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

بررسی الگوریتم‌های فشرده‌سازی و کاربردهای آن‌ها

نگارنده:

محمد مهدی عرفانیان

درس:

ارائه مطالب علمی و فنی

مدرس:

دکتر همت‌یار

۲۰ دی ۱۳۹۸

چکیده

الگوریتم‌های فشرده‌سازی در طی عمر علوم کامپیوتر یکی از مهم‌ترین موضوعات مورد بحث آن بوده اند. در ابتدا با توجه به کندی و ضعف سخت‌افزار و نیاز به انتقال حداقلی داده این الگوریتم‌ها مورد توجه قرار گرفتند؛ سپس با پیشرفت اینترنت و نیاز به انتقال سریع داده، این الگوریتم‌ها دوباره به یکی از مهم‌ترین زمینه‌های تحقیق در علوم کامپیوتر تبدیل شدند، در این زمان بسیاری از الگوریتم‌های مهم فشرده‌سازی داده کشف شدند. فرمت‌هایی هم‌چون ۳mp یا jpeg ـ که از شناخته‌ترین فرمت‌های فشرده‌سازی حال حاضر هستند ـ ماحصل تحقیقات همین دوران اند. الگوریتم‌های فشرده‌سازی نه تنها راه خود را به مدیا باز کردند بلکه در پردازش سیگنال نیز تاثیر عمده‌ای داشتند، تمامی ارتباطات رادیویی با استفاده از این الگوریتم‌ها سعی در بالا بردن امنیت و کاهش هزینه انتقال داده را دارند، در رمزنگاری نیز این الگوریتم‌ها راه‌گشایی می‌کنند و بسیاری از الگوریتم‌های رمزنگاری و درهم‌سازی خود نوعی الگوریتم فشرده‌سازی محسوب می‌شوند.

در این مستند سعی شده است تا مقدمه‌ای برای ورود به دنیای شگفت‌انگیز الگوریتم‌های فشرده‌سازی نوشته شود و خوانندگان را با تعاریف بنیادین و کاربردهای اصلی این الگوریتم‌ها آشنا کند، این مستند مقدمتاً دربارهٔ تعاریف اصلی و انواع الگوریتم‌های فشرده‌سازی و کاربردهای اصلی آن‌ها در دنیای امروز بحث می‌کند و پس از آن به بررسی جزیی‌تر دو فرمت اصلی فشرده‌سازی (۳mp و jpeg) می‌پردازد. امید است تا مطالب این مستند مورد توجه خوانندگان عزیز قرار گیرد.

به عنوان موخره چکیده باید نوشت که این مستند در راستای انجام پروژهٔ مستندسازی درس ارائه مطالب علمی و فنی تهیه شده است و برای سهولت کار استاد محترم درس برای تحصیل اطمینان از درستی مستندسازی و همچنین استفاده دانش‌جویان علاقه‌مند، سیر پیشرفت مستند به همراه کدهای LaTeX در *Github* قرار گرفته اند، لازم به ذکر است که این مستند به صورت متن‌باز ارائه شده و استفاده از آن بدون ذکر منبع برای همگان آزاد است.

کلمات کلیدی: الگوریتم‌های فشرده‌سازی، فشرده‌سازی تصویر، فشرده‌سازی صوت، ۳MP، JPEG

فهرست مطالب

فهرست تصاویر

فهرست جداول

فصل ۱

مقدمه

توضیحی اولیه مبنی بر تعریف کلی فشرده‌سازی، انواع الگوریتم‌ها و کاربردها

۱.۱ تعریف

الگوریتم‌های فشرده‌سازی، الگوریتم‌هایی هستند که می‌توان با استفاده از آن‌ها داده‌ها را طوری رمزنگاری کرد که در تعداد کمتری بیت نسبت به آرایش اولیه قابل ارائه باشند. [۱] برای مثال می‌دانیم که برای ذخیرهٔ هر بیت اسکی [۱] هشت بیت فضا لازم است، می‌توان با استفاده از نگاشتی متشکل از حروف استفاده‌شده در یک متن تعداد بیت‌های مورد نیاز برای نشان دادن هر حرف استفاده‌شده در متن را کاهش داد. با استفاده از این تکنیک در حقیقت متن را در قالب جدیدی فشرده کرده‌ایم.

۲.۱ انواع الگوریتم‌های فشرده‌سازی

در یک دسته‌بندی الگوریتم‌های فشرده‌سازی را به دو نوع زیر افراز می‌کنند.

- بدون هدررفت داده [۲]
- همراه هدررفت داده [۳]

۱.۲.۱ الگوریتم‌های بدون هدررفت داده

در این سری الگوریتم‌ها دادهٔ ورودی بدون هیچ‌گونه هدررفتی از دادهٔ خروجی قابل بازیابی‌ست، الگوریتم‌های این دسته با استفاده از افزونگی آماری تلاش می‌کنند تا نحوهٔ نمایش داده را در نگاشتی به نحوهٔ نمایش دیگری که به فضای کمتری نیاز دارد تبدیل کنند. این الگوریتم‌ها در مواقعی که ثابت ماندن داده در طی فشرده‌سازی الزامی‌ست استفاده می‌شوند، همچنین معمولا برای بازیابی اطلاعات فشرده‌شده نیاز به داده‌هایی خارجی است که با کمک آن عمل بازیابی انجام می‌گیرد، از این رو می‌توان از این نوع الگوریتم‌ها در رمزنگاری نیز استفاده کرد. یکی از کاربردهای اصلی این الگوریتم‌ها در فشرده‌سازی متون است که اشتباه شدن حتی یک حر ف می‌تواند باعث بدخوانی و بدفهمی متن اصلی گردد. الگوریتم‌های مشهور فشرده‌سازی بدون هدررفت داده به شرح زیر اند[۲].

- Run-Length Enconding (RLE)
- Lempel-Ziv (LZ)
- Huffman Encoding
- Burrows Wheeler Transform

البته لازم به ذکر است که در عمل از مجموعه‌ای از الگوریتم‌های فوق برای رسیدن به درصد مطلوب فشرده‌سازی استفاده می‌شود.

نمونه‌های الگوریتم‌های بدون هدررفت داده

در عمل از الگوریتم‌های بدون هدررفت داده در مواقعی که پایداری داده‌های ذخیره‌شده حیاتی‌ست یا این که فایل در آینده به تعداد زیادی بار فشرده و گسترده می‌شود و از دست دادن قسمتی از داده در هربار

[1] ASCII
[2] Lossless
[3] Lossy

شکل ۱.۱: شکل نمونه برای تست میزان کمپرس در فرمت PNG [۳]

جدول ۱.۱: حجم فایل نمونه در فرمت PNG

Format	Size
BMP	۷.۷ مگابایت
PNG	۹۸ کیلوبایت

فشرده‌سازی منجر به اختلاف فاحش نهایی خواهد شد استفاده می‌شود. احتمالا می‌توان مشهورترین فرمت فشرده‌سازی بدون هدررفت داده را فرمت ZIP نامید، این فرمت که برای فشرده‌سازی فایل‌های مختلف استفاده می‌شود به علت گستردگی استفاده در درون خود از الگوریتم‌های بدون هدررفت داده استفاده می‌کند، به جز فرمت ZIP فرمت‌های دیگری که استفاده‌های تخصصی‌تری دارند نیز از این الگوریتم‌ها استفاده می‌کنند که در ادامه به دو مورد مهم ایشان اشاره می‌کنیم.

- PNG

در طراحی فرمت PNG برای فشرده‌سازی تصاویر از الگوریتم Lempel-Ziv-Welch (LZW) که الگوریتی Lossless است استفاده شده. در شکل ۱.۱ و جدول ۱.۱ یک نمونه عکس در حالت فشرده‌نشده و فشرده‌شده با فرمت PNG و مقدار حجم آن در حالت‌های مختلف آورده شده است.

- FLAC [4]

کدک صوتی بدون هدررفت دادهٔ آزاد [۴] یا FLAC الگوریتمی که با استفاده از اطلاعات ذاتی داده‌های صوتی به فشرده‌سازی آن‌ها می‌پردازد، نرخ فشرده‌سازی این الگوریتم با توجه به سطح فشرده‌سازی سازی آن معمولا بین ۴۰ تا ۶۰ درصد می‌باشد اما در حالت بیشینه ممکن است تا ۸۰ درصد هم برسد.

۲.۲.۱ الگوریتم‌های با هدررفت داده

۲معمولا در فشرده‌سازی با هدررفت داده از تبدیل فضایی [۵] استفاده می‌شود که داده‌ها را از فضای حقیقی به فضایی دیگر(معمولا فرکانس) می‌برد و در آنجا از قسمت‌هایی از داده که تاثیرگذاری و حس‌پذیری کمتری

[۴] Free Lossless Audio Codec
[۵] Transformation

نسبت به دیگران دارند صرف نظر می‌شود، سپس وارون تبدیل اجرا شده و داده‌های کوچک‌شدهٔ جدید این بار با الگوریتم‌های بدون هدررفت داده فشرده می‌شوند. یکی از مشهورترین تبدیل‌های فضایی ها تبدیل کسینوسی گسسته [۶] است که در فصل‌های بعدی این مستند بیشتر با آن آشنا خواهیم شد [۶].

نمونه‌های الگوریتم‌های با هدررفت داده

الگوریتم‌های با هدررفت داده با همه‌گیر شدن اینترنت و اشتراک‌گذاری بیشتر مدیا در فضای اینترنت بسیار گسترده شدند، از این رو اکثر این الگوریتم‌ها برای فشرده‌سازی فایل‌های صوتی_تصویری یا به اصطلاح مدیا [۷] استفاده می‌شوند.

- JPEG
- MP3
- MP4
- H.26x

به عنوان نمونهٔ برای الگوریتم باهدررفت داده حالت فشرده‌شدهٔ عکس ۱.۱ با فرمت JPEG با کیفیت‌های مختلف در جدول ۲.۱ آورده شده است.

جدول ۲.۱: حجم فایل نمونه در فرمت JPG

Format	Size	Quality(%)
BMP	۷.۷ مگابایت	۱۰۰
PNG	۹۸ کیلوبایت	۱۰۰
JPG	۸.۹۶ کیلوبایت	۹۰
JPG	۹.۶۲ کیلوبایت	۵۰
JPG	۴۹ کیلوبایت	۲۰

۳.۱ کاربردهای دیگر فشرده‌سازی

الگوریتم‌های فشرده‌سازی داده علاوه بر این که در کاربرد اصلی خود یعنی فشرده کردن فایل‌ها بسیار استفاده می‌شوند اما کاربردهایی دیگری هم در دنیای کامپیوتر دارند، به عنوان مثال یکی از حیاتی‌ترین عناصر پردازش سیگنال [۸] را فشرده‌سازی سیگنال تشکیل می‌دهد. در ارتباطات رادیویی برای بالا بردن امنیت و کم کردن هزینهٔ انتقال از فشرده‌سازی داده استفاده می‌شود.

از دیگر کاربردهای جدید فشرده‌سازی داده ژنتیک است، زشته‌های DNA را معمولا می‌توان به صورت یک رشته متنی نشان داد و با الگوریتم‌های فعلی فشرده‌سازی متن فشرده کرد اما به دلیل این که این رشته‌ها طول و تعداد زیادی دارند بهتر است از روش‌های دیگری برای بازیابی و فشرده‌سازی سریع آن‌ها استفاده کینم،

[۶] Transform Cosine Discrete (DCT)
[۷] Media
[۸] Signal Processing

با رشد سریع ژنتیک و بایوانفورماتیک در جهان و نیاز مبرم به ذخیره‌سازی رشته‌های DNA الگوریتم‌های مختص فشرده‌سازی DNA مختلفی معرفی شده اند[۱۲].

فصل ۲

بررسی نحوهٔ فشرده‌سازی در فرمت JPEG

توضیح نحوهٔ کار فشرده‌سازی در فرمت JPEG، مقدمه‌ای بر DCT و توضیح کلی Huffman encoding

۱.۲ تبدیل کسینوسی گسسته

۱.۱.۲ تعریف

تبدیل کسینوسی گسسته [۱] یا مختصرا DCT ، دنباله‌ای محدود از اعداد (داده ها) را به صورت مجموع توابع کسینوسی با فرکانس های متفاوت نمایش می‌دهد. تبدیل کسینوسی گسسته، شباهت بسیاری به تبدیل فوریه گسسته (DFT) دارد، با این تفاوت که حاصل تبدیل فقط مقادیر حقیقی دارد (بر خلاف تبدیل فوریه که منجر به مقادیر مختلط می شود).

به صورت علمی‌تر می‌توان نوشت که DCT تابعی معکوس‌پذیر و خطی از R^N به R^N است.

فرمول کلی DCT برای فضای یک‌بعدی به شکل زیر است [۶].

$$X_k = \sum_{n=0}^{N-1} x_n \cos[\pi/n(n+1/2)k], k = 0, ..., N-1 \qquad (۱.۲)$$

۲.۲ تبدیل دنباله‌ای از پیکسل‌ها به تابع

هر تصویر در حقیقت به صورت چهار ماتریس دوبعدی ذخیره می‌شود که هر ماتریس مقدار رنگی پیکسل را برای آن کانال رنگی (RBG و α) نشان می‌دهد، برای فشرده کردن یک عکس ماتریس‌های کانال‌های رنگی مختلف را جداگانه فشرده می‌کنیم، کانال رنگی R را در نظر بگیرید، در این حالت یک ماتریس $n*m$ داریم که هر خانهٔ آن عددی بین ۰ تا ۲۵۵ را نشان می‌دهد، برای سادگی یک سطر از این ماتریس را در نظر بگیرید، این سطر معادل با یک آرایه از اعداد می‌باشد، در صورتی که تمامی اعداد را از دامنه ۰ تا ۲۵۵ به دامنه ۱۲۸ـ تا ۱۲۷ ببریم می‌توانیم این دنباله را با مجموعه‌ای از توابع کسینوسی بازتولید کنیم، این ضرایب همان ضرایبی‌ست که با استفاده از DCT می‌خواهیم به دست بیاوریم.

شکل ۱.۲ نمونه‌ای از توصیف عکس با توابع کسینوسی را نشان می‌دهد.

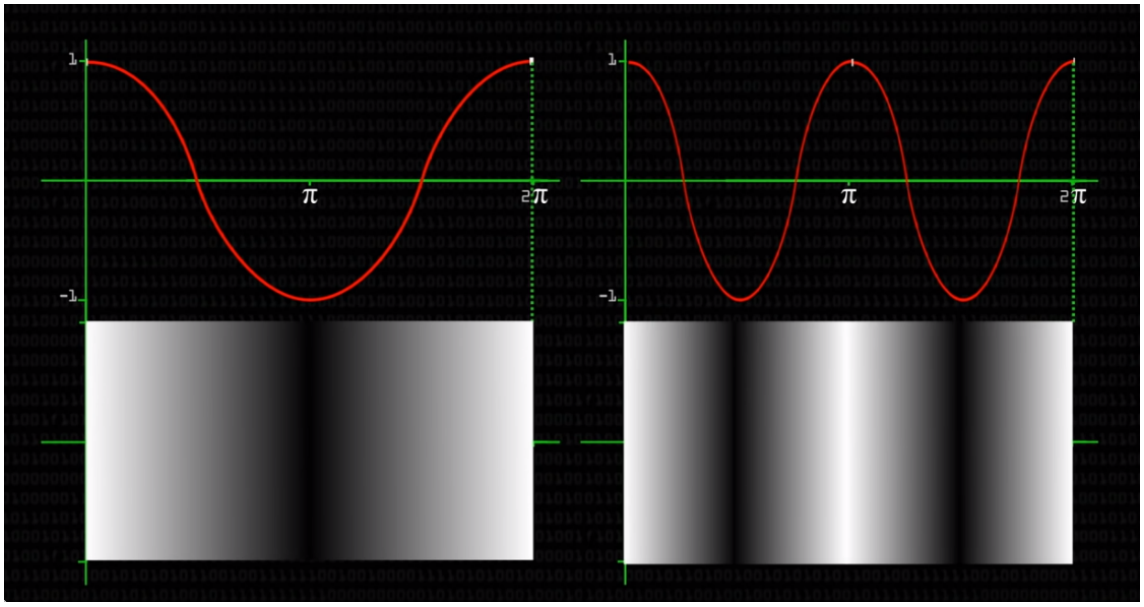
حال اگر دو تابع نشان‌داده شده در شکل ۱.۲ را با هم جمع کرده و میانگین بگیریم، به نمایش فرکانس دیگری می‌رسیم که در شکل ۲.۲ نشان داده شده است. در صورتی که به شکل‌های متفاوت و با ضرابت متفاوت توابع مختلف کسینوسی را با هم جمع کنیم می‌توانیم هر فرکانسی را بسازیم. این کاری‌ست که در عمل الگوریتم DCT برای ما انجام می‌دهد.

۳.۲ نحوهٔ کار JPEG

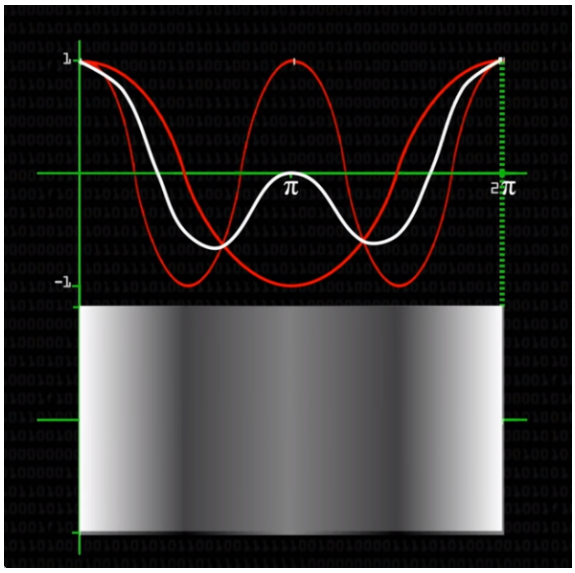
برای فشرده‌سازی یک تصویر در JPEG مراحل زیر انجام می‌شود.

- تبدیل تصویر به بلوک‌های کوچک
- تبدیل هر بلوک به مجموعه‌ای از توابع کسینوسی استاندارد با DCT
- تنظیم ضرایب متناسب برای بلوک‌های DCT
- فشرده کردن ضرایب با استفاده از Huffman

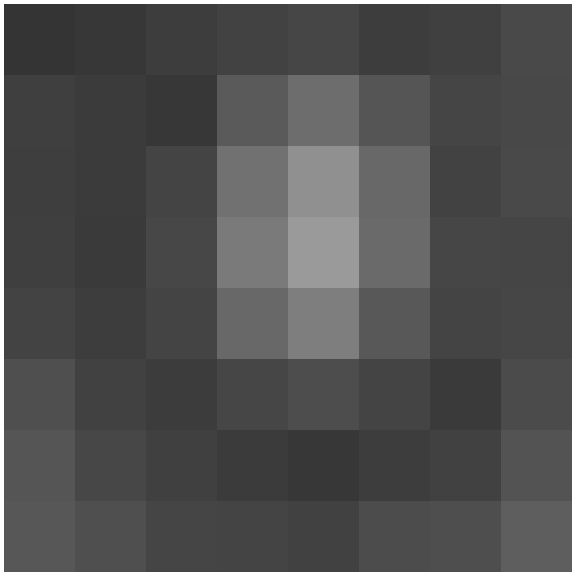
[۱] Discrete cosine transform



شکل ۱.۲: توصیف فرکانسی توابع $\cos(x), \cos(2x)$ [۸]



شکل ۲.۲: توصیف فرکانسی تابع $\cos(x) + \cos(2x)/2$ [۸]



شکل ۳.۲: نمونهٔ تصویر برای فشرده‌سازی [۷]

۴.۲ یک مثال از فشرده‌سازی JPEG

برای درک بهتر مراحل گفته شده تلاش می‌کنیم تا ماتریس ۲.۲ ـ که تصویر کانال خاکستری آن را در شکل ۳.۲ مشاهده می‌شود ـ را با الگوریتم‌های گفته‌شده فشرده‌سازی کنیم.

$$M = \begin{bmatrix} ۵۲ & ۵۵ & ۶۱ & ۶۶ & ۷۰ & ۶۱ & ۶۴ & ۷۳ \\ ۶۳ & ۵۹ & ۵۵ & ۹۰ & ۱۰۹ & ۸۵ & ۶۹ & ۷۲ \\ ۶۲ & ۵۹ & ۶۸ & ۱۱۳ & ۱۴۴ & ۱۰۴ & ۶۶ & ۷۳ \\ ۶۳ & ۵۸ & ۷۱ & ۱۲۲ & ۱۵۴ & ۱۰۶ & ۷۰ & ۶۰ \\ ۶۷ & ۶۱ & ۶۸ & ۱۰۴ & ۱۲۶ & ۸۸ & ۶۸ & ۷۰ \\ ۷۹ & ۶۵ & ۶۰ & ۷۰ & ۷۷ & ۶۸ & ۵۸ & ۷۵ \\ ۸۵ & ۷۱ & ۶۴ & ۵۹ & ۵۵ & ۶۱ & ۶۵ & ۸۳ \\ ۸۷ & ۷۹ & ۶۹ & ۶۸ & ۶۵ & ۷۶ & ۷۸ & ۹۴ \end{bmatrix} \quad (۲.۲)$$

هر عنصر ماتریس ۲.۲ عددی بین ۰ تا ۲۵۵ دارد اما از آنجایی که تابع کسینوسی مقادیر بین ۱ـ تا ۱ می‌گیرد نیازمندیم تا با کم کردن هر عنصر این ماتریس از ۱۲۸ هر عنصر این ماتریس را به عددی بین ۱۲۷ـ تا ۱۲۸

تبدیل کنیم. ماتریس ۳.۲ تبدیل‌شده است.

$$M' = \begin{bmatrix} -۷۶ & -۷۳ & -۶۷ & -۶۲ & -۵۸ & -۶۷ & -۶۴ & -۵۵ \\ -۶۵ & -۶۹ & -۷۳ & -۳۸ & -۱۹ & -۴۳ & -۵۹ & -۵۶ \\ -۶۶ & -۶۹ & -۶۰ & -۱۵ & ۱۶ & -۲۴ & -۶۲ & -۵۵ \\ -۶۵ & -۷۰ & -۵۷ & -۶ & ۲۶ & -۲۲ & -۵۸ & -۵۹ \\ -۶۱ & -۶۷ & -۶۰ & -۲۴ & -۲ & -۴۰ & -۶۰ & -۵۸ \\ -۴۹ & -۶۳ & -۶۸ & -۵۸ & -۵۱ & -۶۰ & -۷۰ & -۵۳ \\ -۴۳ & -۵۷ & -۶۴ & -۶۹ & -۷۳ & -۶۷ & -۶۳ & -۴۵ \\ -۴۱ & -۴۹ & -۵۹ & -۶۰ & -۶۳ & -۵۲ & -۵۰ & -۳۴ \end{bmatrix} \quad (۳.۲)$$

در گام بعدی مقادیر ماتریس تیدیل‌شده را با استفاده از الگوریتم DCT به فضای فرکانس می‌بریم هر مقدار از این ماتریس جدید برابر با ضریب فرکانس معادل در ماتریس استاندارد است. ماتریس ۴.۲ ماتریس تبدیل‌شدهٔ ما خواهد بود.

$$F = \begin{bmatrix} -۴۱۵/۳۸ & -۳۰/۱۹ & -۶۱/۲۰ & ۲۷/۲۴ & ۵۶/۱۲ & -۲۰/۱۰ & -۲/۳۹ & ۰/۴۶ \\ ۴/۴۷ & -۲۱/۲۸ & -۶۰/۷۶ & ۱۰/۲۵ & ۱۳/۱۵ & -۷/۰۹ & -۸/۵۴ & ۴/۸۸ \\ -۴۶/۸۳ & ۷/۳۷ & ۷۷/۱۳ & -۲۴/۵۶ & -۲۸/۹۱ & ۹/۹۳ & ۵/۴۲ & -۵/۶۵ \\ -۴۸/۵۳ & ۱۲/۰۷ & ۳۴/۱۰ & -۱۴/۷۶ & -۱۰/۲۴ & ۶/۳۰ & ۱/۸۳ & ۱/۹۵ \\ ۱۲/۱۲ & -۶/۵۵ & -۱۳/۲۰ & -۳/۹۵ & -۱/۸۷ & ۱/۷۵ & -۲/۷۹ & ۳/۱۴ \\ -۷/۷۳ & ۲/۹۱ & ۲/۳۸ & -۵/۹۴ & -۲/۳۸ & ۰/۹۴ & ۴/۳۰ & ۱/۸۵ \\ -۱/۰۳ & ۰/۱۸ & ۰/۴۲ & -۲/۴۲ & -۰/۸۸ & -۳/۰۲ & ۴/۱۲ & -۰/۶۶ \\ -۰/۱۷ & ۰/۱۴ & -۱/۰۷ & -۴/۱۹ & -۱/۱۷ & -۰/۱۰ & ۰/۵۰ & ۱/۶۸ \end{bmatrix} \quad (۴.۲)$$

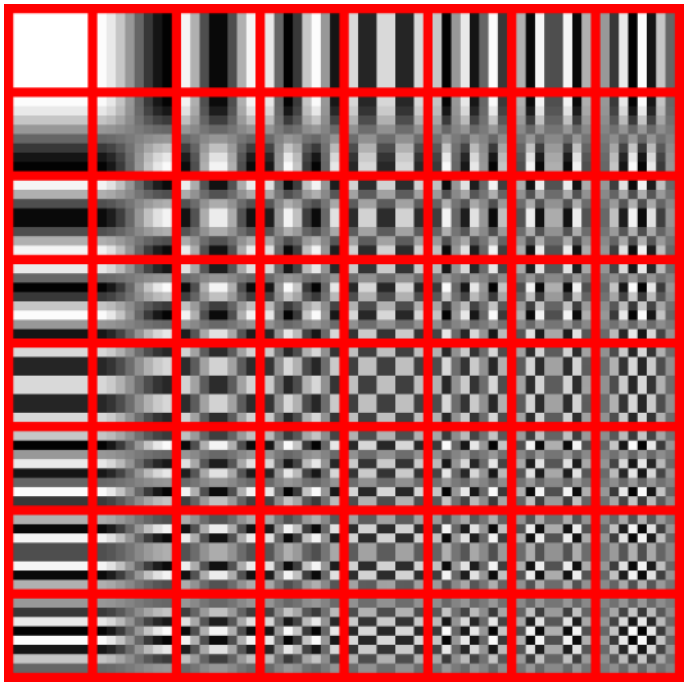
هر کدام از عناصر ماتریس ۴.۲ مقدار ضریب تاثیر فرکانس نشان‌داده‌شده در شکل ۴.۲ می‌باشند.

تا به این لحظه هیچ مقداری از داده را برای فشره‌سازی از دست نداده‌ایم اما در این گام می‌خواهیم قسمت‌هایی از تصویر که برای چشم انسان قابل تشخیص نیستند را حذف کنیم تا مقدار دادهٔ کمتری را ذخیره کنیم. باید توجه داشت که چشم انسان معمولا قادر به تشخیص و تمییز تصاویر با فرکانس‌های بالا در تصاویر نیست و همان‌طور که در ماتریس ۴.۲ هم مشاهده می‌شود ضریب این فرکانس‌ها نسبت به فرکانس‌های پایین بسیار کم است، می‌توانید به مقدار بسیار بزرگ ۴۱۵ برای فرکانس بسیار پایین در راستای x و y در ماتریس توجه کنید تا این نکته روشن شود. حال باید ضرایب فعلی ماتریس را با تقریبی گرد کنیم و تاثیر فرکانس‌های بالا را کمتر از تاثیر فرکانس‌های پایین قرار دهیم، از این رو از جدولی به نام Quantization Table استفاده می‌کنیم.

۱.۴.۲ جدول کوانتیزه کردن

جدول کوانتیزه کردن [۲] جدولی است که در آن مقدار تاثیر هر فرکانس برای هر ردهٔ فشرده‌سازی برای عکس‌های JPEG به صورت جهانی استانداردسازی و تعیین شده‌است. برای مثال برای نرخ فشرده‌سازی ۵۰ درصد جدول کوانتیزه کردن در ماتریس ۵.۲ آمده است. این ماتریس برای کیفیت‌های مختلف فشرده‌سازی

[۲] معادل فارسی برای Quantization یافت نشد.



شکل ۴.۲: ماتریس فرکانس‌های استاندارد [۹]

به صورت استانداردشده وجود دارد.

$$Q = \begin{bmatrix} ۱۶ & ۱۱ & ۱۰ & ۱۶ & ۲۴ & ۴۰ & ۵۱ & ۶۱ \\ ۱۲ & ۱۲ & ۱۴ & ۱۹ & ۲۶ & ۵۸ & ۶۰ & ۵۵ \\ ۱۴ & ۱۳ & ۱۶ & ۲۴ & ۴۰ & ۵۷ & ۶۹ & ۵۶ \\ ۱۴ & ۱۷ & ۲۲ & ۲۹ & ۵۱ & ۸۷ & ۸۰ & ۶۲ \\ ۱۸ & ۲۲ & ۳۷ & ۵۶ & ۶۸ & ۱۰۹ & ۱۰۳ & ۷۷ \\ ۲۴ & ۳۵ & ۵۵ & ۶۴ & ۸۱ & ۱۰۴ & ۱۱۳ & ۹۲ \\ ۴۹ & ۶۴ & ۷۸ & ۸۷ & ۱۰۳ & ۱۲۱ & ۱۲۰ & ۱۰۱ \\ ۷۲ & ۹۲ & ۹۵ & ۹۸ & ۱۱۲ & ۱۰۰ & ۱۰۳ & ۹۹ \end{bmatrix} \tag{۵.۲}$$

حال برای آخرین گام باید مقادیر ماتریس ۴.۲ را بر مقادیر ماتریس ۵.۲ تقسیم کنیم و عدد به دست آمده را گرد کنیم. ماتریس نهایی برای فشرده‌سازی به شکل زیر است.

$$R = \begin{bmatrix} -۲۶ & -۳ & -۶ & ۲ & ۲ & -۱ & ۰ & ۰ \\ ۰ & -۲ & -۴ & ۱ & ۱ & ۰ & ۰ & ۰ \\ -۳ & ۱ & ۵ & -۱ & -۱ & ۰ & ۰ & ۰ \\ -۳ & ۱ & ۲ & -۱ & ۰ & ۰ & ۰ & ۰ \\ ۱ & ۰ & ۰ & ۰ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۰ & ۰ & ۰ & ۰ \end{bmatrix} \tag{۶.۲}$$

همان‌طور که در ماتریس ۶.۲ مشهود است تعداد بسیار زیادی از عناصر ماتریس جدید مقدار صفر دارند (که بیشتر فرکانس‌های بالا را شامل می‌شوند) در ادامهٔ فشرده‌سازی ماتریس ۶.۲ به وسیلهٔ الگوریتم فشرده‌سازی بدون هدررفت داده Huffman فشرده می‌شود و به همراه مقداری دادهٔ افزوده [۳] مانند درصد فشرده‌سازی و... ذخیره می‌شود.

۵.۲ فشرده‌سازی Huffman

برای این که ماتریس مرحلهٔ آخر را به یک رشتهٔ متنی تبدیل کنیم می‌خواهیم از فشرده‌سازی یا الگوریتم Huffman استفاده کنیم. در این قسمت مختصرا الگوریتم Huffman برای فشرده‌سازی یک رشتهٔ متنی شرح داده می‌شود.

۱.۵.۲ تعریف

برای نمایش رشته در حالت ascii برای هر حرف ۸ بیت فضا گرفته می‌شود و تمامی حروف با یک آرایهٔ هشت‌بیتی نمایش داده می‌شوند. رشتهٔ زیر را در نظر بگیرید.

$$S = bananasc$$

در این رشته در حالت ascii به $۶۴ = ۸ * ۸$ بیت فضا نیاز داریم، در صورتی که بخواهیم از طراحی با طول بیت ثابت برای هر حرف در این رشته استفاده کنیم می‌توانیم نظیرسازی زیر را در نظر بگیریم.

جدول ۱.۲: نوعی نظیرسازی حروف با طول ثابت برای هر حرف

حرف	نماد
a	۰۰۱
b	۰۱۰
n	۰۱۱
s	۱۰۰
c	۱۰۱

در این روش برای نشان دادن هر حرف به سه بیت فضا نیاز داریم و در کل به $۲۴ = ۳ * ۸$ بیت فضا نیاز داریم.

اما در صورتی که به صورت دقیق‌تر به رشته نگاه کنیم متوجه می‌شویم که تعداد تکرار هر حرف یکسان نیست و می‌توانیم برای نشان دادن هر حرف از تعداد بیت متغیر استفاده کنیم به شکلی که حروفی که بیشتر تکرار شده‌اند تعداد بیت کمتری بگیرند و حروفی که کمتر تکرار شده اند از تعداد بیت بیشتری برای ذخیره‌سازی استفاده کنند. مثلا نظیرسازی زیر را برای این رشته در نظر بگیرید.

درصورتی که رشتهٔ اصلی را با این نظیرسازی فشرده کنیم به بیت‌های زیر می‌رسیم.

۱۱۰۰۱۰۰۰۱۰۰۱۱۱۱۱۱۰

همان‌طور که در رشتهٔ جدید مشهود است تعداد بیت‌های استفاده‌شده برای نمایش رشتهٔ اصلی به ۱۸ بیت کاهش پیدا کرده، نکته اساسی در این فشرده‌سازی این است که رشته به صورت یکتا قابل بازیابی باشد، در

[۳] meta data

جدول ۲.۲: نوعی نظیرسازی حروف با طول متغیر برای هر حرف

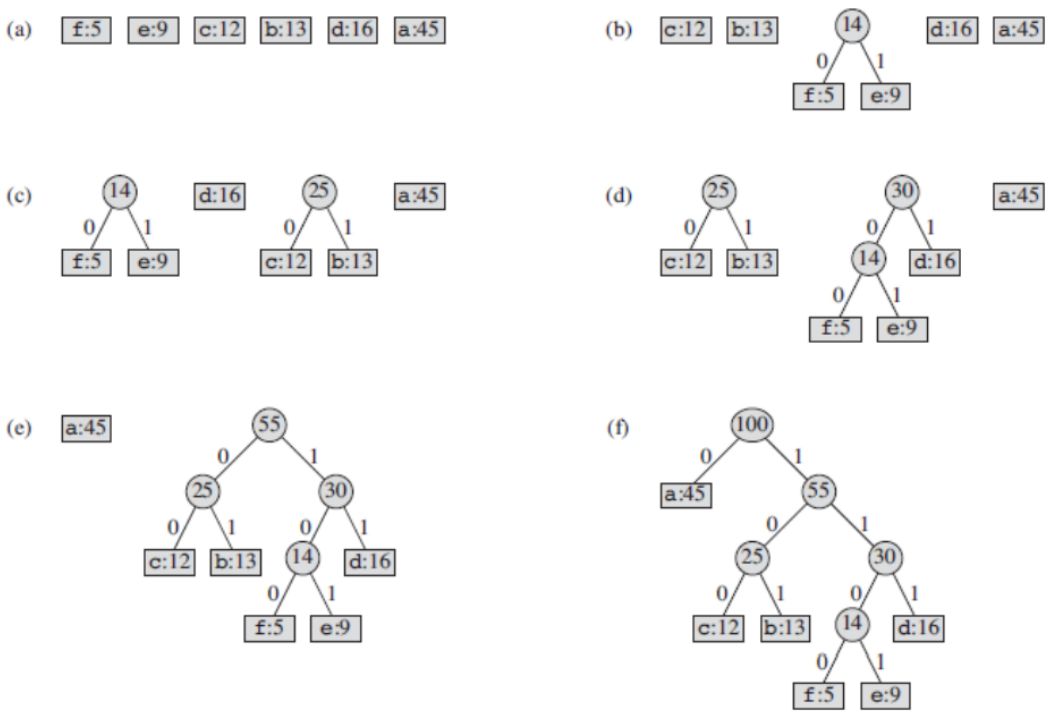
حرف	نماد
a	۰
b	۱۱۰
n	۱۰
s	۱۱۱
c	۱۱۱۰

صورتی که به رشتهٔ بالا دقت کنیم متوجه می‌شویم که تنها حالتی که می‌توان با توجه به حروف جدول برای بازیابی بیت‌ها متصور شد همین حالتی‌ست که به رشتهٔ اصلی منجر می‌شود، این اتفاق به این دلیل رخ می‌دهد که شروع هیچ حرفی زیرمجموعهٔ پیشوندی هیچ رشتهٔ دیگری نیست، برای مثال در صورتی که حرف a با ۰ و حرف b با ۰۱۰ و حرف n با ۱۰ نمایش داده می‌شدندبرای رشتهٔ بیتی ۰۱۰ دو حالت an و b می‌توانستیم متصور شویم، علت این اتفاق این است که رشتهٔ حرف a زیرمجموعهٔ حرف b است.

کلیت الگوریتم Huffman پیدا کردن بهترین کدگذاری برای هر حرف در رشتهٔ اصلی است که طول رشتهٔ نهایی کم‌ترین اندازه را داشته باشد، برای این کار از شبه کد زیر استفاده می‌شود. [۱۰]

```
//Huffman Algorithm
n := |C|;
Q := C;
for i := 1 to n - 1 do
        allocate a new node z
        z.lef t := x := Extract-Min(Q);
        z.right := y := Extract-Min(Q);
        z.freq := x.freq + y.freq;
        Insert(Q, z);
end for
return Extract-Min(Q); {return the root of the tree}
```

شکل ۵.۲ مراحل الگوریتم Huffman را نشان می‌دهد.



شکل ۵.۲: مراحل الگوریتم Huffman [۱۱]

فصل ۳

بررسی نحوهٔ فشرده‌سازی در فرمت MP۳

توضیح نحوهٔ کار فشرده‌سازی در فرمت MP3، تحلیل صدای انسان و توضیح کانال‌های فرکانسی

۱.۳ مقدمه

در اواخر قرن بیستم و با شروع فراگیری اینترنت در جوامع مختلف، اشتراک‌گذاری مدیاهای مختلف مانند صوت، فیلم و تصویر در فضای اینترنت به یکی از خواست‌های همگانی و نیازهای اصلی مردم تبدیل شد اما مشکل اصلی در این میان، حجم زیاد فایل‌های مدیا و سرعت پایین انتقال داده در فضای اینترنت بود، فرمت‌های فشرده‌سازی مختلفی برای حل این معضل پیشنهاد شدند که هر کدام با تکیه بر تحلیل‌های آماری و یا حذف داده‌هایی غیرضروری در راستای کم کردن حجم فایل‌های مدیا می‌کوشیدند. در زمینهٔ فشرده‌سازی فایل‌های صوتی فرمت ۳MP یقینا فراگیرترین و موفق‌ترین فرمت به شمار می‌رود. در این فصل نگاهی کلی به نحوهٔ کار این فرمت و الگوریتم‌های با هدررفت داده به کاررفته در این فرمت خواهیم داشت. برای تقریب اذهان به بزرگی حجم فایل‌های صوتی فشرده‌ناشده لازم به توجه است که هر فایل صوتی در حالتی که هیچ مقداری از داده فشرده نشود در هرثانیه حدود ۱۷۶۰۰۰ بایت فضا می‌گیرد.

۲.۳ نرخ بیتی

برای درک بهتر مفاهیمی که در ادامهٔ متن با آن‌ها سر و کار داریم لازم است تا ابتدا با مفهوم نرخ بیتی [۱] آشنا شویم، در مفاهیم علوم کامپیوتر، نرخ بیتی معمولا مقدار بیتی‌ست که در برای واحد زمانی ذخیره می‌شود، مثلا وقتی bitrate یک موسیقی 320 kb/s اعلام می‌شود به این معنی‌ست که برای هر ثانیه از این صوت ۳۲۰ کیلوبیت داده ذخیره شده است؛ همچنین نرخ بیتی می‌تواند سرعت انتقال داده در یک کانال را بیان کند، مثلا در طراحی مودم‌های شبکه از این مفهوم برای نمایش سرعت انتقال داده استفاده می‌شود.

۳.۳ نحوهٔ کار ۳mp

تمرکز فرمت ۳mp در فشرده‌سازی بر حذف اصواتی‌ست که توسط گوش انسان شنیده نمی‌شوند، همان‌طور که از فصل دو به یاد دارید برای فشرده‌سازی تصاویر می‌توانستیم از بیت‌هایی که نمایانگر تصاویری با فرکانس بالا بودند صرف نظر کنیم زیرا توسط چشم انسان قابل تشخیص نبودند، در فشرده‌سازی صوت نیز می‌توانیم با مطالعهٔ ساختار شنوایی انسان اصواتی که به طور معمول توسط گوش انسان شنیده نمی‌شنود را از صوت حذف می‌کنیم تا حجم فایل کاهش یابد.

به شکل خلاصه می‌توان مراحل فشرده‌سازی در فرمت ۳mp را به شکل زیر خلاصه کرد.

- تبدیل صوت به قسمت‌های کوچک
- حذف صوت‌های خارج از محدودهٔ شنوایی انسان
- نمونه‌برداری از موسیقی با توجه به نرخ بیتی خواسته شده
- اضافه کردن افزونه‌ها و فشرده‌سازی نمونه

[۱] bitrate

۴.۳ بررسی دستگاه شنوایی انسان

برای فشرده‌سازی صوت باید در ابتدا اصواتی که توسط گوش انسان قابل شنیدن نیستند یا گوش انسان با جزییات کمتری آن‌ها را درک می‌کند را حذف کنیم، برای این کار به تحلیل صوت‌شناسی [۲] نیاز دارد. نتایج تحلیل‌های صوت‌شناسی برای دستگاه شنوایی انسان موارد مفید زیر را اعلام می‌کند.

- ضعف شنوایی بزرگسالان
- درک کمتر جزییات صداهای کم
- آستانهٔ شنوایی انسان
- اثر پوشش‌دهی صداهای بلند

به تفصیل موارد بالا و کاربردهای آنان در فشرده‌سازی را بررسی خواهیم کرد.

۱.۴.۳ ضعف شنوایی بزرگسالان

در کودکی انسان‌ها معمولا می‌توانند اصواتی را که بین فرکانس‌های ۱۰ تا ۲۰ کیلوهرتز باشند را بشنوند، اما با بزرگ شدن انسان معمولا حد بالای شنوایی به مقدار ۱۵ کیلوهرتز می‌رسد و در صورتی که به درصد فشرده‌سازی بالایی نیاز داشته باشیم می‌توانیم از فرکانس‌های بیش از ۱۵ کیلوهرتز صرف نظر کنیم.

۲.۴.۳ درک کمتر جزییات صداهای کم

دستگاه شنوایی انسان معمولا به جزییات صداهای بلند بیش از صداهای آرام توجه می‌کند، در نتیجه می‌توانیم برای صداهایی که سطح فشار صوت [۳] کمتری دارند از bitrate پایین‌تری استفاده کنیم.

۳.۴.۳ آستانهٔ شنوایی انسان

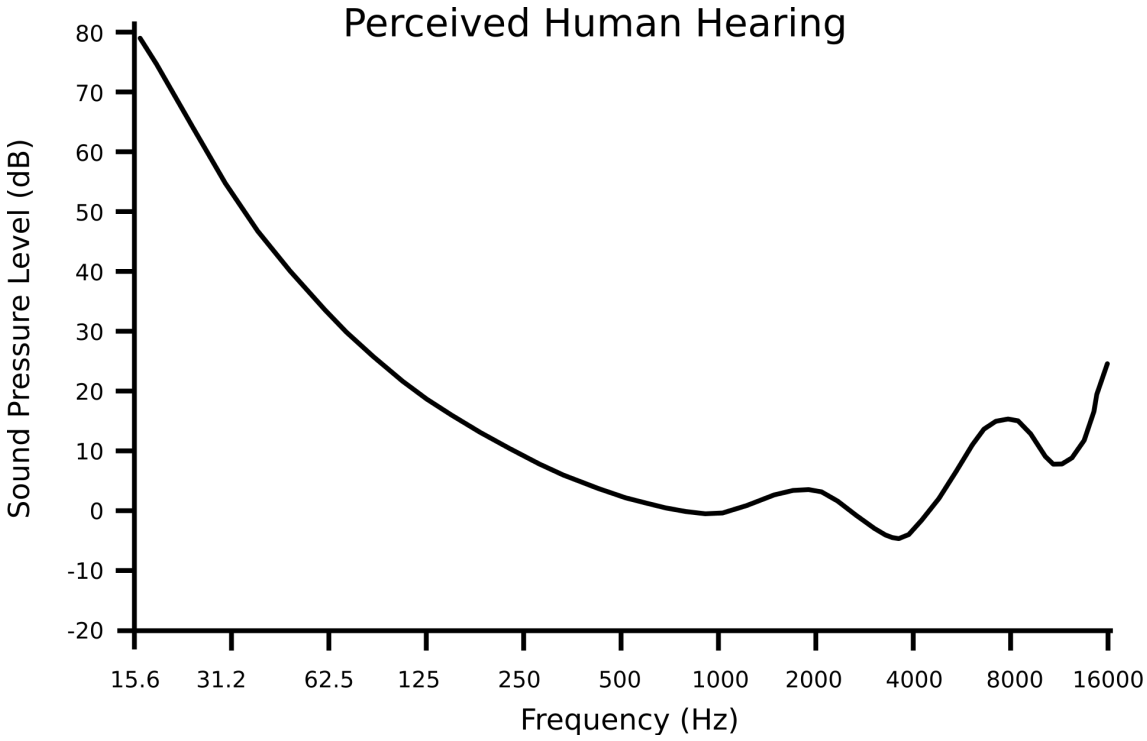
گوش انسان برای هر فرکانس صوتی آستانهٔ شنوایی دارد که اگر سطح فشار صدا از آن کمتر باشد آن را نمی‌شنود، شکل ۱.۳ این آستانه را برای فرکانس‌های مختلف نشان می‌دهد؛ در صورتی که سطح صوتی فرکانسی در هر قسمت از مقدار آستانهٔ آن کمتر باشد آن صدا شنیده نمی‌شود و می‌توان آن را حذف کرد.

۴.۴.۳ اثر پوشش‌دهی صداهای بلند

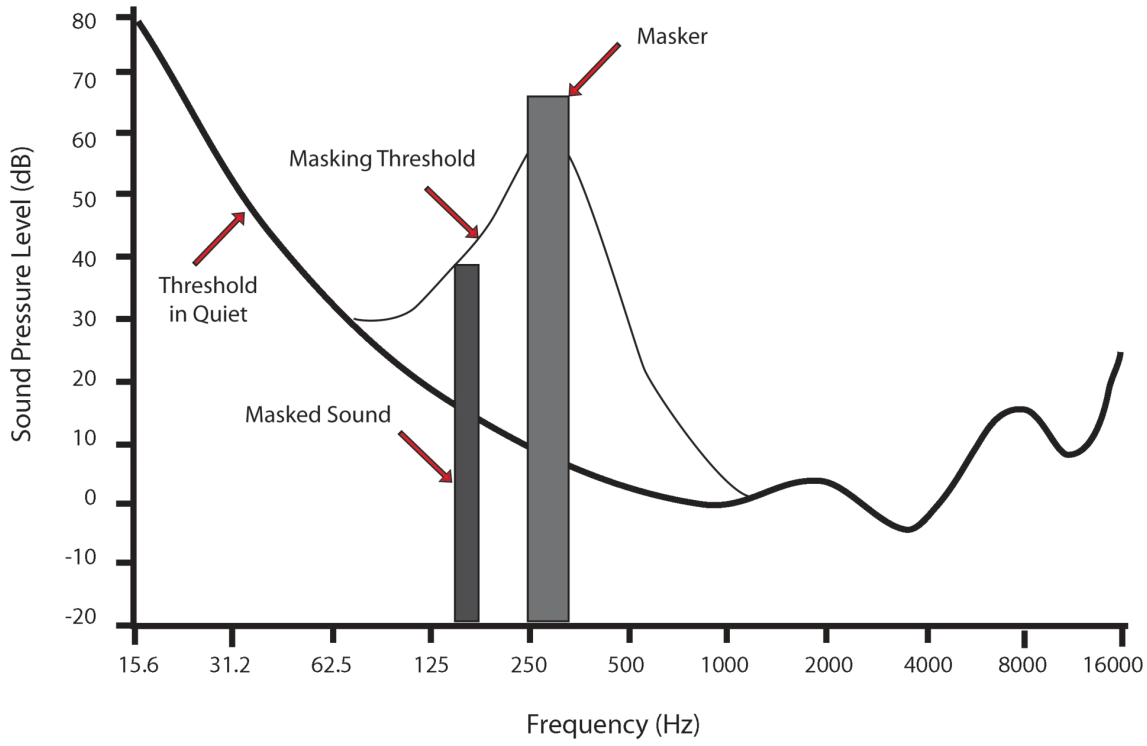
هنگامی که در یک محدودهٔ فرکانسی یک صدای بلند رخ دهد گوش انسان نمی‌تواند صداهای آرام با فرکانس‌های نزدیک به آن را تشخیص دهد و بشنود حتی اگر سطح صوتی آن از آستانهٔ شنوایی انسان بیشتر باشد، به این اثر در گوش انسان اثر پوشش‌دهی [۴] گفته می‌شود. مثالی از اثر پوشش‌دهی در شکل ۲.۳ نشان داده شده است.

---

[۲] Psychoacoustics
[۳] Sound Pressure Level (db)
[۴] Masking Effect



شکل ۱.۳: آستانهٔ شنوایی انسان برای فرکانس‌های مختلف



شکل ۲.۳: اثر پوشش‌دهی

۵.۳ فشرده‌سازی mp3 در عمل

پس از بررسی انتزاعی اتفاقاتی که برای فشرده‌سازی در فرمت ۳mp میافتد نیازمند آنیم تا با یک مثال جزییات فنی پیاده‌سازی این فرمت را نیز بررسی کنیم.

هنگامی که یک صوت برای فشرده شدن انتخاب می‌شود ابتدا به تکه‌های صوتی کوچک‌تر تقسیم یم‌شود و با استفاده از نسخهٔ اندکی تغییریافته الگوریتم DCT که در فصل دوم معرفی شد به فضای فرکانس برده می‌شود، پس از آن اصواتی که فرکانس‌هایی پایین‌تر از آستانهٔ شنوایی انسان دارند حذف می‌شوند و همین‌طور برای هر کانال صوتی عامل پوشش‌دهنده [۵] و عوامل پوشش پذیر [۶] شناسایی می‌شوند و بیت‌های مربوط به عوامل پوشش‌پذیر حذف می‌شوند، همچنین برای صداهای بلند مقداری از جزییات اصوات آرام‌تر را حذف می‌کنیم زیرا توسط ذهن انسان تشخیص داده نمی‌شوند.

پس از این تغییرات دوباره به فضای زمان برمی‌گردیم تا نمونه‌گیری را با توجه به bitrate خواسته شده انجام دهیم؛ دقت کنید که تا این‌جای کار هیچ مقداری از بیت‌هایی که توسط دستگاه شنیداری انسان قابل شنیدن باشند از دست نرفته اند.

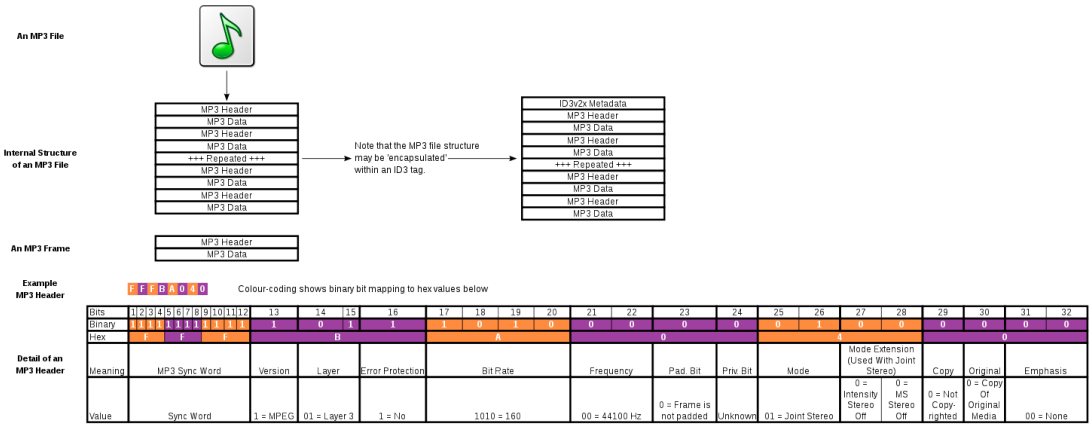
در گام سوم با توجه به bitrate از هر قسمت کوچک ساخته شده نمونه برداری می‌کنیم و سپس به گام آخر می رسد، در این گام اطلاعاتی که برای بازیابی هر قسمت مورد نیاز است به همراه تعدادی بیت که برای تشخیص خطا قرار می‌گیرند را در Header هر قسمت قرار می‌دهیم و سپس داده اصلی را قرار می‌دهیم. مختصرا می‌توان گفت که هر بلوک داده موارد زیر را در خود دارد.

- سرتیتر
  - کد همگام‌سازی
  - نسخه
  - لایه صوتی
  - بیت تشخصی خطا
  - نرخ بیتی
  - فرکانس
  - بیت padding
  - بیت خصوصی
  - حالت صوت
  - دادهٔ کپی‌شده
  - دادهٔ اصلی
  - بیت تاکید
- داده صوتی

موارد بالا به صورت جدولی در شکل ۳.۳ آورده شده‌اند.

---

[۵] Masker
[۶] Maskee



شکل ۳.۳: ساختار فایل ۳MP

۴ مراجع

[1] Mahdi, O.A.; Mohammed, M.A.; Mohamed, A.J. (Marhc 2013) *Implementing a Novel Approach an Convert Audio Compression to Text Coding via Hybrid Technique*. International Journal of Computer Science Issues.

[2] Pujar, J.H.; Kadlaskar, L.M. (May 2010). *"A New Lossless Method of Image Compression and Decompression Using Huffman Coding Techniques" (PDF)*. Journal of Theoretical and Applied Information Technology.

[3] Mahdi, E: *https://github.com/merfanian/DataCompressionDoc/blob/master/LatexFiles/figs/compressed.png*

[4] Web Archive, *https://web.archive.org/web/20090202063734/http://synthetic-soul.co.uk/comparison/lossless/index.asp*

[5] Arcangel, Cory. (March 2013) *On Compression*

[6] Ahmed, Nasir (January 1991). *How I Came Up With the Discrete Cosine Transform*

[7] By en:User:Cburnett - Own work in Inkscape based on the following data:, CC BY-SA 3.0, *https://commons.wikimedia.org/w/index.php?curid=3333955*

[8] Computerphile, *https://www.youtube.com/watch?v=n_uNPbdenRs*

[9] Wikipedia, *https://commons.wikimedia.org/wiki/File:Dctjpeg.png*

[10] Huffman, D. (1952). *"A Method for the Construction of Minimum-Redundancy Codes"*

[11] Radu Trımbiţaş (2012), *Huffamn Codes*

[12] Stéphane Grumbach, FarizaTahi (November 1994), *A new challenge for compression algorithms: Genetic sequences*