2019-2020 Bahar Yarıyılı BLM2512 Veri Yapıları ve Algoritmalar 2. Ödevi

Mesut Şafak Bilici
17011086

Bu ödevde bir *Linked List* yapısı kurup, onu *Inserton Sort* ile sıralayıp, *Huffman Tree* yapısına çevirip Level Order bir şekilde bastırmam gerekiyordu. Bu işlemleri yaptığım fonksiyonlar aşağıdaki gösterilmiştir:

1. void insertLinkedList(NODE**,char);
    - İlk olarak input dosyamdaki harfler için frekans bazlı bir linked linked list oluşturmam gerekiyordu. Buradaki önemli kontrol harfleri teker teker yerleştirirken ve tüm linked list'i kontrol ederken, while geldiğinde eğer en sondaysam, ya next NULL olabilir ya da son eleman input char ile aynıdır.
2. void printLinkedList(NODE**);
    - Basit bir şekilde, current isimli geçici değişkenimle linked list'in sonuna giderken bastırıyoruz. Program içinde kullanıldığı bir yer yok fakat kontrol amaçlı yazıldı.
3. void insertionSortToLinkedList(NODE**);
    - Burada Huffman Tree oluşturmadan önce son işlemimizi yapıyoruz: Linked List'i sıralamak. Verilen ödevlendirme üzerinde kullanılan algoritma Insertion Sort olmuştur. Bilindiği gibi Insertion Sort'ta kontrol işlemleri için iki farklı indis değişkeni, ve bu indis değişkenleri ile iki farklı array bölgesi tutuyoruz. Bu değerleri karşılaştırarak

da sıralamamızı yapıyoruz. Fakat elimizde indis tutma durumu olmadığı için bu işe yarayan değişkenlerimiz *start1* ve *start2* olarak belirlenmiştir. Tıpki basit bir array sıralama işlemi yapıyormuş örneği verilecek olursa, i 1'den başlayacak şekilde start1 = arr[i] ve j=i-1 olacak şekilde start2 = arr[j] olmuştur. En başta belirlenen temp-of-the-temps isimli değişken her bir node'un letter bilgisini swap etmek için tanımlanmıştır. Kullanılan mantık basit bir array işlemi ile aynı fakat elimizde array indexing yerine linked list vardır.

4. void readFromFile(NODE**,char*);

   - fgetc(FILE*) fonksiyonu ile teker teker karakterler okunmuştur. Fakat dosyanın sonuna geldiğimizde new line okuduğu için her zaman içerde buffer'ın new line karakterine eşit olup olmama kontrolü yapıldı.

5. void readFromStdin(NODE**);

   - Elle klavyeden girdi yapmak için fonksiyon. EOF sinyali gelene kadar okuma yapmaya devam eder. (EOF = Ctrl+d)

6. void insertButSorted(NODE**,NODE*);

   - Bu fonksiyon özel olarak Huffman Ağacını oluştururken toplanmış frekanslara sahip olan node'ların yerini belirleyip oraya insert etmek için yazılmıştır. Eğer direkt head'in frekansından düşükse head'in yerine geçer bir ilerisi eski head olur. Eğer ki öyle değilse önceden linked list sıralandığı için büyük olana kadar frekans, linked list içerisinde ilerlenir ve sonrasında olması gereken yere insert edilir.

7. void doThatTree(NODE**);

   - Bu fonksiyonda artık Huffman Tree'yi tam olarak gerçekliyoruz. Elimizdeki linked list doubly linked list olmadığı için bir önceki node'a ulaşmamız biraz güçleşiyor. Bunu engellemek için iki farklı current değişkeni tanımladım. While içinde linked listin sonuna giderek ağacımızı gerçekledim. İlk olarak yeni oluşacak node için dinamik bellek alokasyonu yaptım. Sonra bu newnode'un frekans bilgisine linked listin en küçük iki elemanının frekans bilgisini toplamını atadım. Daha sonra bu newnode'un left ve

right bilgilerine bu iki node'u atadım. Bu işlemi linked list'in sonuna kadar gerçekleştirdim. İstenilen input'un çıktısında boşluk karakteri olduğu için toplam frekansların letter bilgisine * char'ını girdim. Bu şekilde boşluğun konumunu görebileceksiniz.

8. void printSpecificLevel(NODE*,int);

   - Bu fonksiyon verilen level'deki nodeları bastırıyor teker teker. Recursive bir fonksiyon. Eğer girilen NODE* parametresi NULL ise çıkıyoruz fonksiyondan. Eğer NODE* parametresi tree'nin root'u ise bunu kontrol edip root'un frekansını ve letter'ını (root'un letter'ı yok fakat karmaşa olmasın diye *'e eşitledik) bastırıyoruz. Eğer bunlardan hiçbirisi ise recursive bir şekilde leveli azaltarak node'larımızın left ve right bilgilerini basıyoruz .

9. void printThatTree(NODE*);

   - Bu fonskiyonda spesifik level'i bastırdığımız fonksiyonu ve ağacın level sayısını hesapladığımız fonksiyonu birlikte kullanıyoruz. Ağacın uzunluğunu bulup, for döngüsü içinde ve sınır bu uzunluk olacak şekilde teker teker ağacın levellerini bastırıyoruz.

10. int getLevelCount(NODE*);

    - Ağacımızın uzunluğunu / level sayısını hesaplıyoruz. Recursive bir fonksiyon.

Main:

Main fonksiyonu içinde teker teker yukarıda belirtilen fonksiyonlar çağrılmıştır istenilenin yapılması için. Öncesinde kullanıcıya elle input mu gireceksin yoksa file'dan mı okuyacaksın sorusu sorulur. File ise file'ın ismini sorar.

İstenilen çıktı aşağıdaki gibidir:

```
(base) safak@progcu:/media/safak/Data/School/BLM2512 Homework2$ make all
gcc 17011086.c -o 17011086
(base) safak@progcu:/media/safak/Data/School/BLM2512 Homework2$ make exe
./17011086
If you read the text from file PRESS 1, if you read the text from stdin PRESS 0:  1
What is your file name?: v1.txt
46*
18* 28*
8* 10* 12* 16*
4* 4* 5a 5* 6  6* 8* 8*
2g 2t 2r 2* 2* 3m 3n 3s 4o 4i 4* 4*
1u 1p 1e 1l 2h 2f 2c 2d
```