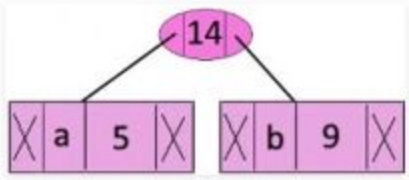
Huffman coding

الگوریتم کد هافمن برای lossless data compression استفاده می شود. در این پروژه یک فایل تکست به شما داده می شود و با استفاده از این الگوریتم , فایل تکست داده شده را به یک رشته بیت انکد می کنید و در یک فایل باینری ذخیره می کنید. همچنین باید برنامه ای جهت دیکد کردن فایل باینری به فایل تکست بنویسید. **این دو برنامه باید از هم جدا باشند.**

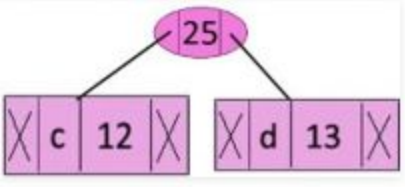
ابتدا برای هر کاراکتر فراوانی آن را در متن بدست می آورید و **حتما دقت کنید کل فرایند بدست آوردن فراوانی باید** $O(n)$ **باشد.** با استفاده از فراوانی ها صف اولویتی تشکیل می دهید. که سر صف کاراکتر با کمترین فراوانی قرار دارد. برای تولید کد هافمن نیاز به ساختمان داده درخت است. برای تولید درخت به نحو زیر عمل می کنیم.

character	Frequency
a	5
b	9
c	12
d	13
e	16
f	45



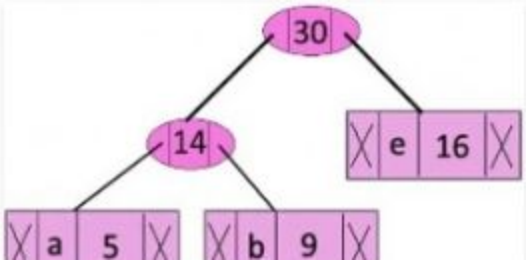
---

character	Frequency
c	12
d	13
Internal Node	14
e	16
f	45



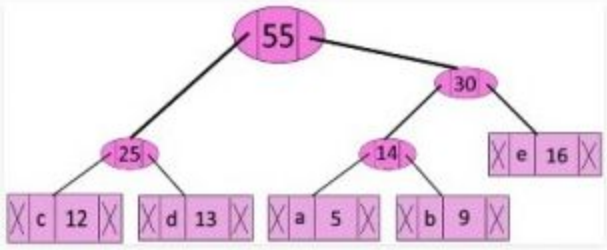
---

character	Frequency
Internal Node	14
e	16
Internal Node	25
f	45



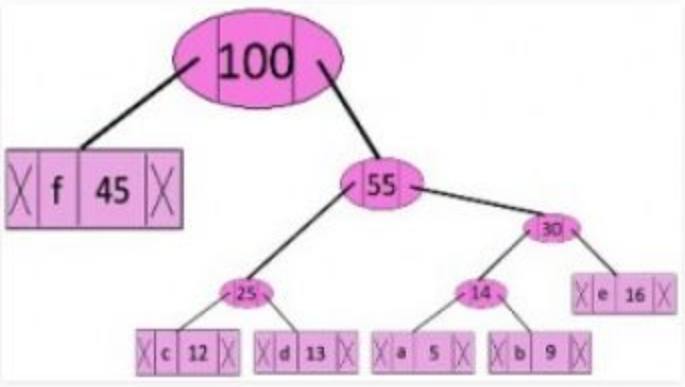
---

character	Frequency
Internal Node	25
Internal Node	30
f	45

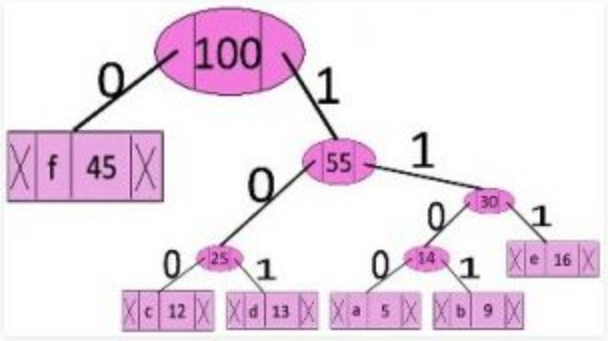


---

character	Frequency
f	45
Internal Node	55



حال هنگام پیمایش درخت برای یال سمت چپ عدد 0 و برای یال سمت راست عدد 1 را در نظر می گیریم. شکل نهایی درخت و کد مخصوص هر کاراکتر به صورت زیر است.



character	code-word
f	0
c	100
d	101
a	1100
b	1101
e	111

با نگاه کوتاهی به کد های بوجود آمده متوجه می شوید کاراکتری که بیشترین فراوانی را دارد کوتاه ترین کد را دارد و این سبب فشرده سازی می شود.

حال به جای استفاده از بایت برای ذخیره کاراکتر ها می توان با چندین بیت این کار را انجام داد.
مثلا رشته ffffcc به رشته 0000100100 تبدیل می شود. برای دیکد کردن رشته بیت هم کافیست درخت کد هافمن را داشته باشید و با پیمایش آن و تطابق آن با درخت هنگام رسیدن به یک برگ در درخت کاراکتر معادل آن بخش پردازش شده از رشته بیتی را با کاراکتر متناسب با آن کد جایگزین کنید.
از آنجایی که برنامه دیکد و انکد از هم جدا هستند, نمی توان درخت هافمن را به صورت مستقیم در اختیار برنامه دیکد قرار داد. برای این کار باید در اول فایل باینری یک قسمت هدر در نظر بگیرید که اطلاعات مربوط به درخت و کاراکتر ها و کد ها را در آن قرار دهید. از آنجایی که این فایل هدر شامل متن اصلی نیست سربار به حساب آمده و برای تولید سربار کمتر باید کد هافمن موجود را به **Canonical Huffman code** تبدیل کرد و از آن برای انکد متن استفاده کرد.و روشی برای ذخیره سازی این کد جدید در هدر یافت.**(دانشجویان می توانند این بخش را از لینک های قرار گرفته در آخر متن پروژه مطالعه کند).**

نمره اضافه

- برای سه کدی که بیشترین فشرده سازی را داشته باشد نمره اضافه در نظر گرفته می شود.

توضیحات تکمیلی و نحوه تحویل

- برای پیاده سازی محدودیت زبان برنامه نویسی وجود ندارد.
- صف اولویت باید با استفاده از مین هیپ پیاده سازی شود و استفاده از صف اولویت موجود در کتابخانه های زبان مورد استفاده مجاز نیست.
- پروژه به صورت ***انفرادی*** است.
- در صورت مشاهده تقلب نمره ***-100*** برای متقلبین در نظر گرفته می شود.
- در یک فایل readme نحوه پیاده سازی به صورت مختصر توضیح داده شود.
- برنامه دیکد و انکد باید از هم جدا باشند و در قالب ۲ برنامه جدا ارسال شوند.
- برنامه انکد و دیکد و readme به صورت یک فایل به فرم DS_project1_studentNumber.zip ارسال شوند.

منابع

https://en.wikipedia.org/wiki/Huffman_coding

https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/

https://en.wikipedia.org/wiki/Canonical_Huffman_code

https://www.geeksforgeeks.org/canonical-huffman-coding/