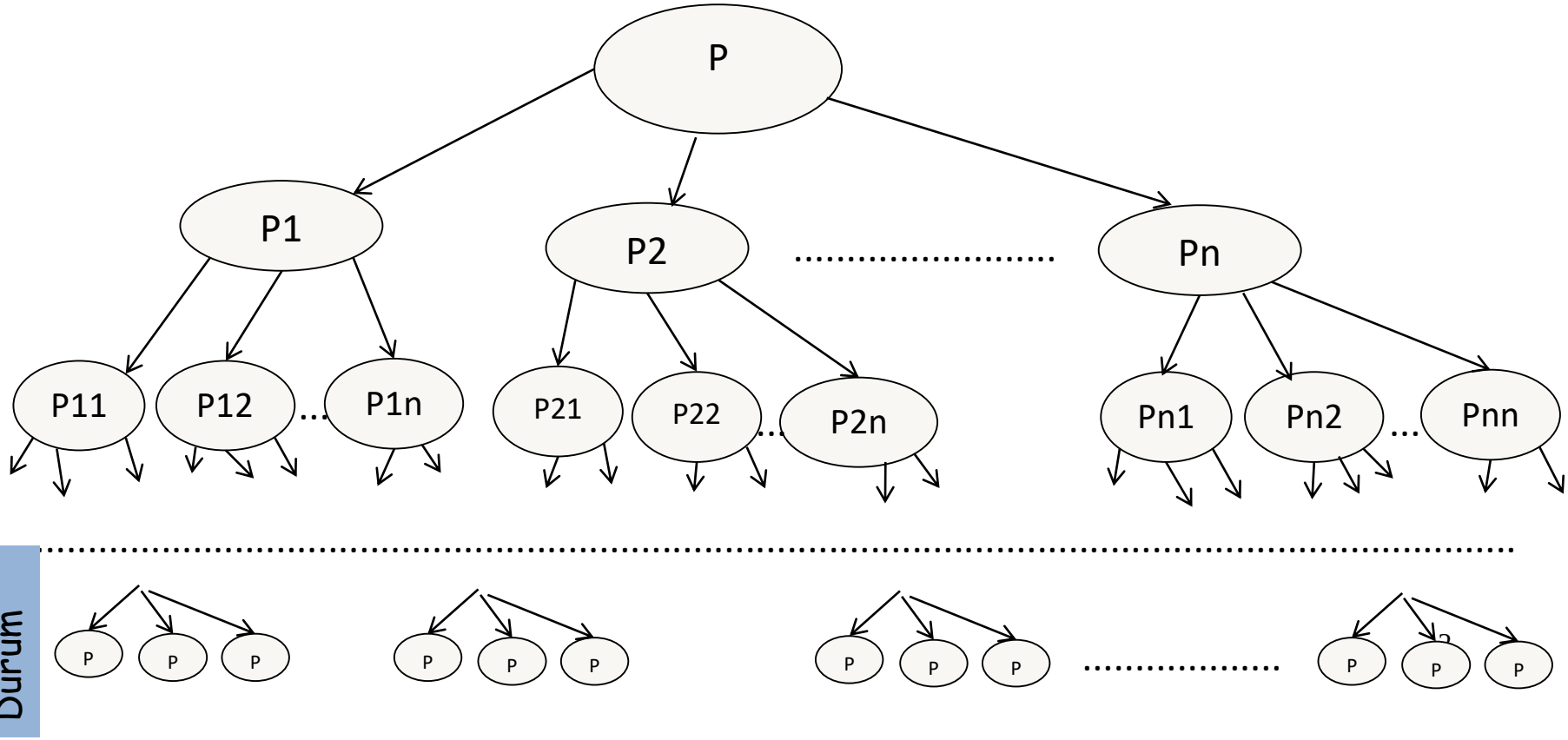
Sıralama Algoritmaları 2

Quick Sort
Merge Sort

Böl & Yönet Stratejisi

1. Problemi küçük parçalara böl
2. Her bir parçayı bağımsız şekilde çöz
3. Parçaları birleştirerek ana problemin çözümüne ulaş



Quick Sort

Bu algoritmada böl ve yönet (divide and conquer) politikasına dayanır, sıralanması istenen dizi belirli bir sınır (pivot) değerine göre iki alt diziye ayrılır. İkiye ayırma işlemi pivottan küçük elemanlar pivotun soluna, pivottan büyük elemanlar pivotun sağına yerleşecek şekilde yer değiştirme işlemi kullanılarak sağlanır. Daha sonra pivotun sağı ve solu olmak üzere elde edilmiş olan 2 dizi içinde yine bir pivot değeri belirlenerek bu işlem diziler parçalanmayacak duruma gelene kadar sürdürülür. En iyi durumu n log n, en kötü durumu n^2 çalışan bir algoritmadır.

Quick Sort

```
void Quicksort(int dizi[],int alt,int ust)
{       int pivot_indisi;
        if (ust>alt)
        {       pivot_indisi=Partition(dizi,alt,ust);
                Quicksort(dizi,alt,pivot_indisi-1);//pivottan küçükler
                Quicksort(dizi,pivot_indisi+1,ust);//pivottan büyükler
        }
}
```

```
int Partition(int dizi[],int alt_indis,int ust_indis)
{       int i,j,pivot_degeri,pivot_indisi,gecici;
        pivot_degeri=dizi[alt_indis]; //pivot en soldaki eleman
        j=alt_indis;
        for(i=alt_indis+1;i<=ust_indis;i++)
        if (dizi[i]<pivot_degeri)  /*pivotun pozisyonu ayarlanıyor*/
        {       j++;
                gecici=dizi[i];dizi[i]=dizi[j];dizi[j]=gecici;
        }
        pivot_indisi=j;  //pivotun olması gereken yeri j. hücredir.
        gecici=dizi[alt_indis];
        dizi[alt_indis]=dizi[pivot_indisi];
        dizi[pivot_indisi]=gecici;
        return pivot_indisi;  //pivotun yeri döndürülür.
}
```

Merge Sort

Bu sıralama yönteminde böl ve yönet yaklaşımına dayanır. Sıralanması istenen dizi elemanları önce 2 alt diziye ayrılır. Alt kümelere ayırma işlemine alt kümelerdeki bir tane eleman kalana kadar devam edilir. Alt kümelerde 1 tane eleman kalmışsa rekürsif çağırma geriye doğru dönmeye başlar ve geri dönülürken alt kümeler elemanları sıralı olacak şekilde birleştirilir. Aynı girdi boyutu için her durumda aynı sürede (Every Case Running Time) çalışan bir algoritmadır. Her durumda n log n zamanda çalışmaktadır.

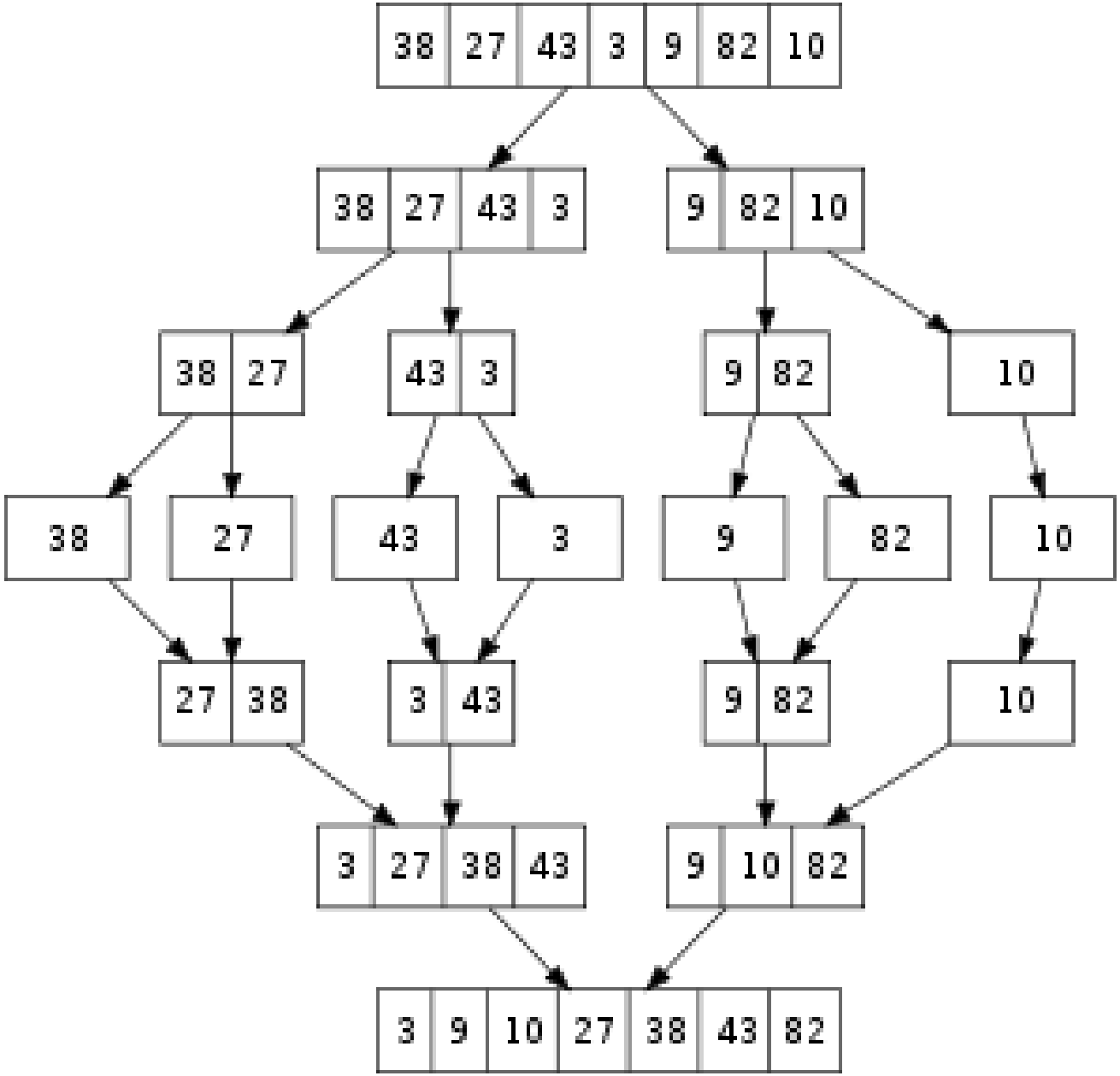
Birleştirmeli sıralama, çalışma zamanını iyileştirmek için şu iki fikir üzerine kuruludur:

1. Küçük bir listeyi sıralamak, büyük bir listeyi sıralamaktan daha az adımda yapılabilir.

2. Sıralı iki alt listeden bir sıralı liste elde etmek, sırasız iki alt listeden bir sıralı liste elde etmekten daha az adımda yapılabilir.

```
void Mergesort(int Dizi [], int ilk, int son)
{       if (ilk < son)
        {       int orta = (ilk + son) / 2;
                Mergesort(Dizi, ilk, orta); // dizi bölünüyor
                Mergesort(Dizi, orta + 1, son); // dizi bölünüyor
                Merge(Dizi, ilk, orta, son); // birleştiriliyor
        }
}
```

```
void Merge(int Dizi[], int ilk, int orta, int son)
{       int tempArray[1000]; // geçici dizi tanımlanıyor
        int ilk1 = ilk; int son1 = orta; // birinci altdizinin başlangıcı ve sonu
        int ilk2 = orta + 1; int son2 = son; // ikinci altdizinin başlangıcı ve sonu
        int index = ilk1; // geçici dizinin indisi ayarlanıyor
        for ( ; (ilk1 <= son1) && (ilk2 <= son2); index++) //Geçici diziye elemanlar sıralı alınıyor
        {       if (Dizi[ilk1] < Dizi[ilk2]) {   tempArray[index] = Dizi[ilk1];
                        ilk1++; }
                else {      tempArray[index] = Dizi[ilk2];
                        ilk2++; }
        }
        for (; ilk1 <= son1; ilk1++, index++) // birinci altdizinin elemanları bitmemişse
                tempArray[index] = Dizi[ilk1];
        for (; ilk2 <= son2; ilk2++, index++) // ikinci altdizinin elemanları bitmemişse
                tempArray[index] = Dizi[ilk2];
        for (index = ilk; index <= son; index++) //gecici dizi orjinal diziye kopyalanır(merge)
        Dizi[index] = tempArray[index];
}
```


38 27 43 3 9 82 10
38 27 43 3
9 82 10
38 27
43 3
9 82
10
38
27
43
3
9
82
10
27 38
3 43
9 82
10
3 27 38 43
9 10 82
3 9 10 27 38 43 82