

دانشکده مهندسی کامپیوتر

ارائه مطالب علمی و فنی

مستند پروژه

---

# بررسی الگوریتم‌های فشرده‌سازی و کاربردهای آنها

---

نگارندگان:  
محمد مهدی عرفانیان

۱۲ دی ۱۳۹۸



## چکیده

در مستندی که پیش روی خواننده عزیز قرار دارد تلاش شده تا مختصراً الگوریتم‌های فشرده‌سازی مختلف و کاربردهای آن‌ها در زمینه‌های مختلف مهندسی کامپیوتر در راستای انجام پروژه درس ارائه مطالب علمی و فنی بررسی شود. این درس در پاییز ۹۸ در دانشکده مهندسی کامپیوتر دانشگاه صنعتی شریف توسط دکتر همت‌یار ارائه شده است.

برای سهولت کار استاد محترم درس برای تحصیل اطمینان از درستی مستندسازی و همچنین استفاده دانش‌جویان علاقه‌مند، سیر پیشرفت مستند به همراه کدهای  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  در *Github* قرار گرفته است، لازم به ذکر است که این مستند به صورت متن‌باز ارائه شده و استفاده از آن بدون ذکر منبع برای همگان آزاد است. در انتها از استاد محترم درس، دستیار آموزشی ایشان و خوانندگان محترم تشکر می‌کنم.

با آرزوی خوش‌وقتی برای تمامی خوانندگان این مستند  
نگارنده

# فهرست مطالب

۴	۱	مقدمه
۵	۱.۱	تعریف
۵	۲.۱	انواع الگوریتم‌های فشرده‌سازی
۵	۱.۲.۱	الگوریتم‌های Lossless
۶	۲.۲.۱	الگوریتم‌های Lossy
۸	۲	بررسی نحوه فشرده‌سازی در فرمت JPEG
۹	۱.۲	DCT
۹	۱.۱.۲	تعریف
۹	۲.۲	تبدیل دنباله‌ای از پیکسل‌ها به تابع
۹	۳.۲	نحوه کار JPEG
۹	۴.۲	یک مثال از فشرده‌سازی JPEG
۱۱	۱.۴.۲	Quantization Table
۱۳	۵.۲	فشرده‌سازی Huffman
۱۳	۱.۵.۲	تعریف

# فهرست تصاویر

۱.۱	نمونه فایل تولیدشده توسط نگارنده برای تست میزان کمپرس در فرمت PNG . . . . .	۶
۱.۲	توصیف فرکانسی توابع $\cos(x), \cos(2x)$ . . . . .	۱۰
۲.۲	توصیف فرکانسی تابع $\cos(x) + \cos(2x)$ . . . . .	۱۰
۳.۲	ماتریس فرکانس‌های استاندارد . . . . .	۱۲

# فهرست جداول

۶	.....	۱.۱	حجم فایل نمونه در فرمت PNG
۷	.....	۲.۱	حجم فایل نمونه در فرمت JPG
۱۳	.....	۱.۲	نوعی نظیرسازی حروف با طول ثابت برای هر حرف
۱۳	.....	۲.۲	نوعی نظیرسازی حروف با طول متغیر برای هر حرف

# فصل ۱

## مقدمه

توضیحی اولیه مبنی بر تعریف کلی فشردده سازی، 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. برای مثال می‌دانیم که برای ذخیره هر بیت اسکی 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.

- Lossless یا بدون هدررفت داده
- Lossy یا همراه هدررفت داده

### ۱.۲.۱ الگوریتم‌های Lossless

در این سری الگوریتم‌ها داده ورودی بدون هیچ‌گونه هدررفتی از داده خروجی قابل بازیابی است، الگوریتم‌های این دسته با استفاده از افزودنی آماري تلاش می‌کنند تا نحوه نمایش داده را در نگاشتی به نحوه نمایش دیگری که به فضای کمتری نیاز دارد تبدیل کنند. این الگوریتم‌ها در مواقعی که ثابت ماندن داده در طی فشرده‌سازی الزامی است استفاده می‌شوند، همچنین معمولاً 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 حرف می‌تواند باعث بدخوانی و بدفهمی متن اصلی گردد. الگوریتم‌های مشهور کمپرس Lossless به شرح زیر اند.

- 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 Lossless

در عمل از الگوریتم‌های Lossless در مواقعی که پایداری داده‌های ذخیره‌شده حیاتی است یا این که فایل در آینده به تعداد زیادی بار فشرده و گسترده می‌شود و از دست دادن قسمتی از داده در هربار فشرده‌سازی منجر به اختلاف فاحش نهایی خواهد شد استفاده می‌شود.

#### • PNG

در طراحی فرمت PNG برای فشرده‌سازی تصاویر از الگوریتم Lempel-Ziv-Welch (LZW) که الگوریتمی 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.

```

۸ ۱ ۸ ۵ ۱ ۰ ۰ ۱ ۱ ۸
۸ ۴ ۴ ۳ ۹ ۱ ۷ ۳ ۰ ۵
۸ ۹ ۷ ۰ ۵ ۲ ۸ ۹ ۲ ۰
۴ ۱ ۶ ۰ ۱ ۰ ۲ ۸ ۶ ۳
۱ ۴ ۷ ۲ ۱ ۵ ۴ ۷ ۵ ۶
۷ ۱ ۲ ۳ ۶ ۷ ۵ ۵ ۹ ۰
۴ ۸ ۲ ۱ ۲ ۲ ۱ ۶ ۲ ۶
۶ ۰ ۵ ۹ ۵ ۴ ۹ ۹ ۶ ۱
۲ ۳ ۹ ۹ ۵ ۴ ۵ ۴ ۰ ۹
۴ ۴ ۴ ۸ ۶ ۶ ۰ ۹ ۹ ۴

```

شکل ۱.۱: نمونه فایل تولیدشده توسط نگارنده برای تست میزان کمپرس در فرمت PNG

جدول ۱.۱: حجم فایل نمونه در فرمت PNG

Format	Size
BMP	۷.۷ مگابایت
PNG	۹۸ کیلوبایت

#### • Free Lossless Audion Codec (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 معمولاً بین ۴۰ تا ۶۰ درصد می‌باشد اما در حالت بیشینه ممکن است تا ۸۰ درصد هم برسد.

### ۲.۲.۱ الگوریتم‌های Lossy

در این الگوریتم‌ها پس از هر بار فشرده‌سازی مقداری از داده‌ها از دست می‌روند، معیار ارزیابی این الگوریتم‌ها مقدار فشرده‌سازی با توجه به میزان هدررفت داده می‌باشد، به علت هدررفت مقداری از داده این الگوریتم‌ها معمولاً در مواردی که هدررفت اندک داده توسط انسان یا ماشین قابل تشخیص نباشد استفاده می‌شوند، مثلاً تکنیک‌های ذخیره‌سازی تصاویر و ویدئوها در کامپیوترها مبتنی بر الگوریتم‌های Lossy است زیرا چشم انسان قادر به تشخیص عوض شدن تعدادی پیکسل در صفحه پس از بازیابی فایل فشرده‌شده نیست.

معمولاً در فشرده‌سازی Lossy از Transform Coding استفاده می‌شود که داده‌ها را از فضای حقیقی به فضایی دیگر (معمولاً فرکانس) می‌برد و در آنجا از قسمت‌هایی از داده که تاثیرگذاری و حس‌پذیری کمتری نسبت به دیگران دارند صرف نظر می‌شود، سپس وارون تبدیل اجرا شده و داده‌های کوچک‌شده جدید با الگوریتم‌های Lossless فشرده می‌شوند. یکی از مشهورترین Transform Coding ها الگوریتم Discrete Cosine Transform (DCT) است که در فصول آتی این مستند بیشتر با آن آشنا خواهیم شد.

#### نمونه‌های الگوریتم‌های Lossy

الگوریتم‌های Lossy با همه‌گیر شدن اینترنت و اشتراک‌گذاری بیشتر مدیا در فضای اینترنت بسیار همه‌گیر شدند، از این رو اکثر این الگوریتم‌ها برای فشرده‌سازی فایل‌های صوتی-تصویری یا به اصطلاح Media



استفاده می‌شوند.

- JPEG
- MP3
- MP4
- H.26x

به عنوان نمونه برای الگوریتم Lossy حالت فشرده‌شده عکس ۱.۱ با فرمت JPEG با کیفیت‌های مختلف در جدول ۲.۱ آورده شده است.<sup>۱</sup>

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

<sup>۱</sup> در صورتی که تمایل به دیدن اصل فایل‌ها دارید می‌توانید به *Github* مستند مراجعه کنید.

## فصل ۲

# بررسی نحوه فشرده‌سازی در فرمت JPEG

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

## ۱.۲ DCT

### ۱.۱.۲ تعریف

تبدیل کسینوسی گسسته (به انگلیسی: Discrete cosine transform)، دنباله ای محدود از اعداد (داده ها) را به صورت مجموع توابع کسینوسی با فرکانس های متفاوت نمایش می دهد. 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 علمی تر می توان نوشت که DCT تابعی معکوس پذیر و خطی از  $R^N$  به  $R^N$  است. فرمول کلی DCT برای فضای یک بعدی به شکل زیر است.

$$X_k = \sum_{n=0}^{N-1} x_n \cos[\pi/n(n+1/2)k], k = 0, \dots, N-1 \quad (1.2)$$

## ۲.۲ تبدیل دنباله ای از پیکسل ها به تابع

هر تصویر در حقیقت به صورت چهار ماتریس دوبعدی ذخیره می شود که هر ماتریس مقدار رنگی پیکسل را برای آن کانال رنگی (RGB) و  $\alpha$  نشان می دهد، برای کمپرس کردن یک عکس ماتریس های کانال های رنگی مختلف را جداگانه فشرده می کنیم، کانال رنگی R را در نظر بگیرید، در این حالت یک ماتریس  $n*m$  داریم که هر خانه آن عددی بین ۰ تا ۲۵۵ را نشان می دهد، برای سادگی یک سطر از این ماتریس را در نظر بگیرید، این سطر معادل با یک آرایه از اعداد می باشد، در صورتی که تمامی اعداد را از دامنه ۰ تا ۲۵۵ به دامنه ۱۲۸ - ۱۲۷ ببریم می توانیم این دنباله را با مجموعه ای از توابع کسینوسی بازتولید کنیم، این ضرایب همان ضرایبی ست که با استفاده از DCT می خواهیم به دست بیاوریم.  
شکل ۱.۲ نمونه ای از توصیف عکس با توابع کسینوسی را نشان می دهد.

حال اگر دو تابع نشان داده شده در شکل ۱.۲ را با هم جمع کرده و میانگین بگیریم، به نمایش فرکانس دیگری می رسیم که در شکل ۲.۲ نشان داده شده است. در صورتی که به شکل های متفاوت و با ضرایب متفاوت توابع مختلف کسینوسی را با هم جمع کنیم می توانیم هر فرکانسی را بسازیم. این کاری ست که در عمل الگوریتم DCT برای ما انجام می دهد.

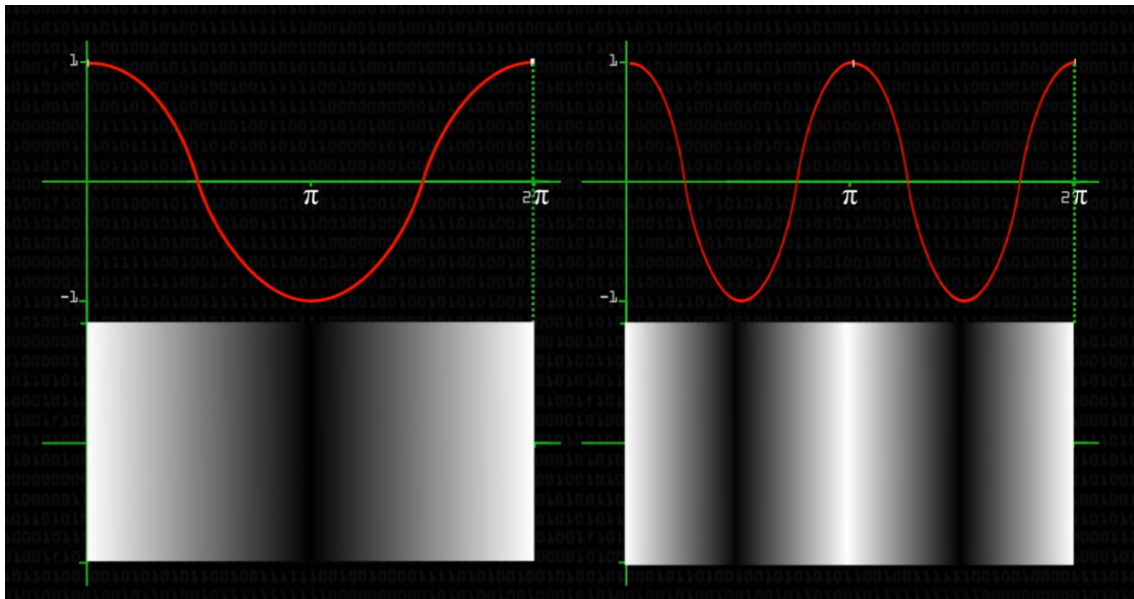
## ۳.۲ نحوه کار JPEG

برای فشرده سازی یک تصویر در JPEG مراحل زیر انجام می شود.

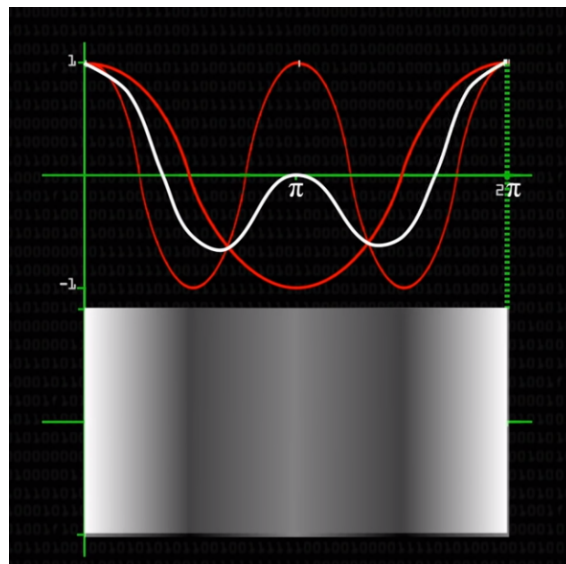
- تبدیل تصویر به بلوک های کوچک
- تبدیل هر بلوک به مجموعه ای از توابع کسینوسی استاندارد با DCT
- تنظیم ضرایب متناسب برای بلوک های DCT
- کمپرس کردن ضرایب با استفاده از Huffman

## ۴.۲ یک مثال از فشرده سازی JPEG

برای درک بهتر مراحل گفته شده تلاش می کنیم تا ماتریس زیر را (که در حقیقت می تواند یک کانال رنگی از قسمتی از یک تصویر باشد) را با الگوریتم های گفته شده فشرده سازی کنیم.



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



شکل ۲.۲: توصیف فرکانسی تابع  $\cos(x) + \cos(2x)$

$$M = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 60 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

هر عنصر این ماتریس عددی بین ۰ تا ۲۵۵ دارد اما از آنجایی که تابع کسینوسی مقادیر بین ۱- تا ۱ می‌گیرد نیازمندیم تا با کم کردن هر عنصر این ماتریس از ۱۲۸ هر عنصر این ماتریس را به عددی بین ۱۲۷- تا ۱۲۸ تبدیل کنیم. ماتریس تبدیل‌شده به شکل زیر است.

$$M_{shifted} = \begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

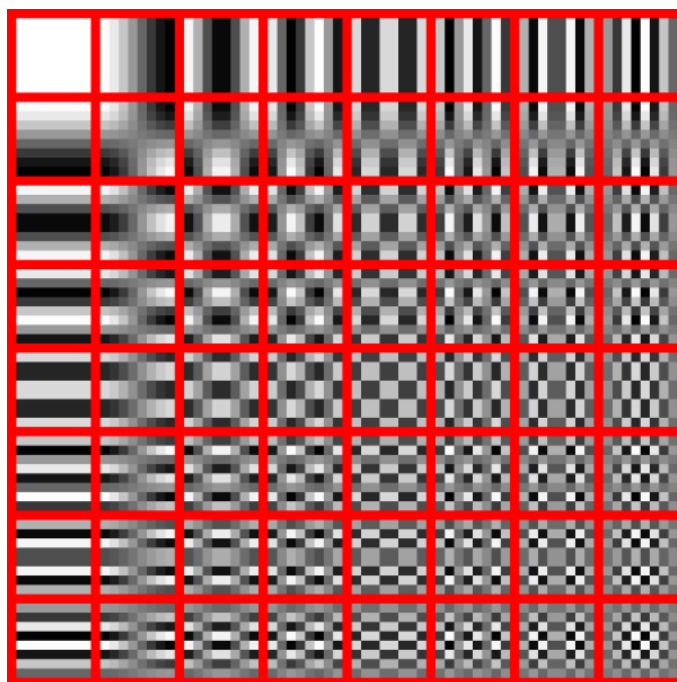
در گام بعدی مقادیر این ماتریس را با استفاده از الگوریتم DCT به فضای فرکانس می‌بریم هر مقدار از این ماتریس جدید برابر با ضریب فرکانس معادل در ماتریس استاندارد است.

$$F = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.28 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

هر کدام از عناصر ماتریس بالا مقدار ضریب تاثیر فرکانس نشان داده‌شده در شکل ۳.۲ می‌باشند. تا به این لحظه هیچ مقداری از داده را برای فشرده‌سازی از دست نداده‌ایم اما در این گام می‌خواهیم قسمت‌هایی از تصویر که برای چشم انسان قابل تشخیص نیستند را حذف کنیم تا مقدار داده کمتری را ذخیره کنیم. باید توجه داشت که چشم انسان معمولاً قادر به تشخیص و تمیز تصاویر با فرکانس‌های بالا در تصاویر نیست و همان‌طور که در ماتریس 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.

## Quantization Table ۱۰.۴.۲

Quantization table جدولی است که در آن مقدار تاثیر هر فرکانس برای هر رده فشرده‌سازی برای



شکل ۳.۲: ماتریس فرکانس‌های استاندارد

عکس‌های JPEG به صورت جهانی استانداردسازی و تعیین شده‌است. برای مثال برای نرخ کمپرس ۵۰ درصد Quantization Table به شکل زیر است.

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

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

$$R = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

همان‌طور که در ماتریس R مشهود است تعداد بسیار زیادی از عناصر ماتریس جدید مقدار صفر دارند (که بیشتر فرکانس‌های بالا را شامل می‌شوند) در ادامه فشرده‌سازی این ماتریس به وسیله الگوریتم فشرده‌سازی بدون هدررفت داده Huffman فشرده می‌شود و به همراه مقداری داده افزوده<sup>۱</sup> مانند درصد فشرده‌سازی

<sup>۱</sup> meta data

و... ذخیره می‌شود.

## ۵.۲ فشرده‌سازی Huffman

برای این که ماتریس مرحله آخر را به یک رشته متنی تبدیل کنیم می‌خواهیم از فشرده‌سازی یا الگوریتم Huffman استفاده کنیم. در این قسمت مختصراً الگوریتم Huffman برای فشرده‌سازی یک رشته 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 آرایه هشت‌بیتی نمایش داده می‌شوند. رشته زیر را در نظر بگیرید.

$$S = \text{bananasc}$$

در این رشته در حالت ascii به  $8 * 8 = 64$  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

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

در این روش برای نشان دادن هر حرف به سه بیت فضا نیاز داریم و در کل به  $8 * 3 = 24$  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 شده‌اند از تعداد بیت بیشتری برای ذخیره‌سازی استفاده کنند. مثلاً نظیرسازی زیر را برای این رشته در نظر بگیرید.

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

نماد	حرف
۰	a
۱۱۰	b
۱۰	n
۱۱۱	s
۱۱۱۰	c

در صورتی که رشته اصلی را با این نظیرسازی فشرده کنیم به بیت‌های زیر می‌رسیم.

110010001001111110

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

صورتی که به رشته بالا دقت کنیم متوجه می‌شویم که تنها حالتی که می‌توان با توجه به حروف جدول برای بازیابی بیت‌ها متصور شد همین حالتی است که به رشته اصلی منجر می‌شود، این اتفاق به این دلیل رخ می‌دهد که شروع هیچ حرفی زیرمجموعه پیشوندی هیچ رشته دیگری نیست، برای مثال در صورتی که حرف  $a$  با  $0$  و حرف  $b$  با  $010$  و حرف  $n$  با  $10$  نمایش داده می‌شدند برای رشته  $010$  دو حالت  $an$  و  $b$  می‌توانستیم متصور شویم، علت این اتفاق این است که رشته حرف  $a$  زیرمجموعه حرف  $b$  است.