삽입정렬
(Insertion Sort)

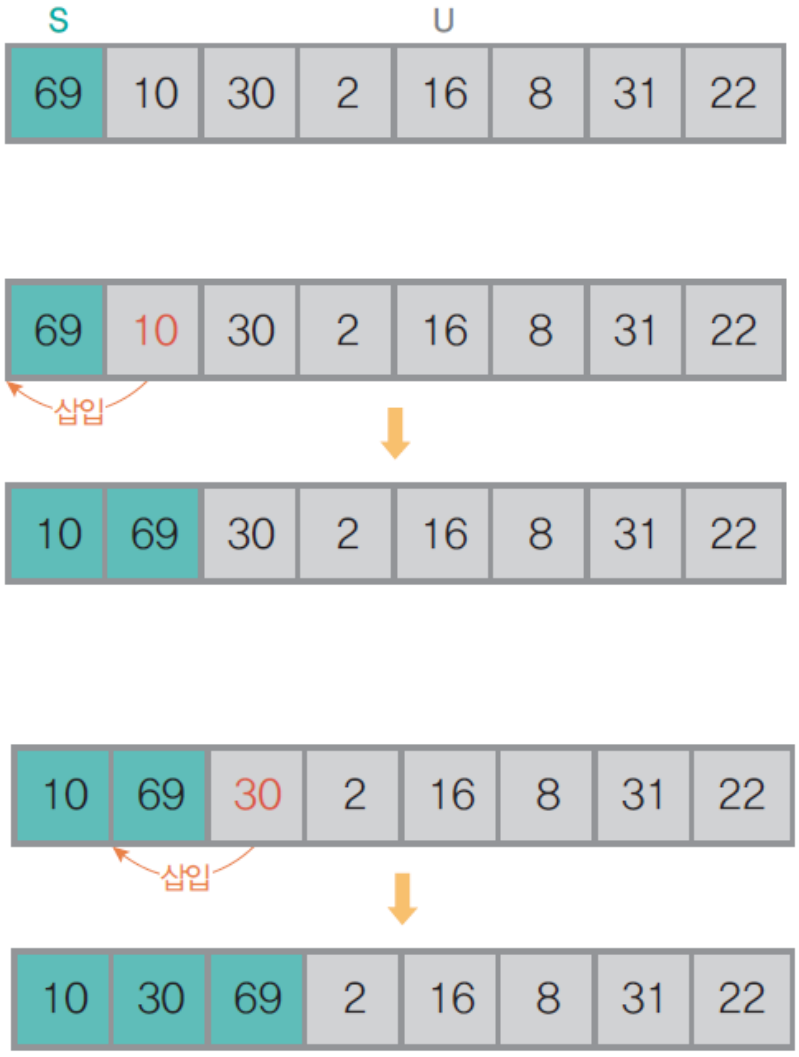
삽입정렬

❖ 삽입 정렬의 이해

- 정렬되어있는 부분집합에 정렬할 새로운 원소의 위치를 찾아 삽입하는 방법
- 정렬할 자료를 두 개의 부분집합으로 구분
  - S(Sorted Subset): 정렬된 앞부분의 원소들
  - U(Unsorted Subset) : 아직 정렬되지 않은 나머지 원소들

- 정렬되지 않은 부분집합 U의 원소를 하나씩 꺼내서 이미 정렬되어있는 부분집합 S의 마지막 원소부터 비교하면서 위치를 찾아 삽입
- 삽입 정렬을 반복하면서 부분집합 S의 원소는 하나씩 늘리고 부분집합 U의 원소는 하나씩 감소하게 함.
- 부분집합 U가 공집합이 되면 삽입 정렬이 완성

삽입정렬

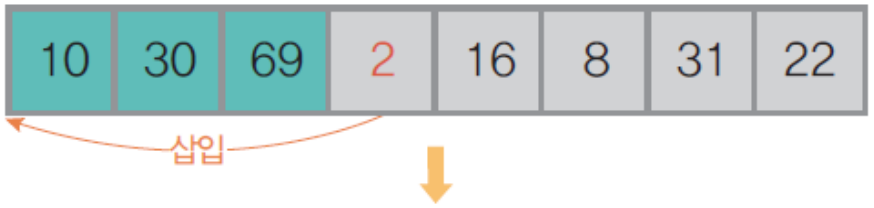


S = {69}
U = {10, 30, 2, 16, 8, 31, 22}

S = {10, 69}
U = {30, 2, 16, 8, 31, 22}

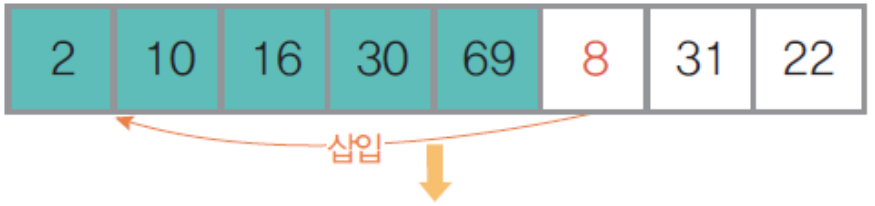
S = {10, 30, 69}
U = {2, 16, 8, 31, 22}

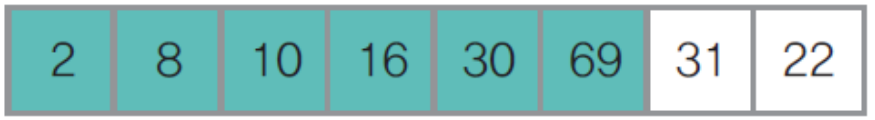
삽입정렬



S = {2, 10, 30, 69}
U = {16, 8, 31, 22}

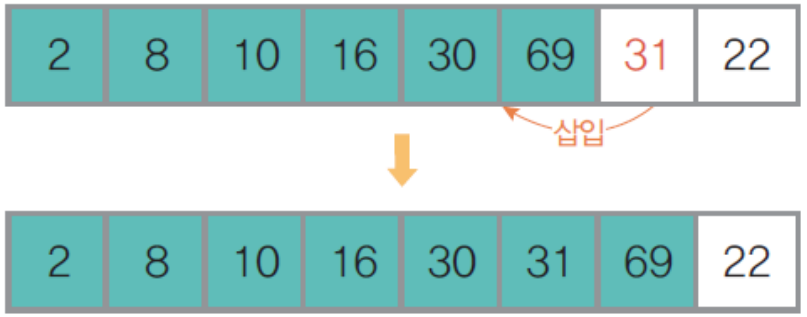
S = {2, 10, 16, 30, 69}
U = {8, 31, 22}



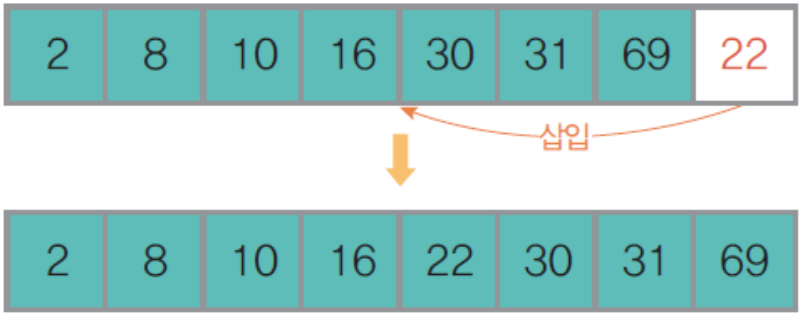


S = {2, 8, 10, 16, 30, 69}
U = {31, 22}

삽입정렬



S = {2, 8, 10, 16, 30, 31, 69}
U = {22}



S = {2, 8, 10, 16, 22, 30, 31, 69}
U = { }

삽입정렬

❖ 삽입정렬 소스

```
public static int[] sortByInsertion(int[] items) {

    for (int i = 1; i < items.length; i++) {
        int data = items[i];      // 기준

        for (int aux = i - 1; data < items[aux]; aux--) {
            // 비교대상이 큰 경우 오른쪽으로 밀어냄
            items[aux + 1] = items[aux];        }

        items[aux + 1] = data;  // 기준값 저장
    }

     return items;
 }
```