Algoritmalar

Doğrusal Zamanda Sıralama

Doğrusal Zamanda Sıralama

- Daha önce gördüğümüz tüm sıralama algoritmaları karşılaştırma tabanlı sıralama algoritmalarıdır ve bu tür algoritmalarda elde edilebilecek en düşük çalışma zamanı $\Omega(n\lg n)$'dir.
- Bu değerden daha düşük bir zamanda sıralama yapılması gerekiyorsa kullanılacak yöntemin karşılaştırma tabanlı olmaması gerekmektedir.
- Ayrıca karşılaştırma yapmadan sıralama yapabilmek için bazı özel şartların sağlanması gerekmektedir.
- Doğrusal zamanda sıralama yapan algoritmaların en bilinen örneği Sayma Sıralama algoritmasıdır (Counting Sort).

Sayma Sıralama Algoritması

- Sayma Sıralama algoritmasının doğrusal zamanda çalışabilmesi için sıralanacak olan **dizideki elamanların değerlerinin dizinin boyutundan büyük olmaması** gerekmektedir.
- Bir başka ifade ile eğer sıralanacak olan dizi ***x*** elemandan oluşuyorsa dizi içindeki sayıların her biri **0** ve ***x*** arasında olmalıdır.
- Ayrıca Sayma Sıralaması sıralama işlemi için ek yere ihtiyaç duymaktadır.
- Bu şartlar sağlandığında Sayma Sıralaması algoritması O(*n*) çalışma zamanına sahip olur.

Sayma Sıralama Algoritması

- Algoritmanın çalışma mantığı oldukça basittir.
- Öncelikle, sıralanacak dizideki her bir *k* değeri için dizide *k*'den küçük kaç tane sayı olduğunu bulunur.
- Bu bilgi ile *k* değerine sahip elemanın sıralı dizi içindeki yeri belirlenmiş olur. Örneğin dizi içerisinde *k* değerinden küçük 5 tane sayı varsa *k* değeri sıralanmış dizi içinde 6. sırada yer alacaktır.
- Bu noktada algoritmanın aynı değere sahip değerleri sıralarken hepsini aynı yere koymaması gerektiği açıktır.

Sayma Sıralama Algoritması

- Sayma Sıralama algoritmasının sözde kodu aşağıda verilmiştir.
- Sözde koddan görüleceği üzere algoritma sayılan değerlerin tutulması ve sıralanmış dizi için ekstra iki diziye (C ve B dizisi) ihtiyaç duymaktadır.
- İlk iki adımda C dizisi 0 değerleri ile doldurulur. 4. adımda $A[j]$ değerinden kaç tane olduğunu C dizisindeki $A[j]$ indeksli pozisyona yerleştirilir.
- C dizisi tamamen dolduktan sonra 6. adımda ise C dizisi kümülatif hale getirilir.
- 8. Adımda $A[j]$ değerleri C dizisindeki sayma değerlerine göre B dizisindeki sıralı yerine alınır.

Sayma Sıralaması (*A*, *B*, *k*)

Adım 1. **for** $i \leftarrow 1$ to k **do**

Adım 2. $\quad C[i] \leftarrow 0$

Adım 3. **for** $j \leftarrow 1$ to n **do**

Adım 4. $\quad C[A[j]] \leftarrow C[A[j]] + 1$

Adım 5. **for** $i \leftarrow 2$ to k **do**

Adım 6. $\quad C[i] \leftarrow C[i] + C[i-1]$

Adım 7. **for** $j \leftarrow n$ **downto** 1 **do**

Adım 8. $\quad B[C[A[j]]] \leftarrow A[j]$

Adım 9. $\quad C[A[j]] \leftarrow C[A[j]] - 1$

Örnek – Sayma Sıralaması

- Aşağıda verilen A dizisinin Sayma Sıralaması algoritması ile sıralanması isteniyor.

A dizisi:

1	2	3	4	5	6	7	8
3	5	4	1	3	4	0	4

C Dizisi:

0	1	2	3	4	5
0	0	0	0	0	0

- Önce adım 3 ve 4 yapılan A[j] değerinden kaç tane olduğunun C dizisindeki *A*[*j*] indeksli pozisyona yerleştirmesi yapılır.

Örnek – Sayma Sıralaması

- j=8 olduğunda C dizisi aşağıdaki şekilde olur:

j=8

0	1	2	3	4	5
1	1	0	2	3	1

Örnek – Sayma Sıralaması

- Daha sonra 5. ve 6. Adımlarda olduğu gibi C dizisi kümülatif hale getirilir. Bu sayede A dizisinde *k* değerine eşit ve *k*'den küçük kaç tane değer olduğu ortaya çıkarılmış olur.

Kümülatif C dizisi:

0	1	2	3	4	5
1	2	2	4	7	8

- Diziden anlaşılacağı görüleceği üzere *A* dizisinde 3 den küçük 2 değer vardır, veya 5 den küçük değer sayısı 7'dir.
- Son aşamada C dizisinden elde edilen bilgiye göre *A* dizisi *B* dizisine sıralı olarak aktarılır.

j=8 iterasyonu

A dizisi:

1	2	3	4	5	6	7	8
3	5	4	1	3	4	0	4

8. Adımda

B[C[A[8]]] ← A[8]

B[C[4]] ← A[8]

B[7] ← A[8]

B[7] ← 4

B dizisi:

1	2	3	4	5	6	7	8
						4	

C Dizisi:

0	1	2	3	4	5
1	2	2	4	7	8

9. Adımda

C[A[8]] ← C[A[8]] – 1

C[4] ← C[4] - 1

C[4] ← 7 – 1

C[4] ← 6

j=7 iterasyonu

A dizisi:

1	2	3	4	5	6	7	8
3	5	4	1	3	4	0	4

8. Adımda

B[C[A[7]]] ← A[7]

B[C[0]] ← A[7]

B[1] ← A[7]

B[7] ← 0

B dizisi:

1	2	3	4	5	6	7	8
0						4	

C Dizisi:

0	1	2	3	4	5
1	2	2	4	6	8

9. Adımda

C[A[7]] ← C[A[7]] – 1

C[0] ← C[0] - 1

C[0] ← 1 – 1

C[0] ← 0

Sayma Sıralama Algoritması

- Aynı işlemler j=1'e kadar yapıldığında *A* dizisi aşağıdaki gibi sıralanmış olur.
- Sayma Sıralamasının bir başka özelliği ise kararlı (stable) bir sıralama algoritması olmasıdır.
- Kararlı sıralama algoritmaları sıralama esnasında aynı değere sahip elamanların sırasının orijinal dizideki sıra ile aynı olmasını garanti eder.
- Sayma Sıralamasında eğer *A* dizisinde birden çok *x* değeri varsa, *A* dizisinde ilk görülen *x* değeri *B* dizisinde de ilk olarak görülmektedir.
- Bu özellikle sıralanan değerlere bağlı başka verilerin olduğu durumlar için önemlidir.

Sayma Sıralaması Çalışma Zamanı Analizi

Algoritmanın çalışma zamanı analizi oldukça basittir, eğer A dizisinde n eleman bulunuyorsa ve bu elamanların hepsi k değerinden küçük veya eşitse;

1. ve 2. adımlardaki döngü $O(k)$ zaman alacaktır
3. ve 4. adımlardaki döngü $O(n)$ zaman alacaktır
5. ve 6. adımlardaki döngü $O(k)$ zaman alacaktır
7. ve 9. adımlardaki döngü $O(n)$ zaman alacaktır

Bu nedenle Sayma Sıralamasının toplam çalışma zamanı

$$O(k) + O(n) + O(k) + O(n) = O(k + n)$$

değerine eşit olur. k değeri n değerinden çok daha küçük olacağı için bu ifade $O(n)$ olarak kısaltılabilir.

Basamak Sıralaması

- Radix sort olarak bilinir
- İlk olarak IBM makinalarında kullanılan delikli kartlarda kullanılmıştır
  - Sayım sonuçları 1890

Basamak Sıralaması

- Ana fikir: en sağ basamaktan başlamak üzere tekrarlı olarak basamak bazında sıralama yap
- Her basamaktaki en büyük değer 9 olabilir
- Basamaklar «kararlı» bir şekilde sıralanmalıdır
  - Sayma sıralaması?
- Basamak sayısı kadar sayma sıralaması yapılarak dizi doğrusal zamanda sıralanabilir.

Örnek

3	2	9		7	2	0		7	2	0		3	2	9
4	5	7		3	5	5		3	2	9		3	5	5
6	5	7		4	3	6		4	3	6		4	3	6
8	3	9	→	4	5	7	→	8	3	9	→	4	5	7
4	3	6		6	5	7		3	5	5		6	5	7
7	2	0		3	2	9		4	5	7		7	2	0
3	5	5		8	3	9		6	5	7		8	3	9
		sort				sorted			sorted				sorted	

Çalışma zamanı

- Basamaklar için sayma sıralaması kullanılabilir.
  - Doğrusal zamanda çalışır
  - Kararlı bir sıralama algoritmasıdır
- Her basmak $\mathrm{O}(n)$ zamanda sıralanır.
- d basamak varsa $\mathrm{O}(dn)$ zaman alır.
- $\mathrm{O}(n)$ olarak sadeleştirilebilir.