2017-2018 Güz Yarıyılı
Algoritma Analizi 2. Ödevi

Konu : Dinamik Programlama

Problem: Bu ödevde harf yazım kontrolü (spell checker) işlemi yapan bir sistem tasarlanacaktır. Sistem, verilen bir kelimenin **sözlükte** olup olmadığını kontrol edecek, kelimeyi sözlükte bulamazsa sözlükte bulunan kelimelerden doğru kelime önerilerini verecektir.

Ödev iki ana bölümden oluşmaktadır :

1. **Sözlüğe hızlı erişim için yapılacak ön işlemler :** Doğru yazılmış kelimelerin saklandığı sözlük dosyasına hızlı erişmek için kelimeleri sözlükte ***hashing*** yöntemi ile saklayınız. Bunun için, aşağıdaki işlemleri yapınız:
   1) Bu ödevde sözlük olarak CMU sözlüğü **(http://svn.code.sf.net/p/cmusphinx/code/trunk/cmudict/cmudict-0.7b)** kullanılacaktır. **Sözlük A harfi ile başlamaktadır.** Dokümanın başındaki satırları siliniz. satırlarını siliniz. Sözlükte her satırda ilk kelimeden sonraki kelime parçaları, kelimenin okunuşunu göstermektedir. Siz sadece her satırdaki **ilk kelime** ile ilgileniniz. Satırdaki diğer kelimeleri elle veya programla silebilirsiniz.
   2) Kelimeleri saklamak için **sözlükteki kelimesayısı*2** uzunluklu bir diziye ihtiyacınız var. Bunun için kelimesayısı*2 değerine **en yakın asal sayıyı(m)** elle(herhangi bir program yazmanıza gerek yok) belirleyiniz. Bu şekilde, tamamen NULL ile ilklendirilmiş **dictionary[m]** dizisini tanımlayınız.
   3) Sözlükteki büyük harfleri küçük harfe çeviriniz.
   4) CMU sözlüğündeki her kelimenin sizin hazırladığınız *dictionary* dizisinde hangi adrese yerleşeceğini aşağıdaki şekilde hesaplayınız:

      ```
      for(i=addr=0; i < length(word); i++)
            addr  +=  (word[i] – 'a' )*26*i ;
      addr = addr % m;
      ```

      Eğer hesapladığınız *addr* değeri için *dictionary[addr]* gözünde başka bir kelime varsa boş göz bulana kadar aşağıdaki işlemi yaparak kelimenin yerleşeceği adresi belirleyiniz:

      ```
      addr  =  (addr++) % m;
      ```

   5) Dictionary dizisini programı her çalıştırışınızda yeniden oluşturmak gereksiz zaman alıcı bir işlemdir. Dictionary dizisini bir kere oluşturup, gerektikçe kullanmak için bir dosyaya yazınız. Kullanırken ise, dosya erişimi işlemlerinden ötürü programınızın yavaş çalışmaması için *dictionary* dizisini önce dosyadan belleğe alıp sonra kullanınız.

2. **Sözlükte bulunmayan kelime yerine benzer kelime önerme :** Sözlükte bulunmayan kelime yerine kelime önerme işlemi için **Edit Distance(Levenshtein) yöntemi** ile kelimenizin sözlükteki kelime ile benzerliğini ölçünüz. Bunun için aşağıdaki işlemleri yapınız.
   1) Kelimenin doğru yazılıp yazılmadığını anlamak için **önce** *dictionary* dizisinde olup olmadığına bakınız. Önce kelimedeki büyük harfleri küçük harfe çeviriniz. Kelimenin *dictionary* dizisinde olabileceği adresi yine yukarıda kullandığınız bağıntı ile hesaplayınız:

      ```
      for(i=addr=0; i < length(word); i++)
            addr  +=  (word[i] – 'a' )*26*i ;
      addr = addr % m;
      ```

      Eğer kelime bu adreste yoksa, **addr = (addr++) % m;** bağıntısı ile en fazla **dictionary dizisinin uzunluğu kadar** adımda aramaya devam ediniz. Kelime dizide yok ise yanlış

yazılmış demektir. Bu durumda, edit distance ile benzer kelimeleri arama işlemine geçiniz.

2) Kelime sözlükte bulunamadı ise sözlükteki **bütün kelimelere benzerliğini edit distance(Levenhstein)** yöntemi ile ölçünüz. Sözlükte bu kelimeye **en çok benzeyen kelimeyi(**birden fazla seçenek varsa birini vermeniz yeterli**)** öneriniz.
3) Farklılıklar değiştirme(change), araya ekleme(insert) ve silme(delete) olabilir. Değiştirme için ceza puanını 2, ekleme ve silme işlemleri için ise 1 olarak alınız.

Giriş/Çıkış işlemleri : Giriş/Çıkış işlemlerini aşağıdaki gibi yapınız.

1. Giriş bilgisini dosyadan okuyunuz. Dosyadaki kelimelerin boşluklarla ayrıldığını kabul ediniz.
2. Çıkış bilgisini de bir dosyaya yazınız. Her satırda **ilk kelime** giriş dosyasındaki kelime, **ikinci kelime** doğru için OK, yanlış fakat benzer kelime bulunamadı ise NONE, yanlış ve benzer kelime bulundu ise benzer kelime ve **üçüncü olarak** da iki kelime arasındaki değişimleri gösteriniz.

   Örnek : İlk örnekde *tah* kelimesinin doğru hali *that* ve tah'dan that'e dönüşüm *h* harfinin *t*'den sonra insert edilmesi ve *h* harfinin *t* harfi olarak değişmesi ile elde edilmiştir. İkinci örnekde happy kelimesi sözlükte olan bir kelimedir. 3. örnekdeki xxyywwwzzzz kelimesi ise sözlükte yoktur.

   tah that t h(insert) a t(change)
   happy OK
   xxyywwwzzzz NONE

Not : Sözlük çok büyük olduğu için benzer bulma işlemi dakika mertebesinde sürebilir. Olabildiğince efektif kod yazmaya özen gösteriniz.

Teslim Edilecekler: Aşağıdaki bilgileri içeren dokümanları teslim ediniz.

1. Ekteki test.txt dosyasından seçeceğiniz 10 örnek kelime için elde ettiğiniz sonuçları ekran çıktısı olarak gösteriniz.
2. test.txt dosyası için testout.txt dosyasını çıkış olarak oluşturunuz ve teslim ediniz. test.txt dosyası **http://aspell.net/test/cur/** adresindeki http://aspell.net/test/cur/batch0.tab dosyasından elde edilmiştir. O dosyada kelimelerin doğru yazılmış hallerini de görebilirsiniz.
3. Algoritmanızın **C dilinde** yazılmış programını veriniz.

Önemli : Algoritmanızı kendiniz tasarlayınız. İstenilen algoritmaların hazır kodunu internetten bulabilirsiniz. Fakat tasarımı kendiniz yaparsanız hem daha iyi öğrenirsiniz hem de edindiğiniz tecrübe öğrendiklerinizin kalıcı olmasını sağlar. Ayrıca internette bulunan hazır bir koda belli bir eşik seviyesinden fazla benzeyen kodlar **kopya** olarak değerlendirilecektir ☹ .

Teslim İşlemleri:
Ödevler **30 Ekim 2017 tarihinde** yapılacak laboratuvarda gösterilecektir. **Sunum ile ilgili açıklamalar için** Arş. Grv. Zeynep Banu Özger'in sayfasını takip ediniz.

Değerlendirme:Ödeviniz aşağıdaki gibi değerlendirilecektir:

Algoritma Tasarımı ve Programın Çalışması: (%60)

1. Ödev, istenilen işlerin tamamını yerine getirmelidir.
2. Gereksiz kontrollerden ve işlemlerden arınmış bir tasarım yapılmalıdır.
3. Programda gerekli alt modüller belirlenerek her modül ayrı fonksiyon olarak yazılmalıdır.

4. Program hatasız çalışmalıdır.
5. Programın çalışması sırasında, konuyu bilmeyen kişilerin rahatlıkla anlayabilmesi için, giriş ve çıkışlarda mesajlarla bilgi verilmelidir.

Rapor Dokümantasyonu: (%40)

1. Raporun ilk sayfasında, dersin adı, öğrencinin ad, soyad ve numarası, ödev konusu bilgileri yer almalıdır.
2. Kaynak kodda değişken deklerasyonu yapılırken her değişken tek satırda tanımlanmalı, tanımın yanına değişkenin ne için kullanılacağı açıklama olarak yazılmalıdır.
3. Değişken ve fonksiyon(veya metod) isimleri anlamlı olmalıdır.
4. Her fonksiyonun (veya metodun) yaptığı iş, parametreleri ve dönüş değeri açıklanmalıdır.
5. Gerekli yerlerde açıklama satırları ile kodda yapılan işlemler açıklanmadır.
6. Gereksiz kod tekrarı olmamalıdır.
7. Kaynak kodun formatı düzgün olmalıdır.