Bölüm 8: Çevrim İçi Algoritmalar

Bölüm 8: Çevrim İçi Algoritmalar · Çalışma Soruları

Çevrim içi model | Rekabet oranı | Kayak kiralama | Sayfalama: FIFO, LRU, OPT | Liste güncelleme | Veri akışları | Rezervuar örnekleme

Sercan KÜLCÜ · Giresun Üniversitesi · Bilgisayar Mühendisliği

© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü:
https://sercankulcu.github.io/files/advanced_algorithms/slides/Bolum_08_Cevrim_Ici_Algoritmalar_Calisma_Sorulari.html

ÇOKTAN SEÇMELİ · ÇEVRİM İÇİ MODEL

Soru 1: Hangisi Çevrim İçi Çalışabilir?

Öğeler tek tek geliyor ve her öğe geldiğinde o ana kadarki öğelerin sıralı hâli hazır olmalı. Hangi algoritma bu **çevrim içi** modele doğal olarak uyar?

A) Seçimli sıralama: her turda kalanların en küçüğünü seçer

B) Eklemeli sıralama: gelen öğeyi sıralı kısma yerleştirir

C) Birleştirme sıralaması: diziyi ikiye bölüp birleştirir

D) Rastgele QuickSort: rastgele pivotla parçalar

ÇOKTAN SEÇMELİ · ÇEVRİM İÇİ MODEL

Soru 2: Rekabet Oranının Tanımı

Bir en küçükleme probleminde çevrim içi algoritma ALG için "c-rekabetçidir" ifadesi ne anlama gelir?

A) Her istek dizisi σ için $ALG(\sigma) \leq c \cdot OPT(\sigma) + \alpha$ (α sabit) sağlanır.

B) Rastgele seçilen istek dizilerinde ALG ortalama olarak $c \cdot OPT$ kadar öder.

C) ALG'nin çalışma süresi en iyi çevrim dışı algoritmanın en çok c katıdır.

D) Yalnızca en kötü istek dizisinde $ALG(\sigma) = c \cdot OPT(\sigma)$ eşitliği olur.

ÇOKTAN SEÇMELİ · ÇEVRİM İÇİ MODEL

Soru 3: Düşman Türleri

Rastgele bir sayfalama algoritmasını analiz ederken **habersiz (oblivious) düşman** için hangisi doğrudur?

A) Algoritmanın kodunu bilmez; yalnızca önceki istekleri görür.

B) Her adımda algoritmanın rastgele kararını görüp sonraki isteği seçer.

C) İstek dizisini baştan seçer; algoritmanın yazı-turalarını göremez.

D) OPT yerine algoritmanın kendisiyle karşılaştırma yapar.

Soru 4: Başa Baş Stratejisi

`kayakMaliyet(5, 5, 7)` : B = 5, satın alma günü k = 5, gerçek gün sayısı d = 7. Maliyet nasıl birikir, rekabet oranı nedir?

```
1 int kayakMaliyet(int B, int k, int d) {
▶   int maliyet = 0;
3   for (int gun = 1; gun <= d; gun++) {
4     if (gun == k) { maliyet += B; break; }
  // satın al
5     maliyet += 1;  // kirala
6   }
7   return maliyet;
8 }
```



Satın alma 5 TL, kira günlüğü 1 TL. Strateji: k − 1 = 4 gün kiralanır, 5. gün hâlâ kayılıyorsa satın alınır. Sezon d = 7 gün sürecek (algoritma bunu bilmiyor).

Adım 1 / 7

DEĞİŞKENLER

`B = 5` `k = 5` `d = 7` `gun = –` `maliyet = 0`

Soru 4: Başa Baş Stratejisi

`kayakMaliyet(5, 5, 7)` : B = 5, satın alma günü k = 5, gerçek gün sayısı d = 7. Maliyet nasıl birikir, rekabet oranı nedir?

```
1 int kayakMaliyet(int B, int k, int d) {
2   int maliyet = 0;
    for (int gun = 1; gun <= d; gun++) {
      if (gun == k) { maliyet += B; break; }
  // satın al
      maliyet += 1;  // kirala
6   }
7   return maliyet;
8 }
```



gün = 1 ≠ k → kiralanır: `maliyet = 1` .

Adım 2 / 7

DEĞİŞKENLER

`B = 5` `k = 5` `d = 7` `gun = 1` `maliyet = 1`

Soru 4: Başa Baş Stratejisi

`kayakMaliyet(5, 5, 7)` : B = 5, satın alma günü k = 5, gerçek gün sayısı d = 7. Maliyet nasıl birikir, rekabet oranı nedir?

```
1 int kayakMaliyet(int B, int k, int d) {
2   int maliyet = 0;
▶   for (int gun = 1; gun <= d; gun++) {
▶     if (gun == k) { maliyet += B; break; }
    // satın al
▶     maliyet += 1;  // kirala
6   }
7   return maliyet;
8 }
```



gün = 2 ≠ k → kiralanır: `maliyet = 2` .

Adım 3 / 7

DEĞİŞKENLER

`B = 5` `k = 5` `d = 7` `gun = 2` `maliyet = 2`

Soru 4: Başa Baş Stratejisi

`kayakMaliyet(5, 5, 7)` : B = 5, satın alma günü k = 5, gerçek gün sayısı d = 7. Maliyet nasıl birikir, rekabet oranı nedir?

```
1 int kayakMaliyet(int B, int k, int d) {
2   int maliyet = 0;
▶   for (int gun = 1; gun <= d; gun++) {
▶     if (gun == k) { maliyet += B; break; }
    // satın al
▶     maliyet += 1;  // kirala
6   }
7   return maliyet;
8 }
```



gün = 3 ≠ k → kiralanır: `maliyet = 3` .

Adım 4 / 7

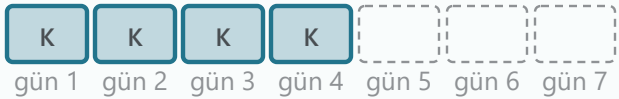
DEĞİŞKENLER

`B = 5` `k = 5` `d = 7` `gun = 3` `maliyet = 3`

Soru 4: Başa Baş Stratejisi

`kayakMaliyet(5, 5, 7)` : B = 5, satın alma günü k = 5, gerçek gün sayısı d = 7. Maliyet nasıl birikir, rekabet oranı nedir?

```
1 int kayakMaliyet(int B, int k, int d) {
2   int maliyet = 0;
▶   for (int gun = 1; gun <= d; gun++) {
▶     if (gun == k) { maliyet += B; break; }
    // satın al
▶     maliyet += 1;  // kirala
6   }
7   return maliyet;
8 }
```



gün = 4 ≠ k → kiralanır: `maliyet = 4` .

Adım 5 / 7

DEĞİŞKENLER

`B = 5` `k = 5` `d = 7` `gun = 4` `maliyet = 4`

Soru 4: Başa Baş Stratejisi

`kayakMaliyet(5, 5, 7)` : B = 5, satın alma günü k = 5, gerçek gün sayısı d = 7. Maliyet nasıl birikir, rekabet oranı nedir?

```
1 int kayakMaliyet(int B, int k, int d) {
2   int maliyet = 0;
▶   for (int gun = 1; gun <= d; gun++) {
▶     if (gun == k) { maliyet += B; break; }
  // satın al
5     maliyet += 1;  // kirala
6   }
7   return maliyet;
8 }
```



gün = 5 = k → satın alınır: maliyet = 4 + 5 = 9, `break` . Kalan günler (6, 7) bedavadır.

Adım 6 / 7

DEĞİŞKENLER

`B = 5` `k = 5` `d = 7` `gun = 5` `maliyet = 9`

Soru 4: Başa Baş Stratejisi

`kayakMaliyet(5, 5, 7)` : B = 5, satın alma günü k = 5, gerçek gün sayısı d = 7. Maliyet nasıl birikir, rekabet oranı nedir?

```
1 int kayakMaliyet(int B, int k, int d) {
2   int maliyet = 0;
3   for (int gun = 1; gun <= d; gun++) {
4     if (gun == k) { maliyet += B; break; }
    // satın al
5     maliyet += 1;  // kirala
6   }
▶   return maliyet;
8 }
```



ALG = **9**, OPT = min(d, B) = min(7, 5) = 5 → oran 1,8 = 2 − 1/B. d = 5 olsaydı da ALG 9, OPT 5 olurdu: düşmanın en iyi seçimi d = k'dir.

Adım 7 / 7

DEĞİŞKENLER

`B = 5` `k = 5` `d = 7` `gun = 5` `maliyet = 9`

ÇIKTI SORUSU · KAYAK KİRALAMA

Soru 5: B = 12 için Oranlar

Kayak kiralamada satın alma fiyatı B = 12 TL. (a) Başa baş stratejisinin (11 gün kirala, 12. gün al) rekabet oranı nedir? (b) **4. gün** satın alan stratejinin en kötü durum oranı nedir; düşman hangi d'yi seçer?

▶ **Cevabı göster**

KLASİK SORU · KAYAK KİRALAMA

Soru 6: Neden 2 − 1/B'den İyisi Yok?

Kayak kiralamada hiçbir deterministik çevrim içi stratejinin 2 − 1/B'den iyi rekabet oranına sahip olamayacağını düşman argümanıyla gösterin. Rastgelelik bu sınırı nasıl değiştirir?

▶ **Cevabı göster**

Soru 7: LRU Sayfalama İzleme

`lruHata(new int[]{2, 3, 2, 1, 5, 2, 4, 5, 3, 2}, 3)` çağrısında `onb` listesi nasıl değişir, kaç sayfa hatası olur?

```
1  int lruHata(int[] istekler, int k) {
▶    var onb = new LinkedList<Integer>();
▶    int hata = 0;
4    for (int sayfa : istekler) {
5      if (onb.remove((Integer) sayfa)) {
6        onb.addLast(sayfa);  // en yeni yap
7      } else {
8        hata++;
9        if (onb.size() == k)
10         onb.removeFirst();  // en eskiyi at
11       onb.addLast(sayfa);
12     }
13   }
14   return hata;
15 }
```



baş: en eski

k = 3, liste boş. İstekler: 2 3 2 1 5 2 4 5 3 2. Liste başı en uzun süredir kullanılmayan sayfadır.

Adım 1 / 12

DEĞİŞKENLER

`sayfa = –` `hata = 0` `onb = []`

Soru 7: LRU Sayfalama İzleme

`lruHata(new int[]{2, 3, 2, 1, 5, 2, 4, 5, 3, 2}, 3)` çağrısında `onb` listesi nasıl değişir, kaç sayfa hatası olur?

```
1  int lruHata(int[] istekler, int k) {
2    var onb = new LinkedList<Integer>();
3    int hata = 0;
4    for (int sayfa : istekler) {
5      if (onb.remove((Integer) sayfa)) {
6        onb.addLast(sayfa);  // en yeni yap
7      } else {
▶        hata++;
▶        if (onb.size() == k)
10         onb.removeFirst();  // en eskiyi at
▶        onb.addLast(sayfa);
12     }
13   }
14   return hata;
15 }
```



Sayfa 2 yok → hata = 1. Yer var.

Adım 2 / 12

DEĞİŞKENLER

`sayfa = 2` `hata = 1` `onb = [2]`

Soru 7: LRU Sayfalama İzleme

`lruHata(new int[]{2, 3, 2, 1, 5, 2, 4, 5, 3, 2}, 3)` çağrısında `onb` listesi nasıl değişir, kaç sayfa hatası olur?

```
1  int lruHata(int[] istekler, int k) {
2    var onb = new LinkedList<Integer>();
3    int hata = 0;
4    for (int sayfa : istekler) {
5      if (onb.remove((Integer) sayfa)) {
6        onb.addLast(sayfa);  // en yeni yap
7      } else {
▶        hata++;
▶        if (onb.size() == k)
10         onb.removeFirst();  // en eskiyi at
▶        onb.addLast(sayfa);
12     }
13   }
14   return hata;
15 }
```



Sayfa 3 yok → hata = 2. Yer var.

Adım 3 / 12

DEĞİŞKENLER

`sayfa = 3` `hata = 2` `onb = [2, 3]`

Soru 7: LRU Sayfalama İzleme

`lruHata(new int[]{2, 3, 2, 1, 5, 2, 4, 5, 3, 2}, 3)` çağrısında `onb` listesi nasıl değişir, kaç sayfa hatası olur?

```
1  int lruHata(int[] istekler, int k) {
2    var onb = new LinkedList<Integer>();
3    int hata = 0;
4    for (int sayfa : istekler) {
▶      if (onb.remove((Integer) sayfa)) {
▶        onb.addLast(sayfa);  // en yeni yap
7      } else {
8        hata++;
9        if (onb.size() == k)
10         onb.removeFirst();  // en eskiyi at
11       onb.addLast(sayfa);
12     }
13   }
14   return hata;
15 }
```



Sayfa 2 listede: remove true → **isabet**; sona eklenir, artık en yeni kullanılan.

Adım 4 / 12

DEĞİŞKENLER

`sayfa = 2` `hata = 2` `onb = [3, 2]`

Soru 7: LRU Sayfalama İzleme

`lruHata(new int[]{2, 3, 2, 1, 5, 2, 4, 5, 3, 2}, 3)` çağrısında `onb` listesi nasıl değişir, kaç sayfa hatası olur?

```
1  int lruHata(int[] istekler, int k) {
2    var onb = new LinkedList<Integer>();
3    int hata = 0;
4    for (int sayfa : istekler) {
5      if (onb.remove((Integer) sayfa)) {
6        onb.addLast(sayfa);  // en yeni yap
7      } else {
▶        hata++;
▶        if (onb.size() == k)
10         onb.removeFirst();  // en eskiyi at
▶        onb.addLast(sayfa);
12     }
13   }
14   return hata;
15 }
```



Sayfa 1 yok → hata = 3. Yer var.

Adım 5 / 12

DEĞİŞKENLER

`sayfa = 1` `hata = 3` `onb = [3, 2, 1]`

Soru 7: LRU Sayfalama İzleme

`lruHata(new int[]{2, 3, 2, 1, 5, 2, 4, 5, 3, 2}, 3)` çağrısında `onb` listesi nasıl değişir, kaç sayfa hatası olur?

```
1  int lruHata(int[] istekler, int k) {
2    var onb = new LinkedList<Integer>();
3    int hata = 0;
4    for (int sayfa : istekler) {
5      if (onb.remove((Integer) sayfa)) {
6        onb.addLast(sayfa);  // en yeni yap
7      } else {
▶        hata++;
▶        if (onb.size() == k)
▶          onb.removeFirst();  // en eskiyi at
▶        onb.addLast(sayfa);
12     }
13   }
14   return hata;
15 }
```



3 çıktı

Sayfa 5 yok → hata = 4. Dolu → baştaki (en eski kullanılan) 3 atılır.

Adım 6 / 12

DEĞİŞKENLER

`sayfa = 5` `hata = 4` `onb = [2, 1, 5]`

Soru 7: LRU Sayfalama İzleme

`lruHata(new int[]{2, 3, 2, 1, 5, 2, 4, 5, 3, 2}, 3)` çağrısında `onb` listesi nasıl değişir, kaç sayfa hatası olur?

```
1  int lruHata(int[] istekler, int k) {
2    var onb = new LinkedList<Integer>();
3    int hata = 0;
4    for (int sayfa : istekler) {
▶      if (onb.remove((Integer) sayfa)) {
▶        onb.addLast(sayfa);  // en yeni yap
7      } else {
8        hata++;
9        if (onb.size() == k)
10         onb.removeFirst();  // en eskiyi at
11       onb.addLast(sayfa);
12     }
13   }
14   return hata;
15 }
```



Sayfa 2 listede: remove true → **isabet**; sona eklenir, artık en yeni kullanılan.

Adım 7 / 12

DEĞİŞKENLER

`sayfa = 2` `hata = 4` `onb = [1, 5, 2]`

Soru 7: LRU Sayfalama İzleme

`lruHata(new int[]{2, 3, 2, 1, 5, 2, 4, 5, 3, 2}, 3)` çağrısında `onb` listesi nasıl değişir, kaç sayfa hatası olur?

```
1  int lruHata(int[] istekler, int k) {
2    var onb = new LinkedList<Integer>();
3    int hata = 0;
4    for (int sayfa : istekler) {
5      if (onb.remove((Integer) sayfa)) {
6        onb.addLast(sayfa);  // en yeni yap
7      } else {
▶        hata++;
▶        if (onb.size() == k)
▶          onb.removeFirst();  // en eskiyi at
▶        onb.addLast(sayfa);
12     }
13   }
14   return hata;
15 }
```



1 çıktı

Sayfa 4 yok → hata = 5. Dolu → baştaki (en eski kullanılan) 1 atılır.

Adım 8 / 12

DEĞİŞKENLER

`sayfa = 4` `hata = 5` `onb = [5, 2, 4]`

Soru 7: LRU Sayfalama İzleme

`lruHata(new int[]{2, 3, 2, 1, 5, 2, 4, 5, 3, 2}, 3)` çağrısında `onb` listesi nasıl değişir, kaç sayfa hatası olur?

```
1  int lruHata(int[] istekler, int k) {
2    var onb = new LinkedList<Integer>();
3    int hata = 0;
4    for (int sayfa : istekler) {
▶      if (onb.remove((Integer) sayfa)) {
▶        onb.addLast(sayfa);  // en yeni yap
7      } else {
8        hata++;
9        if (onb.size() == k)
10         onb.removeFirst();  // en eskiyi at
11       onb.addLast(sayfa);
12     }
13   }
14   return hata;
15 }
```



Sayfa 5 listede: remove true → **isabet**; sona eklenir, artık en yeni kullanılan.

Adım 9 / 12

DEĞİŞKENLER

`sayfa = 5` `hata = 5` `onb = [2, 4, 5]`

Soru 7: LRU Sayfalama İzleme

`lruHata(new int[]{2, 3, 2, 1, 5, 2, 4, 5, 3, 2}, 3)` çağrısında `onb` listesi nasıl değişir, kaç sayfa hatası olur?

```
1  int lruHata(int[] istekler, int k) {
2    var onb = new LinkedList<Integer>();
3    int hata = 0;
4    for (int sayfa : istekler) {
5      if (onb.remove((Integer) sayfa)) {
6        onb.addLast(sayfa);  // en yeni yap
7      } else {
▶        hata++;
▶        if (onb.size() == k)
▶          onb.removeFirst();  // en eskiyi at
▶        onb.addLast(sayfa);
12     }
13   }
14   return hata;
15 }
```



2 çıktı

Sayfa 3 yok → hata = 6. Dolu → baştaki (en eski kullanılan) 2 atılır.

Adım 10 / 12

DEĞİŞKENLER

`sayfa = 3` `hata = 6` `onb = [4, 5, 3]`

Soru 7: LRU Sayfalama İzleme

`lruHata(new int[]{2, 3, 2, 1, 5, 2, 4, 5, 3, 2}, 3)` çağrısında `onb` listesi nasıl değişir, kaç sayfa hatası olur?

```
1  int lruHata(int[] istekler, int k) {
2    var onb = new LinkedList<Integer>();
3    int hata = 0;
4    for (int sayfa : istekler) {
5      if (onb.remove((Integer) sayfa)) {
6        onb.addLast(sayfa);  // en yeni yap
7      } else {
▶        hata++;
▶        if (onb.size() == k)
▶          onb.removeFirst();  // en eskiyi at
▶        onb.addLast(sayfa);
12     }
13   }
14   return hata;
15 }
```



4 çıktı

Sayfa 2 yok → hata = 7. Dolu → baştaki (en eski kullanılan) 4 atılır.

Adım 11 / 12

DEĞİŞKENLER

`sayfa = 2` `hata = 7` `onb = [5, 3, 2]`

Soru 7: LRU Sayfalama İzleme

`lruHata(new int[]{2, 3, 2, 1, 5, 2, 4, 5, 3, 2}, 3)` çağrısında `onb` listesi nasıl değişir, kaç sayfa hatası olur?

```
1  int lruHata(int[] istekler, int k) {
2    var onb = new LinkedList<Integer>();
3    int hata = 0;
4    for (int sayfa : istekler) {
5      if (onb.remove((Integer) sayfa)) {
6        onb.addLast(sayfa);  // en yeni yap
7      } else {
8        hata++;
9        if (onb.size() == k)
10         onb.removeFirst();  // en eskiyi at
11       onb.addLast(sayfa);
12     }
13   }
▶    return hata;
15 }
```



10 istekte **7 hata**, 3 isabet. remove listeyi taradığı için her istek O(k); LinkedHashMap ile O(1) olur.

Adım 12 / 12

DEĞİŞKENLER

`sayfa = –` `hata = 7` `onb = [5, 3, 2]`

ÇIKTI SORUSU · SAYFALAMA

Soru 8: Belady'nin OPT Politikası

İstek dizisi 2 3 2 1 5 2 4 5 3 2 ve k = 3 için Belady'nin OPT politikası (bir sonraki kullanımı en uzak olanı çıkar) kaç sayfa hatası yapar? Her çıkarmada hangi sayfa atılır?

▶ **Cevabı göster**

ÇIKTI SORUSU · SAYFALAMA

Soru 9: Belady Anomalisi

İstek dizisi 0 1 2 3 0 1 4 0 1 2 3 4. FIFO ve LRU, k = 3 ve k = 4 çerçeveyle kaçar sayfa hatası yapar? Sonucu yorumlayın.

▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · SAYFALAMA

Soru 10: FIFO ile LRU Farkı

Önbellekte 3, 5, 8 sayfaları var (FIFO yükleme sırası 3, 5, 8; LRU kullanım sırası da başlangıçta 3, 5, 8). Sırasıyla 3 (isabet) ve 9 (hata) isteniyor. FIFO ve LRU 9 için hangi sayfaları çıkarır?

A) FIFO 3'ü, LRU 5'i çıkarır

B) İkisi de 3'ü çıkarır

C) FIFO 5'i, LRU 3'ü çıkarır

D) İkisi de 5'i çıkarır

ÇOKTAN SEÇMELİ · SAYFALAMA

Soru 11: İşaretleme Algoritmasının Oranı

Rastgele işaretleme (marking) algoritması habersiz düşmana karşı $2H_k$-rekabetçidir. **k = 4** için bu oran yaklaşık kaçtır?

A) 2,08

B) 4,00

C) 8,00

D) 4,17

KLASİK SORU · SAYFALAMA

Soru 12: Deterministik Alt Sınır k

Hiçbir deterministik sayfalama algoritmasının k'den iyi rekabet oranına sahip olamayacağını, yalnızca k + 1 farklı sayfa kullanan bir düşman dizisiyle gösterin.

▶ **Cevabı göster**

Soru 13: Öne Taşı İzleme

`mtfMaliyet`, A B C D listesi ve D D C D C D istekleriyle çağrılıyor. Her istekte maliyet ve liste nasıl değişir?

```
1  int mtfMaliyet(LinkedList<Character> liste,
   char[] istekler) {
▶    int toplam = 0;
3    for (char x : istekler) {
4      int konum = liste.indexOf(x) + 1;  // 1
   tabanlı
5      toplam += konum;
6      liste.remove((Character) x);
7      liste.addFirst(x);  // öne taşı
8    }
9    return toplam;
10 }
```



Liste A B C D; istekler D D C D C D. i. konumdaki öğeye erişmek i birim tutar (baştan taranır).

Adım 1 / 8

DEĞİŞKENLER

`x = –` `konum = –` `toplam = 0`

`liste = [A, B, C, D]`

Soru 13: Öne Taşı İzleme

`mtfMaliyet`, A B C D listesi ve D D C D C D istekleriyle çağrılıyor. Her istekte maliyet ve liste nasıl değişir?

```
1  int mtfMaliyet(LinkedList<Character> liste,
   char[] istekler) {
2    int toplam = 0;
3    for (char x : istekler) {
       int konum = liste.indexOf(x) + 1;  // 1
   tabanlı
       toplam += konum;
       liste.remove((Character) x);
       liste.addFirst(x);  // öne taşı
8    }
9    return toplam;
10 }
```



D: 4. konumda → maliyet 4, toplam **4**; D başa taşınır.

Adım 2 / 8

DEĞİŞKENLER

`x = D` `konum = 4` `toplam = 4`

`liste = [D, A, B, C]`

Soru 13: Öne Taşı İzleme

`mtfMaliyet`, A B C D listesi ve D D C D C D istekleriyle çağrılıyor. Her istekte maliyet ve liste nasıl değişir?

```
1  int mtfMaliyet(LinkedList<Character> liste,
   char[] istekler) {
2    int toplam = 0;
3    for (char x : istekler) {
▶      int konum = liste.indexOf(x) + 1;  // 1
   tabanlı
▶      toplam += konum;
▶      liste.remove((Character) x);
▶      liste.addFirst(x);  // öne taşı
8    }
9    return toplam;
10 }
```



D: 1. konumda → maliyet 1, toplam **5**; D başa taşınır.

Adım 3 / 8

DEĞİŞKENLER

`x = D` `konum = 1` `toplam = 5`

`liste = [D, A, B, C]`

Soru 13: Öne Taşı İzleme

`mtfMaliyet`, A B C D listesi ve D D C D C D istekleriyle çağrılıyor. Her istekte maliyet ve liste nasıl değişir?

```
1  int mtfMaliyet(LinkedList<Character> liste,
   char[] istekler) {
2    int toplam = 0;
3    for (char x : istekler) {
▶      int konum = liste.indexOf(x) + 1;  // 1
   tabanlı
▶      toplam += konum;
▶      liste.remove((Character) x);
▶      liste.addFirst(x);  // öne taşı
8    }
9    return toplam;
10 }
```



C: 4. konumda → maliyet 4, toplam **9**; C başa taşınır.

Adım 4 / 8

DEĞİŞKENLER

`x = C` `konum = 4` `toplam = 9`

`liste = [C, D, A, B]`

Soru 13: Öne Taşı İzleme

`mtfMaliyet`, A B C D listesi ve D D C D C D istekleriyle çağrılıyor. Her istekte maliyet ve liste nasıl değişir?

```
1  int mtfMaliyet(LinkedList<Character> liste,
   char[] istekler) {
2    int toplam = 0;
3    for (char x : istekler) {
▶      int konum = liste.indexOf(x) + 1;  // 1
   tabanlı
▶      toplam += konum;
▶      liste.remove((Character) x);
▶      liste.addFirst(x);  // öne taşı
8    }
9    return toplam;
10 }
```



D: 2. konumda → maliyet 2, toplam **11**; D başa taşınır.

Adım 5 / 8

DEĞİŞKENLER

`x = D` `konum = 2` `toplam = 11`

`liste = [D, C, A, B]`

Soru 13: Öne Taşı İzleme

`mtfMaliyet`, A B C D listesi ve D D C D C D istekleriyle çağrılıyor. Her istekte maliyet ve liste nasıl değişir?

```
1  int mtfMaliyet(LinkedList<Character> liste,
   char[] istekler) {
2    int toplam = 0;
3    for (char x : istekler) {
▶      int konum = liste.indexOf(x) + 1;  // 1
   tabanlı
▶      toplam += konum;
▶      liste.remove((Character) x);
▶      liste.addFirst(x);  // öne taşı
8    }
9    return toplam;
10 }
```



C: 2. konumda → maliyet 2, toplam **13**; C başa taşınır.

Adım 6 / 8

DEĞİŞKENLER

`x = C` `konum = 2` `toplam = 13`

`liste = [C, D, A, B]`

Soru 13: Öne Taşı İzleme

`mtfMaliyet`, A B C D listesi ve D D C D C D istekleriyle çağrılıyor. Her istekte maliyet ve liste nasıl değişir?

```
1  int mtfMaliyet(LinkedList<Character> liste,
   char[] istekler) {
2    int toplam = 0;
3    for (char x : istekler) {
▶      int konum = liste.indexOf(x) + 1;  // 1
   tabanlı
▶      toplam += konum;
▶      liste.remove((Character) x);
▶      liste.addFirst(x);  // öne taşı
8    }
9    return toplam;
10 }
```



D: 2. konumda → maliyet 2, toplam **15**; D başa taşınır.

Adım 7 / 8

DEĞİŞKENLER

`x = D` `konum = 2` `toplam = 15`

`liste = [D, C, A, B]`

Soru 13: Öne Taşı İzleme

`mtfMaliyet`, A B C D listesi ve D D C D C D istekleriyle çağrılıyor. Her istekte maliyet ve liste nasıl değişir?

```
1  int mtfMaliyet(LinkedList<Character> liste,
   char[] istekler) {
2    int toplam = 0;
3    for (char x : istekler) {
4      int konum = liste.indexOf(x) + 1;  // 1
   tabanlı
5      toplam += konum;
6      liste.remove((Character) x);
7      liste.addFirst(x);  // öne taşı
8    }
▶    return toplam;
10 }
```



MTF toplamı **15**. Aynı dizide hiç taşımayan liste 4+4+3+4+3+4 = 22, transpoz (bir öne al) 17 öder: sık istenen D ve C hızla öne gelir.

Adım 8 / 8

DEĞİŞKENLER

`x = –` `konum = –` `toplam = 15`

`liste = [D, C, A, B]`

ÇOKTAN SEÇMELİ · LİSTE GÜNCELLEME

Soru 14: Liste Güncelleme Stratejileri

Liste güncelleme problemindeki stratejiler için hangisi doğrudur?

A) Transpoz 2-rekabetçidir, çünkü öğeyi yalnızca bir adım taşır.

B) Sıklık sayacı rekabetçidir; sık istenen öğeler hep öndedir.

C) MTF 2-rekabetçidir; kanıt, ters sıralı çift sayısını potansiyel alır.

D) MTF her dizide hiç taşımayan listeden daha az öder.

ÇIKTI SORUSU · VERİ AKIŞLARI

Soru 15: Çevrim İçi Ortanca

İki heap'li çevrim içi ortanca yapısına (sol: max-heap, sag: min-heap) sırasıyla 7, 3, 9, 1, 5 ekleniyor ve her eklemeden sonra `System.out.print(ortanca() + " ")` çalıştırılıyor. Çıktı nedir? En sonda heap'lerde ne vardır?

```
void ekle(int x) {
  if (sol.isEmpty() || x <= sol.peek())
    sol.offer(x);  // küçük yarı
  else
    sag.offer(x);  // büyük yarı
  if (sol.size() > sag.size() + 1)
    sag.offer(sol.poll());
  else if (sag.size() > sol.size())
    sol.offer(sag.poll());
}
double ortanca() {
  if (sol.size() == sag.size())
    return (sol.peek() + sag.peek()) / 2.0;
  return sol.peek();
}
```

▶ **Cevabı göster**

ÇIKTI SORUSU · VERİ AKIŞLARI

Soru 16: Rezervuar Örnekleme

`rezervuar(new int[]{5, 8, 2, 9, 4, 7}, 2)` çağrılıyor; `rastgele.nextInt(i + 1)` sırasıyla 1, 3, 0, 4 döndürüyor. Dönen liste nedir? 0 tabanlı 3. öğe (9) rezervuara hangi olasılıkla girer, akış sonunda hangi olasılıkla rezervuardadır?

```
List<Integer> rezervuar(int[] akis, int k) {
  List<Integer> r = new ArrayList<>();
  for (int i = 0; i < Math.min(k, akis.length); i++)
    r.add(akis[i]);
  for (int i = k; i < akis.length; i++) {
    int j = rastgele.nextInt(i + 1);  // 0..i
    if (j < k)
      r.set(j, akis[i]);
  }
  return r;
}
```

▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · VERİ AKIŞLARI

Soru 17: Rezervuarda Kalma Olasılığı

k = 3'lük rezervuar örneklemede 0 tabanlı i = 11. öğe gelince rezervuara hangi olasılıkla girer? Akış toplam n = 20 öğeyle biterse bu öğe sonda hangi olasılıkla rezervuardadır?

A) 3/12 ve 1/20

B) 3/11 ve 3/20

C) 1/4 ve 1/4

D) 1/4 ve 3/20

ÇOKTAN SEÇMELİ · VERİ AKIŞLARI

Soru 18: nextInt(i) Hatası

Rezervuar kodunda `rastgele.nextInt(i + 1)` yerine `rastgele.nextInt(i)` yazılırsa (i ≥ k) ne olur?

A) Değişiklik olmaz; j yine 0 … i aralığında kalır.

B) Öğe k/i olasılıkla girer; örneklem yanlı olur.

C) Yeni öğeler artık hiçbir zaman rezervuara giremez.

D) İlk k öğe her zaman rezervuarda kalır.

Son

Bölüm 8: Çevrim İçi Algoritmalar: Çalışma Soruları

Konu anlatımı: Çevrim İçi Algoritmalar sunumu · Terim sözlüğü