TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ALGORİTMA ANALİZİ 2.ÖDEV :DİNAMİK PROGRAMLAMA

17011604 — Havvanur Dervişoğlu
—

ALGORİTMA ANALİZİ ÖDEVİ RAPORU

Danışman
Dr.Öğr.ÜyesiM. Amaç Güvensan

Aralık, 2018

İÇİNDEKİLER

1
YÖNTEM

1.1 Problem Nedir?

Tanıtım sloganımız var bu sloganı en fazla M karakter uzunluklu olacak şekilde satırlara yerleştirmek istiyoruz.Bu sloganımızdaki toplam kelime sayısını N ile ifade ederken, çözüm yolu kısaca aşağıdaki gibidir:

Kelime uzunlukları I1,I2,...In olsun ,yani sloganımızdaki her bir kelimenin uzunluğu, mesela; 1.kelimenin uzunkluğu I1, 2.kelimenin uzunluğu I2 gibi.

Her kelimeyi satırlara yerleştirirken aralarında birer boşluk olacak şekilde yerleştirmeliyiz.

1. Bir satıra i. kelimeden j.kelimeye yerleştirdiğimizde satırın sonuna kalan boşluk sayısı, yani satırda i. ve j. kelime varken ki geriye kalan boşluk sayısı : space[i][j]=M-j+i-$\sum_{k=i}^{j} Ik$
2. Bir satıra i. kelimeden j.kelimeye yerleştirdiğimizde bu satırdaki boşluk bedeli: spaceCost[i][j]=$space[i,j]^3$

SpaceCost matrisini oluştururken bazı şeyleri dikkat etmemiz gerekiyor:

- eğer space[i,j]<0 ise spaceCost=INF
- eğer space[i,j]=0 ise spaceCost=0
- eğer space[i,j]>0 ise $space[i,j]^3$
- son satırın boşluk değeri 0'dır

Peki bu kelimeleri satırlara yerleştirme işlemini nasıl yapacağız:

1. kelimeden j.kelimeye kadar satırlara yerleştirdiğimizde bunu gösteren minimum değer cost[j]'dir.Örnek vermek gerekirse; cost[3] 1.kelimeden 3. kelimeye kadar yerleştirdiğimizde oluşan minumum cost değeridir.

cost[j]=0 eğer j=0
cost[j]=$min_{1<=i<=j}$(cost[i-1]+spaceCost[i,j] eğer j>0

Bu cost değerini ,bir satırın hangi kelime ile başalaması durumunda bedelin en düşük olacağını belirlemek için kullanıyoruz.

Ornek **:** 3 tane satır var diyelim.

1. bir satır 3 tane space'e sahip , diğerlerinde ise 0 tane space var:
   spaceCost= 3^3 +0^3 +0^3 = 27
2. her 3 satırdada 1 tane space var :
   spaceCost= 1^3 +1^3 +1^3 = 3

->2.seneryo seçilir çünkü bedelin düşük olduğu durumdur.

"lineAdr" dizisi ise bir satıra hangi kelime ile başlayacağımız bilgisini tutar.Bu diziyi cost değerine göre oluşturacağız.

1.2 Neden Dinamik Programlama?

Bu problemi dinamik programlama yöntemiyle çözdük çünkü; cost[5] değeri için cost[4],cost[3] gibi önceki değerlere ihtiyacamız olduğu gibi cost[4] değeri içinde cost[3] değeri lazım yani aynı "n" değerleri için cost değerleri tekrar tekrar hesaplamak gerekecek bunu önlemek için dinamik programlama yöntemi ile bir matriste önceki değeri kaydedip gerektiğinde oradan alıp kullanabiliriz.

Problem **:**Aynı n değerleri için tekrar tekrar cost değerinin hesaplanması gereksinimi.
Yontem **:**Dinamik programlama kullanarak bir önceki bilgiyi gerektiğinde sakladığı yerden alarak kullanmak.

2
UYGULAMA

2.1 Input Açıklaması

"The DEMO is below my."
Bu örnek için N=5 ve M=15.

Uygulama aşamasında ilk olarak cümleyi kelime kelime ayırdım ve sonrasında her bir kelimenin uzunluğunu hesapladım:

Kelimelerin uzunluğu:
I1=3,I2=4,I3=5,I4=2 ve I5=2'dir.

2.2 Space Matrisi

space[i][j]=M-j+i-$\sum_{k=i}^{j} Ik$

```
Space matrisi:
7       2       -4      -7      -10
-       6       0       -3      -6
-       -       5       2       -1
-       -       -       8       5
-       -       -       -       8
```

M=10 , lenghtOfWords=3,4,5,2,2

- $i = 1$:
  space[1][1]=10-1+1-3=10-3=7
  space[1][2]=10-2+1-(3+4)=space[1][1]-4-1=2
  space[1][3]=10-3+1-(3+4+5)=space[1][2]-5-1=-4
  space[1][4]=10-4+1-(3+4+5+2)=space[1][3]-2-1=-7
  space[1][5]=10-5+1-(3+4+5+2+2)=space[1][4]-2-1=-10

- $i = 2$:
  space[2][2]=10-2+2-(4)=10-4=6
  space[2][3]=10-3+2-(4+5)=space[2][2]-5-1=0
  space[2][4]=10-4+2-(4+5+2)=space[2][3]-2-1=-3
  space[2][5]=10-5+2-(4+5+2+2)=space[2][4]-2-1=-6

- $i = 3$:
  space[3][3]=10-3+3-(5)=10-5=5
  space[3][4]=10-4+3-(5+2)=space[3][3]-2-1=2
  space[3][5]=10-5+3-(5+2+2)=space[3][4]-2-1=-1

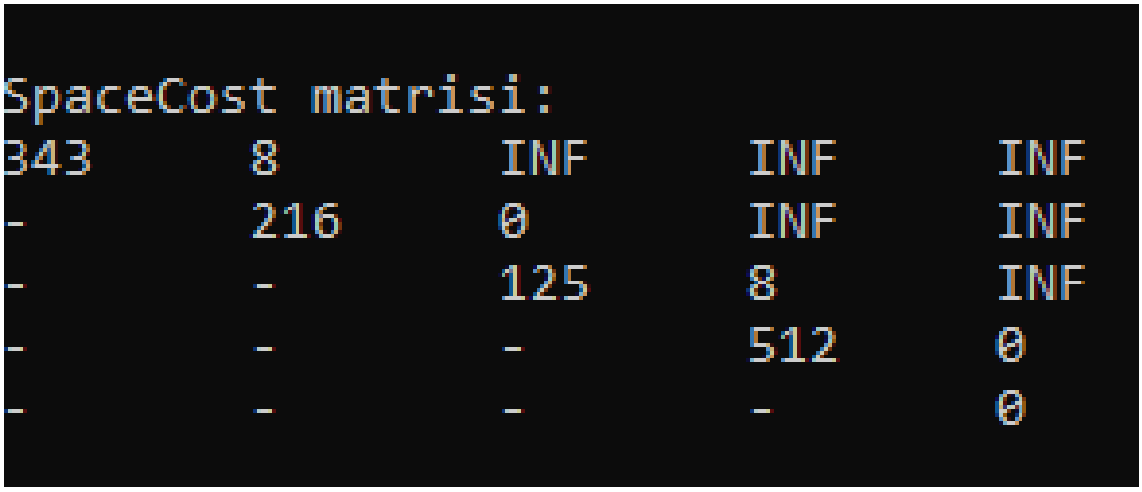
- $i = 4$:
  space[4][4]=10-4+4-(2)=10-2=8
  space[4][5]=10-5+4-(2+2)=space[4][4]-2-1=5

- $i = 5$:
  space[5][5]=10-5+5-(2)=10-2=8

2.3 SpaceCost Matrisi

SpaceCost matrisini oluştururken bazı şeyleri dikkat etmemiz gerekiyor:

- eğer space[i,j] < 0 ise spaceCost=INF
- eğer space[i,j] $= 0$ ise spaceCost=0
- eğer space[i,j] > 0 ise $space[i,j]^3$
- son satırın boşluk değeri 0'dır



- $i = 1$:
  spaceCost[1][1]=$space[1,1]^3$=7^3=343
  spaceCost[1][2]=$space[1,2]^3$=2^3=8
  spaceCost[1][3]=INF çünkü; space[1][3] < 0
  spaceCost[1][4]=INF çünkü; space[1][4] < 0
  spaceCost[1][5]=INF çünkü; space[1][5] < 0

- $i = 2$:
  spaceCost[2][2]=$space[2,2]^3$=6^3=216
  spaceCost[2][3]=0 çünkü; space[2][3] $= 0$
  spaceCost[2][4]=INF çünkü; space[2][4] < 0
  spaceCost[2][5]=INF çünkü; space[2][5] < 0

- $i = 3$:
  spaceCost[3][3]=$space[3,3]^3$=5^3=125
  spaceCost[3][4]=$space[3,4]^3$=2^3=8
  spaceCost[3][5]=INF çünkü; space[3][5] < 0

- $i = 4$:
  spaceCost[4][4]=$space[4,4]^3$=8^3=512
  spaceCost[4][5]=0 çünkü; son satırın boşluk değeri 0

- $i = 5$:
  spaceCost[5][5]=0 çünkü; son satırın boşluk değeri 0

2.4 Cost ve LineAdr Dizileri

Cost değerini ,bir satırın hangi kelime ile başalaması durumunda bedelin en düşük olacağını belirlemek için kullanıyoruz.

cost[j]=0 eğer j=0
cost[j]=$min_{1<=i<=j}$(cost[i-1]+spaceCost[i,j] eğer j>0

"lineAdr" dizisi ise bir satıra hangi kelime ile başlayacağımız bilgisini tutar.Bu diziyi cost değerine göre oluşturacağız.

```
Cost dizisi:
0       343     8       133     16      16
```

```
LineAdr dizisi:
1       1       3       3       5
```

cost[0]=0

- $j = 1$: 1.kelimenin satırda olduğu durumdaki cost değeri:
  cost[1]=INF ;if koşulunda karşılaştırma için gerekli.

  => i=1 için;
  eğer spaceCost[1][1] != INF(satıra sığmıyorsa alma) ve cost[1] > cost[0]+spaceCost[1][1](önceki cost[1] dağerinden küçük olanı almak için):
  İki koşulu da sağlıyor o zaman ;
  cost[1]=cost[0]+spaceCost[1][1]=0+343=343
  lineAdr[1]=1

- $j = 2$: 1. kelimeden 2.kelimeye minimum cost değerinin bulunması :
  cost[2]=INF

  => i=1 için; (bu durumda bir ve ikinci kelimeler aynı satırda olacak yani satır 1.kelime ile başlayacak ve 2.kelime ile bitecek)
  eğer spaceCost[1][2] != INF ve cost[2] > cost[0]+spaceCost[1][2]
  cost[2]=cost[0]+spaceCost[1][2]=0+8=8
  lineAdr[1]=1

  => i=2 için; (bu durumda birinci kelimeye kadar olan minimum cost ile yeni satır 2. kelime ile başlayacak durumudur)
  eğer spaceCost[1][2] != INF ve cost[2] > cost[1]+spaceCost[2][2]
  burada ikinci koşul sağlanmıyor.
  cost[1]+spaceCost[2][2]=343+216 > 8

- $j = 3$: 1. kelimeden 3.kelimeye minimum cost değerinin bulunması :
  cost[3]=INF

  => i=1 için; (satır 1.kelime ile başlayacak ve 3.kelime ile bitecek)
  spaceCost[1][3] = INF bu yüzden almıyoruz.

  => i=2 için; (birinci kelime yerleşti , 2. ve 3. kelimeler aynı satırda durumu)
  cost[3]=cost[1]+spaceCost[2][3]=343+8=351

=> i=3 için; (1. ve 2. kelimeler yerleşmiş , yeni satır 3. kelime ile başlaması durumu)
cost[3]=cost[2]+spaceCost[3][3]=8+125=133 (öncekinden küçük o yüzden bunu alıyoruz)
lineAdr[3]=3

- $j = 4$ **:** 1. kelimeden 4.kelimeye minimum cost değerinin bulunması :
cost[4]=INF

  => i=1 için; (satır 1.kelime ile başlayacak ve 4.kelime ile bitecek)
  spaceCost[1][4] = INF bu yüzden almıyoruz.

  => i=2 için; (birinci kelime yerleşti , 2.'den 4'e kelimeler aynı satırda durumu)
  spaceCost[2][4] = INF bu yüzden almıyoruz.

  => i=3 için; (1. ve 2. kelimeler yerleşmiş , yeni satır 3. kelime ile başlaması durumu)
  cost[4]=cost[2]+spaceCost[3][4]=8+8=16
  lineAdr[4]=3

  => i=4 için;
  cost[3]+spaceCost[4][4]=133+512 > 3.kelime ile başlama durumundan cost[3] , o yüzden bu durumu almıyoruz.

- $j = 5$ **:** 1. kelimeden 5.kelimeye minimum cost değerinin bulunması :
cost[5]=INF

  => i=1 için; (satır 1.kelime ile başlayacak ve 5.kelime ile bitecek)
  spaceCost[1][5] = INF bu yüzden almıyoruz.

  => i=2 için; (birinci kelime yerleşti , 2.'den 5'e kelimeler aynı satırda durumu)
  spaceCost[2][5] = INF bu yüzden almıyoruz.

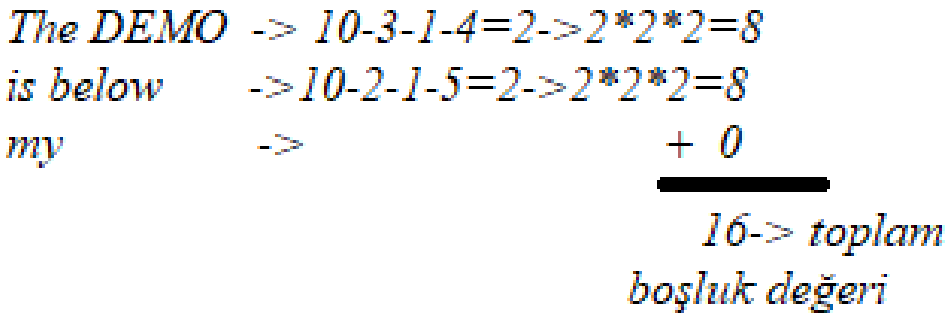
  => i=3 için;
  spaceCost[3][5] = INF bu yüzden almıyoruz.

  => i=4 için;
  cost[5]=cost[3]+spaceCost[4][5]=133+0=133

=> i=5 için;
cost[5]=cost[4]+spaceCost[5][5]=16+0=16
lineAdr[5]=5

2.5 Yazının Yazılması ve Toplam Boşluk Değeri

Toplam boşluk değeri bu örnekte "16" dır , yani cost[5]'in değeridir.1. kelimeden 5. keliemeye kadar satırlara yerleştirdiğimizde ortaya çıkan cost'tur.



3
SONUÇ

Bir text metnindeki kelimeleri satırlara ,minimum boşluk bedeli oluşacak şekilde ,her satır maksimum W karakter olacak, yerleştirdiğimizde oluşan yeni kelime düzenini ve oluşan toplam boşluk bedelini elde etmeye çalıştık. Bu problemi dinamik programlama yöntemiyle çözdük çünkü; aynı "n" değerleri için cost değerleri tekrar tekrar hesaplamak gerekiyordu bunu önlemek için dinamik programlama yöntemi ile kaydedilen önceki bilgiyi gerektiğinde saklandığı yerden alıp kullandık.

Bu şekilde zaman ve yerden kazandık.

Bu problemde kullandığımız yolun karmaşıklığı O(n^2)'dir.N burada kullanılan cümledeki kelime miktarıdır.

```
Stringteki kelimelerin uzunlugu:
 3        4        5        2        2
stringteki toplam kelime sayisi=5
Stringteki kelimeler:
 The
 demo
 below
 is
 my

Space matrisi:
7        2        -4       -7       -10
-        6        0        -3       -6
-        -        5        2        -1
-        -        -        8        5
-        -        -        -        8

SpaceCost matrisi:
343      8        INF      INF      INF
-        216      0        INF      INF
-        -        125      8        INF
-        -        -        512      0
-        -        -        -        0

Cost dizisi:
0        343      8        133      16       16

Toplam bosluk bedeli: 16

LineAdr dizisi:
1        1        3        3        5

Sonuc:

The demo
below is
my
```

A
KOD

```
//problem : dinamik programlamayla verilen slogandaki kelimeleri her satir en fazla
M karakter olacak sekilde satirlara yerlestirmek
//17011604-havanur dervioglu
include<stdio.h>
include <limits.h>
include<stdlib.h>
include<string.h>
define INF INT_MAX
define SIZE 100
//len kelimelerin uzunluklarinin tutuldugu dizi
//n kac tane kelime var stringte onun sayisi
//m ise her satirda en fazla kaç karakter olabilir onun sayisi
int yazdir(int lineAdr[],int n);
char *words[SIZE];
void DP(int len[],int n,int m)
{
//space bir satirda i den j ye kelime yazinca -aralarinda bir boslukla- kalan bosluk
sayisi
//spaceCost bir satirda i den j ye kelime yazinca bu satirdaki bosluk bedeli
//cost ise her satirin hangi kelime ile baslamasi durumda minimum bedel
int space[n+1][n+1],spaceCost[n+1][n+1],cost[n+1];
int lineAdr[n+1];//her satir kacinci kelime ile baslayacak onun tutuldugu dizi
int i,j;//input stringte ki kelimelerin indisi


//satir sonuna kalan bosluk hesabi
for(i=1;i<=n;i++)
{
space[i][i]=m-len[i-1];
```

```
for(j=i+1;j<=n;j++)
space[i][j]=space[i][j-1]-len[j-1]-1;
}
printf("Space matrisi:");
for(i=1;i<=n;i++)
{
j=0;
while(j<i-1) {
j++;
printf("-");
}
for(j=i;j<=n;j++)
{
printf("%d",space[i][j]);
}
printf("");
}
//spaceCost matrisi
for(i=1;i<=n;i++)
{
for(j=i;j<=n;j++)
{
if(space[i][j]<0)
spaceCost[i][j]=INF;
else if (j == n  space[i][j] >= 0)
spaceCost[i][j] = 0;
else
spaceCost[i][j]=space[i][j]*space[i][j]*space[i][j];
}
}
printf("matrisi:");
for(i=1;i<=n;i++)
{
j=0;
while(j<i-1) {
j++;
printf("-");
}
for(j=i;j<=n;j++)
```

```
{
if(spaceCost[i][j]==INF) printf("INF");
else printf("%d",spaceCost[i][j]);
}
printf("");
}
//cost degerinin hesaplanmasi
cost[0]=0;
for(j=1;j<=n;j++)
{
cost[j]=INF;
for(i=1;i<=j;i++)
{
if(cost[i-1]!=INF  spaceCost[i][j]!=INF
cost[j]>cost[i-1]+spaceCost[i][j])
cost[j]=cost[i-1]+spaceCost[i][j];
lineAdr[j]=i;
}
}

}


printf("dizisi:");
for(i=0;i<=n;i++) printf("%d",cost[i]);
printf("bosluk bedeli:%d",cost[n]);
printf("dizisi:");
for(i=1;i<=n;i++) printf("%d",lineAdr[i]);
printf("");
printf(":");
yazdir(lineAdr,n);


} int yazdir(int lineAdr[],int n)
{
int line,i;
if(lineAdr[n]==1)
line=1;
else
```

```
line=yazdir(lineAdr,lineAdr[n]-1)+1;
for(i=lineAdr[n]-1;i<n;i++) printf("%s ",words[i]);
printf("");
return line;
}
void main()
{
int len[SIZE];
char str[] = "The demo below is my";
int n=0,i,m=10,count=0;
//ilk kelime
char *kelime ;
kelime = strtok(str, ",;. ");


//str bitene kadar kelimeleri ayikla
while (kelime != NULL)
{
words[n]=kelime;
len[n] = strlen(kelime);
n++;
kelime = strtok(NULL, " ");
}
printf("Stringteki kelimelerin uzunlugu:");
for( i=0;i<n;i++) {
printf(" %d",len[i]);
}
printf("toplam kelime sayisi=%d",n);
printf("Stringteki kelimeler:");
for( i=0;i<n;i++) {
printf(" %s",words[i]);
}
printf("");
DP(len,n,m);
}
```