Güz-2020

Ankara Üniversitesi Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü
BLM-3067 Algoritmalar

Ödev6

Son Gönderim Tarihi: 29.12.2020 23:59
ÖNEMLİ NOT: Gönderdiğiniz kodlarda kopya kontrolü yapılacaktır.

1. Aşağıdaki problemi 1. Algoritma ile çözen C kodunu yazınız.
2. Aşağıdaki problemi 2. Algoritma ile çözen C kodunu yazınız.

Problem: Her i=1,2,..n sayısı için bir malın i. gün için satış fiyatı olan P[i] sayısı veriliyor. Siz bu malı bir gün alarak başka gün satabiliyorsunuz. Sattıktan sonra başka bir gün aynı malı tekrardan alarak sonraki günlerde satabilirsiniz. Aldığınız malı satmadan yenisini alamıyorsunuz, yani stoklama yapamıyorsunuz. Amacınız C dilinde maksimum karı elde edebilen algoritma tasarlamaktır.

Örnek 1. Girdi**=>** 7, 1, 5, 3, 6, 4 // Malın i. gün için satış fiyatları

Çıktı=> 7 // çıktıda elde edilen maksimum kar yazılacaktır.

Fiyatlar sırasıyla 7, 1, 5, 3, 6, 4 olsun. Bu dizi için cevap 7 olur. Bunu elde etmek için 2. Gün 1 liradan alır, 3. Gün 5 liradan satarız. Sonra 4. Gün 3 liradan alır, 5. Gün 6 liradan satarız ve karımız 5-1+6-3=7 olur.

Örnek 2. Girdi=> 1, 2, 3, 4, 5

Çıktı=> 4

Fiyatlar sırasıyla 1, 2, 3, 4, 5 olsun. Bu dizi için kar 4 lira olur, bunu elde etmek için 1. Gün 1 liradan alır, 5. Gün 5 liradan satarız, kar 5-1=4 lira olur.

Örnek 3. Girdi=>7, 6, 4, 3, 1

Çıktı=> 0

Fiyatlar sırasıyla 7, 6, 4, 3, 1 olsun. Bu dizi için kar 0 lira olur, malı hangi gün alırsak, diğer günlerdeki fiyat daha küçük olacağından satıştan kar edemeyiz.

Bu soru için bir sonraki sayfada 2 farklı algoritma önereceğim. (Belki bir az kendiniz düşünürsünüz diye çözümü bir sonraki sayfaya yazdım.)

1. **Algoritma:** (Bu algoritmayı kaba kuvvet yöntemi ile tasarlayacağım ama sözde-kodu bir az daha azalt-yönet yöntemine benzer yolla yapacağım.) Ana fikrimiz çok basit, sıra ile her gün için (i. gün) malı bu gün aldığımızı varsayalım, sonra da bu günden sonraki daha yüksek fiyatlı her gün için satalım. Sattıktan sonraki günlerde de aynı işleme devam edelim, karı hesaplayalım ve bu kar önceki karımızdan daha fazla ise karımızı güncelleyelim. Örnek1 üzerinde anlatayım. Başlangıçta kar=0 olsun. İ=1 için fiyat 7 dir. Bu fiyattan alalım. Diğer günlere bakarsak fiyatlar 7 den daha az olduğu için satamayız ve kar=0 kalır. İ=2 için fiyat=1 dir. Bu fiyattan alalım. Şimdi kar edebileceğimiz her gün için satmayı deneyelim. 3. Gün fiyat 5 ve sattığımızda kar=5-1=4 tür. Şimdi 4. Gün 3 liradan alırsak 5. Ve 6. Günlerde satabiliriz. 5. Gün satarsak toplam kar 4+6-3=7, 6. Gün satarsak 4+4-3=5 olacağından yeni karımız 7 oldu. Ama i=2 durumu bitmedi. Şimdi 2. Gün aldığımız malı 4. Gün satalım kar 3-1=2 olur ve sonra bu malı 5. Gün alsak da kar elde edemiyoruz. Bu durumda toplam kar 2 olacağından kar güncellenmesi olmaz. Yine de i=2 durumu bitmedi. 2. Gün aldığımız malı 5. Gün satarsak kar=6-1=5 olur. Yine de 5<7 olduğundan kar güncellenmesi olmaz. İ=2 için son durum malın 6. Gün satılmasıdır, bu zaman da kar 4-1=3<7 olur. Yani sonuçta 2. Gün malı alırsak toplamda 7 lira karımız olur. Sonra i=3, 4,5 için aynı işlemlere devam ediyoruz.

   Şimdi bu anlattığım algoritmayı bir az daha akıllı uygulamaya çalışalım. İlk günümüz b, son günümüz s olsun. (Başlangıçta b=1, s=n olur, n P dizisinin boyutudur.) Biz i. günde aldığımızı ve P[j]>P[i] koşulunu sağlayan j. günde satmak istiyoruz. [k, m] aralığında maksimum karı hesaplayabildiğimiz bir MaxKar(P, k, m) fonksiyonumuz olsun. Doğal olarak eğer k>=m ise MaxKar(P, k, m)=0 dır. Bu durumda [b, s] aralığında i. günde alıp j. günde satmaktan elde edeceğimiz

   toplam kar=MaxKar(P, b, i-1)+P[j]-P[i]+MaxKar(P, j+1, s] olacaktır.

   Algoritmanın sözde kodu aşağıdadır. Bu kodun ilk çağrısı MaxKar(P, 1, n) dir.

```
MaxKar (P, b, s)
if s<=b
      then return 0
Kar←0
for  i←b to s
     for j←i+1 to s
         if  P[j]>P[i]
               then yeni_kar←P[j]-P[i]+MaxKar(P, b, i-1)+MaxKar(P, j+1, s)
                    if yeni_kar>Kar
                          then Kar←yeni_kar
return Kar
```

   Şimdi Örnek1 üzerinde (P=7, 1, 5, 3, 6, 4) bu kodu tekrar anlatayım. MaxKar(P, 1,n) çağrıldı. Yani b=1, s=n oldu. İlk if koşulu sağlanmadı. Kar=0 oldu. i=1, n döngüsü başladı. i=1 için j=2, n döngüsü başlar ama P[1]=7 olduğu için hiçbir j için P[j]>P[i] koşulu sağlanmaz. i=2 için j=3, n döngüsü başlar. j=3 için yeni_kar=P[3]-P[2]+MaxKar(P, 1, 1)+MaxKar(P, 4, 6)=5-1+0+MaxKar(P, 4, 6)=4+MaxKar(P,4,6) olur. Bu sayıyı hesaplamak için kod kendini b=4, s=6 için çağıracaktır. Yine ilk if koşulu sağlanmaz. Bu çağrının içinde Kar=0 olur. i=4, 6 döngüsü başlar. i=4 için j=5, 6 döngüsü başlar. j=5 için P[5]>P[4] olduğu için bu çağrının içinde yeni_kar=P[5]-P[4]+MaxKar(P, 4,

3)+MaxKar(P,6,6)=3 olur. Bu çağrının içinde yeni_kar=3>Kar=0 olduğundan Kar=3 olur. j=6 için yeni_kar=P[6]-P[4]+MaxKar(P, 4, 3)+MaxKar(P, 7, 6)=1 olur. yeni_kar=1<Kar=3 olduğundan Kar=3 kalır. i=5 için P[5] den daha düşük fiyat olmadığı için yeni bir değer hesaplanmaz. İ=6 için j=7, 6 döngüsü çalışmaz. Sonuçta bu çağrıdan geriye Kar=3 geri döner. Bu da MaxKar(P, 1, 6) çağrısında i=2 ve j=3 için geri döner ve burada yeni_kar=4+3=7 olur. Sonra benzer işlemler j=4, 5, 6 için yapılır. Sonra da i=3, 4,5, 6 için benzer işlemler yapılır.

2. **Algoritma:** (Bu da aslında bir az daha akıllıca yapılmış kaba kuvvet algoritmasıdır.) Şöyle düşünelim. Eğer size sadece bir defa almak ve sadece bir defa satmak izni verilseydi, en düşük fiyattan alarak en yüksek fiyattan satmak isterdiniz, yani j>i olmak üzere P[j]-P[i] farklarının pozitif olanlarının en büyüğü maksimum karımız olurdu. (Bu farkların hiçbiri pozitif değilse kar=0 olurdu.) Bu da bu dizinin en küçük elemanı (min), en büyük elemanının(max) solunda olduğu durumda max-min olurdu. Yani bu durum varsa max ve min bulmak yeterli olurdu. Bu fikri bizim probleme uyarlamaya çalışalım. Bunun için yerel minimum ve yerel maksimum kavramlarını tanımlayalım. Aslında bu kavramları fonksiyonlar için biliyorsunuz. Örnekler üzerinde anlatayım. Örneğin 5,4,3,6,7,8,1,9 olsun. Bu dizide 3 ve 1 yerel minimumlardır. 5, 8 ve 9 ise yerel maksimumlardır. (İlk elemandan bir sonraki eleman ilk elemandan küçükse ilk eleman yerel maksimum, tersi ise ilk eleman yerel minimumdur. Son elemandan bir önceki eleman son elemandan küçükse son eleman yerel maksimum, aksi ise son eleman yerel minimumdur. İlk ve son olmayan bir i elemanı için P[i-1]<P[i]>P[i+1] ise P[i] yerel maksimumdur. Eğer P[i-1]>P[i]<P[i+1] ise P[i] yerel minimumdur.)

   Algoritmamız şöyledir:

   Kar=0 olsun

   İlk günden başla ve tüm günler bitene kadar aşağıdaki 3 adımı tekrarla

      Kalan günlerde P dizisinin ilk yerel minimumunu bul ve o gün almayı planla

      Sonraki günlerde P dizisinin ilk yerel maksimumunu bul ve o gün satmayı planla. (Yerel maksimum yoksa alma.)

      Bu alışverişten kazancı hesapla ve Kar'a ekle.

   Her 3 örnek üzerinde anlatayım. Örnek1 de P dizisi 7, 1, 5, 3, 6, 4 tür. İlk yerel min=1 dir. Bu fiyattan alıyoruz. Bu günden sonraki ilk yerel maksimum 5 tir. Bu fiyattan satıyoruz. Kar=4 oldu. Bu günden sonraki ilk yerel min=3 ve daha sonraki günlerde ilk yerel maksimum 6 dır. Bu alışverişten 6-3=3 kazanıyoruz ve toplam karımız=4+3=7 olur. Sonraki yerel min=4 ama yerel maksimum olmadığına göre bu alışı yapmıyoruz.

   Örnek2 de P dizisi 1,2,3,4,5 tir. Yerel min=1 ve yerel maksimum=5 ve 5-1=4 kar

   Örnek 3 te 7, 6, 4, 3, 1 ilk yerel minimum 1 ve sonra yerel maksimum yok, buna göre de alış yapmıyoruz. Kar=0 kalıyor.

   Bu algoritmanın sözde kodunu yazmak çok zor olmadığı için sözde kodu yazmayı size bırakıyorum.