Chương 1
THUẬT TOÁN MERGE SORT

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

1. BÀI TOÁN DẪN NHẬP

- Bài toán: Cho 2 mảng một chiều các số nguyên a và b được sắp thứ tự tăng dần. Hãy trộn mảng a và mảng b lại với nhau để được mảng c sắp thứ tự tăng dần.

1. BÀI TOÁN DẪN NHẬP

- Bài toán: Cho 2 mảng một chiều các số nguyên a và b được sắp thứ tự tăng dần. Hãy trộn mảng a và mảng b lại với nhau để được mảng c sắp thứ tự tăng dần.
- Ví dụ:

Mảng a	15	27	32	64

Mảng b	-5	1	18	29	37	56	80	93

Mảng c	-5	1	15	18	27	29	32	37	56	64	80	93

1. BÀI TOÁN DẪN NHẬP

Tron

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

1. BÀI TOÁN DẪN NHẬP

```
11.void Tron
```

1. BÀI TOÁN DẪN NHẬP

```
11.void Tron(int a[],int n
```

1. BÀI TOÁN DẪN NHẬP

```
11.void Tron(int a[],int n,
             int b[],int m
```

1. BÀI TOÁN DẪN NHẬP

```
11.void Tron(int a[],int n,
             int b[],int m,
             int c[],int  p)
```

1. BÀI TOÁN DẪN NHẬP

```
11.void Tron(int a[],int n,
             int b[],int m,
             int c[],int &p)
```

1. BÀI TOÁN DẪN NHẬP

```
void Tron(int a[],int n,
          int b[],int m,
          int c[],int &p)
{
}
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(                  )
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
void Tron(int a[],int n,
          int b[],int m,
          int c[],int &p)
{
    int i = 0;
    int j = 0;
    p = 0;
    while(                    )
    {

    }
}
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(                    )
17.     {
            if(              a[i]<b[j])
        }
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(                    )
17.     {
            if(              a[i]<b[j])


19.           c[p  ] = a[i  ];


        }
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(                    )
17.     {
            if(              a[i]<b[j])


19.           c[p++] = a[i   ];


        }
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(                    )
17.     {
            if(              a[i]<b[j])


19.           c[p++] = a[i++];


        }
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(                    )
17.     {
            if(              a[i]<b[j])


19.           c[p++] = a[i++];
20.         else

        }
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(                    )
17.     {
          if(              a[i]<b[j])


19.         c[p++] = a[i++];
20.       else
21.         c[p  ] = b[j  ];
        }
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(                    )
17.     {
          if(              a[i]<b[j])


19.         c[p++] = a[i++];
20.       else
21.         c[p++] = b[j  ];
        }
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(                    )
17.     {
            if(              a[i]<b[j])


19.           c[p++] = a[i++];
20.         else
21.           c[p++] = b[j++];
        }
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(  i>=n          )
17.     {
            if(           a[i]<b[j])


19.           c[p++] = a[i++];
20.         else
21.           c[p++] = b[j++];
        }
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(  i>=n     j>=m )
17.     {
          if(              a[i]<b[j])


19.         c[p++] = a[i++];
20.       else
21.         c[p++] = b[j++];
        }
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(  i>=n && j>=m )
17.     {
            if(              a[i]<b[j])

19.           c[p++] = a[i++];
20.         else
21.           c[p++] = b[j++];
        }
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while( (i>=n && j>=m))
17.     {
            if(              a[i]<b[j])


19.           c[p++] = a[i++];
20.         else
21.           c[p++] = b[j++];
        }
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(!(i>=n && j>=m))
17.     {
            if(              a[i]<b[j])

19.           c[p++] = a[i++];
20.         else
21.           c[p++] = b[j++];
        }
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(!(i>=n && j>=m))
17.     {
            if((i<n&&j<m&&a[i]<b[j])

19.           c[p++] = a[i++];
20.         else
21.           c[p++] = b[j++];
        }
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(!(i>=n && j>=m))
17.     {
            if((i<n&&j<m&&a[i]<b[j])
                ||          )
19.           c[p++] = a[i++];
20.         else
21.           c[p++] = b[j++];
        }
23. }
```

1. BÀI TOÁN DẪN NHẬP

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.     int i = 0;
14.     int j = 0;
15.     p = 0;
16.     while(!(i>=n && j>=m))
17.     {
18.       if((i<n&&j<m&&a[i]<b[j])
              ||(j>=m))
19.         c[p++] = a[i++];
20.       else
21.         c[p++] = b[j++];
22.     }
23. }
```

2. Ý TƯỞNG THUẬT TOÁN

- Thuật toán Merge Sort chia không gian cần sắp xếp thành 2 không gian con.
  - Nếu không gian con thứ nhất có nhiều hơn một phần tử thì sắp xếp không gian con này bằng thuật toán Merge Sort.
  - Nếu không gian con thứ hai có nhiều hơn một phần tử thì sắp xếp không gian con này bằng thuật toán Merge Sort.
- Trộn 2 không gian con đã được sắp xếp lại với nhau.

3. ÁP DỤNG THUẬT TOÁN

78 53 62 59 89 11 29 50

78 53 62 59 89 11 29 50

78 53 62 59 89 11 29 50

78 53 62 59 89 11 29 50

53 78 59 62 11 89 29 50

53 59 62 78 11 29 50 89

11 29 50 53 59 62 78 89

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

3. ÁP DỤNG THUẬT TOÁN

78 53 62 59 89 11 29 50

78 53 62 59 89 11 29 50

78 53 62 59 89 11 29 50

78 53 62 59 89 11 29 50

53 78 59 62 11 89 29 50

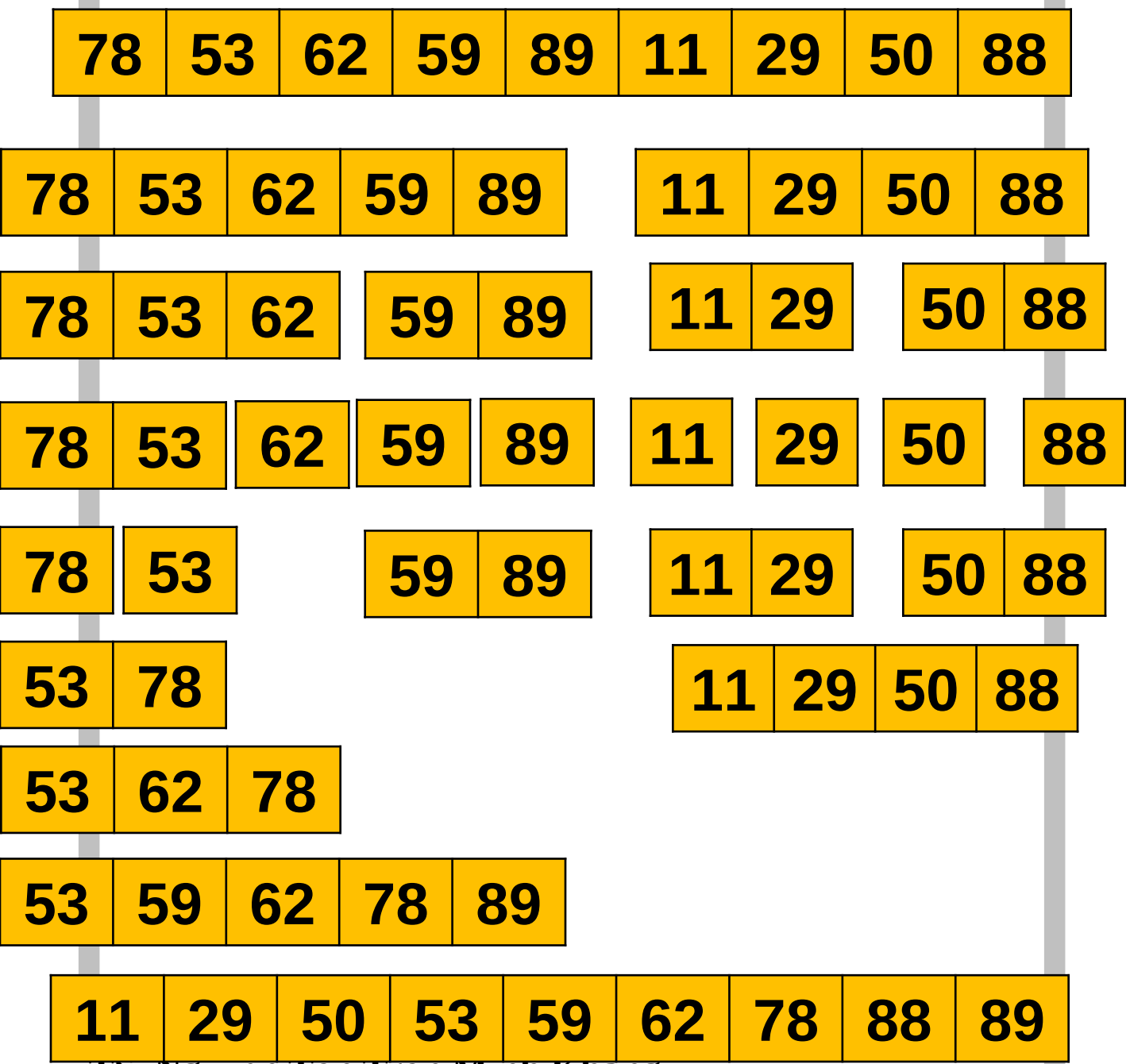
53 59 62 78 11 29 50 89

11 29 50 53 59 62 78 89

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

3. ÁP DỤNG THUẬT TOÁN



4. HÀM CÀI ĐẶT

```
1. void SapTang(int a[], int n)
2. {
3.     MergeSort(a,0,n-1);
4. }
```

4. HÀM CÀI ĐẶT

```
11. void MergeSort(int a[],
                   int Left,int Right)
12. {
13.   if(Left<Right)
14.   //Mảng có nhiều hơn 1 phần tử
15.   {
16.     int Mid = (Left+Right)/2;
17.     // Sắp xếp mảng bên trái
18.     MergeSort(a,Left,Mid);
19.     // Sắp xếp mảng bên phải
20.     MergeSort(a,Mid+1,Right);
21.     // Trộn 2 mảng lại với nhau
22.     Merge(a,Left,Mid,Right);
23.   }
24. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int b[100];int k;
14.   int c[100];int ℓ;
15.   int temp[100];int n;
16.   k = 0;
17.   for(int i=Left;i<=Mid;i++)
18.     b[k++] = a[i];
19.   ℓ = 0;
20.   for(int i=Mid+1;i<=Right;i++)
21.     c[ℓ++] = a[i];
22.   Tron(b,k,c,ℓ,temp,n);
23.   for(int i=0;i<n;i++)
24.     a[Left+i] = temp[i];
25. }
```

4. HÀM CÀI ĐẶT

```
11. void Tron(int a[],int n,
              int b[],int m,
              int c[],int &p)
12. {
13.   int i = 0;
14.   int j = 0;
15.   p = 0;
16.   while(!(i>=n && j>=m))
17.   {
18.     if((i<n&&j<m&&a[i]<b[j])||
           (j>=m))
19.       c[p++] = a[i++];
20.     else
21.       c[p++] = b[j++];
22.   }
23. }
```

5. CẢI TIẾN HÀM CÀI ĐẶT

```
1. void SapTang(int a[], int n)
2. {
3.     MergeSort(a,0,n-1);
4. }
```

5. CẢI TIẾN HÀM CÀI ĐẶT

```
11. void MergeSort(int a[],
                   int Left,int Right)
12. {
13.   if(Left<Right)
14.   //Đoạn có nhiều hơn 1 phần tử
15.   {
16.     int Mid = (Left+Right)/2;
17.     // Sắp xếp đoạn bên trái
18.     MergeSort(a,Left,Mid);
19.     // Sắp xếp đoạn bên phải
20.     MergeSort(a,Mid+1,Right);
21.     // Trộn 2 đoạn lại với nhau
22.     Merge(a,Left,Mid,Right);
23.   }
24. }
```

```
void Merge(int a[], int Left,
                int Mid,int Right)
{

}
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.     int temp[100];
14.     int n;
28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.     int temp[100];
14.     int n;

15.     int i = Left;//Left…Mid
28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.     int temp[100];
14.     int n;

15.     int i = Left;//Left…Mid
16.     int j = Mid+1;//Mid+1…Right

28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.     int temp[100];
14.     int n;

15.     int i = Left;//Left…Mid
16.     int j = Mid+1;//Mid+1…Right
17.     n = 0;

28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.     int temp[100];
14.     int n;

15.     int i = Left;//Left…Mid
16.     int j = Mid+1;//Mid+1…Right
17.     n = 0;
18.     while(                    )
19.     {
25.     }

28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(                    )
19.   {
20.     if(
21.        a[i]<a[j])




25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(                    )
19.   {
20.     if(
21.        a[i]<a[j])
22.       temp[n  ]=a[i  ];

25.   }

28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(                    )
19.   {
20.     if(
21.        a[i]<a[j])
22.       temp[n++]=a[i   ];




25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(                    )
19.   {
20.     if(
21.        a[i]<a[j])
22.       temp[n++]=a[i++];




25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(                    )
19.   {
20.     if(
21.        a[i]<a[j])
22.       temp[n++]=a[i++];
23.     else


25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(                    )
19.   {
20.     if(
21.        a[i]<a[j])
22.       temp[n++]=a[i++];
23.     else
24.       temp[n  ]=a[j  ];
25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(                    )
19.   {
20.     if(
21.        a[i]<a[j])
22.       temp[n++]=a[i++];
23.     else
24.       temp[n++]=a[j  ];
25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(                    )
19.   {
20.     if(
21.        a[i]<a[j])
22.       temp[n++]=a[i++];
23.     else
24.       temp[n++]=a[j++];
25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(                    )
19.   {
20.     if( i<=Mid
21.        a[i]<a[j])
22.       temp[n++]=a[i++];
23.     else
24.       temp[n++]=a[j++];
25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(                    )
19.   {
20.     if( i<=Mid&&
21.         a[i]<a[j])
22.       temp[n++]=a[i++];
23.     else
24.       temp[n++]=a[j++];
25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(                    )
19.   {
20.     if( i<=Mid&&j<=Right&&
21.         a[i]<a[j])
22.       temp[n++]=a[i++];
23.     else
24.       temp[n++]=a[j++];
25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(                    )
19.   {
20.     if((i<=Mid&&j<=Right&&
21.        a[i]<a[j]) ||
22.       temp[n++]=a[i++];
23.     else
24.       temp[n++]=a[j++];
25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(                    )
19.   {
20.     if((i<=Mid&&j<=Right&&
21.        a[i]<a[j]) || (j>Right))
22.       temp[n++]=a[i++];
23.     else
24.       temp[n++]=a[j++];
25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(  i>Mid                )
19.   {
20.     if((i<=Mid&&j<=Right&&
21.        a[i]<a[j]) || (j>Right))
22.       temp[n++]=a[i++];
23.     else
24.       temp[n++]=a[j++];
25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(  i>Mid    j>Right )
19.   {
20.     if((i<=Mid&&j<=Right&&
21.        a[i]<a[j]) || (j>Right))
22.       temp[n++]=a[i++];
23.     else
24.       temp[n++]=a[j++];
25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while( (i>Mid && j>Right))
19.   {
20.     if((i<=Mid&&j<=Right&&
21.        a[i]<a[j]) || (j>Right))
22.       temp[n++]=a[i++];
23.     else
24.       temp[n++]=a[j++];
25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(!(i>Mid && j>Right))
19.   {
20.     if((i<=Mid&&j<=Right&&
21.        a[i]<a[j]) || (j>Right))
22.       temp[n++]=a[i++];
23.     else
24.       temp[n++]=a[j++];
25.   }


28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(!(i>Mid && j>Right))
19.   {
20.     if((i<=Mid&&j<=Right&&
21.        a[i]<a[j]) || (j>Right))
22.       temp[n++]=a[i++];
23.     else
24.       temp[n++]=a[j++];
25.   }
26.   for(i=0;i<n;i++)

28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(!(i>Mid && j>Right))
19.   {
20.     if((i<=Mid&&j<=Right&&
21.        a[i]<a[j]) || (j>Right))
22.       temp[n++]=a[i++];
23.     else
24.       temp[n++]=a[j++];
25.   }
26.   for(i=0;i<n;i++)
27.     a[     +i] = temp[i];
28. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int temp[100];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(!(i>Mid && j>Right))
19.   {
20.     if((i<=Mid&&j<=Right&&
21.        a[i]<a[j]) || (j>Right))
22.       temp[n++]=a[i++];
23.     else
24.       temp[n++]=a[j++];
25.   }
26.   for(i=0;i<n;i++)
27.     a[Left+i] = temp[i];
28. }
```

6. TINH CHẾ HÀM CÀI ĐẶT

- Trong hàm cài đặt trên mảng temp có thể lâm vào tình huống dư hoặc thiếu. Do đó ta sử dung kỹ thuật con trỏ để cấp phát đủ số lượng bộ nhớ mà mảng temp cần.

6. TINH CHẾ HÀM CÀI ĐẶT

```
1. void SapTang(int a[], int n)
2. {
3.     MergeSort(a,0,n-1);
4. }
```

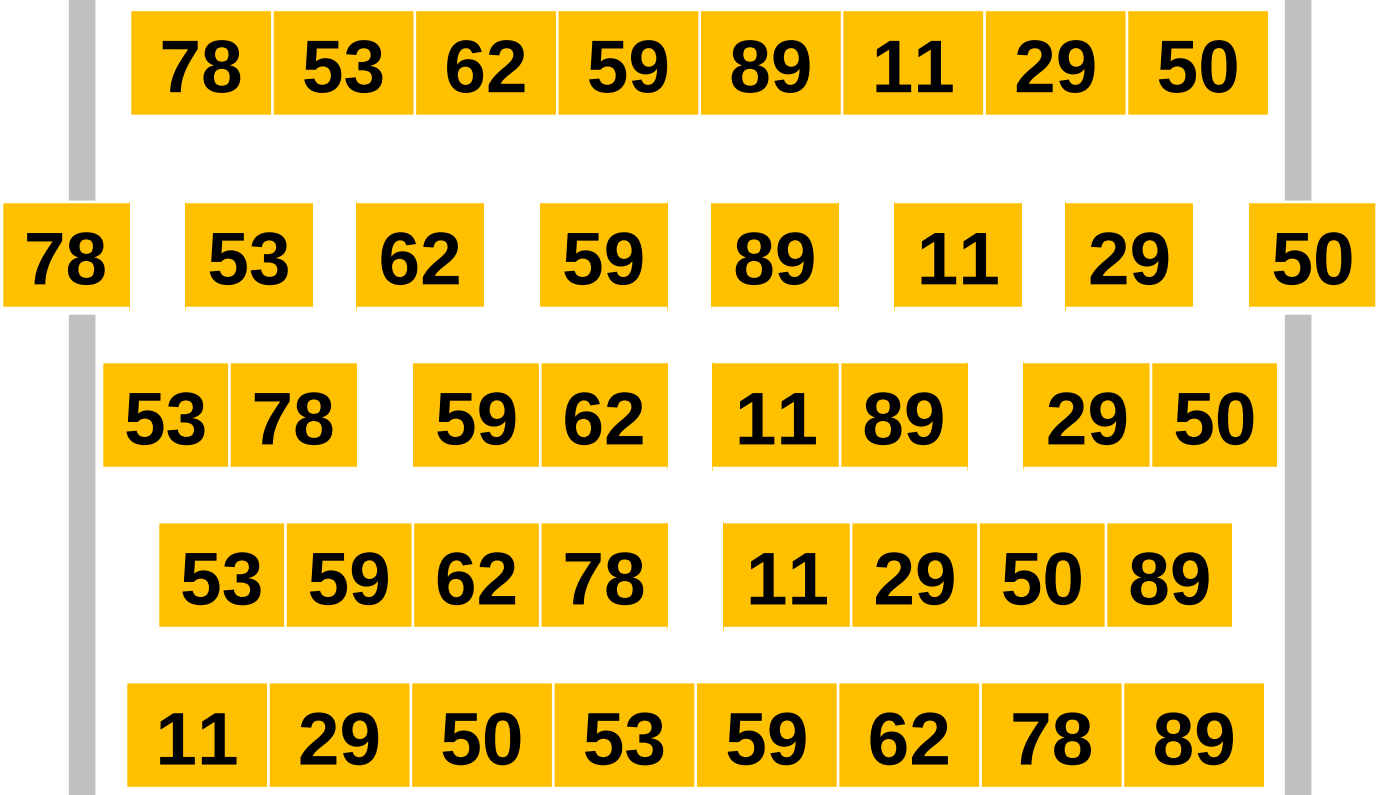
6. TINH CHẾ HÀM CÀI ĐẶT

```
11. void MergeSort(int a[],
                   int Left,int Right)
12. {
13.   if(Left<Right)
14.   //Mảng có nhiều hơn 1 phần tử
15.   {
16.     int Mid = (Left+Right)/2;
17.     // Sắp xếp mảng bên trái
18.     MergeSort(a,Left,Mid);
19.     // Sắp xếp mảng bên phải
20.     MergeSort(a,Mid+1,Right);
21.     // Trộn 2 mảng lại với nhau
22.     Merge(a,Left,Mid,Right);
23.   }
24. }
```

```
11. void Merge(int a[], int Left,
                int Mid,int Right)
12. {
13.   int*temp=new int[Right-Left+1];
14.   int n;

15.   int i = Left;//Left…Mid
16.   int j = Mid+1;//Mid+1…Right
17.   n = 0;
18.   while(!(i>Mid && j>Right))
19.   {
20.     if((i<=Mid&&j<=Right&&
21.        a[i]<a[j]) || (j>Right))
22.       temp[n++]=a[i++];
23.     else
24.       temp[n++]=a[j++];
25.   }
26.   for(i=0;i<n;i++)
27.     a[Left+i] = temp[i];
28.   delete []temp;
29. }
```

7. KHỬ ĐỆ QUY



TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

7. KHỬ ĐỆ QUY

```
11. void MergeSort(int a[], int n)
12. {
13.     int ℓ = 1;
14.     while(ℓ<n)
15.     {
16.         MergeRun(a,n,ℓ);
17.         ℓ = ℓ * 2;
18.     }
19. }
```

7. KHỬ ĐỆ QUY

78	53	62	59	89	11	29	50
78	53	62	59	89	11	29	50
53	78	59	62	11	89	29	50
53	59	62	78	11	29	50	89
11	29	50	53	59	62	78	89

7. KHỬ ĐỆ QUY

```
11. void MergeRun(int a[],int n,
                  int ℓ)
12. {
13.     int vt = 0;
14.     while(vt<n)
15.     {
16.         Merge2Run(a,n,vt,ℓ);
17.         vt = vt + 2*ℓ;
18.     }
19. }
```

```
11. void Merge2Run(int a[],int n,
                   int vt,int ℓ)
12. {
13.   int *temp = new int[2*ℓ];
14.   int k;
15.   int i = vt;    //vt…vt+ℓ-1
16.   int j = vt+ℓ; //vt+ℓ…vt+2*ℓ-1
17.   k = 0;
18.   while(!(!(i<vt+ℓ && i<n)&&
19.           !(j<vt+2*ℓ && j<n)))
20.   {
21.     if((i<vt+ℓ  && i<n &&
22.         j<vt+2*ℓ  && j<n &&
23.         a[i]<a[j]) ||
24.         !(j<vt+2*ℓ && j<n))
25.       temp[k++] = a[i++];
26.     else
27.       temp[k++] = a[j++];
28.   }
29.   for(i=0;i<k;i++)
30.     a[vt+i]=temp[i];
31.   delete []temp;
32. }
```

32	17	45	12	96	78	83	33	27	88
32	17	45	12	96	78	83	33	27	88
17	32	12	45	78	96	33	83	27	88
17	32	12	45	78	96	33	83	27	88
12	17	32	45	33	78	83	96	27	88
12	17	32	45	33	78	83	96	27	88
12	17	32	33	45	78	83	96	27	88
12	17	32	33	45	78	83	96	27	88
12	17	27	32	33	45	78	83	88	96
12	17	27	32	33	45	78	83	88	96