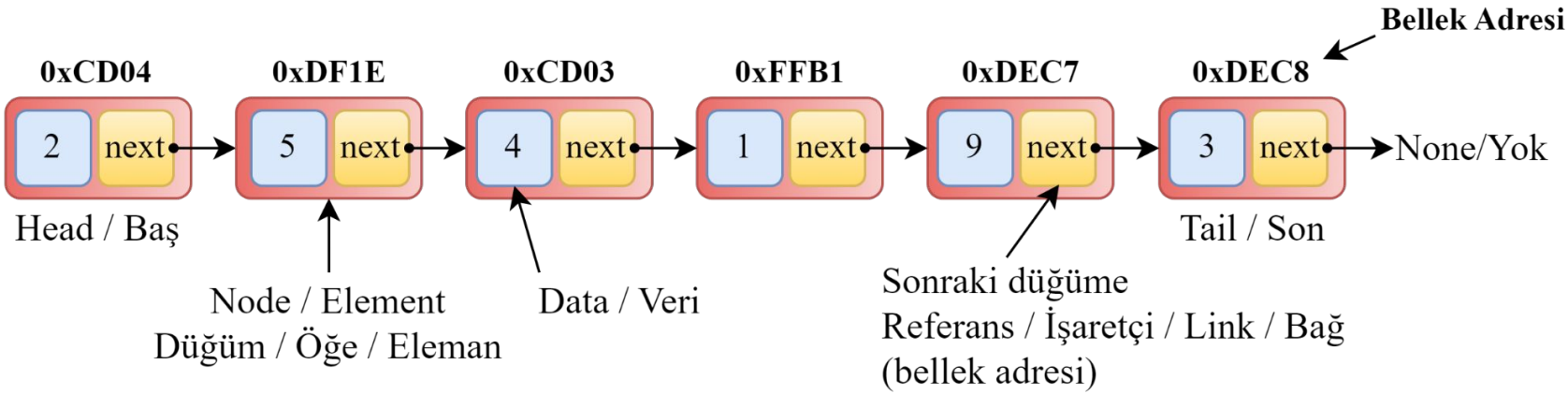
Bağlı Listeler

Dr. Hakan TEMİZ

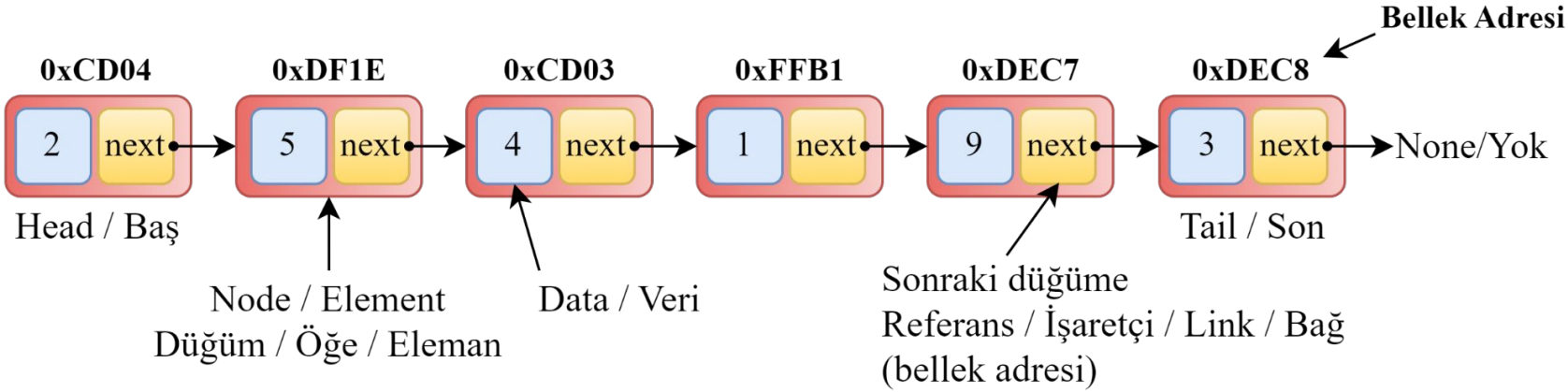
Bağlı Liste

- Bilginin ardışık konumlanmadığı doğrusal ve dinamik bir veri yapısıdır.
- Bağlı listeler, öğelerden meydana gelir. Öğelere, eleman (element), düğüm (node) isimleri de verilir.
- Örneğin, [2, 5, 4, 1, 9, 3] dizisinin bağlı liste yapısı:



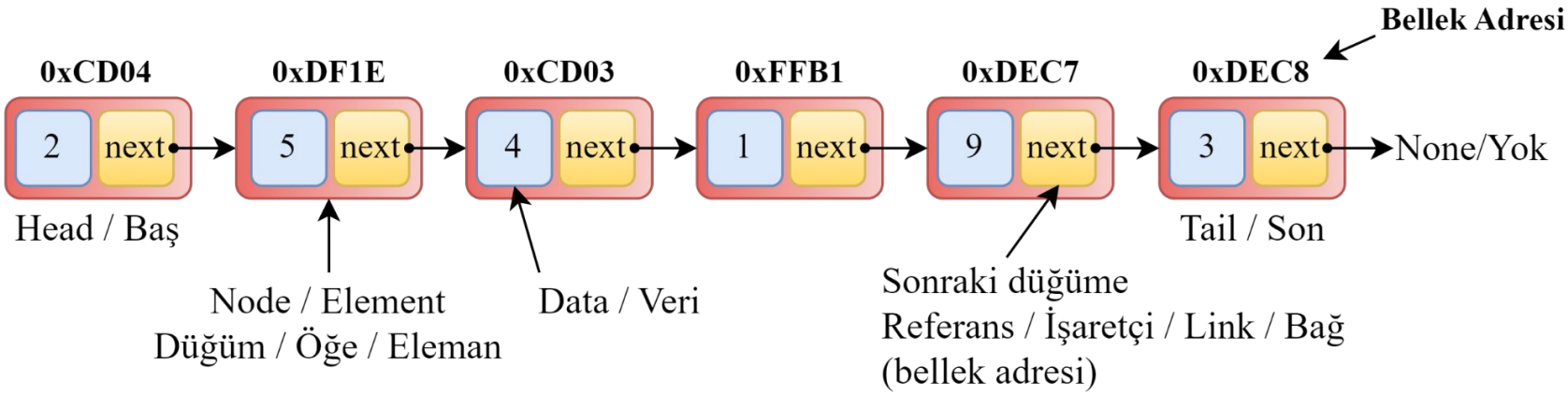
Bağlı Liste

- Listenin düğümleri, geliş sırasına göre birbirinin ardına eklenir.
- Elemanlar bir özelliğe göre sıralanarak oluşturuluyor veya düzenleniyorsa Sıralı Bağlantılı Liste (Ordered Linked List) olarak adlandırılır.
- Mavi kutucuklar veri (data); turuncu (next) kutucuklar sonraki elemanın adresine referanstır, yani işaretçidir (pointer). Listenin ilk elemanı Baş (Head) ve sonu Son (End) veya Kuyruk (Tail) olarak adlandırılır.



Bağlı Liste

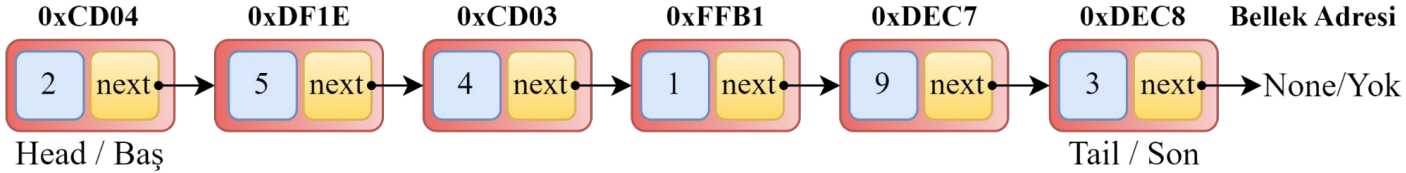
- Listenin her öğesi bir veri alanı bir de sonraki elemana referans (sonraki öğenin bellek adresi) bilgilerini barındırır.
- Adres bilgisini tutan yapıya işaretçi (pointer) adı da verilir.
- Öğeler, içerdiği bilgiye ilave olarak, sonraki öğenin adresini saklar. Böylece, birbirini izleyen öğelere, sonraki öğenin bellek adres bilgisi (referans) üzerinden erişilir. Örneğin, [5] 'in "next" niteliği [4] 'ün adresini (0xCD03) saklar, ...



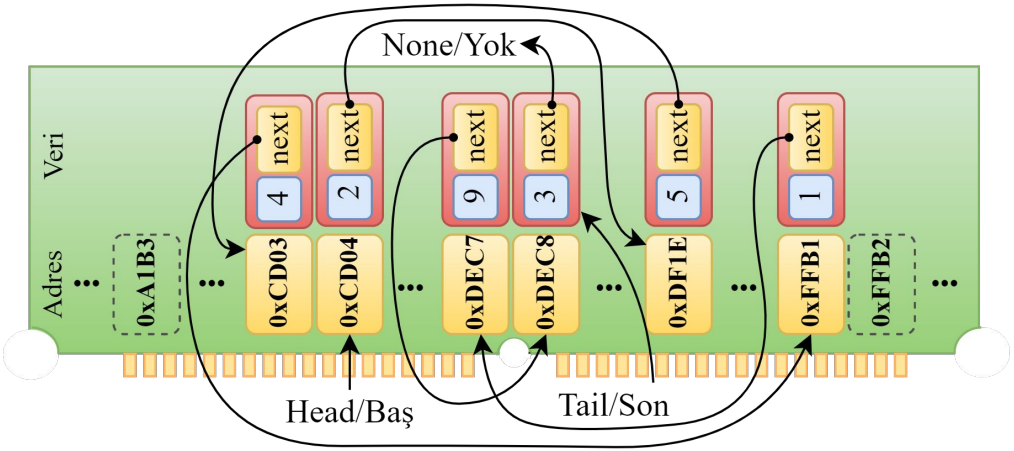
Bağlı Liste - Mantıksal Yapı ve Bellek Yerleşimi

- Öğeler bellekte farklı konumlarda bulunur, nerede bulunduklarının bir önemi yoktur.

Mantıksal Yapısı



Bellekteki Yerleşim



Bağlı Liste - Özellikler & İşlemler

Bağlı Listelerin Özellikleri:

- Dinamiktir : Düğüm ekleyerek veya çıkararak kolayca yeniden boyutlandırılabilir.
- Bitişik Değildir : Düğümler rastgele bellek konumlarında saklanır ve referanslarla (işaretçilerle) birbirine bağlanır.
- Sıralı Erişilir : Düğümlere yalnızca listenin başından başlayarak sıralı olarak erişilebilir.

Bağlı Listelerdeki İşlemler:

- Oluşturma : Yeni bir bağlı liste oluşturma veya mevcut bir listeye yeni bir öğe ekleme.
- Gezinme : Listede yineleme yaparak her öğeye erişme.
- Ekleme : Listede belirli bir konuma (en başa, araya, sona) yeni bir düğüm ekleme.
- Silme : Listeden bir düğümü (baştan, aradan, sondan) kaldırma.
- Arama : Listede belirli bir bilgiye sahip düğümü bulma.
- Uzunluk : Listenin eleman sayısını bulma.

Bağlı Liste - Avantajlar

- Genellikle ekleme ve silme işlemlerinde etkili ve hızlıdır. Ortadaki bir öğeyi eklemek (veya silmek) için yalnızca birkaç referansı değiştirmek yeterlidir.
- Herhangi bir konumda ekleme ve silme işlemi $O(1)$ zaman alır. Bir dizi veri yapısında ise ortada ekleme/silme işlemi $O(n)$ zaman alır. Bu veri yapısı basittir ve yığın, kuyruk ve diğer soyut veri yapılarını uygulamak için de kullanılabilir.
- Kuyruk veri yapılarının uygulanması, dizi yapısı üzerinde uygulanmasından çok daha verimli, kolay ve basittir. Bu nedenle çoğu dil kütüphanesi kuyruk tarzı veri yapılarını uygulamak için dahili olarak Bağlı Liste kullanır.
- Bağlı Liste, eleman sayısının önceden tahmin edilemediği durumlarda dizilere kıyasla daha fazla alan tasarrufu sağlayabilir. Dizi veri yapısında, belki de çok az miktarda bir veri kümesi için çok büyük miktarda alan diziye tahsis edilir. Buna karşın bağlı liste dinamik arttığı için, sadece ihtiyaç duyulan miktarda bellek tahsis edilir.

Bağlı Liste - Dezavantajlar

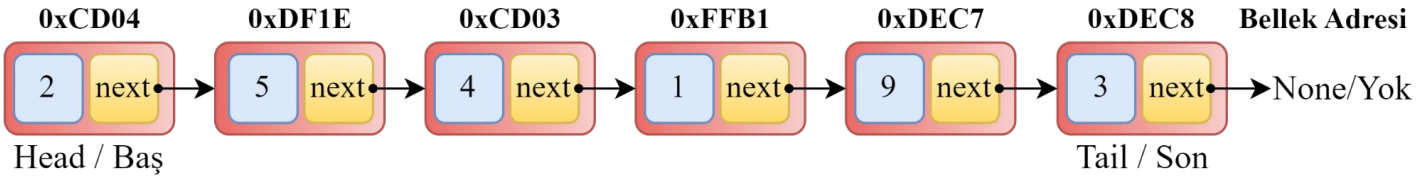
- Yavaş Erişim Süresi: Aranan öğeyi bulmak için listenin gezilmesi (ki bu O(n) işlemdir) gerektiğinden, öğelere erişim yavaş olabilir. Dolayısıyla öğelere erişimin hızlı olması gereken durumlar için pek uygun bir veri olamayabilir.
- İşaretçiler veya Referanslar: Bir sonraki düğüme erişmek için kullanılan işaretçi veya referanslara dayalı yapı, dizilere kıyasla anlaşılması ve kullanılması daha karmaşık bir yapıdır. Bu durum, listelerin hata ayıklamasını ve bakımını zorlaştırabilir.
- Ek Yük: Sonraki düğümlere erişmek için kullanılan işaretçi veya referanslar nedeniyle, veri yapısında ilave alan kullanılması gerekir.
- Önbellek Verimsizliği: Bitişik olmayan veri yapısı, bağlı listeleri önbellek açısından verimsiz kılar.

Bağlı Liste - Türler

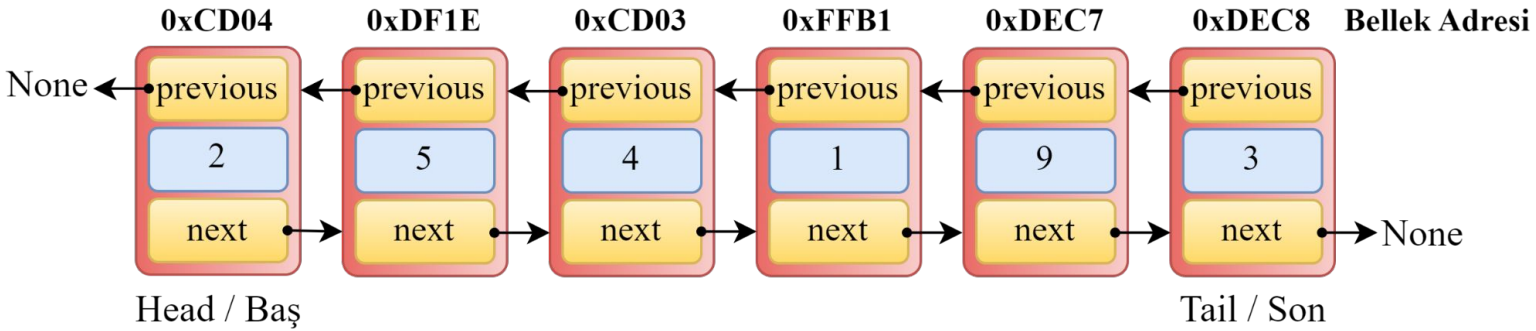
- Yaygın bağlı liste türleri şunlardır:
  - Doğrusal Bağlı Liste
    - Tek Yönlü Bağlı Liste - Singly Linked List
    - Çift Yönlü Bağlı Liste - Doubly Linked List
  - Çevrimsel Bağlı Liste - Circular Linked List
    - Tek Yönlü Çevrimsel Liste - Singly Circular Linked List
    - Çift Yönlü Çevrimsel Liste - Doubly Circular Linked List
- Çevrimsel listelerin son elemanlarının referansı boş (None) değil; listenin Baş elemanıdır.
- Böylece, çevrimsel listelerde, listenin sonuna gelindiğinde, son elemandaki referans vasıtası ile tekrar başa dönmek mümkün olur.

Bağlı Liste - Türler - Doğrusal Listeler

- Tek Yönlü Doğrusal Bağlı Liste

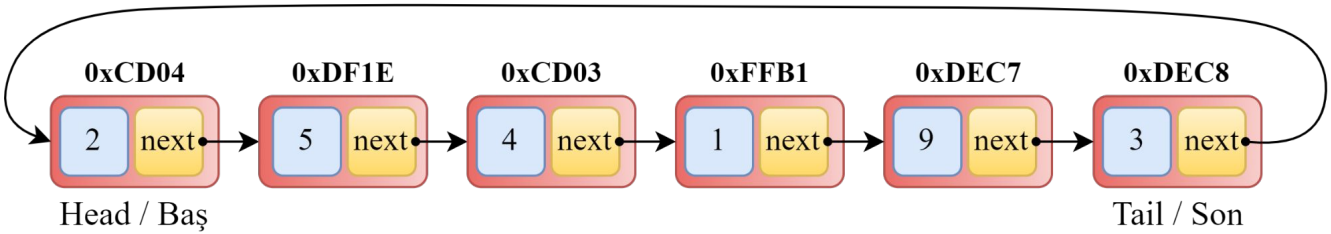


- Çift Yönlü Doğrusal Bağlı Liste

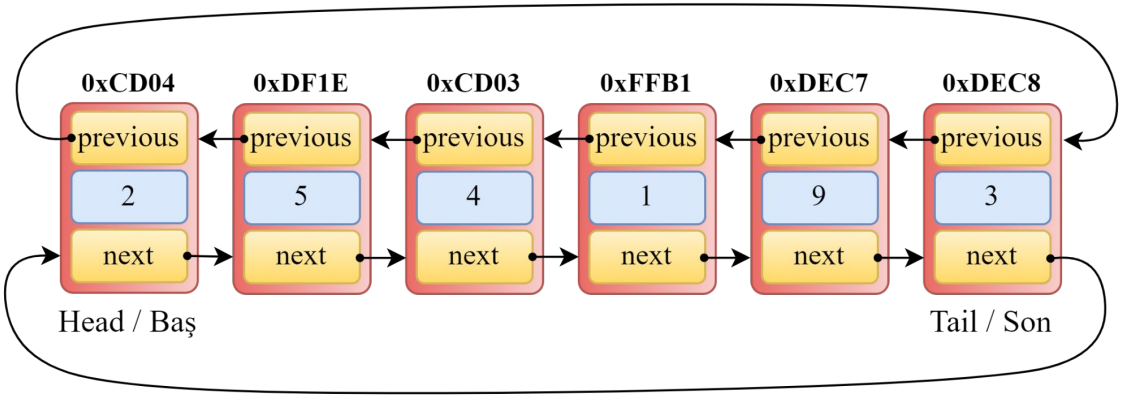


Bağlı Liste - Türler - Çevrimsel Listeler

- Tek Yönlü Çevrimsel Liste

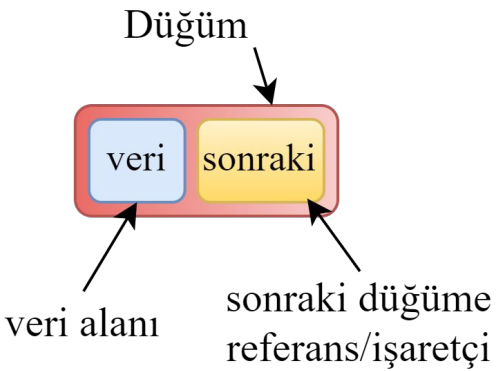


- Çift Yönlü Çevrimsel Liste



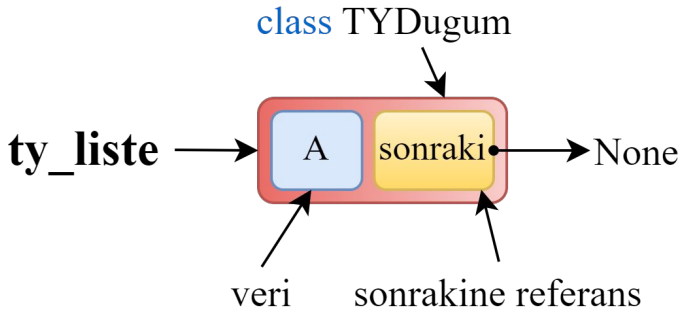
Bağlı Liste - Düğüm / Öğe / Node Tasarımı

```
class TYDugum(): # Tek Yönlü Düğüm
  """Öğe sınıfı"""
  def __init__(self, veri):
    self.veri = veri
    self.sonraki = None # bir sonraki öğeye referans
```



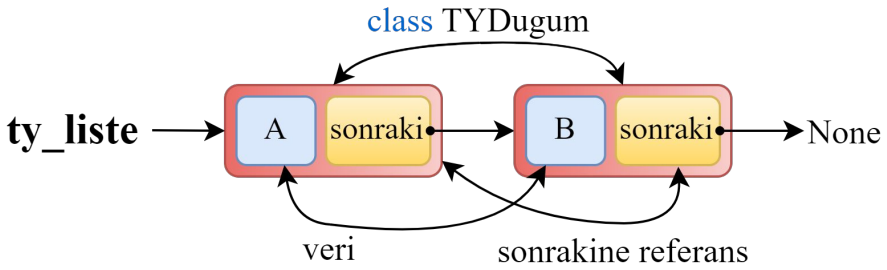
Bağlı Liste - Liste Oluşturma

```
class TYDugum(): # Tek Yönlü Düğüm
  """Öğe sınıfı"""
  def __init__(self, veri):
    self.veri = veri
    self.sonraki = None # bir sonraki öğeye referans


# ty_liste değişkeni listenin başlangıç elemanına referanstır
ty_liste = TYDugum("A") # liste oluşturuldu.
```



Bağlı Liste - Listeye Öğe Ekleme

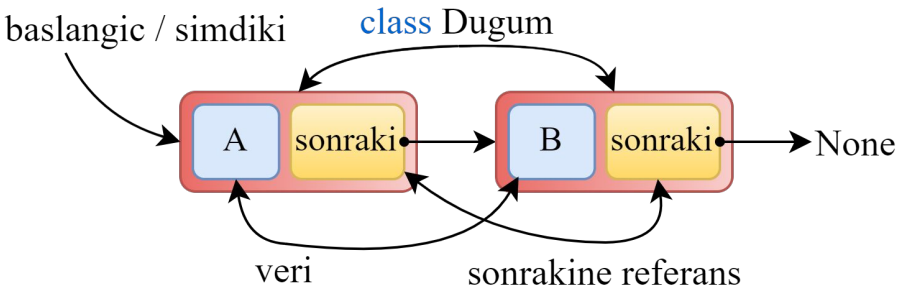
```
class TYDugum(): # Tek Yönlü Düğüm
  """Öğe sınıfı"""
  def __init__(self, veri):
    self.veri = veri
    self.sonraki = None # bir sonraki öğeye referans


# ty_liste değişkeni listenin başlangıç elemanına referanstır
ty_liste = TYDugum("A") # liste oluşturuldu.

yeni = TYDugum("B") # "B" verisi içeren yeni bir düğüm
ty_liste.sonraki = yeni # baş düğümün referansı yeni düğüm adresine atandı.
```



Bağlı Liste - Listede Gezinme

```
# Listeyi gezmek için fonksiyon
def liste_gez(baslangic):
    simdiki = baslangic # referansı listenin Baş/Head düğümüne ata

    # şimdiki düğüm None olmadığı müddetçe
    while simdiki:
        # şimdiki düğümdeki veriyi yazdır.
        print(simdiki.veri, end=" ") # end=" ", yazdırmaya aynı satırda devam et


        # bir sonraki düğüme geç
        simdiki = simdiki.sonraki
```

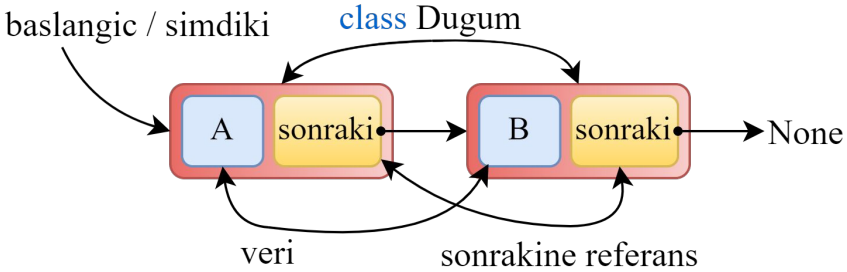
```
>>>liste_gez(ty_liste)
A B
```



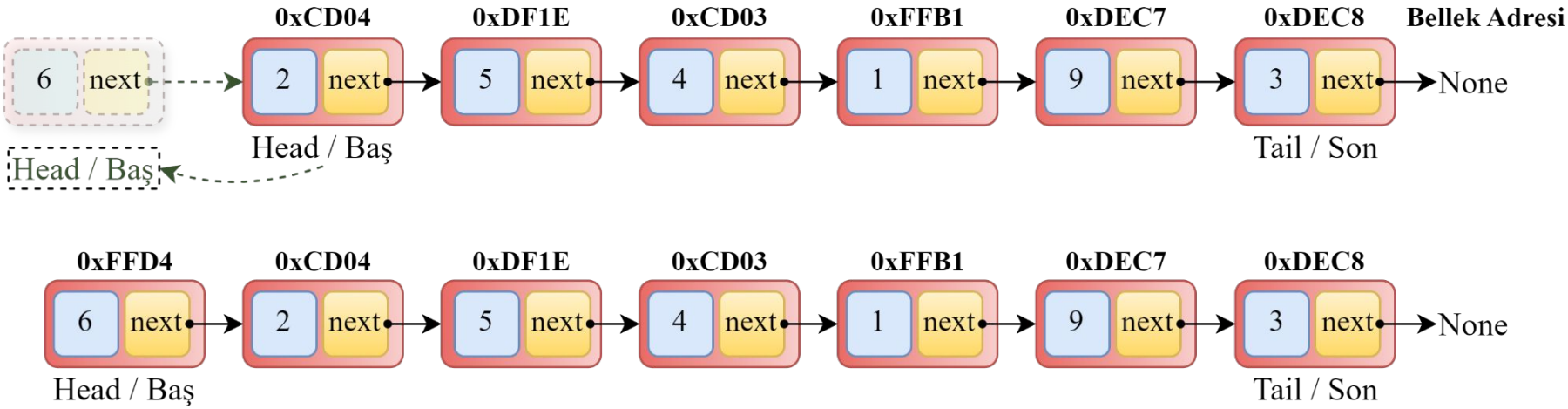
Bağlı Liste - Listede Arama

```
# Liste bir veriyi aramak için fonksiyon
def listede_ara(baslangic, aranan):
    simdiki = baslangic # şimdiki düğüm, listenin Baş/Head düğümü

    while simdiki:# şimdiki düğüm None olmadığı müddetçe
        if simdiki.veri == aranan:
          return True # aranan bulundu

        simdiki = simdiki.sonraki

    return False # aranan, listede yok
```

```
>>>listede_ara(ty_liste, "C")
False

>>>listede_ara(ty_liste, "A")
True
```

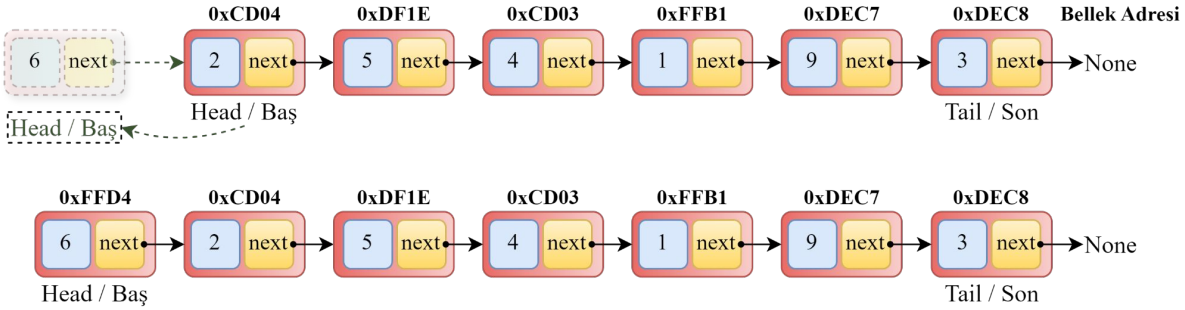


Tek Yönlü Bağlı Liste - Başa Öğe Ekleme



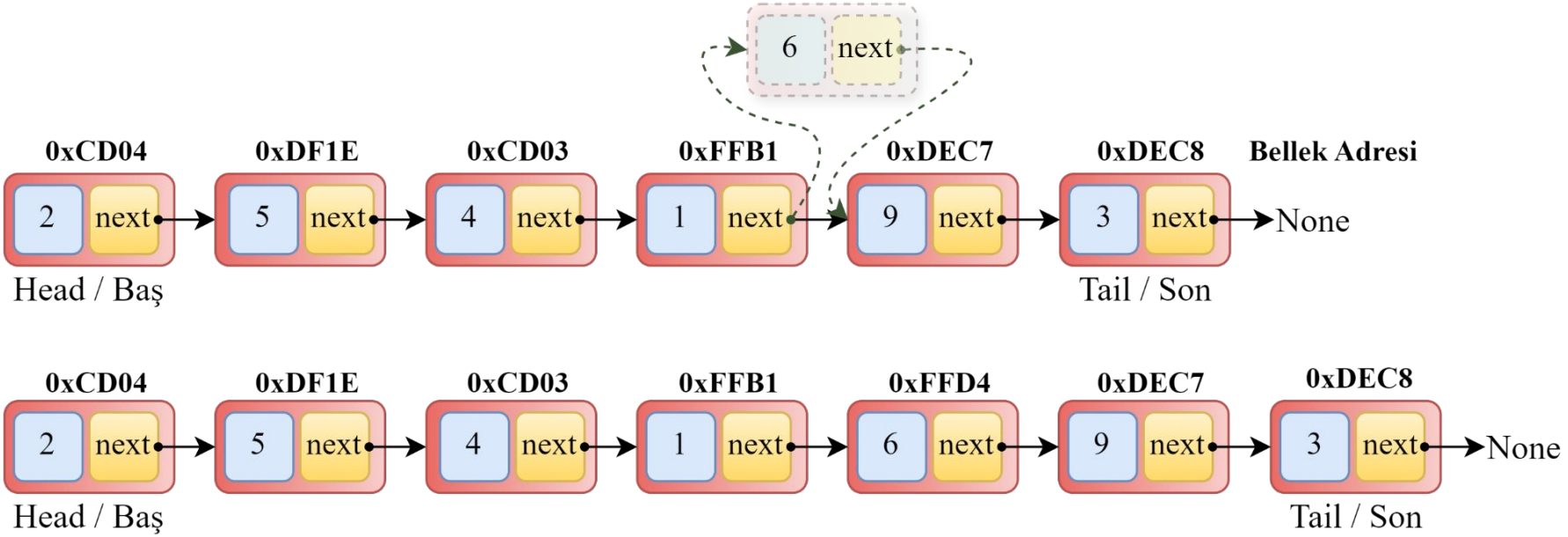
Tek Yönlü Bağlı Liste - Başa Öğe Ekleme

```
# ty_liste = [2, 5, 4, 1, 9, 3] olsun.
yeni = TYDugum(6)
yeni.sonraki = ty_liste
ty_liste = yeni

# listeyi yazdır
liste_gez(ty_liste)

6 2 5 4 1 9 3
```




```
# [2 5 4 1 9 3] verisine sahip
# Tek Yönlü Bir Liste Oluştur
def ty_liste_olustur():
  d2, d5, d4, d1, d9, d3 = \
  TYDugum(2), TYDugum(5), TYDugum(4),
  TYDugum(1), TYDugum(9), TYDugum(3)
  d9.sonraki = d3
  d1.sonraki = d9
  d4.sonraki = d1
  d5.sonraki = d4
  d2.sonraki = d5
  return d2

ty_liste = ty_liste_olustur()
# listeyi yazdır
liste_gez(ty_liste)
2 5 4 1 9 3
```

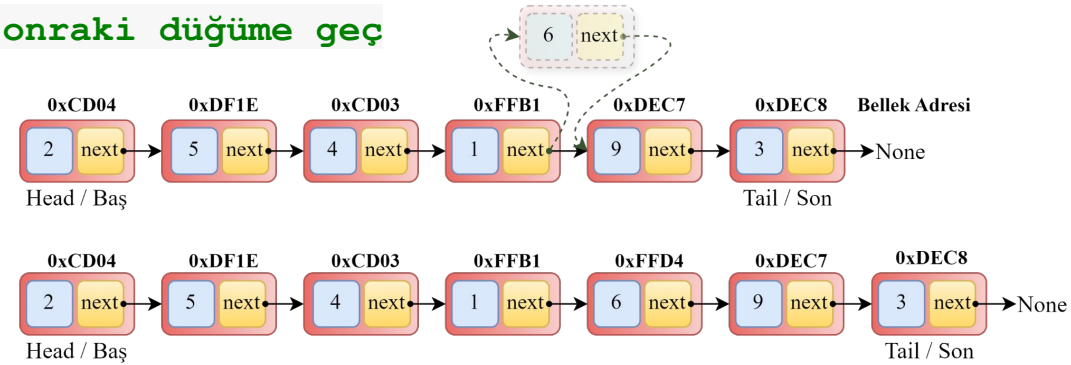
Tek Yönlü Bağlı Liste - Araya Öğe Ekleme



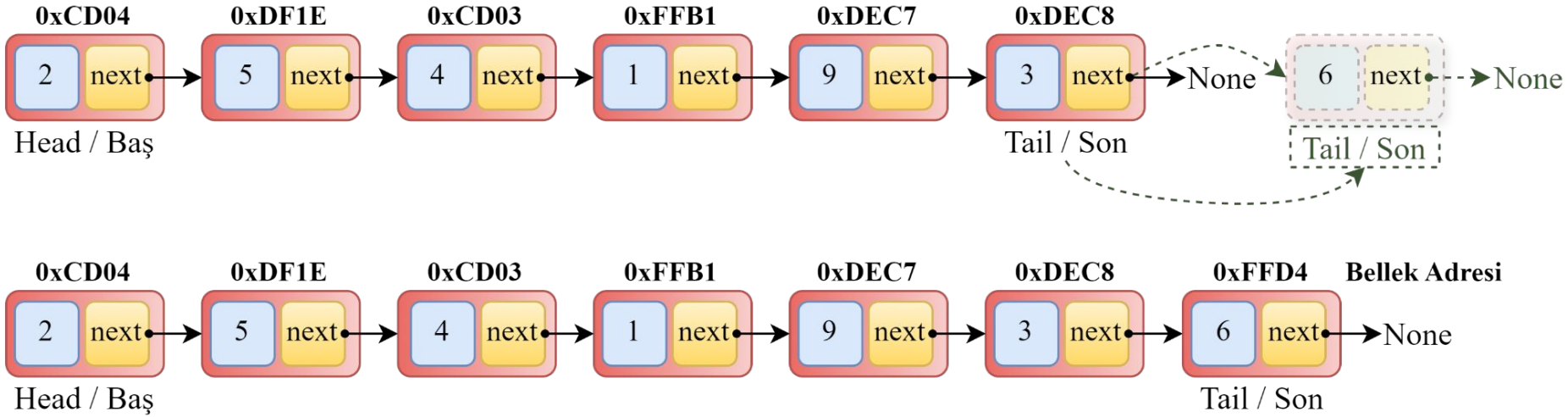
Tek Yönlü Bağlı Liste - Araya Öğe Ekleme

```
# ty_liste = [2, 5, 4, 1, 9, 3] olsun.
aranan = 1
simdiki = ty_liste # aktif düğüm en baştaki

while simdiki: # bu bir öğe ise devam et
  if simdiki.veri == aranan:  # bu öğe ardına eklenecek
    yeni = Dugum(6)
    yeni.sonraki = simdiki.sonraki # eklenen öğeyi sonraki öğeye bağla
    simdiki.sonraki = yeni # önceki öğeyi eklenen öğeye bağla
    break
  else: # aranan bulunamadı
    simdiki = simdiki.sonraki # sonraki düğüme geç
```

```
# listeyi yazdır
liste_gez(ty_liste)
2 5 4 1 6 9 3
```



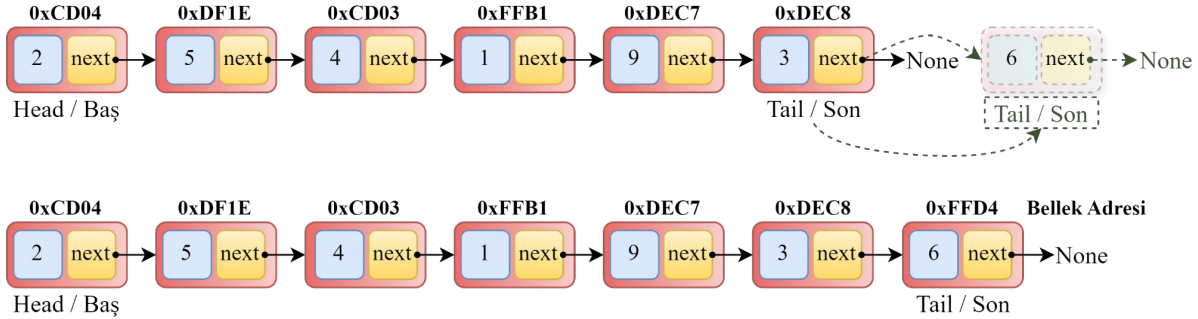
Tek Yönlü Bağlı Liste - Sona Öğe Ekleme



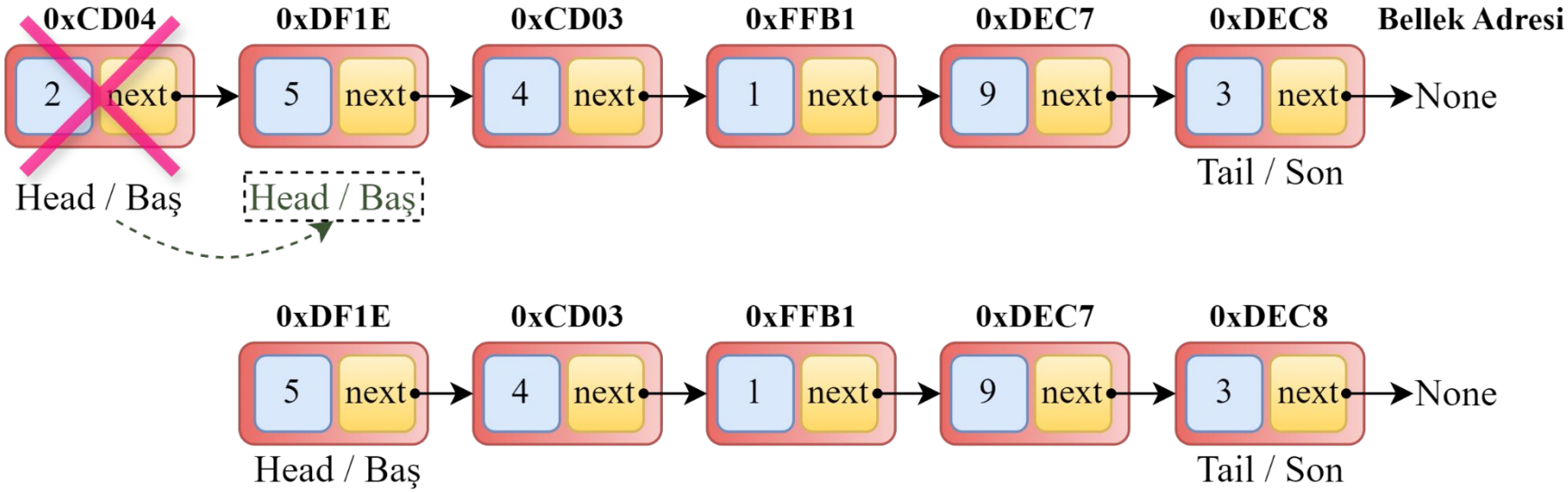
Tek Yönlü Bağlı Liste - Sona Öğe Ekleme

```
# ty_liste = [2, 5, 4, 1, 9, 3] olsun.
simdiki = ty_liste

while simdiki:
  if simdiki.sonraki is None: # son düğümdeyiz
     yeni = Dugum(6)
     simdiki.sonraki = yeni
     break
  else :
     simdiki = simdiki.sonraki
```

```
# listeyi yazdır
liste_gez(ty_liste)
2 5 4 1 9 3 6
```



Tek Yönlü Bağlı Liste - Baştan Öğe Silme

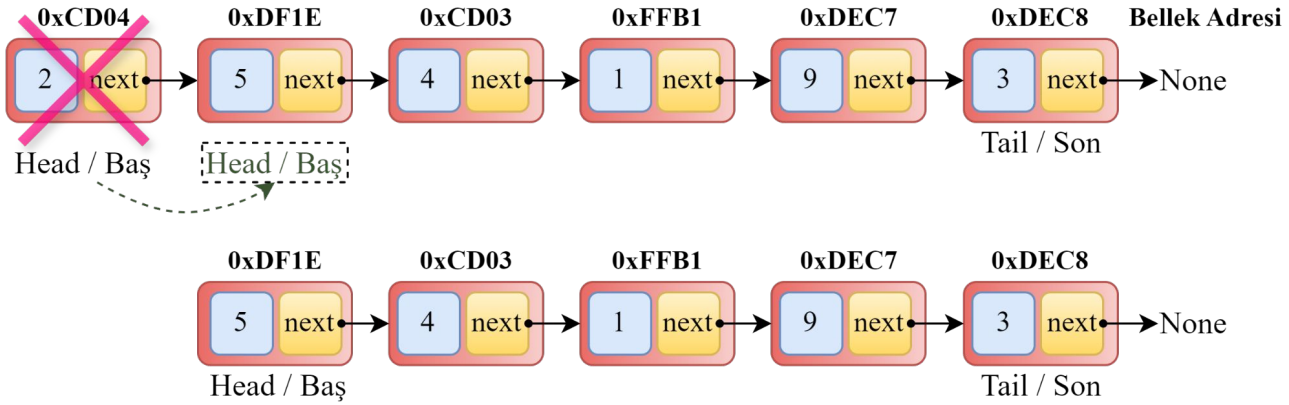


Tek Yönlü Bağlı Liste - Baştan Öğe Silme

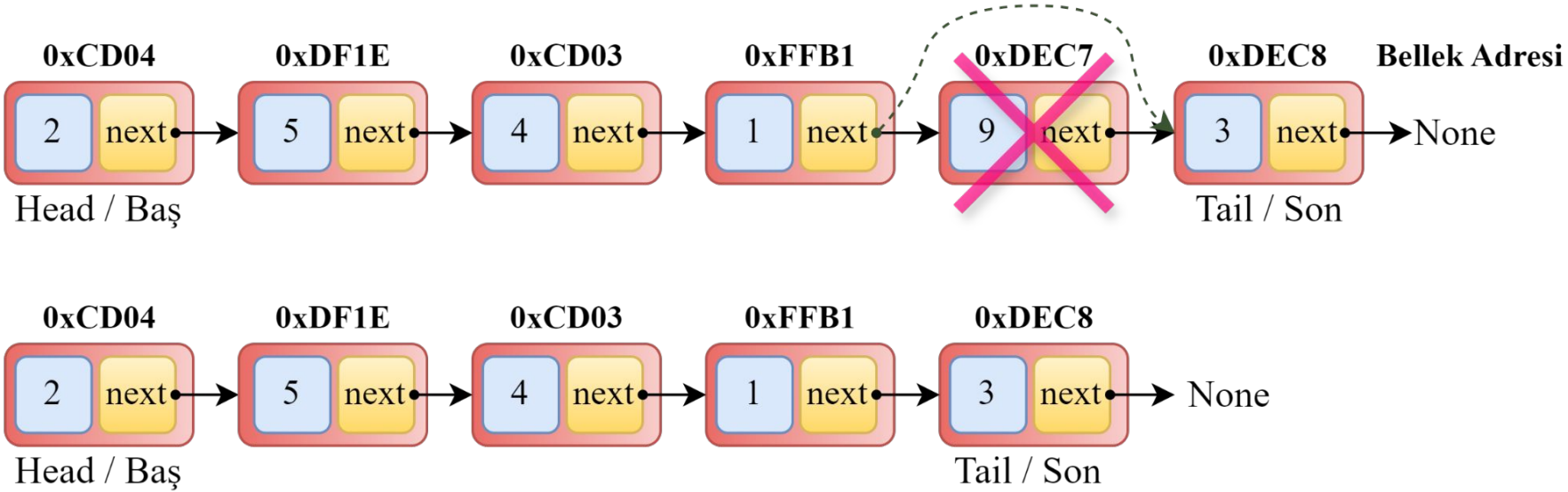
```
# ty_liste = [2, 5, 4, 1, 9, 3] olsun.
if ty_liste and ty_liste.sonraki:
  ty_liste = ty_liste.sonraki
else:
  ty_liste = None
```

```
# listeyi yazdır
liste_gez(ty_liste)
```

```
5 4 1 9 3
```

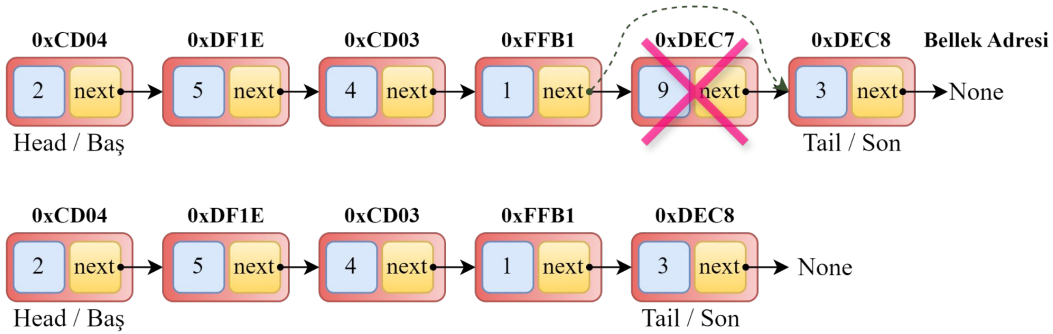


Tek Yönlü Bağlı Liste - Aradan Öğe Silme

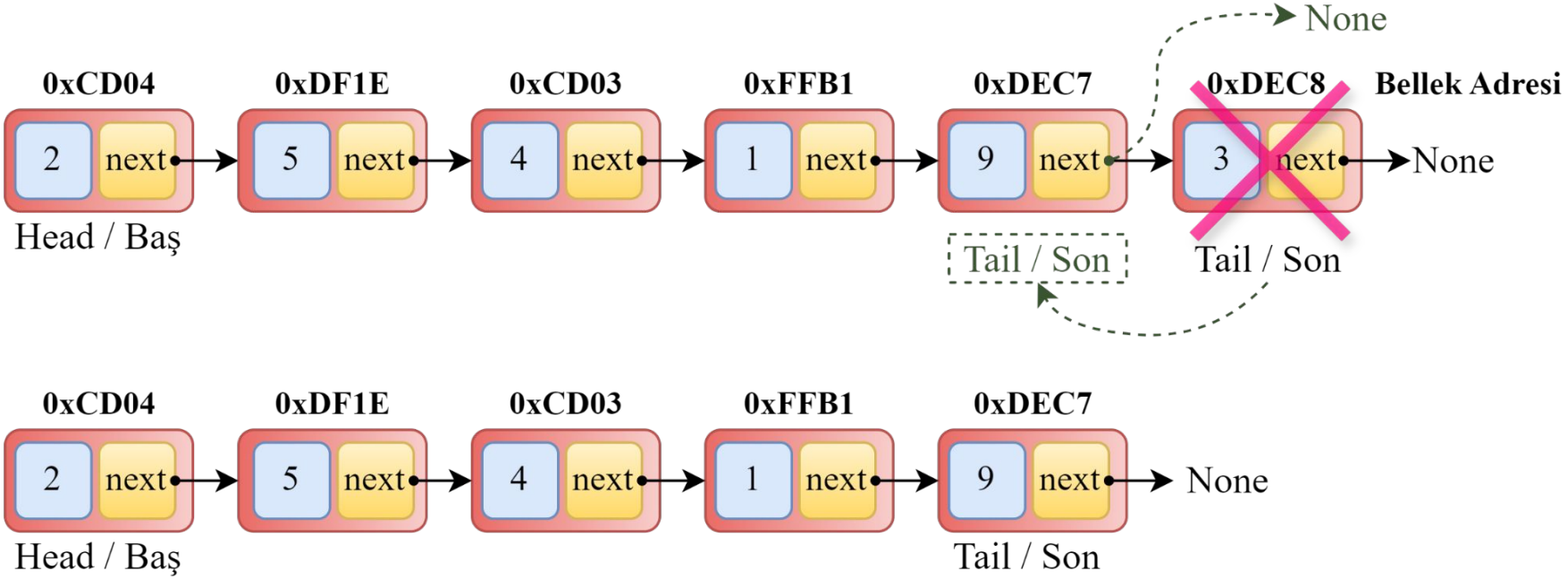


Tek Yönlü Bağlı Liste - Aradan Öğe Silme

```
# ty_liste = [2, 5, 4, 1, 9, 3] olsun.
# 1'den sonraki 9 düğümünü çıkar
simdiki = ty_liste

while simdiki:
  if simdiki.veri == 1:
     # simdiki.sonraki.sonraki, şimdikinden 2 sonraki öğedir.
     simdiki.sonraki = simdiki.sonraki.sonraki
     break
  else:
     simdiki = simdiki.sonraki
# listeyi yazdır
liste_gez(ty_liste)
2 5 4 1 3
```



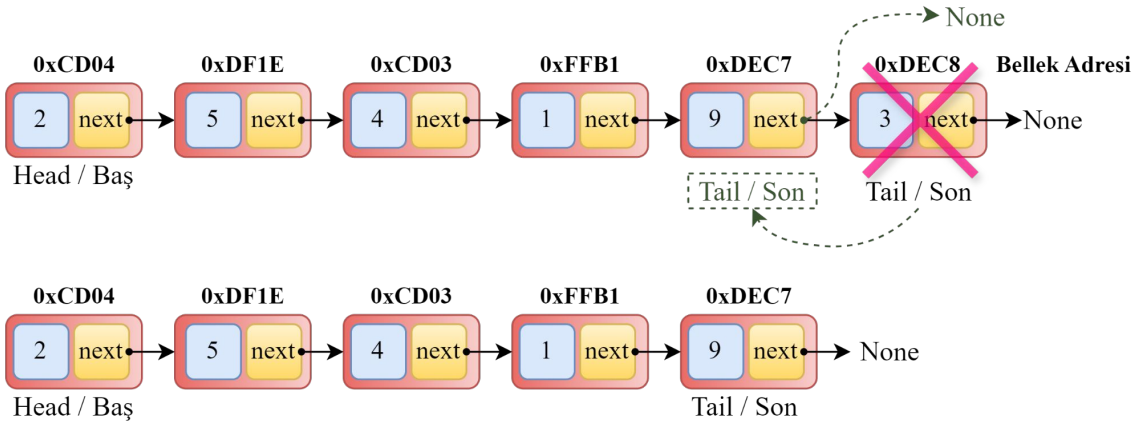
Tek Yönlü Bağlı Liste - Sondan Öğe Silme



Tek Yönlü Bağlı Liste - Sondan Öğe Silme

```
# ty_liste = [2, 5, 4, 1, 9, 3] olsun.
simdiki = ty_liste
onceki = None
while simdiki and simdiki.sonraki: # bundan sonra öğe varsa
  onceki = simdiki
  simdiki = simdiki.sonraki

if onceki:
  onceki.sonraki = None

# listeyi yazdır
liste_gez(ty_liste)
```

```
2 5 4 1 9
```



Tek Yönlü Bağlı Listelerde Karmaşıklık

Parametre	**Bağlı Liste**	**Dizi**	**Dinamik Dizi**
İndeksleme	O(n)	O(1)	O(1)
Başa ekleme / silme	O(1)	O(n), dizi dolmamışsa	O(n)
Sona ekleme	O(n)	O(1), dizi dolmamışsa	O(1), dizi dolmamışsa O(n), dizi doluysa
Sondan silme	O(n)	O(1)	O(n)
Araya ekleme	O(n)	O(n), dizi dolmamışsa	O(n)
Aradan silme	O(n)	O(1), dizi dolmamışsa	O(n)
İlave alan	O(n), referanslar için	0	O(n)

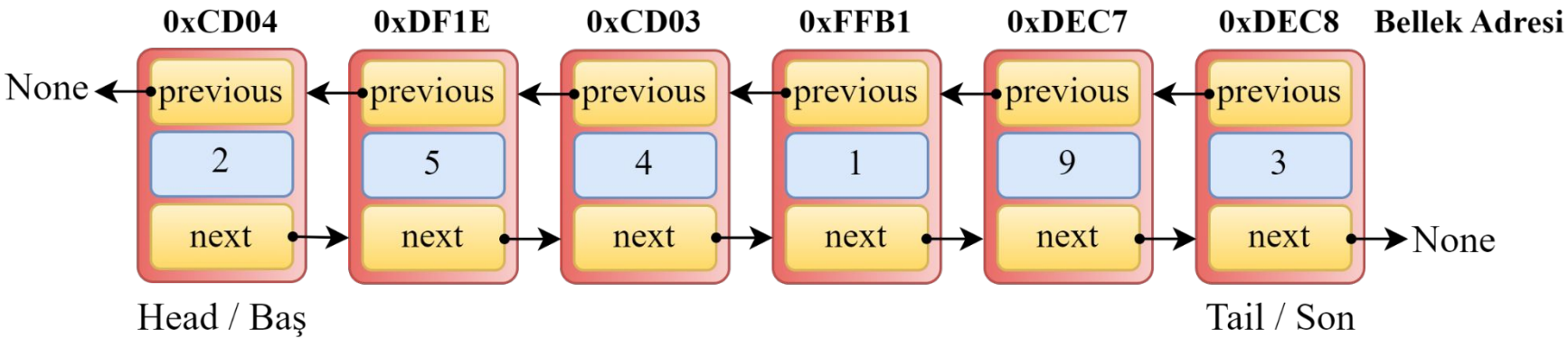
Tek Yönlü Bağlı Liste - Uygulama

```
Tek Yönlü Bağlı Liste'yi Tüm Özellikler ve İşlemleriyle Bir class ile
gerçekleştiriniz
```

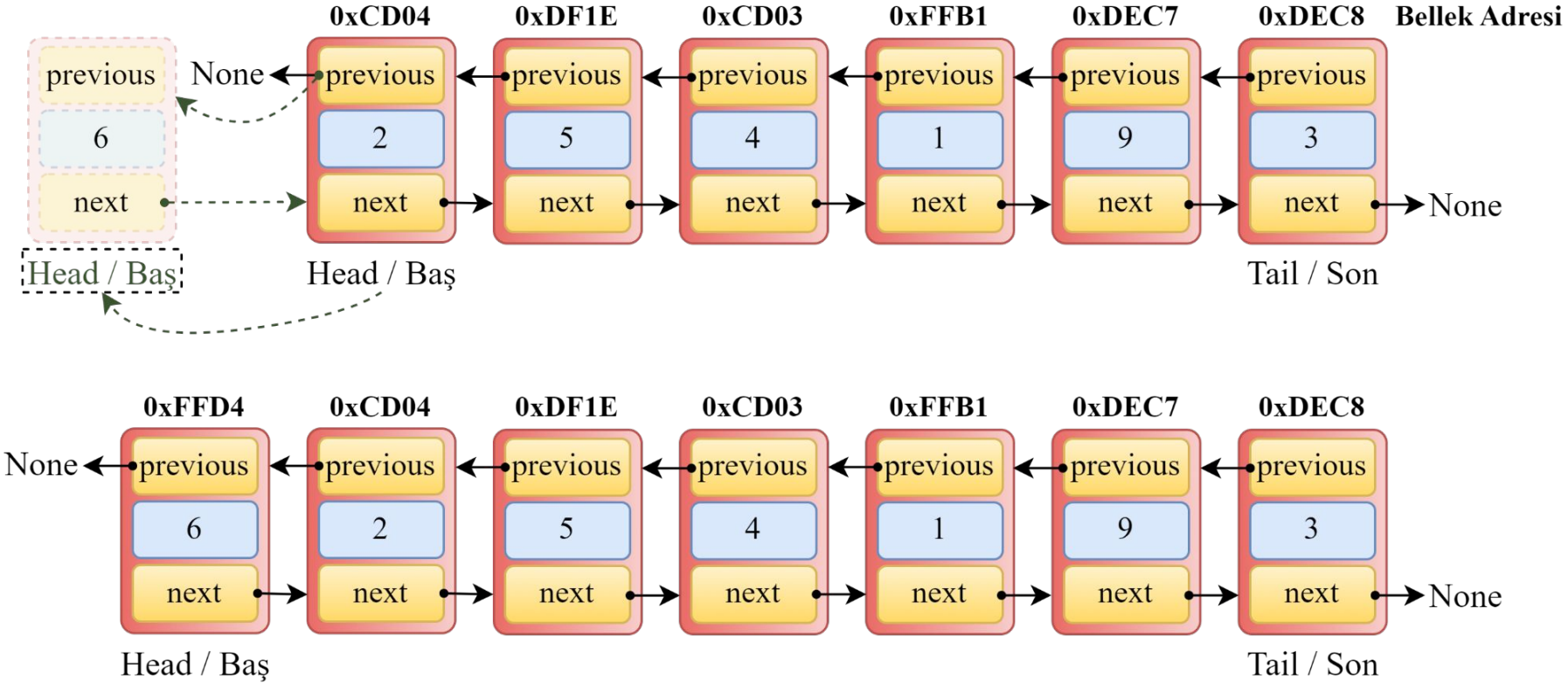
```
class TYListe:

…
```

Çift Yönlü Bağlı Liste

- Çift yönlü bağlı listenin düğümleri sonraki öğeye ilave olarak, önceki öğenin de adresine referansı (previous - önceki) barındırır.
- Baş düğümün previous niteliği ve Son düğümün next niteliği None'dır.
- İki yönde de gezinmek mümkün ve çok kolaydır.

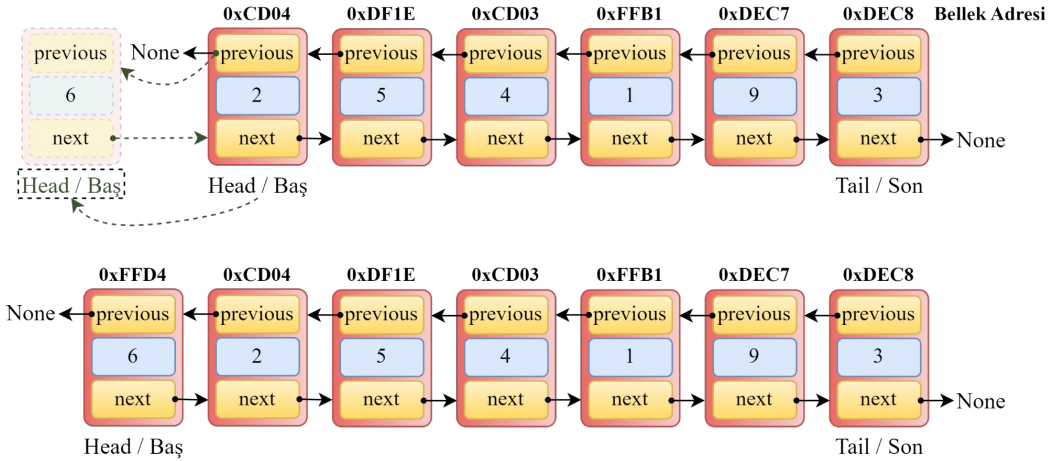


Çift Yönlü Bağlı Liste - Başa Öğe Ekleme

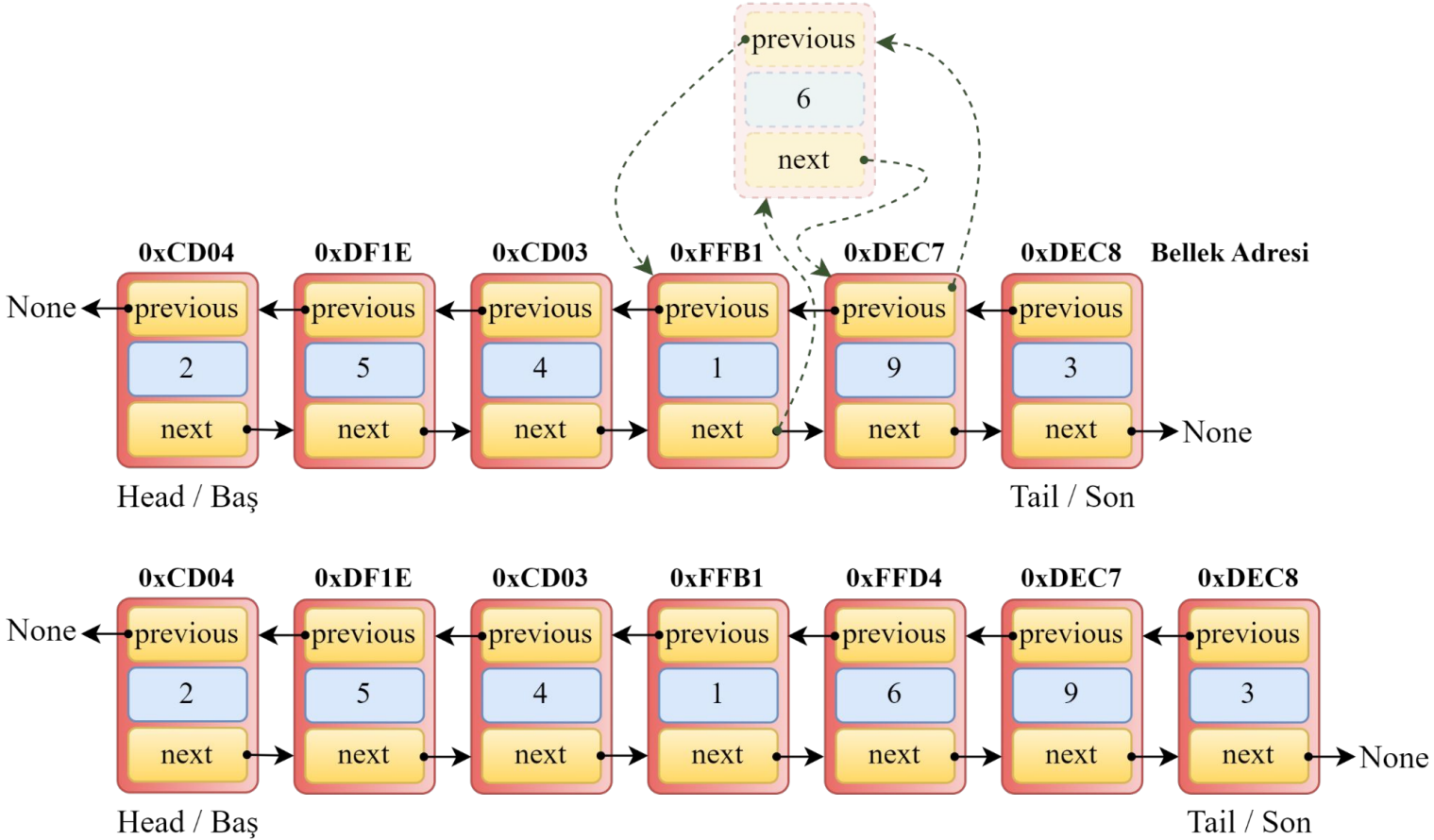


Çift Yönlü Bağlı Liste - Başa Öğe Ekleme

Başa öğe eklemek için program?

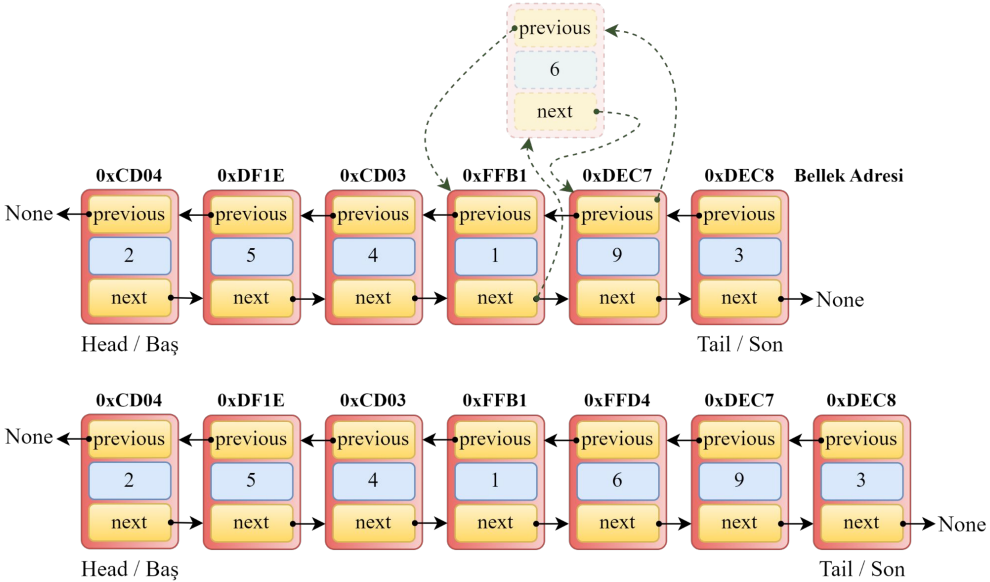


Çift Yönlü Bağlı Liste - Araya Öğe Ekleme

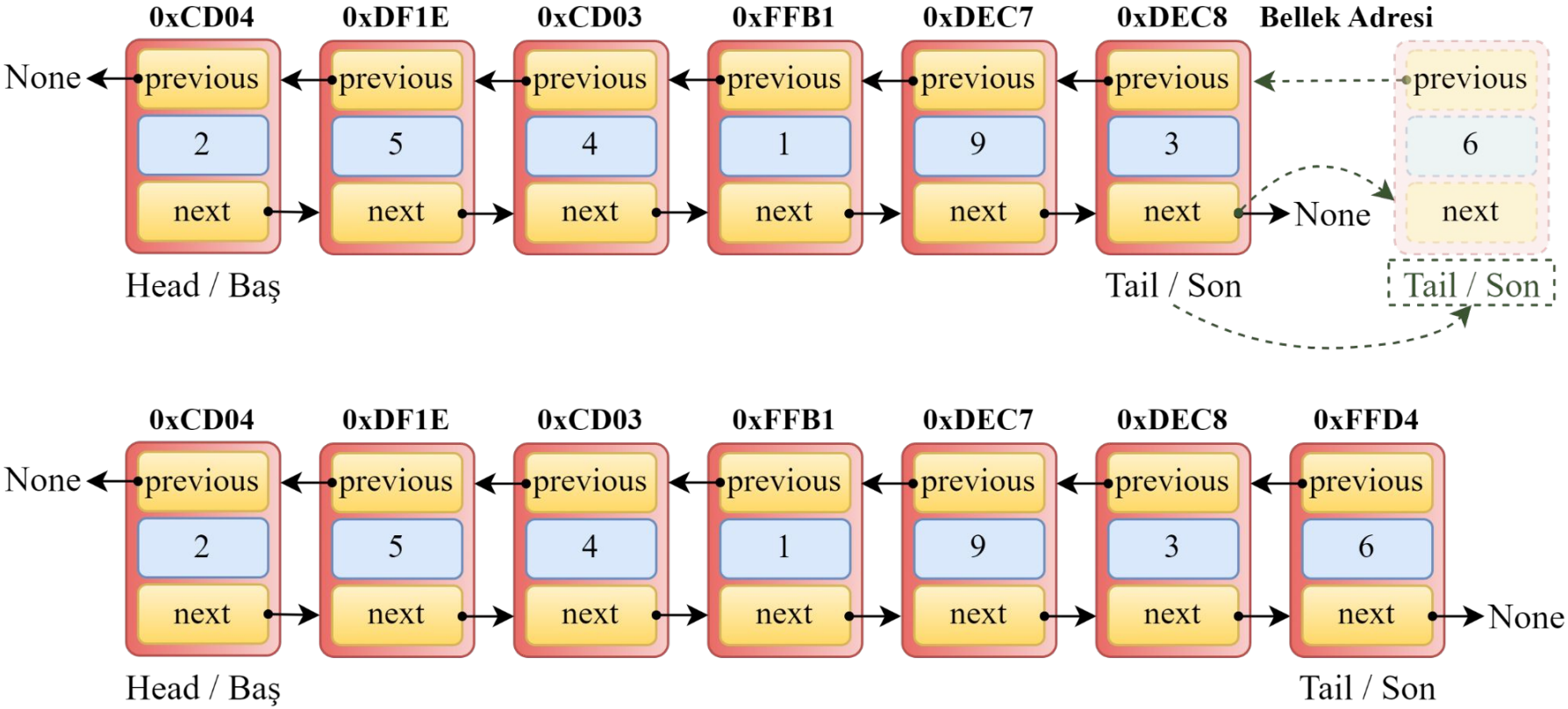


Çift Yönlü Bağlı Liste - Araya Öğe Ekleme

Araya öğe eklemek için program?

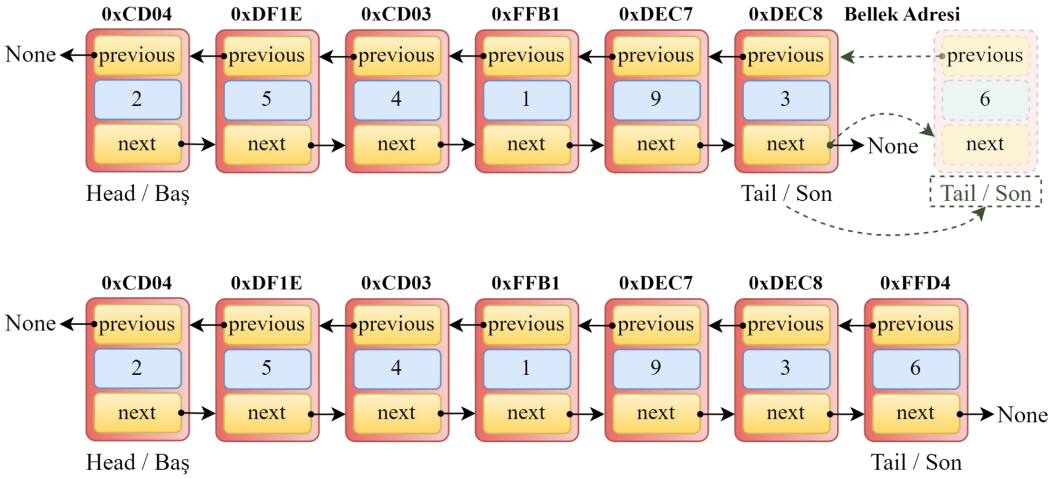


Çift Yönlü Bağlı Liste - Sona Öğe Ekleme

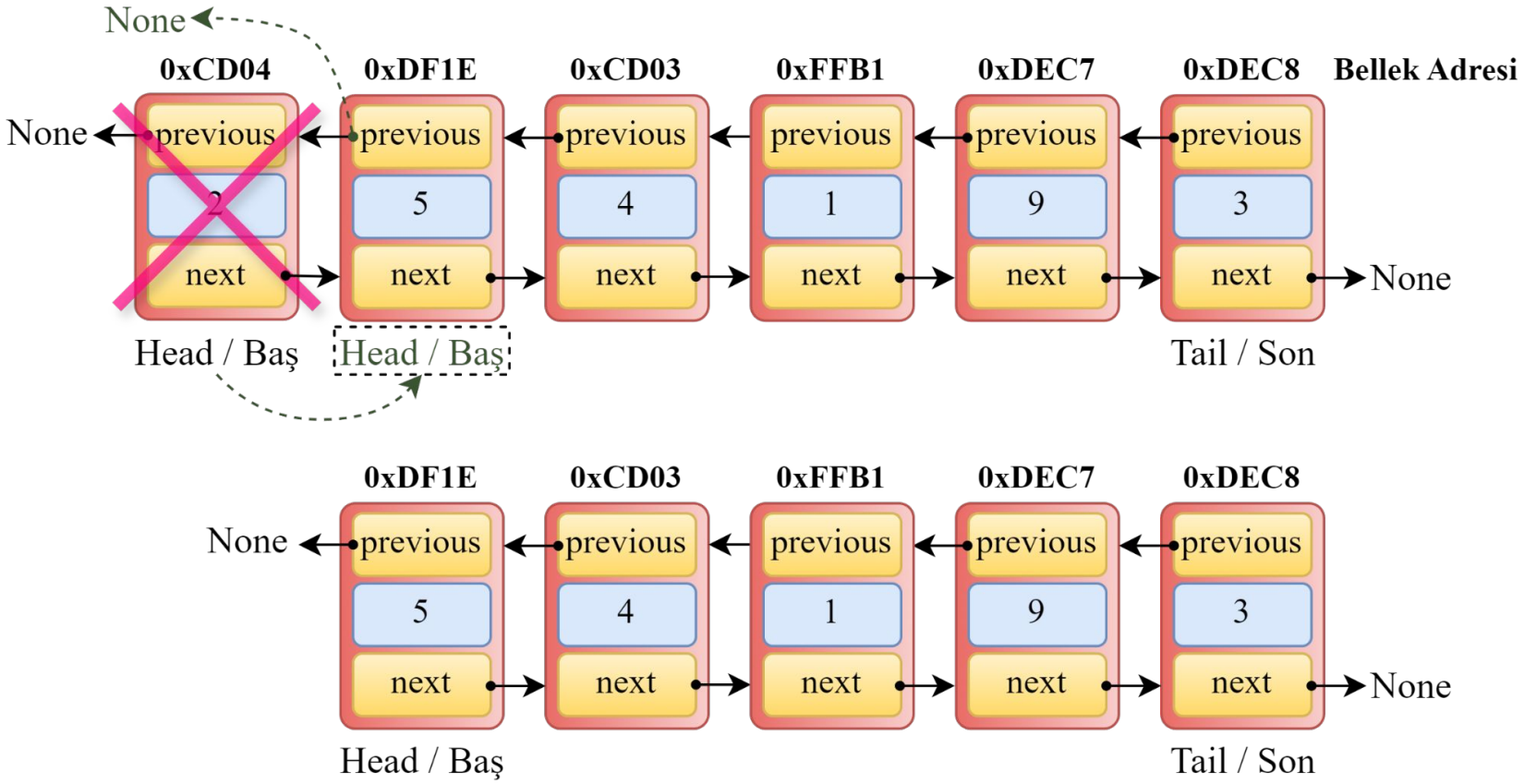


Çift Yönlü Bağlı Liste - Sona Öğe Ekleme

Sona öğe eklemek için program?

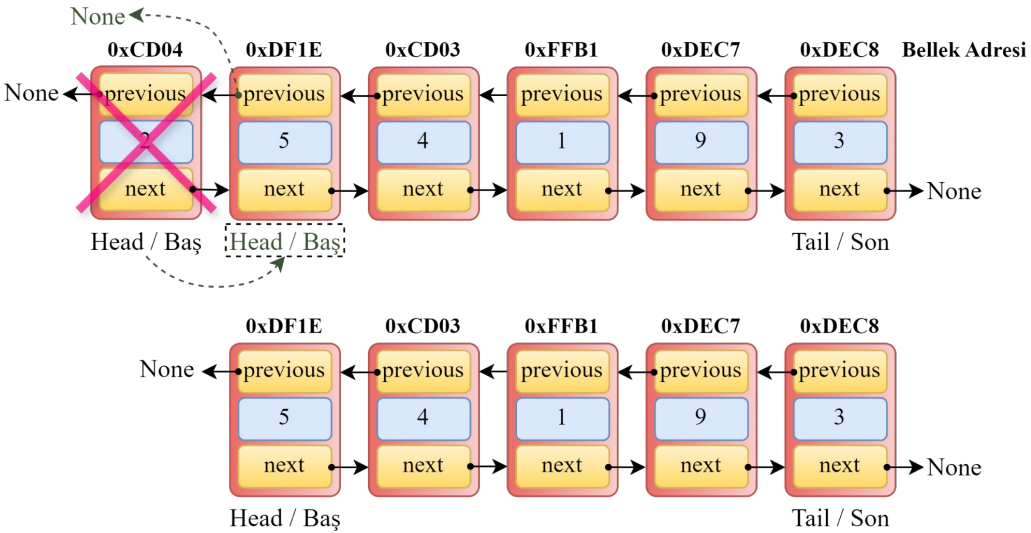


Çift Yönlü Bağlı Liste - Baştan Öğe Silme

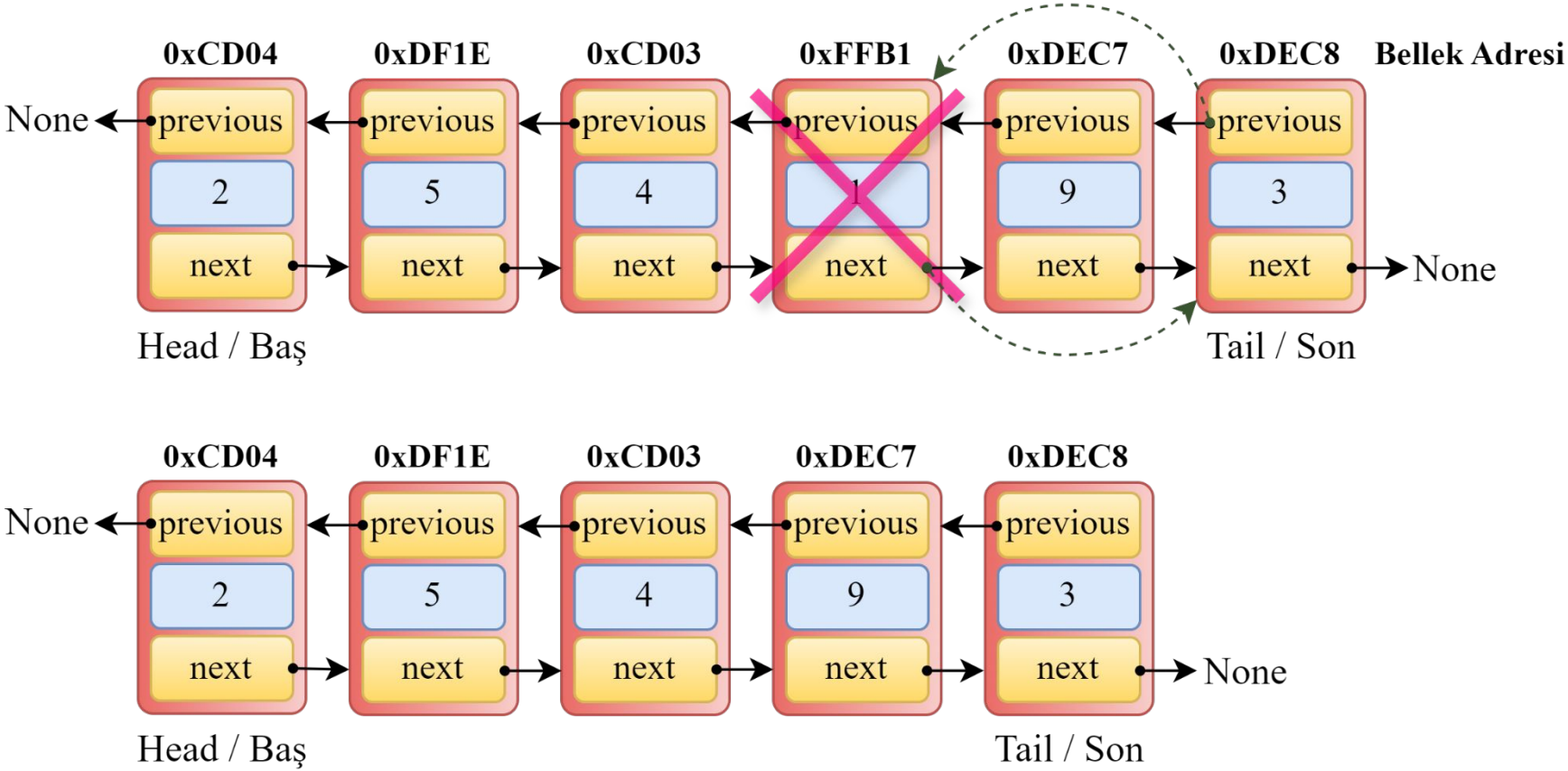


Çift Yönlü Bağlı Liste - Baştan Öğe Silme

Baştan öğe silmek için program?

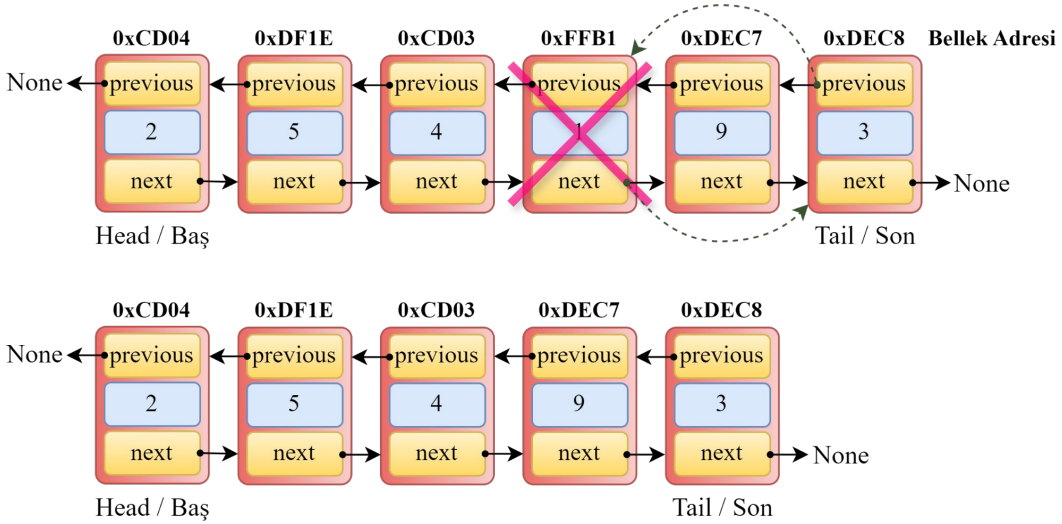


Çift Yönlü Bağlı Liste - Aradan Öğe Silme

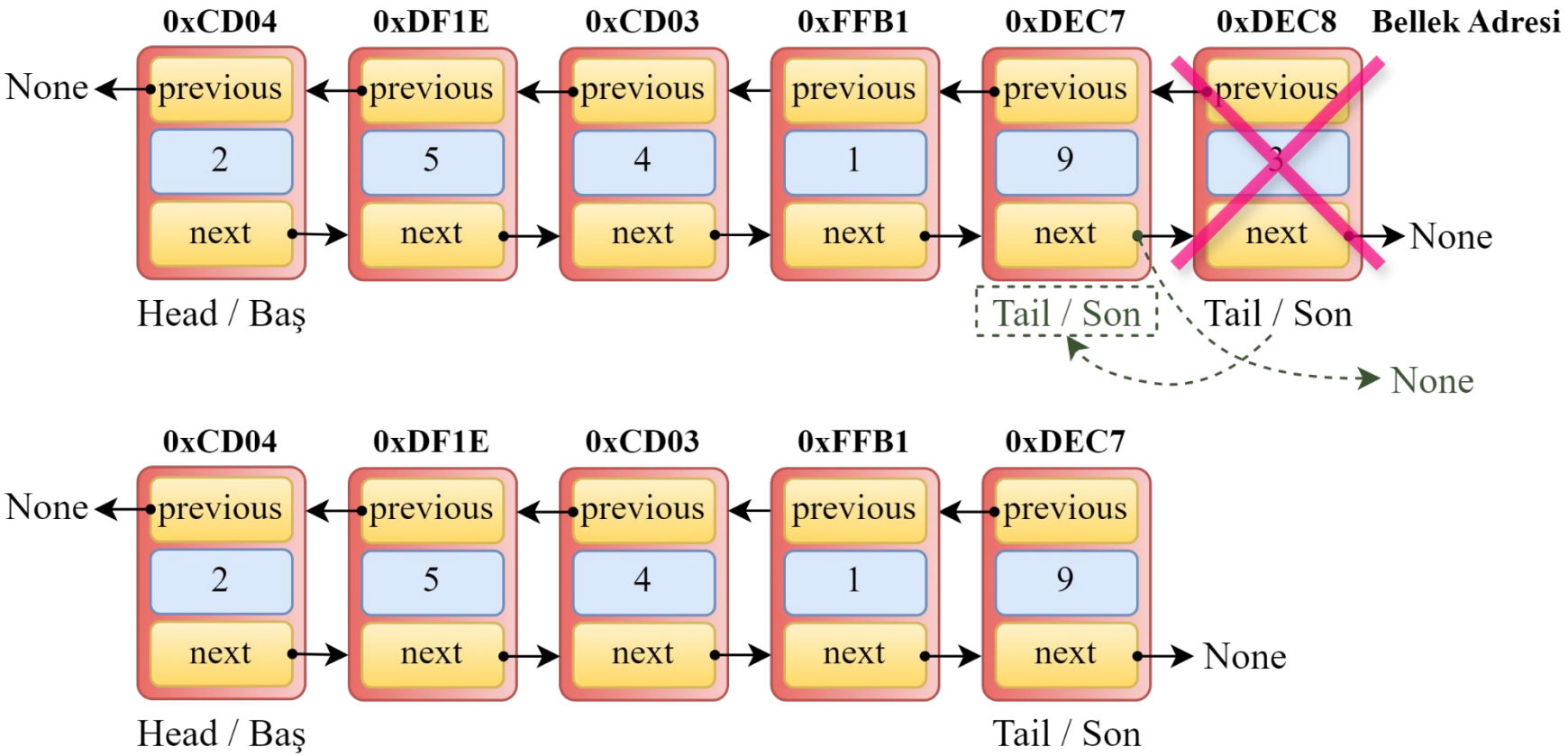


Çift Yönlü Bağlı Liste - Aradan Öğe Silme

Aradan öğe silmek için program?

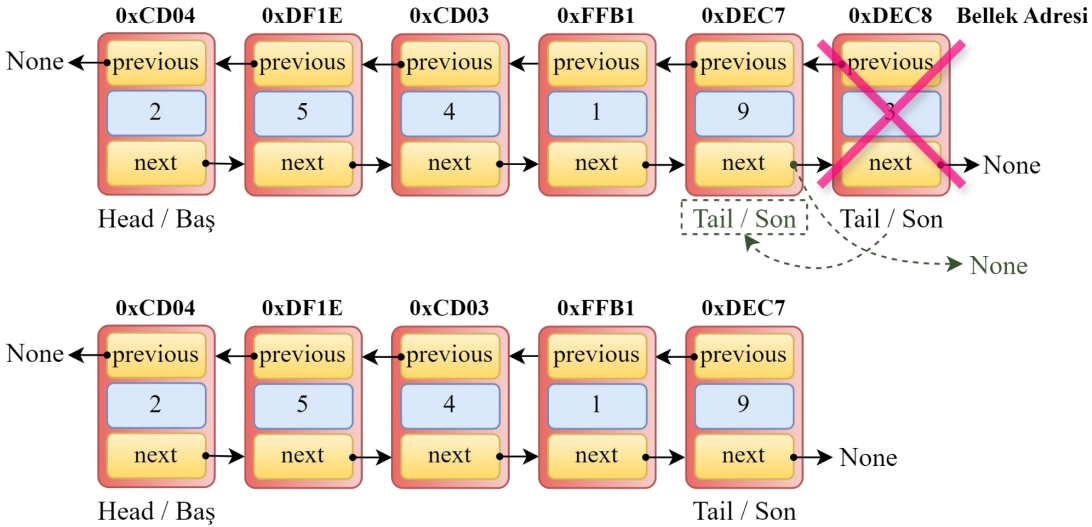


Çift Yönlü Bağlı Liste - Sondan Öğe Silme



Çift Yönlü Bağlı Liste - Sondan Öğe Silme

Sondan öğe silmek için program?



Çift Yönlü Bağlı Liste - Uygulama

```
Çift Yönlü Bağlı Liste'yi Tüm Özellikler ve İşlemleriyle Bir class ile
gerçekleştiriniz
```

```
class CYListe:

…
```