فشرده‌سازی داده‌های متنی

پروژه پایان‌ترم درس ساختمان داده‌ها و طراحی الگوریتم

دانشکده برق و کامپیوتر

ترم اول تحصیلی ۹۷-۹۶

- پروژه باید به صورت تک نفره انجام شود.
- برنامه نوشته شده توسط هر فرد، تحویل حضوری نیز خواهد داشت.
- تمام ساختمان داده‌های مورد استفاده در این پروژه باید توسط خود شما پیاده سازی شوند.
- هیچ محدودیتی در نوع زبان مورد استفاده وجود ندارد.

۱ مقدمه

فردی قصد دارد یک فایل متنی ،که شامل یک داستان کوتاه به زبان انگلیسی است، را از نقطه s به نقطه d انتقال دهد. تنها راه انتقال این فایل متنی استفاده از یک خط انتقال دیجیتال است. بدین ترتیب محتویات فایل متنی، یا همان کاراکترها، باید به رشته‌ای از اعداد ۰ و ۱ تبدیل شوند تا بتوان آنها را بر روی خط انتقال دیجیتال قرار داد.

یک روش برای تبدیل کاراکترها به رشته‌ای از ۰ و ۱ استفاده از روش کدگذاری اسکی[۱] است. در این روش به هر کاراکتر یک کد ۸ بیتی اختصاص می‌یابد. به طور مثال کد اسکی کاراکتر A برابر با ۰۱۰۰۰۰۰۱ است. با استفاده

[۱] ASCII

از این روش کدگذاری اگر فرض کنیم فایل مورد نظر دارای ۳۰۰۰۰۰ کاراکتر است آنگاه با توجه به اینکه هر کاراکتر نیاز به ۸ بیت دارد حجمی از داده که باید از s به d منتقل شود برابر است با $۳۰۰۰۰۰ \times ۸ = ۲۴۰۰۰۰۰$ بیت.

با در نظر گرفتن یک نکته می‌توان حجم داده انتقالی را تا حد زیادی کاهش داد. در زبان انگلیسی برخی از حروف مانند e و t بسیار پرکاربرد هستند در حالیکه حرفی مانند z در کلمات معدودی مورد استفاده قرار می‌گیرد. بدین ترتیب با توجه به اینکه می‌دانیم برخی از حروف نسبت به برخی دیگر تعداد تکرار بیشتری دارند می‌توانیم با استفاده از یک روش کدگذاری به نام FooCode طوری به هر کاراکتر یک کد اختصاص دهیم که برای کاراکترهایی که بیشتر در یک متن تکرار می‌شوند کدی کوتاه‌تر و برای کاراکترهایی که به ندرت در یک متن تکرار می‌شوند کدی طولانی‌تر اختصاص دهیم. برای درک بهتر این موضوع به مثالی که در ادامه آمده است توجه کنید.

فرض کنید قصد داریم دنباله‌ی کاراکتر زیر را انتقال دهیم:

AAABETRAAAREE

بیشترین کاراکتر به کار رفته در این دنباله کاراکتر A است با ۶ تکرار، سپس کاراکتر E با ۳ تکرار، بعد کاراکتر R با ۲ تکرار و در نهایت کاراکترهای B و T هر یک با ۱ تکرار. اگر از کد اسکی برای کد کردن این دنباله استفاده کنیم آنگاه با توجه به اینکه دارای ۱۳ کاراکتر هستیم و هر کاراکتر نیاز به ۸ بیت دارد در کل باید ۱۰۴ بیت اطلاعات را انتقال دهیم. اما اگر از روش FooCode استفاده کنیم آنگاه به کاراکتر A کد دودویی ۰، به کاراکتر B کد دودویی ۱۱۰۰، به کاراکتر E کد ۱۰، به کاراکتر R کد ۱۱۱ و به کاراکتر T کد ۱۱۰۱ نسبت داده می‌شود. در نتیجه طول کدگذاری حاصل از این روش کدگذاری برابر خواهد بود با:

$$\underbrace{(۶ \times ۱)}_{A} + \underbrace{(۱ \times ۴)}_{B} + \underbrace{(۳ \times ۳)}_{E} + \underbrace{(۲ \times ۳)}_{R} + \underbrace{(۱ \times ۴)}_{T} = ۳۱$$

بدین ترتیب مشاهده می‌شود که مقدار اطلاعاتی که باید منتقل شود را از ۱۰۴ بیت به ۳۱ بیت کاهش یافته است.

۲ نحوه به دست آوردن کدگذاری FooCode

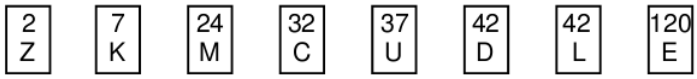
در این بخش به روش به دست آوردن کدگذاری FooCode پرداخته می‌شود. فرض کنید فایلی متنی از ورودی خوانده شده است و تعداد رخداد هر یک از کاراکترهای این فایل در شکل ۱ آمده است.

Letter	C	D	E	K	L	M	U	Z
Frequency	32	42	120	7	42	24	37	2

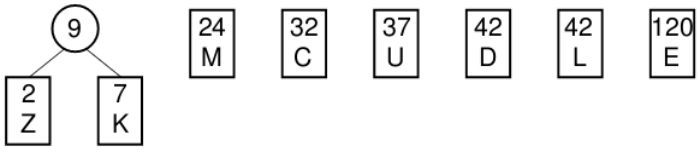
شکل (۱): تعداد رخداد کاراکترها

طبیعتاً عدم وجود کاراکترهای دیگر بدین معنی است که تعداد رخداد این کاراکترها برابر با صفر است.

به دست آوردن کدگذاری FooCode دارای دو گام اساس است. در مرحله اول یک درخت دودویی خاص ساخته می‌شود و در مرحله دوم با توجه به درخت ساخته شده در مرحله اول به هر یک از کاراکترها یک کد دودویی اختصاص می‌یابد.



شکل (۲): گام اول ساخت درخت



شکل (۳): گام دوم ساخت درخت

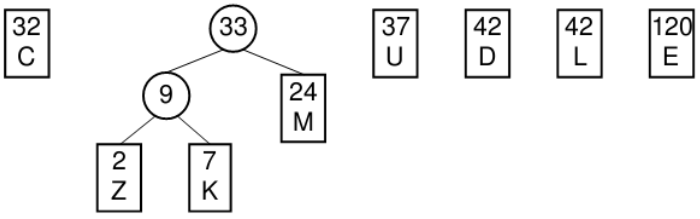
۱.۲ ساخت درخت کاراکترها

در اولین قدم به ازای هر یک از کاراکترها یک گره می‌سازیم به طوری که محتویات هر گره شامل کاراکتر مورد نظر و وزن آن کاراکتر است (منظور از وزن، تعداد رخداد کاراکتر است). به بیان دیگر می‌توان گفت دارای ۸ درخت هستیم که هر یک دارای تنها یک گره است (وزن هر درخت برابر با مقدار عددی موجود در ریشه آن است). سپس تمام درختان ساخته شده را در یک صف اولویت نزولی درج می‌کنیم. به شکل ۲ توجه کنید.

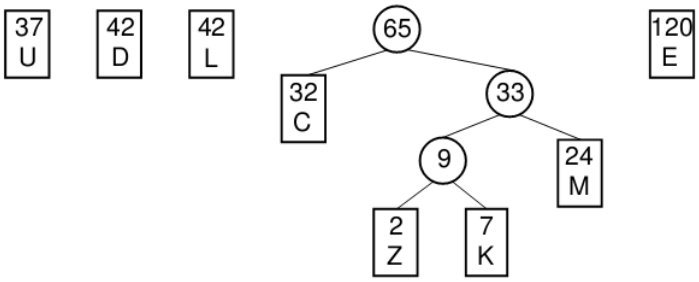
در گام دوم دو درختی که دارای کمترین وزن هستند را از صف اولویت حذف کرده و آنها را با یکدیگر ترکیب می‌کنیم. این دو درخت را t_1 و t_2 می‌نامیم. عمل ترکیب بدین صورت انجام می‌شود که یک گره جدید مانند n می‌سازیم. درخت با وزن کمتر (فرض کنید درخت t_1 دارای وزن کمتر است) را به عنوان زیردرخت چپ گره n و درخت t_2 را به عنوان زیردرخت راست گره n قرار می‌دهیم و مقدار گره n برابر با مجموع وزن ریشه دو درخت t_1 و t_2 قرار داده می‌شود. درخت ساخته شده جدید را در صف اولویت درج می‌کنیم. شکل ۳ صف اولویت پس از انجام مرحله دوم را نشان می‌دهد.

در گام سوم مجدداً دو درختی که دارای کمترین وزن هستند را از صف اولویت حذف کرده و دو درخت را مانند گام قبل با یکدیگر ترکیب می‌کنیم و درخت ساخته شده را در صف اولویت درج می‌کنیم. به شکل ۴ توجه کنید.

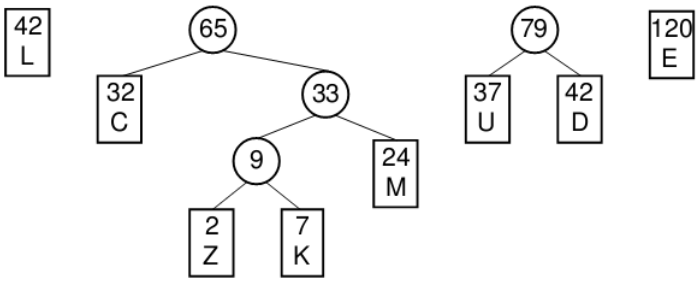
این روند به همین ترتیب ادامه می‌یابد تا در نهایت در صف اولویت تنها یک درخت باقی بماند. شکل ۵ گام چهارم از مراحل ساخت درخت و شکل ۶ گام پنجم را نشان می‌دهد. شکل ۷ نیز درخت نهایی را نشان می‌دهد.



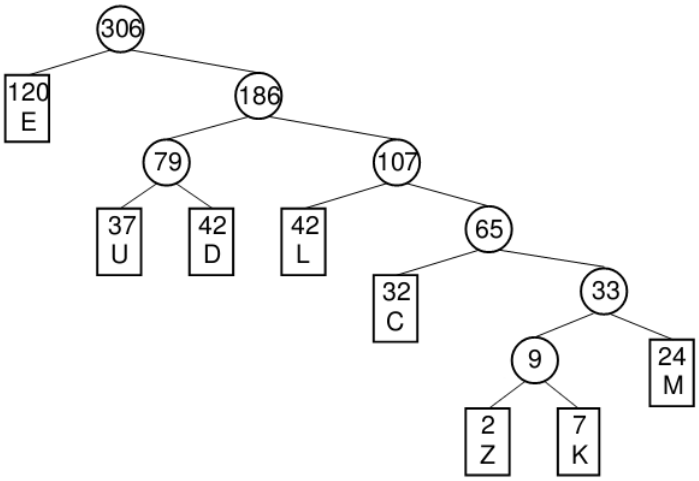
شکل (۴): گام سوم ساخت درخت



شکل (۵): گام چهارم ساخت درخت



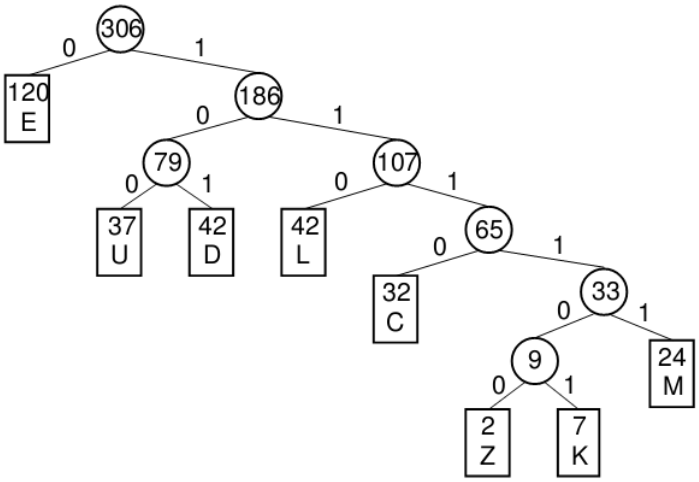
شکل (۶): گام پنجم ساخت درخت



شکل (۷): درخت نهایی

۲.۲ اختصاص کد دودویی به کاراکترها

پس از انجام مرحله اول و ساخت درخت دودویی نوبت به انتساب اعداد صفر و یک به یال‌های درخت و به دست آوردن کد دودویی هر کاراکتر می‌رسد. برای انتساب عدد صفر یا یک به یال‌ها به این صورت عمل می‌کنیم که به یال سمت چپ هر گره عدد صفر و به یال سمت راست هر گره عدد یک را اختصاص می‌دهیم. به این ترتیب درخت به دست آمده در مرحله قبل به درخت موجود در شکل ۸ تبدیل می‌شود.



شکل (۸): درخت حاصل پس از انتساب اعداد صفر و یک به یال‌ها

برای به دست آوردن کد هر کاراکتر از گره ریشه به سمت گره حاوی کاراکتر مورد نظر حرکت کرده و اعدادی که بر روی یال‌های مسیر طی شده وجود دارند را در کنار هم قرار می‌دهیم. بدین ترتیب کد دودویی کاراکتر مورد نظر به دست می‌آید. برای مثال کد دودویی کاراکتر E برابر با ۰، کد دودویی کاراکتر U برابر با ۱۰۰ و کد دودویی کاراکتر M برابر با ۱۱۱۱۱ است. همانطور که انتظار داشتیم به کاراکترهای با تکرار بیشتر کدی کوتاه‌تر و به کاراکترهای با تکرار کمتر کدی طولانی‌تر اختصاص یافت.

۳ شرح پروژه

شما در این پروژه به پیاده‌سازی روش کدگذاری FooCode خواهید پرداخت. توجه داشته باشید که تمام ساختمان داده‌های مورد نیاز را باید خود پیاده‌سازی کرده و مجاز به استفاده از کتابخانه‌های موجود در زبانهای برنامه‌نویسی نیستید.

برای پیاده‌سازی این روش کدگذاری به دو ساختمان داده نیاز است:

- صف اولویت نزولی
- درخت دودویی کاراکترها

از صف اولویت برای نگهداری اشاره‌گر به گره ریشه درختان استفاده خواهد شد. از این صف در مرحله ساخت درخت کاراکترها استفاده می‌شود. دو عمل مورد نیاز برای این ساختمان داده عبارت اند از: حذف و درج. در

صورت نیاز می‌توانید اعمال کمکی دیگری را نیز برای این ساختمان داده تعریف کنید. این صف باید طوری طراحی شود که با انجام عمل حذف از صف، ریشه درختی برگردانده شود که دارای کمترین وزن در بین تمام درختان موجود در صف است.

درخت دودویی کاراکترها نیز درختی است که به تدریج و با ترکیب درختان کوچکتر ساخته می‌شود. به عبارت دیگر درخت دودویی کاراکترها خروجی مرحله اول اجرای الگوریتم خواهد بود. در این درخت دارای دو نوع گره هستیم. یکی گره‌های برگ که حاوی یک کاراکتر و وزن آن هستند و دیگری گره‌های داخلی که فقط شامل وزن هستند.

با توجه به توضیحات فوق می‌توان از تعاریف زیر برای گره‌ها استفاده کرد (گرچه تعاریف زیر به زبان Java هستند اما به راحتی قابل تبدیل به زبان C++ یا هر زبان شیء گرای دیگری خواهند بود).

```
1  // Base class
2  public interface BaseNode{
3          public boolean isLeaf();
4          public int getWeight();
5  }
6  // Leaf node
7  class LeafNode implements BaseNode{
8          private char element; // Element for this node
9          private int weight; // Weight for this node
10         // constructor
11         public LeafNode( char e, int w){
12                 element = e;
13                 weight = w;
14         }
15         public char getElement(){
16                 return element;
17         }
18         public int getWeight(){
19                 return weight;
20         }
21         public boolean isLeaf(){
22                 return true;
23         }
24 } // end class LeafNode
25 // internal node
26 class InternalNode implements BaseNode{
27         private int weight; // Weight (sum of children)
28         private BaseNode leftChild; // Pointer to left child
29         private BaseNode rightChild; // Pointer to right child
```

```
30          // constructor
31          public InternalNode( BaseNode lc, BaseNode rc, int w )
32          {
33                  leftChild = lc;
34                  rightChild = rc;
35                  weight = w;
36          }
37          public BaseNode getLeftChild(){
38                  return leftChild;
39          }
40          public BaseNode getRightChild(){
41                  return rightChild;
42          }
43          public int getWeight(){
44                  return weight;
45          }
46          public boolean isLeaf(){
47                  return false;
48          }
49  } // end class InternalNode
```

صف اولویت شما باید بتواند هم گره‌های از نوع LeafNode و هم گره‌های از نوع InternalNode را در خود ذخیره کنید. با توجه به اینکه دو نوع گره مذکور از کلاس BaseNode ارث‌بری دارند پس باید نوع عناصر صف اولویت از نوع BaseNode تعریف شود.

توجه کنید که لزومی ندارد در برنامه فوق حتماً از کلاس‌هایی با تعاریف فوق استفاده کنید. تعاریف فوق صرفاً به عنوان یک پیشنهاد برای پیاده‌سازی آورده شده است.

۴ ورودی و خروجی برنامه

برنامه شما به عنوان ورودی باید مسیر یک فایل متنی را دریافت کند. برای مثال پیغام Enter file name: نمایش داده شده و کاربر مسیر فایل متنی را وارد می‌کند.

Enter file name: `C:\sample.txt`

چنین فایلی تنها شامل حروف بزرگ انگلیسی و فاقد کاراکتر فاصله (space) است. برای مثال محتویات یک فایل متنی نمونه می‌تواند به صورت زیر باشد:

AAABETRAAAREE

خروجی برنامه شما باید به صورت زیر باشد:

$\mathrm{A} = 0$
$\mathrm{B} = 1100$
$\mathrm{E} = 10$
$\mathrm{R} = 111$
$\mathrm{T} = 1101$

به بیان دیگر خروجی برنامه شما نشان می‌دهد که به هر کاراکتر موجود در فایل ورودی چه کدی اختصاص یافته است.

موفق باشید