هو العلیم



|  |
| --- |
| **طراحی و تحلیل الگوریتم­ها** |
| **نیمسال دوم سال تحصیلی 1401 – 1400** |
| **تمرینات برنامه­نویسی 1** |
| **مریم رضائی** |

**1.** می­خواهیم مبتنی بر الگوریتم هافمن، برنامه­ای برای کدگذاری (فشرده­سازی) متون و برنامه­ای مستقل نیز برای کدگشایی (منبسط­سازی) متون فشرده بنویسیم. فرض کنید متونی که قرار است فشرده شوند با کدهای اَسکی[[1]](#footnote-1) در رایانه ذخیره شده باشد.

برنامه‌ی کدگذار باید یک فایل متنی را به عنوان ورودی بخواند و یک فایل متنی را به عنوان خروجی بنویسید. به بیان دقیق­تر، برنامه فشرده­ساز باید

* برای بار اول متن ورودی را بخواند و فراوانی (تعداد دفعات تکرار) هر یک از نویسه­های (کاراکترهای) آن را تعیین کند؛
* درخت کدگذاری هافمن را بسازد؛
* جدول حاوی کلمات کد (رشته­های بیتی) هر یک از کاراکترهای متن را بسازد.
* درخت هافمن را به شکل یک رشته بیتی (در ابتدای فایل خروجی) بنویسید.
* تعداد کاراکترهای متن ورودی را به شکل یک رشته ­بیتی (بعد از رشته بیتی درخت) بنویسید.
* برای بار دوم متن ورودی را بخواند و با استفاده از جدول کلمات کد، به جای هر کاراکتر در متن، رشته بیتی معادل با آن را بگذارد تا رشته بیتی فشرده­ی معادل با متن اصلی را به دست آورد. و این رشته بیتی را (بعد از رشته بیتی معادل با تعداد کاراکترهای متن ورودی) در فایل خروجی بنویسید.
* بعد از فشرده کردن متن اصلی و ذخیره کردن آن در فایل خروجی، هم متن اصلی و هم متن فشرده و هم نسبت فشرده­سازی (تعداد بیت­های نمایشگر متن فشرده به تعداد بیت­های نمایشگر متن اصلی) را در پنجره خروجی چاپ کند.

برنامه کدگشا باید یک فایل متنی را که قبلاً با برنامه کدگذار فشرده شده است، به عنوان ورودی بخواند و با پردازش آن، فایل متنی اصلی را بازیابی کند. به بیان دقیق­تر، برنامه منبسط­ساز باید

* درخت هافمن را (که به شکل یک رشته بیتی در ابتدای متن فایل ورودی قرار گرفته است) بسازد.
* با خواندن قسمتی دیگر از متن، تعداد کاراکترهایی را که باید آنها را کدگشایی کند تعیین کند.
* با استفاده از درخت، قسمت باقیمانده متن را (که در واقع، همان متن فشرده است) منبسط کند و آن را در یک فایل جدید ذخیره کند.
* بعد از منبسط کردن متن اصلی و ذخیره کردن آن در یک فایل، هم متن فشرده و هم متن اصلی را در پنجره خروجی چاپ کند.

**جواب**:

**توصیف کلی کار برنامه‌ها:** در دو برنامه مجزا، کدگذاری و کدگشایی را می‌نویسیم به طوری که برنامه کدگشا نام یک فایل متنی در پوشه برنامه را گرفته و آن را به صورت یک رشته می‌خواند و در خروجی فایلی در پوشه می‌سازد یا تغییر می‌دهد که دارای کد شده‌ی درخت، طول متن و کد شده‌ی متن در سه سطر مجراست. برای این کار، برنامه از توابع متفاوت استفاده می‌کند که هر کدام در برنامه با کامنت‌ها توضیح داده شده‌اند. در دنباله آن، برنامه کد گشا نام یک فایل متنی در همان پوشه که حاوی سه سطر کد درخت و کد متن است را گرفته و ابتدا درخت را می‌سازد و سپس با استفاده از آن، متن را کد گشایی می‌کند تا در نهایت آن را در فایلی دیگر ذخیره کند.

**مراجع**:

* [**https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3**](https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/)
* [**https://www.geeksforgeeks.org/python-frequency-of-each-character-in-string**](https://www.geeksforgeeks.org/python-frequency-of-each-character-in-string/)

**2.** فرض کنید می­خواهیم در یک شبکه رایانه­ای، از رایانه­ای خاص به عنوان رایانه مبدأ، جریانی پیوسته از داده­ها را با حداکثر سرعت ممکن به رایانه­ای دیگر به عنوان رایانه مقصد، منتقل کنیم. برای آنکه بتوانیم میزان بیشتری از داده­ها را در واحد زمان ارسال کنیم، می­توانیم داده­ها را به بسته­هایی تقسیم کنیم و آن بسته­ها را از طریق مسیرهای مختلف از مبدأ به مقصد بفرستیم. از طرف دیگر، برای ارسال داده­ها هم با این قید مواجه­ایم که نمی­توان جریان داده­ها را با سرعتی بیشتر از یک مقدار مشخص از هر خط ارتباطی بین دو رایانه عبور داد و و هم با این قید مواجه‌ایم که نمی­توان جریان داده­ها را با سرعتی بیشتر از یک مقدار مشخص از هر رایانه میانی (هر مسیریاب) عبور داد.

این شبکه رایانه­ای را می­توان با یک گراف‌جهت­دارِ وزن­دار نمایش داد: هر رایانه عضو شبکه را می­توان رأسی از رئوس گراف در نظر گرفت و هر خط ارتباطیِ یک طرفـه بین دو رایانه را می­توان یالی از یال­های جهت­دار گراف دانست. پهنای باند هر خط ارتباطی (که حداکثر تعداد بایت­هایی است که می­توان در یک ثانیه از آن خط عبور داد) در شبکه، وزن یک یال جهت­دار در گراف را مشخص می­کنـد و پهنای باند هر رایـانه در شبکـه، وزن یک رأس در گراف را مشخص می­کنـد.

برنامه­ای برای حل حالت کلی این مسأله بنویسید. این برنامه باید گراف جهت­دارِ وزن­دار و دو رأس مبدأ و مقصد را بگیـرد و حداکثـر میزان داده­هایی را که می­توان در واحد زمـان از رایانـه به رایانه منتقل کرد، و همچنین مسیرهای ارسال بسته­های داده­ را تعیین کند.

**جواب**:

**توصیف راه:** در اصل سوال همان مسئله جریان بیشینه است با این تفاوت که رئوس وزن داشته و مسیرها را نیز باید خروجی دهیم. به این علت در پیش روی و یافتن راه، کافیست برای هر راه یافت شده با پیمایش سطحی، مسیر را در آرایه‌ای ذخیره کنیم. همچنین برای حل مشکل وزن رأس، می‌توانیم رئوس را در اصل یالی دیگر در نظر گرفته (یعنی یال‌های اصلی جایگزین شده و یال‌های گراف جدید دو برابر شوند) و برای این گراف جدید مسئله جریان بیشینه را حل کنیم. برای خروجی، جریان یکسان خواهد بود و فقط کافیست رئوس مسیر یافت شده را به رئوس گراف اصلی تبدیل کنیم که برای این کار می‌دانیم رئوس جدید دو برابر رئوس قبلی هستند و می‌توان رأس معادل را با محاسبات یافت.

**مشکل برنامه:** برنامه در بخش محاسبه جریان بیشینه گراف تبدیل شده به مشکلی برخورده و با در نظر گرفتن یال جایگزینی به عنوان یک مسیر محدود، از آن گذشته و از جریان باقی‌مانده کم می‌کند و بدین طور جلوی پیشرفت خود از یال اول را می‌گیرد. متاسفانه فرصت فکر و تصحیح این ایراد را نداشتم.

**مراجع**:

* [**https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem**](https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem)

1. ASCII [↑](#footnote-ref-1)