Ordinamento su Array: Selection Sort e Merge Sort

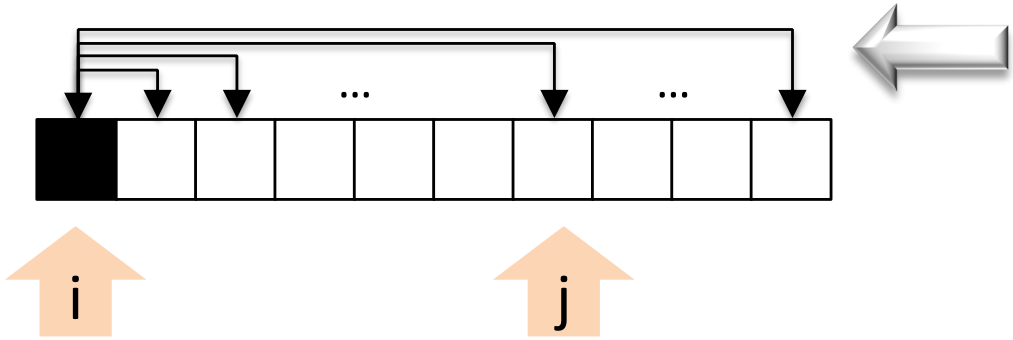
Informatica - CdL Triennale in Fisica

Carlo Mereghetti

`Sito del corso:  mereghetti.di.unimi.it/inf`

`Pagina personale: mereghetti.di.unimi.it`

Implementazione di `selSort`




```
if ( A[j] < A[i] )
    swap( A[i], A[j] );
```

```
void swap( int &x, int &y ) {
    int t = x;
    x = y;
    y = t;
}
```

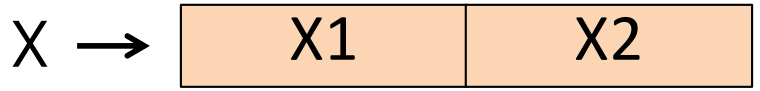
Complessità in tempo

$T(n) = n^2$

```
void selSort( int *X, int dim ) {
    for ( int i = 0, i < dim - 1; i++ )
        for ( int j = i + 1; j < dim; j++ )
            if ( X[j] < X[i] )
                swap( X[i], X[j] );
}
```

Implementazione di `mergeSort`

```
mergeSort( X ) {
    if ( dim(X) > 1 ) {
        mergeSort( X1 );
        mergeSort( X2 );
        X ← merge( X1, X2 );
    }
}
```

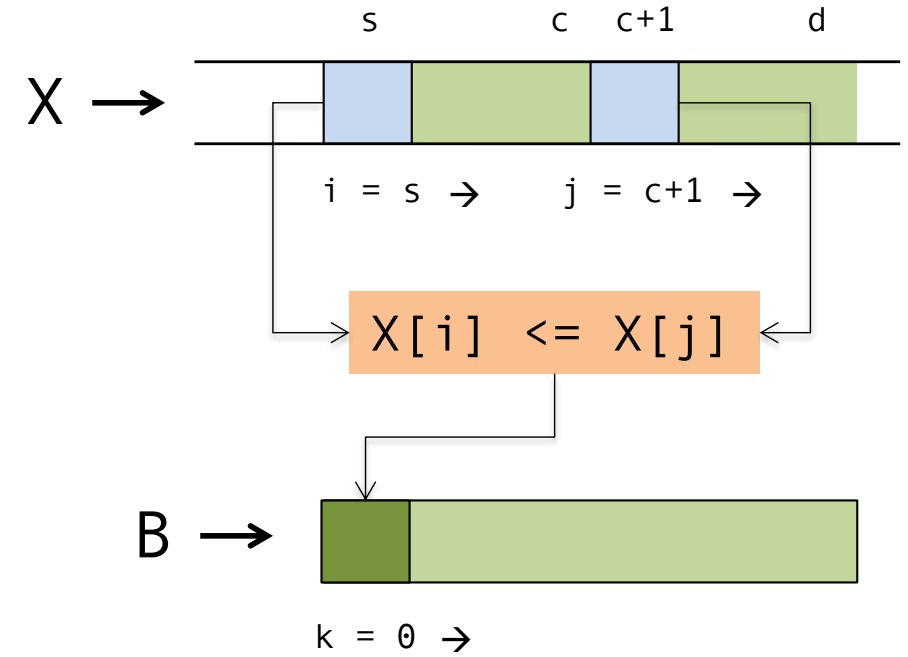



```
int main() {
    ...
    int A[100];
    ...
    mergeSort( A, 0, 99 );
    ...
}
```

```
void mergeSort( int *X, int s, int d ) {
    if ( s < d ) {
        mergeSort( X, s, (s+d)/2 );
        mergeSort( X, (s+d)/2 + 1, d );
        merge( X, s, d );
    }
}
```

Implementazione di merge

```
void merge( int *X, int s, int d ) {
    int *B = new int[ d - s + 1 ];
    int i = s, c = (s+d)/2, j = c+1, k = 0;
    while ( i <= c && j <= d ) {
        if ( X[i] <= X[j] ) {
            B[k] = X[i];
            i++;
        }
        else {
            B[k] = X[j];
            j++;
        }
        k++;
    }
    for ( ; i <= c; i++, k++ )
        B[k] = X[i];
    for ( ; j <= d; j++, k++ )
        B[k] = X[j];
    for ( i = s, k = 0; i <= d; i++, k++ )
        X[i] = B[k];
    delete [] B;
}
```



Algoritmo mergeSort completo

```
void mergeSort( int *X, int s, int d ) {
     if ( s < d ) {
          mergeSort( X, s, (s+d)/2 );
          mergeSort( X, (s+d)/2 + 1, d );
          merge( X, s, d );
     }
}
```

```
void merge( int *X, int s, int d ) {
     int *B = new int[ d - s + 1 ];
     int i = s, c = (s+d)/2, j = c+1, k = 0;
     while ( i <= c && j <= d ) {
          if ( X[i] <= X[j] ) {
               B[k] = X[i];
               i++;
          }
          else {
               B[k] = X[j];
               j++;
          }
          k++;
     }
     for ( ; i <= c; i++, k++ )
          B[k] = X[i];
     for ( ; j <= d; j++, k++ )
          B[k] = X[j];
     for ( i = s, k = 0; i <= d; i++, k++ )
          X[i] = B[k];
     delete [] B;
}
```

```
PROTOTIPI

void mergeSort( int*, int, int );
void merge( int*, int, int );
```

Complessità in tempo

$T(n) = n \log n$