

# 자료구조를 구조하자

2019.11.28

32153180 이상민

32162436 신창우 32163006 이건욱

32164420 조정민 32164959 허전진

# 학습 내용

**합병 정렬을 위한 패스 함수**

합병 정렬을 위한 패스 함수는 두 배열을 하나의 배열로 합치는 과정이다. 이 과정은 재귀적으로 호출되며, 배열의 크기가 1 이하일 때까지 반복된다.

**7.6 히프 정렬**

히프 정렬은 완전 이진 트리를 이용하여 배열을 정렬하는 알고리즘이다. 이 과정은 재귀적으로 호출되며, 배열의 크기가 1 이하일 때까지 반복된다.

**조정 함수**

조정 함수는 히프 정렬을 위한 보조 함수로, 배열의 크기가 1 이하일 때까지 반복된다.

**7.7 여러 키에 의한 정렬**

여러 키에 의한 정렬은 여러 개의 키를 사용하여 배열을 정렬하는 알고리즘이다. 이 과정은 재귀적으로 호출되며, 배열의 크기가 1 이하일 때까지 반복된다.

**기수 정렬(radix sort) = bin sort**

기수 정렬은 여러 개의 키를 사용하여 배열을 정렬하는 알고리즘이다. 이 과정은 재귀적으로 호출되며, 배열의 크기가 1 이하일 때까지 반복된다.

**기수 정렬의 예**

기수 정렬의 예는 여러 개의 키를 사용하여 배열을 정렬하는 알고리즘이다. 이 과정은 재귀적으로 호출되며, 배열의 크기가 1 이하일 때까지 반복된다.

**기수 정렬의 예(계속)**

기수 정렬의 예(계속)는 여러 개의 키를 사용하여 배열을 정렬하는 알고리즘이다. 이 과정은 재귀적으로 호출되며, 배열의 크기가 1 이하일 때까지 반복된다.

**히프 정렬을 위한 2단계**

히프 정렬을 위한 2단계는 배열의 크기가 1 이하일 때까지 반복된다.

**히프 정렬 과정(계속)**

히프 정렬 과정(계속)는 배열의 크기가 1 이하일 때까지 반복된다.

**히프 정렬 함수와 분석**

히프 정렬 함수와 분석은 배열의 크기가 1 이하일 때까지 반복된다.

신창우

# 학습 내용

## 제 7 장 질점(계속)

### 7.5 합병 정렬

[illegible]

합별 함수(프로그램 7.7)

[illegible]
$$L \sim \frac{1}{2} \frac{d}{dt} \left( \frac{d}{dt} \right) \left( \frac{d}{dt} \right)$$

## (2) 바둑 함범 진법

[illegible]

### 복 정렬의 예제

[illegible][illegible]

## 복소 정칙 함수

```
void Queue::dequeue() { // 删除队首元素，并返回其值
// 删除队首元素，并返回其值
if (!isEmpty()) { // 如果队列不为空
    int x = front;
    front++;
    if (front == rear)
        rear = rear + 1;
    return x;
} else {
    cout << "Queue is Empty\n";
    return -1;
}
}
```

## 권정렬의 본심

1. 2차 방정식 판별식:  $\Delta = b^2 - 4ac$  (1, 2)  
 2. 2차 방정식 판별식 판별 결과:  $\Delta > 0$  (2개 실근),  $\Delta = 0$  (1개 실근),  $\Delta < 0$  (2개 허근)  
 3. 2차 방정식 판별식 판별 결과:  $\Delta > 0$  (2개 실근),  $\Delta = 0$  (1개 실근),  $\Delta < 0$  (2개 허근)  
 4. 2차 방정식 판별식 판별 결과:  $\Delta > 0$  (2개 실근),  $\Delta = 0$  (1개 실근),  $\Delta < 0$  (2개 허근)  
 5. 2차 방정식 판별식 판별 결과:  $\Delta > 0$  (2개 실근),  $\Delta = 0$  (1개 실근),  $\Delta < 0$  (2개 허근)  
 6. 2차 방정식 판별식 판별 결과:  $\Delta > 0$  (2개 실근),  $\Delta = 0$  (1개 실근),  $\Delta < 0$  (2개 허근)  
 7. 2차 방정식 판별식 판별 결과:  $\Delta > 0$  (2개 실근),  $\Delta = 0$  (1개 실근),  $\Delta < 0$  (2개 허근)  
 8. 2차 방정식 판별식 판별 결과:  $\Delta > 0$  (2개 실근),  $\Delta = 0$  (1개 실근),  $\Delta < 0$  (2개 허근)  
 9. 2차 방정식 판별식 판별 결과:  $\Delta > 0$  (2개 실근),  $\Delta = 0$  (1개 실근),  $\Delta < 0$  (2개 허근)  
 10. 2차 방정식 판별식 판별 결과:  $\Delta > 0$  (2개 실근),  $\Delta = 0$  (1개 실근),  $\Delta < 0$  (2개 허근)

$\frac{1}{2} \times 10 = 5$

## 한변 정렬을 위한 피스 함수

[illegible]

全 部 和 平

[illegible]

## 7.5 히프 정렬

- `std::vector`의 크기를 고정해 줌으로써 사용한다.
- `std::vector`의 `size()`가 가져오는 값을 주로 사용하는 부분. 크기를 고정해 줌으로써 얻는다. `std::vector`의 `capacity()` 값의 경우, 무한히 사용가능하는 것이 특징이다.



## 조절 함수

100% 的准确率, 那么  $\text{acc} = 1.0$ ,  $\text{precision} = 1.0$ ,  $\text{recall} = 1.0$   
 7. 在机器学习中, 我们经常会遇到一些数据集, 其中某些类别的样本数量远少于其他类别, 这会导致模型在训练过程中偏向于多数类, 从而降低对少数类的识别能力。为了解决这个问题, 我们可以采用一些方法来平衡数据集, 例如: 过采样 (oversampling) 和欠采样 (undersampling)。过采样是指通过复制或生成新的样本来增加少数类的数量, 而欠采样则是通过减少多数类的数量来达到平衡。这两种方法各有优缺点, 需要根据具体情况选择使用。

8. 在机器学习中, 我们经常需要评估模型的性能。除了准确率 (accuracy) 之外, 还有很多其他的评估指标, 例如: 精确率 (precision)、召回率 (recall)、F1 分数 (F1 score) 等。这些指标可以帮助我们更全面地了解模型的表现, 从而选择出最适合的模型。

9. 在机器学习中, 我们经常需要处理大量的数据。为了提高模型的训练效率, 我们可以采用一些优化技术, 例如: 批量归一化 (batch normalization)、学习率衰减 (learning rate decay) 等。这些技术可以帮助我们更好地利用数据, 从而提高模型的训练效率和性能。

10. 在机器学习中, 我们经常需要处理一些复杂的问题, 例如: 图像识别、自然语言处理等。为了解决这些问题, 我们可以采用一些深度学习模型, 例如: 卷积神经网络 (CNN)、循环神经网络 (RNN) 等。这些模型具有强大的学习能力, 可以帮助我们更好地理解和处理复杂的数据。

2014.3.14

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	right
26	5	37	1	61	11	57	15	48	19	400	1	10
130		1							1			
(130)		1							1			
		19							37			
				1								
				1								
				15								
					1	1						
					1	1						
11	5	19	1	15	16	57	11	48	37			

3-15

이건 뭐

# 학습 내용

2019-11-25

### 7.5 합병 정렬

(1) 합병

- 의미: 정렬된 2개의 리스트를 1개의 정렬된 리스트로 만드는 것
- 예

↓ 이는 merge  
하는 과정

array1 [10]      array2 [15]

3	4	8	10	2	6	9
↑ ①				↑ ②		↑ ③
11				m+1		n
				12		

← merged list:2    3    4    6    8    9    10

11 < 12 일 때, 3은 12에 들어감

2가 배열 array2의 index + 1

현재 array2도 + 1

- 배열을 이진할 경우, 좌측 곱공률 또 다른 배열에 저장해야 한다.
- 프로그램 7.7: for 루프를 반복할 때마다 iResult와 j만큼 증가하므로  $O(n \rightarrow n)$ 이다.

### 한번 함수 (프로그래밍 7.7)

```

various items()
and Major() item = 1;
various items() {
  for (var i = 0; i < items.length; i++) {
    if (items[i] < 100) {
      items[i] = 100;
    }
  }
}

```

이 함수는 items 배열의 모든 요소를 100 이하로 제한한다. 만약 100 이상이면 100으로 대체한다.

### (2) 반복 알고리즘

이 함수는 items 배열의 모든 요소를 100 이하로 제한한다. 만약 100 이상이면 100으로 대체한다.

## 합병 정렬을 위한 패스 함수

• 둘미스 와 연결된 서브리스트를 찾아볼 필요가 있다.  
→ `0`과 `1`의 인덱스만 찾으면 된다.

```

void MergeSort(int List, T *resultList, const int n, const int s)
// 합병 정렬의 본 메소드를 수행한다. s의 인덱스는 서브리스트의 시작을 나타낸다.
// resultList은 합병된 그 결과를 resultList에 넣는다. n은 리스트의 크기이다.
// 호출 수이다.
for (int i = 1; i < n; i++)
{
    // 이 단계는 두 서브리스트 중에서 최대의 리스트로 분할하여 처리
    // 1. 1과 2의 인덱스만 찾으면 된다.
    // 2. 1과 2의 인덱스만 찾으면 된다.
    MergeSort(List, resultList, (n+1)/2, s);
    MergeSort(List, resultList, (n+1)/2, s+1);
}
// 길이가 2보다 작거나 같은 리스트를 반환한다.
if (n < 2) MergeSort(List, resultList, s, s+1);
else for (int i = 0; i < n; i++) result[i] = resultList[i];
}

```

• `0`과 `1`의 인덱스만 찾으면 된다.

• `0`과 `1`의 인덱스만 찾으면 된다.

• `0`과 `1`의 인덱스만 찾으면 된다.

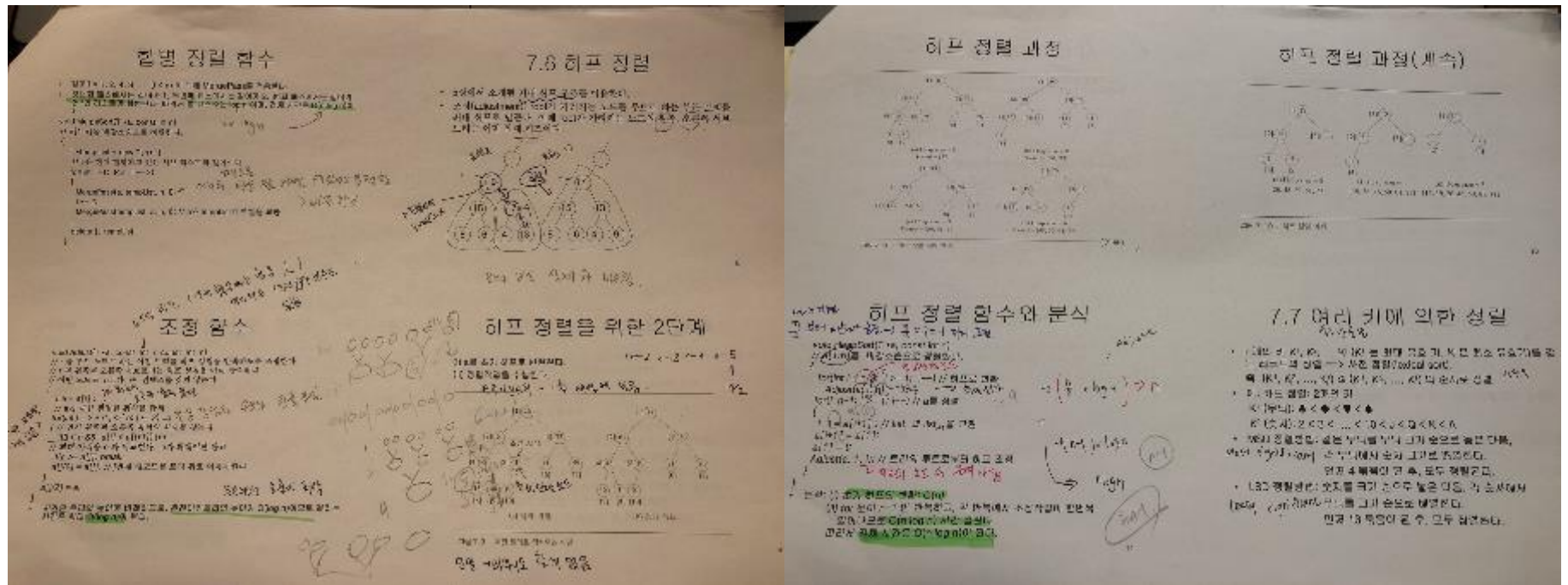
### 7.6 힙프 정렬

• 3장에서 소개된 최대 힙프 구조를 이용한다.

• `1`과 `2`의 인덱스만 찾으면 된다.

조정민

# 학습 내용



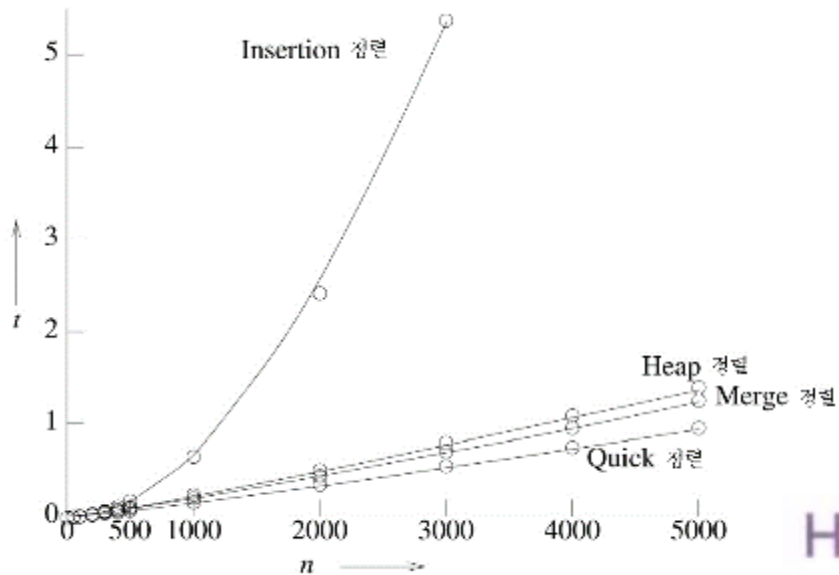
허전진



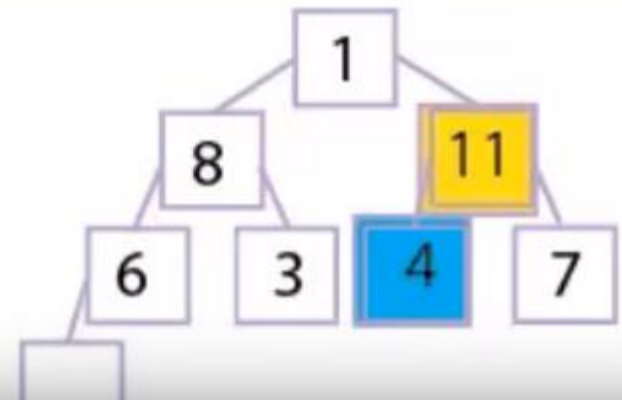
# 참고자료

[https://www.youtube.com/watch?v=QAyl79dCO\\_k](https://www.youtube.com/watch?v=QAyl79dCO_k)

<https://www.youtube.com/watch?v=EreoMaOBTzE>



## Heap Sort



## Merge Sort?

