**고급소프트웨어 실습 2주차 과제**

20171666 이예은

1. Average 필터링을 효율적으로 구현할 수 있는 Moving Average에 대하여 공식을 이용하여 설명한다.

* 일반적인 Average 필터링은 매 픽셀마다 필터 사이즈만큼의 연산이 필요하다. 하지만 이 경우에는 겹치는 계산이 많아 효율적이지 않다. 이를 보완하기 위해 나온 방법이 moving average 방식이다. Moving average의 가장 간단한 공식은 다음과 같다.

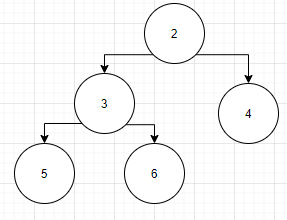
 여기서 는 구한 평균 값들이고, x는 각각의 픽셀 값으로 볼 수 있다. 즉, 이전 평균값에서 이전의 가장 왼쪽 픽셀 값을 빼고, 새로 추가되는 픽셀 값을 더하는 방식이다. 이렇게 하면 중간에 겹치는 부분을 다시 계산할 필요가 없다. 이를 이미지 필터링에 적용하면, 먼저 각 줄의 가장 왼쪽 픽셀은 그냥 average 필터링으로 계산한다. 그리고 필터를 오른쪽으로 옮기며 바로 전 왼쪽 픽셀의 average 값에서 필터의 가장 왼쪽 열의 픽셀 값들을 빼고, 새로 추가되는 현재 픽셀 필터에서 가장 오른쪽 열의 픽셀 값들을 더한다. 이런 방식으로 진행하게 되면, 이전엔 무조건 픽셀 하나 당 필터크기^2 만큼의 연산이 들어갔다면 이젠 처음에만 필터크기^2의 연산이 들어가고 이후부턴 2\*필터크기 만큼의 연산만 들어가게 된다.

1. Median 필터링을 효율적으로 구현할 수 있는 방법에 관하여 도식적으로 설명해 보자. 3x3 윈도우 사이즈를 사용하는 경우를 가정하여 구체적으로 설명한다.

우선 median 필터링의 시간이 오래 걸리는 가장 큰 요인은 sorting이다. 그러므로 median 필터링을 효율적으로 구현하고 싶다면, 이 sorting을 효율적으로 하여 중앙 값을 뽑아내야 한다. 가장 생각하기 쉬운 방법은, 가장 효율적인 sorting 알고리즘을 사용하는 것이다. 일반적인 경우에선 quick sort가 가장 빠르므로 이를 사용해 3\*3 필터의 경우 각 픽셀 당 9개의 픽셀 값을 sorting 한다.

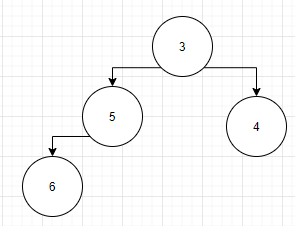
또다른 방법은 우선순위 큐를 이용하는 것이다. Median 필터링은 중간 값을 얻는 게 목적이지 전체 sorting된 값을 알 필요가 없다. 이 방법은 우선 3\*3/2를 올림한 값인 5의 크기를 가지는, 가장 작은 수가 top인 우선순위 큐를 생성한다. 그리고 사이즈가 다 찰 때까지 픽셀 값들을 받는다. 그럼 6번째 부터는 이제 top의 픽셀 값과 비교한다. 만약 이후 넣는 값이 top의 픽셀 값보다 작다면 여전히 top이 5번 째로 큰 중앙 값이므로 넘어간다. 하지만 넣는 값이 top의 픽셀보다 크다면, top은 최소 6번째로 큰 값으로 중앙 값이 아니다. 그러므로 이 top의 원소를 pop 하고 비교한 수를 push 한다. 이 방식으로 3\*3 모든 필터로 픽셀 값들을 큐에 넣으면, 3\*3의 연산이 끝났을 때, 이 우선순위 큐의 top에는 중앙 값이 들어가있다. 그러므로 이를 바로 넣으면 된다.

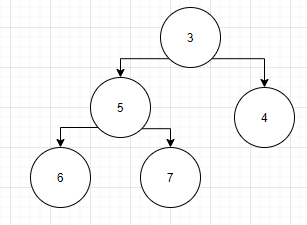
위의 sorting 방식을 사용할 경우, 3\*3의 연산 이후 다시 O(nlogn)의 시간이 또 걸리지만, 이 경우에는 우선순위 큐를 재정립하는데 O(logn)의 시간만 걸린다. 이를 도식적으로 보면 다음과 같다.

 2,3,4,5,6의 픽셀 값이 큐에 있을 경우. 2가 5번째 큰 수로 현재 중앙 값이다.

여기에 1이 들어온다 치면, 1은 이미 6번째 큰 수기 때문에 생각할 필요가 없다.

7이 들어온다면, 이제 2는 6번째 큰 수가 되었기 때문에 2를 삭제한다.

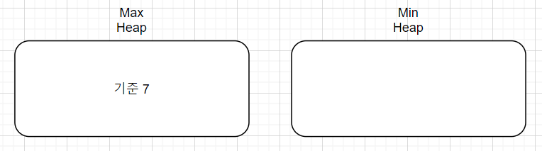
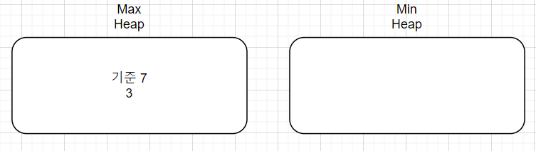
 그럼 다음과 같게 되고, 이제 7을 삽입한다.

 이제 3이 5번째 큰 수가 되어 중앙 값이 된다.

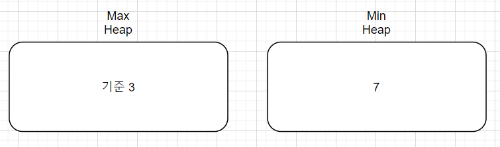
우선순위 큐를 2개 사용하는 방법도 있다. 필터 크기에 따라 우선순위 큐를 미리 만들지 않고 Min heap과 max heap을 둘 다 사용하는 방법이다. 이 경우에는 min, max 둘 중 하나에 중앙 값을 두고, 이보다 작은 건 max heap에, 큰 건 min heap에 넣는다. 그리고 두 heap의 크기가 기준 값을 제외하고 차이가 날 경우 기준 값을 바꾸고 차이를 맞춘다.

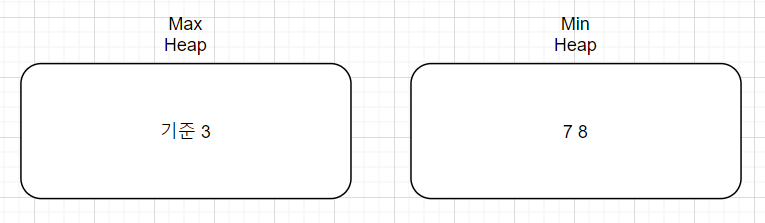
기준 값을 max heap에, 7 3 8 2 라는 수가 들어왔다고 가정하자.

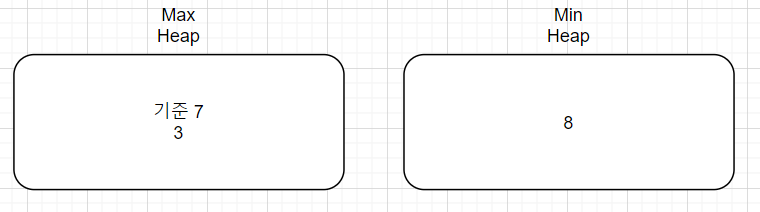
가장 처음 들어온 값은 바로 기준 값이 된다. 그리고 3을 넣으면,

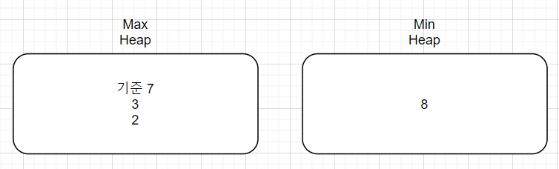
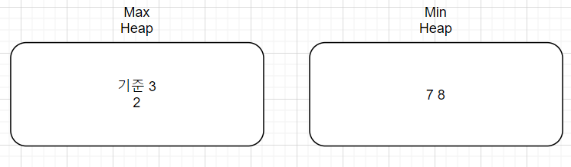
 

다음과 같이 되지만, 크기 차이가 2가 나므로 min으로 하나 옮겨준다.

 그리고 8을 삽입하면,

다음과 같고, 기준을 제외하고 크기 차이가 또 존재하므로 min의 top을 max 로 옮겨 기준 값으로 만든다.

 그리고 2를 넣으면 또 크기 차이가 나므로

다음처럼 변경된다. 여기서 기준을 뽑으면 3으로 중앙값이 된다. 이 방법 또한 우선순위 큐를 사용하므로 시간복잡도는 O(logn)으로 구현이 가능하다. 또한 3\*3 필터링을 할 경우 9개의 입력을 이 알고리즘에 넣으면 효율적으로 중앙 값을 찾을 수 있다.