YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

BLM-1012—YAPISAL PROGRAMLAMAYA GİRİŞ FİNAL PROJESİ

BİTONİC SORT

Ad Soyad : **Ali Kaan TUNÇEL**
Öğrenci no : [redacted]
Öğrenci mail : [redacted]
Ders yürütücüsü : [redacted]

İÇİNDEKİLER

Videolu anlatım linki :

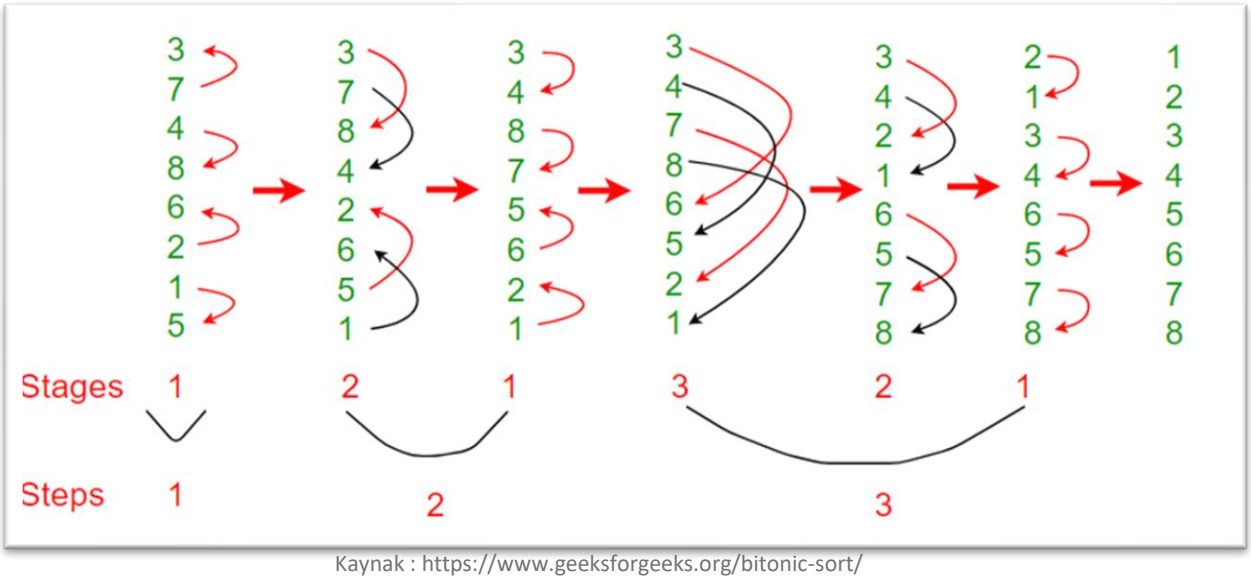
https://www.youtube.com[redacted]

Bitonic Sort nedir? Ne işe yarar? Nasıl Çalışır ?

- Bitonic sort bir sıralama algoritmasıdır. Karışık şekilde verilen bir diziyi sıralamakta kullanılır.
- Bitonic sort ‘bitonic sequence’ şeklinde isimlendirilen diziler üzerinde çalışır.

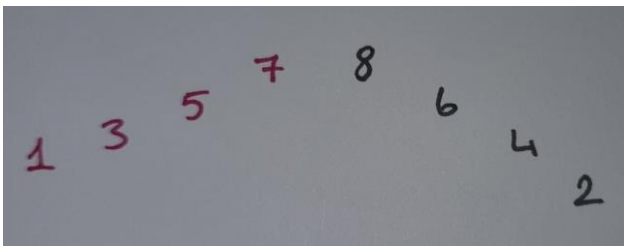
Bitonic sequence:
Bitonic sequence bir dizinin önce artan daha sonra azalan formda ilerlemesidir.
Ör : 1, 3, 5, 8, 6, 4, 2 ➔ bir bitonic sequencedir.

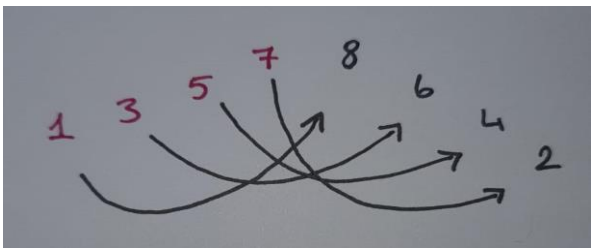
- Algoritamamız ilk yarısı artan ikinci yarısı azalan şekildeki dizileri sıralamaktadır.
- Bunun için karışık verilen diziyi 2 parçaya bölüp ilk kısmını artan ikinci kısmını azalan yönde sıralamak üzere recrusive biçimde çalışır.
- Eleman sayısı 2 olana kadar kendini recrusive olarak çağırır daha sonra alttaki işlemi gerçekleştirir.
- Diziyi iki parça olarak düşünürsek, sıralama yaparken artan kısmın n. Elemanı ile azalan kısmın n. elemanını kıyaslayarak gerekirse yer değiştirir.

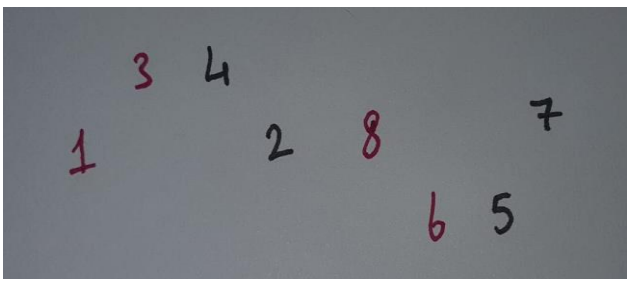


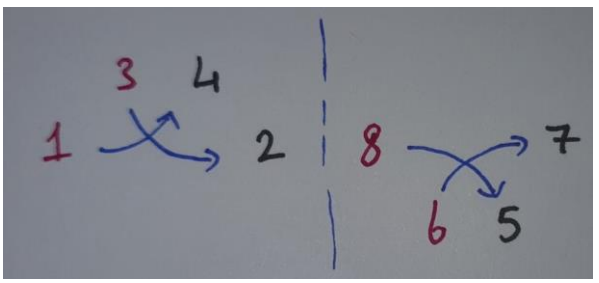
Kaynak : https://www.geeksforgeeks.org/bitonic-sort/

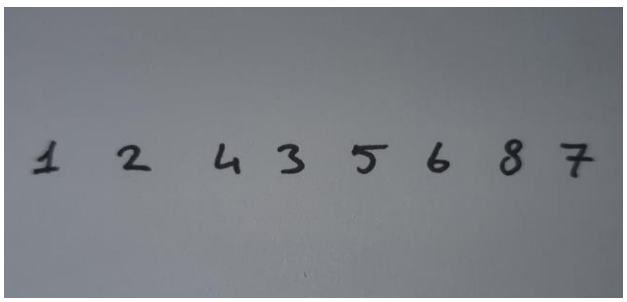
- Yukarıda 8 elemanlı bir dizinin önce bitonic sequence haline getirilip daha sonra sıralandığını görebiliriz.

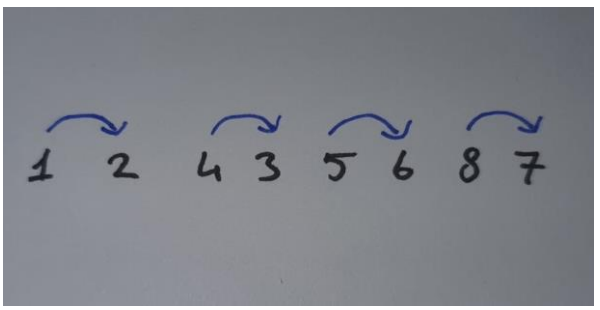


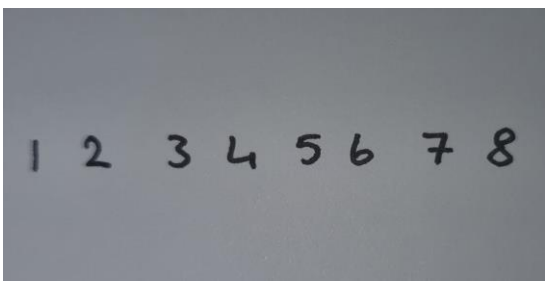












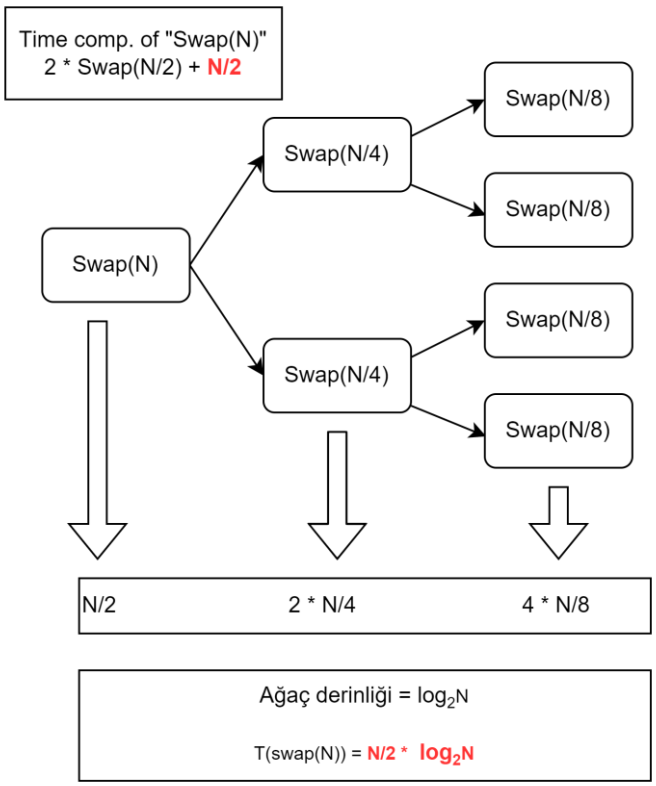
Kullanım yerleri, avantaj ve dezavantajları

- Bitonic sort, paralel bir algoritmadır. Bu nedenle ‘paralel computing’ yapmak üzere birden fazla işlemcinin kullanıldığı sistemlerde kullanılmaya uygundur.
- Özelliklerinde eleman sayısı 2 olana kadar 2’ye bölünerek recrusive olarak kendini çağıracağını belirtmiştim. Bunun yapılabilmesinin yolu sıralanacak olan dizinin eleman sayısının 2’nin tam kuvveti olmasıdır.
- Time complexity’si mergesort ve quicksort gibi algoritmalardan daha büyüktür. Fakat space complexity’si daha düşüktür.

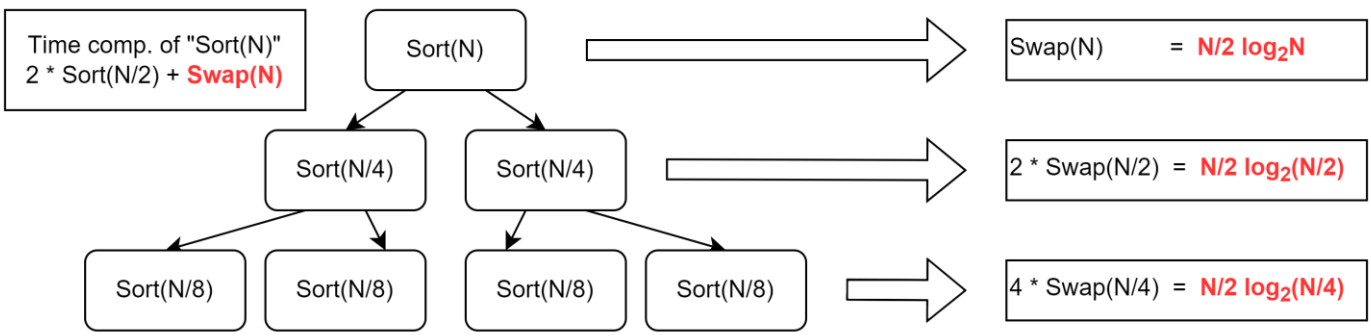
Name	Best	Average	Worst	Memory
Quicksort	$n \log n$	$n \log n$	n^2	$\log n$
Merge sort	$n \log n$	$n \log n$	$n \log n$	n
Bitonic sorter	$\log^2 n$ parallel	$\log^2 n$ parallel	$n \log^2 n$ non-parallel	1

Kaynak : https://en.wikipedia.org/wiki/Sorting_algorithm#:~:text=In%20computer%20science%2C%20a%20sorting,and%20either%20ascending%20or%20descending.

Karmaşıklık Analizi



- Solda swapper fonksiyonu için zaman karmaşıklığı bulunmaktadır.
- Altta ise bu fonksiyondan gelen değer ile sorter fonksiyonu hesaplanmaktadır.



$$T(Sort(N)) = N/2 * (\log_2 N + \log_2(N/2) + \ldots + \log_2 2) = N/2 * \frac{(\log_2 N) * (\log_2 N + 1)}{2}$$

Bitonic sort zaman karmaşıklığı = $\frac{N}{4}\left[\log_2^2(N) + \log_2(N)\right]$

C dilinde bitonic sort kodu

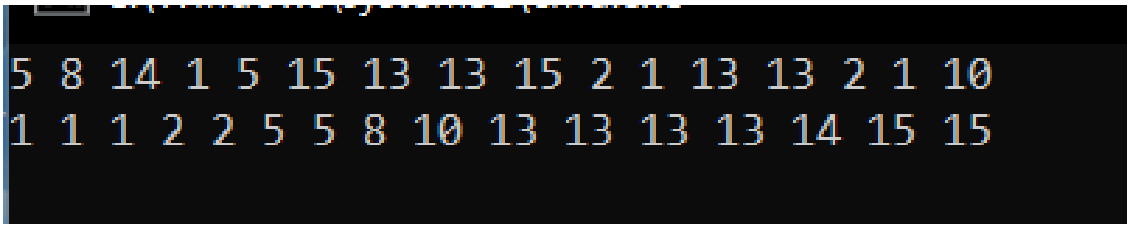
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 100000

int main() {
    srand(time(NULL));

    int i;
    int A[MAX];  //{4, 15, 14, 3, 9, 7, 12, 17, 8, 5, 16, 11, 1, 2, 10, 6}
    int N = 16;

    random(A, N);
    diziYaz(A, N);  // Sirali olmayan diziyi yaz.

    sorter(A, N, 0, 1);  // Artan yonde siralar.}

    diziYaz(A, N);  // Sirali diziyi yaz.

    return 0;
}
```

```c
// artan parametresi = 1 icin artan dizi, = 0 icin azalan dizi olsuturur.
void sorter(int A[], int N, int konum, int artan) {
    if (N != 2) {  // Diziyi once artan sonra azalan sekline getirir. Kisaca bitonic dizi olsuturur.

        sorter(A, N / 2, konum, 1);          // ilk yarisini artan yonde sirala
        sorter(A, N / 2, konum + N / 2, 0);  // ikinci yarisini azalan yonde sirala
    }

    swapper(A, N, konum, artan);  // Ust fonksiyonu ile ayni yonde(artan-azalan) swapper cagir.
}

void swapper(int A[], int N, int konum, int artan) {
    int i, tmp;

    for (i = 0; i < N / 2; i++) {
        if (!((A[konum + i] > A[konum + i + N / 2]) ^ (artan == 1))) {  // Xor gate ile swap gerekli mi kontrolu.
            tmp = A[konum + i];
            A[konum + i] = A[konum + i + N / 2];
            A[konum + i + N / 2] = tmp;
        }
    }
    if (N != 2) {
        swapper(A, N / 2, konum, artan);          // Ust fonksiyonu ile ayni yonde swap islemi uygula
        swapper(A, N / 2, konum + N / 2, artan);  // Ust fonksiyonu ile ayni yonde ikinci bolume swap islemi uygula
    }
}
```

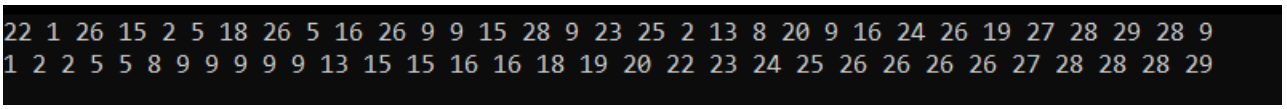
```c
void diziYaz(int A[], int N) {  // Dizi yazdirma
    int i;
    for (i = 0; i < N; i++) {
        printf("%d ", A[i]);
    }
    printf("\n");
}

void random(int A[], int N) {
    int i;
    for (i = 0; i < N; i++) {  // Random dizi olustur.
        A[i] = rand() % N + 1;
    }
}
```

Ekran görüntüleri

16 eleman için



32 eleman için



Kaynaklar

- https://people.cs.rutgers.edu/~venugopa/parallel_summer2012/bitonic_overview.html#:~:text=Bitonic%20sort%20is%20a%20comparison,bitonic%20sequence%20are%20also%20bitonic
- https://www.geeksforgeeks.org/bitonic-sort/
- https://www.youtube.com/watch?v=uEfieI0MumY
- https://sortvisualizer.com/bitonicsort/
- https://en.wikipedia.org/wiki/Bitonic_sorter