자료구조

L09 Sorting

2022년 1학기

국민대학교 소프트웨어학부

Overview

- **Sorting**
- Three $\Theta(n^2)$ algorithms
- Shellsort

정렬 Sorting

- 정렬: 리스트의 아이템들을 특정 순서대로 놓는 것 (증가 혹은 감소)
  - 다양한 응용: 점수, 문서, 검색 결과, …
  - 컴퓨터 과학에서 가장 기초적인 작업 중 하나
- 일상에서의 정렬

정렬 Sorting

- 살펴볼 정렬 알고리즘
  - insertion sort, bubble sort, selection sort, shell sort
  - merge sort, quicksort, heap sort, bin sort, radix sort

- Measure of cost:
  - The number of comparisons
  - The number of swaps

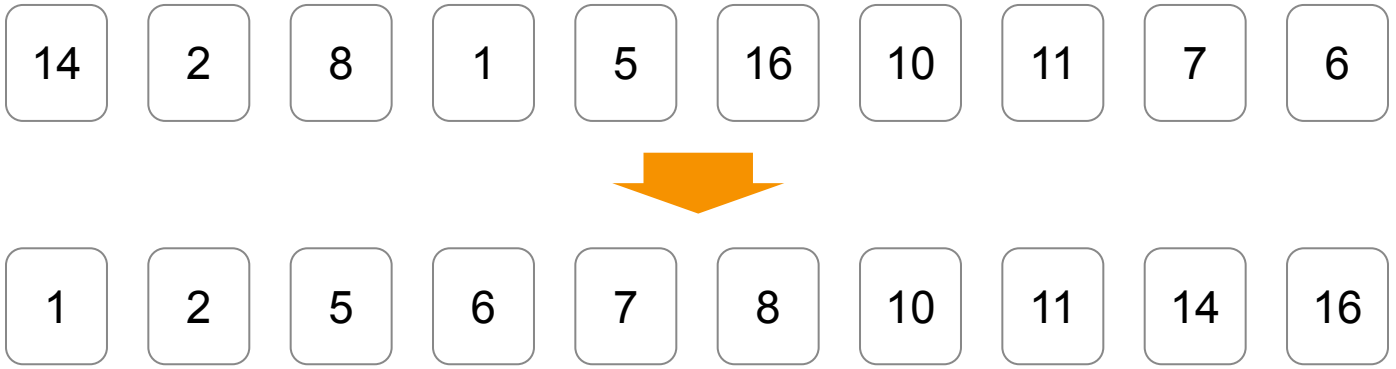
Overview

- Sorting
- **Three $\Theta(n^2)$ algorithms**
- Shellsort

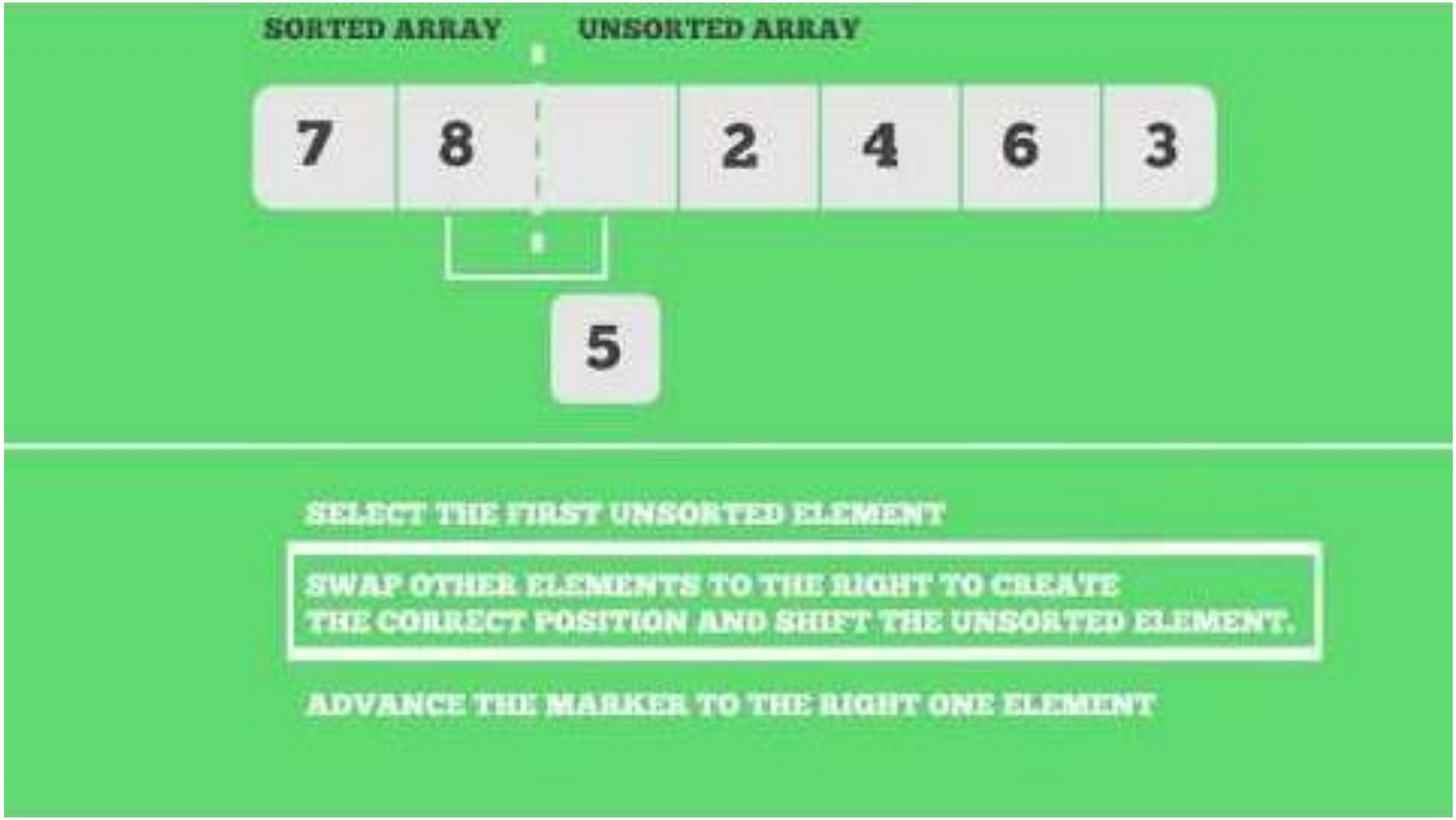
Three $\Theta(n^2)$ Sorting Algorithms

숫자카드가 바닥에 일렬로 놓여있다.
→ 오름차순으로 정렬하는 방법?

- Insertion Sort: 일단 카드 하나 집고, 적절한 위치에 넣자!
- Selection Sort: 가장 작은 카드를 가장 왼쪽에 놓자!
- Bubble Sort: 두개씩 보고 큰 것을 오른쪽으로 보내자!



Insertion Sort



https://youtu.be/OGzPmgsI-pQ

Insertion Sort

```
static <E extends Comparable<? super E>> void Sort(E[] A) {
  for (int i=1; i<A.length; i++)
    for (int j=i; j > 0 && A[j].compareTo(A[j-1]) < 0 ; j--)
      swap(A, j, j-1);
}
```

swaps, # comparisons

- Best Case:
- Worst Case:
- Average Case:

Insertion Sort

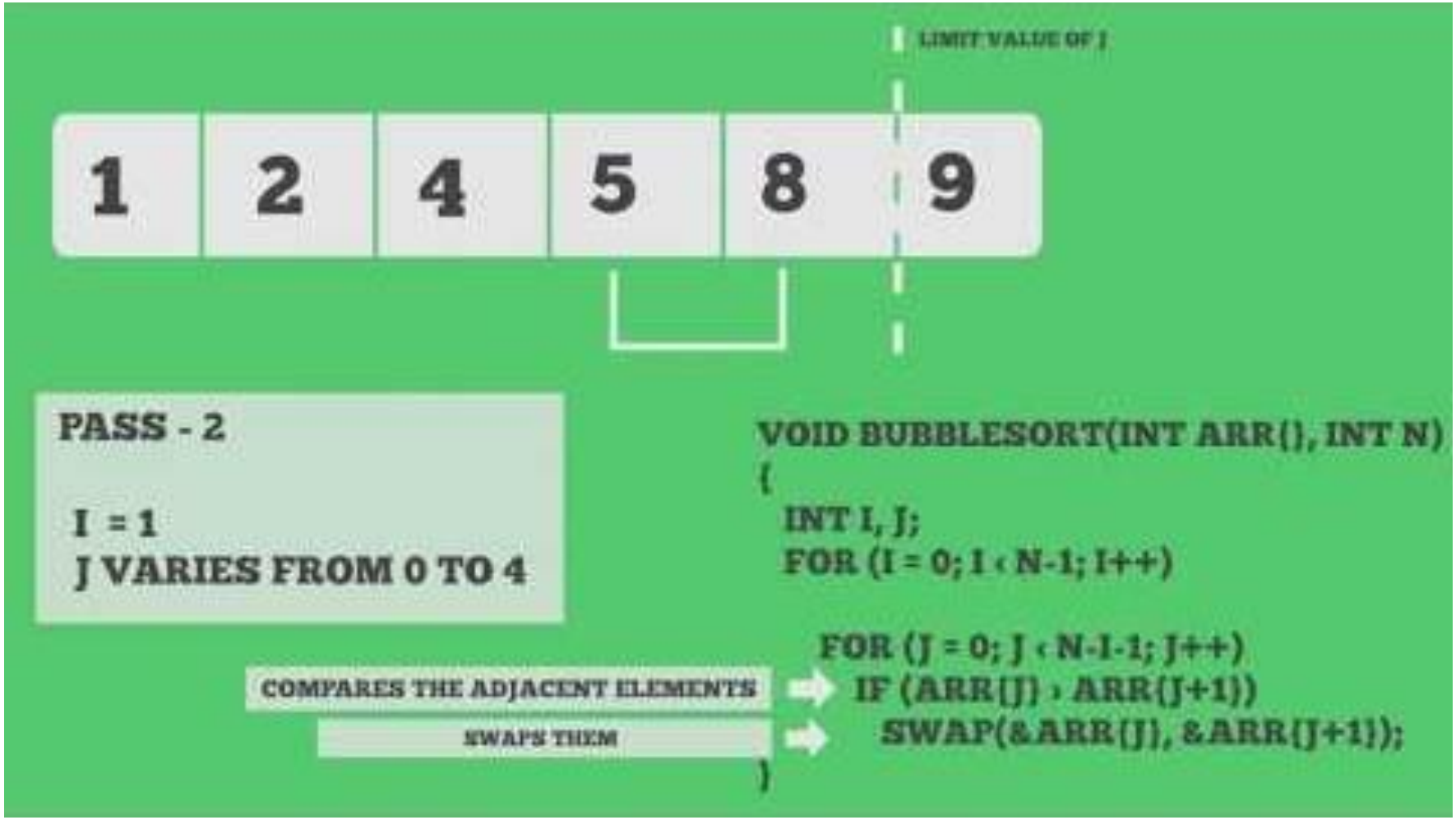
```
static <E extends Comparable<? super E>> void Sort(E[] A) {
  for (int i=1; i<A.length; i++)
    for (int j=i; j > 0 && A[j].compareTo(A[j-1]) < 0 ; j--)
      swap(A, j, j-1);
}
```

swaps, # comparisons

- Best Case: 0 swaps, n-1 comparisons
- Worst Case: $n^2/2$ swaps and comparisons
- Average Case: $n^2/2$ swaps and comparisons

Insertion Sort는 배열이 거의 정렬된 상태일 때 매우 효율적임.
이러한 특징은 다른 정렬 알고리즘에서 활용됨.

Bubble Sort



https://youtu.be/nmhjrI-aW5o

Bubble Sort

```
static <E extends Comparable<? super E>> void Sort(E[] A) {
  for (int i=0; i<A.length-1; i++)
    for (int j=0; j<A.length-1-i; j++)
      if ((A[j].compareTo(A[j+1]) > 0))
        swap(A, j, j+1);
}
```

swaps, # comparisons

- Best Case:
- Worst Case:
- Average Case:

Bubble Sort

```
static <E extends Comparable<? super E>> void Sort(E[] A) {
  for (int i=0; i<A.length-1; i++)
    for (int j=0; j<A.length-1-i; j++)
      if ((A[j].compareTo(A[j+1]) > 0))
        swap(A, j, j+1);
}
```

swaps, # comparisons

- Best Case: 0 swaps and $n^2/2$ comparisons
- Worst Case: $n^2/2$ swaps and comparisons
- Average Case: $n^2/4$ swaps and $n^2/2$ comparisons

Selection Sort

https://youtu.be/xWBP4lzkoyM

Selection Sort

```
static <E extends Comparable<? super E>> void Sort(E[] A) {
    for (int i = 0; i < A.length - 1; i++) {
        int lowindex = i;
        for (int j = A.length - 1; j > i; j--)
            if (A[j].compareTo(A[lowindex]) < 0)
                lowindex = j;
        swap(A, i, lowindex);
    }
}
```

swaps, # comparisons

- Best Case:
- Worst Case:
- Average Case:

Selection Sort

```
static <E extends Comparable<? super E>> void Sort(E[] A) {
    for (int i = 0; i < A.length - 1; i++) {
        int lowindex = i;
        for (int j = A.length - 1; j > i; j--)
            if (A[j].compareTo(A[lowindex]) < 0)
                lowindex = j;
        swap(A, i, lowindex);
    }
}
```

swaps, # comparisons

- Best Case: 0 swaps (n-1 swaps for bad swap()), $n^2/2$ comparisons
- Worst Case: n-1 swaps and $n^2/2$ comparisons
- Average Case: O(n) swaps and $n^2/2$ comparisons

Better than Bubble sort, since # swaps is much smaller

Summary

	Insertion	Bubble	Selection
Comparisons			
Best Case	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$
Average Case	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$
Worst Case	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$
Swaps			
Best Case	0	0	0
Average Case	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n)$
Worst Case	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n)$

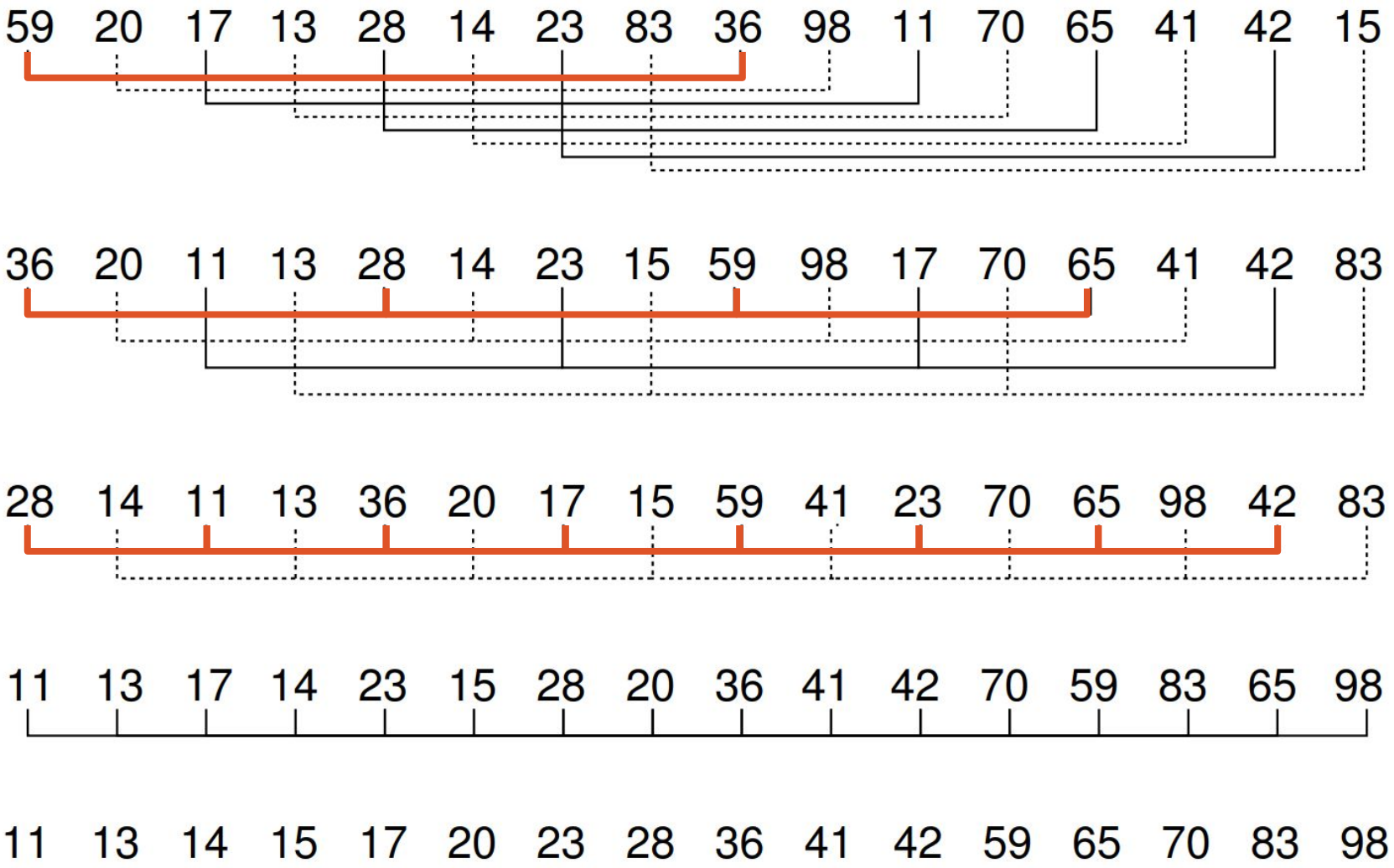
Overview

- Sorting
- Three $\Theta(n^2)$ algorithms
- **Shellsort**

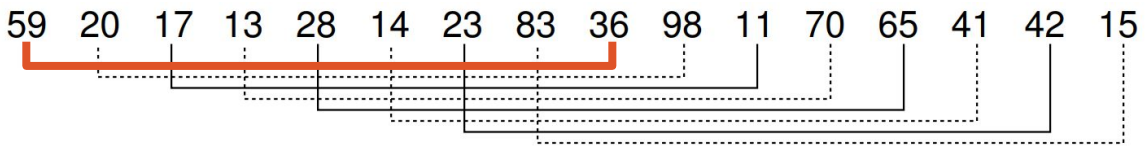
Shellsort

- D. L. Shell이 개발
- 핵심 아이디어
  - Insertion sort는 '거의 정렬된 상태'에서 효율적이다.
  - 리스트를 '거의 정렬된 상태'로 만들고, insertion sort를 돌리자.
- 최악의 경우 시간 복잡도가 $O(n^2)$ 보다 작음
- 평균 시간 복잡도 $O(n^{1.5})$

Shellsort


59 20 17 13 28 14 23 83 36 98 11 70 65 41 42 15
36 20 11 13 28 14 23 15 59 98 17 70 65 41 42 83
28 14 11 13 36 20 17 15 59 41 23 70 65 98 42 83
11 13 17 14 23 15 28 20 36 41 42 70 59 83 65 98
11 13 14 15 17 20 23 28 36 41 42 59 65 70 83 98

Shellsort



Procedure

- Pass 1
  - 위치가 n/2만큼 떨어진 값을 2개씩 묶어 n/2개의 서브리스트를 생성
    - 예) n = 16이면, 8개의 서브리스트 생성: (0,8), (1,9), …, (7, 15)
  - 각 서브리스트를 Insertion Sort로 정렬
- Pass 2
  - 위치가 n/**4**만큼 떨어진 값을 **4**개씩 묶어 n/**4**개의 서브리스트를 생성
    - 예) n = 16이면,4개의 서브리스트 생성: (0,4,8,12), (1,5,9,13), …
  - 각 서브리스트를 Insertion Sort로 정렬
- …

Shellsort

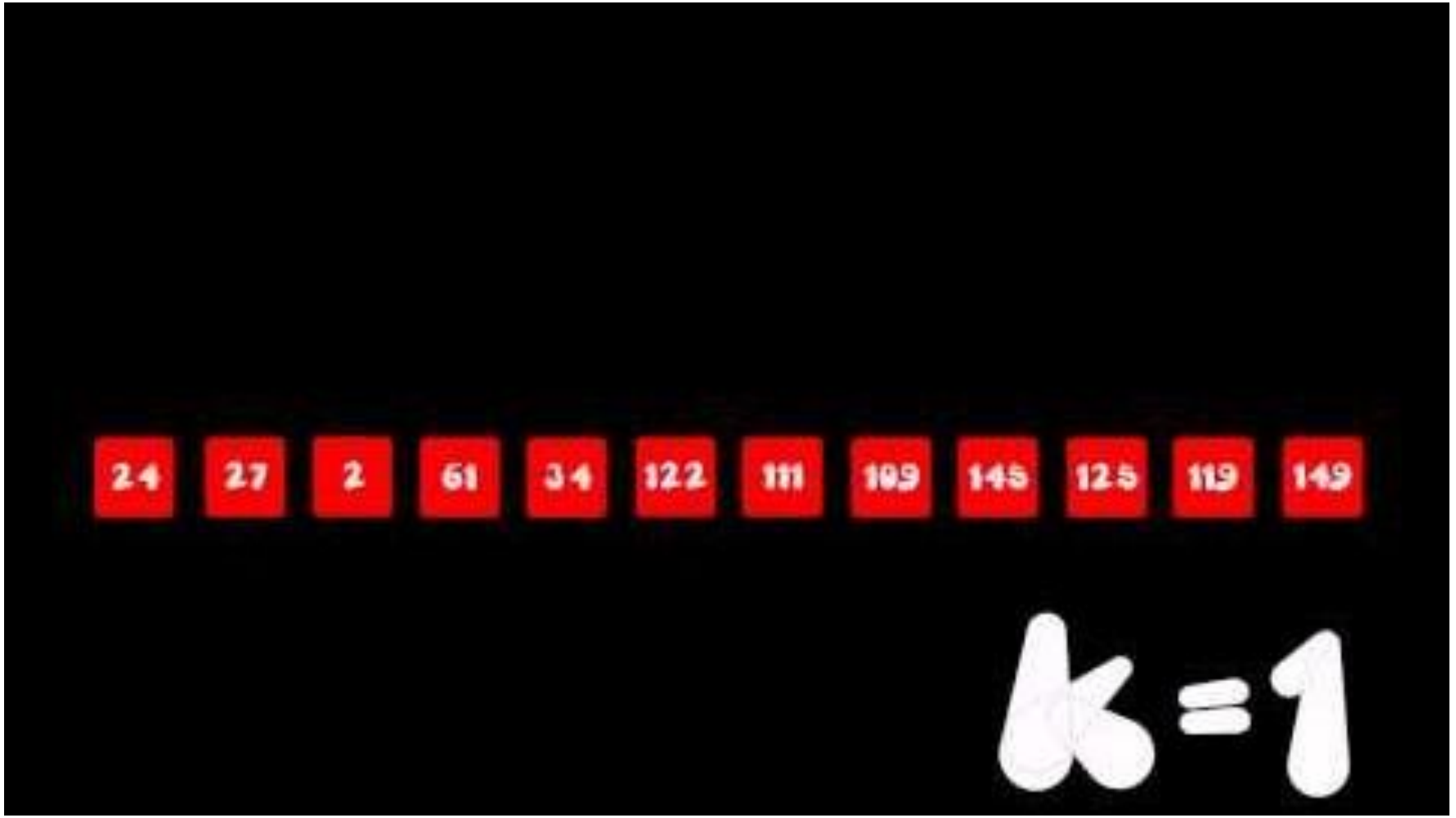
Procedure

- Pass 3
  - 위치가 n/8만큼 떨어진 값을 8개씩 묶어 n/8개의 서브리스트를 생성
    - 예) n = 16이면,2개의 서브리스트 생성: (0,4,8,12), (1,5,9,13), …
  - 각 서브리스트를 Insertion Sort로 정렬
- … Final Pass (Pass (log n))
  - 1개의 서브리스트를 생성하여(=아무것도 안함), 서브리스트를 Insertion Sort로 정렬
  - 즉, 일반 Insertion Sort를 전체 배열에서 실행

Shellsort

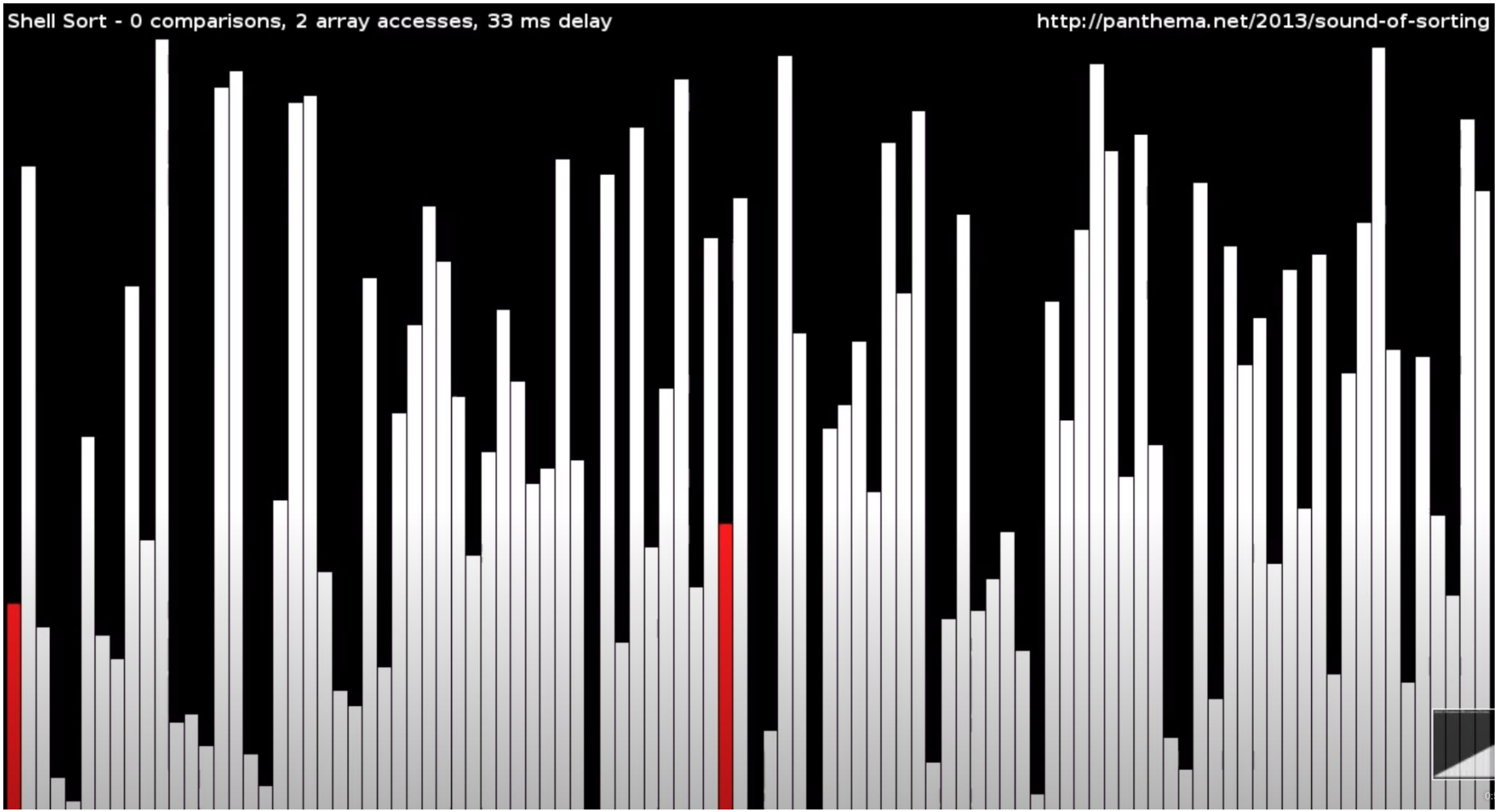
```
static <E extends Comparable<? super E>> void shellsort(E[] A) {
  for (int i = A.length / 2; i > 2; i /= 2) // For each increment
    for (int j = 0; j < i; j++) // Sort each sublist
      inssort2(A, j, i);
  inssort2(A, 0, 1); // Could call regular inssort here
}

/** Modified Insertion Sort for varying increments */
static <E extends Comparable<? super E>>
                        void inssort2(E[] A, int start, int incr) {
  for (int i = start + incr; i < A.length; i += incr)
    for (int j = i;
         j >= incr && A[j].compareTo(A[j - incr]) < 0;
         j -= incr)
      swap(A, j, j - incr);
}
```

Shellsort



https://youtu.be/M9YCh-ZeC7Y

Shellsort



https://youtu.be/n4sk-SzGvZA

Questions?