퍼즐게임챌린지 lv2.

-이진탐색 하는 법 이진탐색은 기본적으로 업 다운 해서 맞추는 방법으로 숫자를 맞춘다.

그냥 배열에서 원하는 숫자를 찾는다면, binary search()를 통해 할 수 있다.

binary\_search(vec.begin(), vec.end(), 9445); 이런식의 활용이 가능하다.

int arr[500]; //짝수인 경우를 봐보자🡪 mid를 구할 때 사이에 2개가 남아도 절삭하고 한번 더 하면 1회 더 구하면 결국 구해진다. 짝수이거나 홀수이거나 상관없이 구할 수 있다.

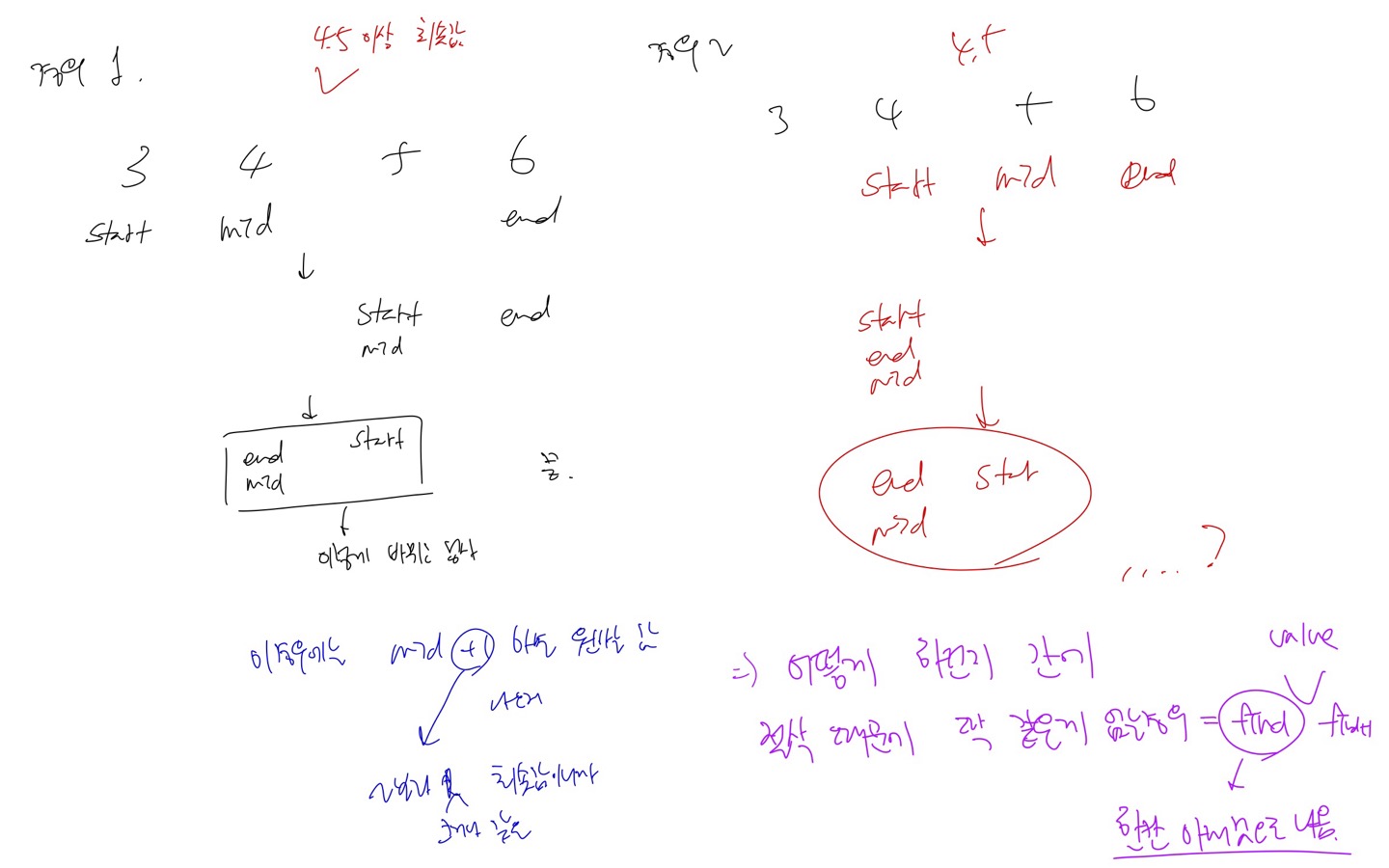
for(int i=1;i<500;i++); … 그리고 계속 이진 탐색을 할 때 mid=right-1, mid=left+1;로 하나씩 더 줄여주는 편이 좋다. 그런데 mid값을 위의 방법처럼 조작하면서 절삭하면, right나 left에 딱 원하는 값이 걸리는 경우가 생기는 경우가 있다.

🡪그런 경우에는 계속 하다보면 결국에는 right와 left가 같아지고 mid 만들면

mid = right = left가 되니까 찾게 된다.

🡪또한 while문으로 찾을 때 까지.. 조건문은 while(left<=right) 이렇게 만들면 된

다. 이렇게 하면 저것을 만족시키는 동안 무조건 찾게 되고 사실 뭐.. 찾은 후에 한번 더 하면 left랑 right의 위치가 역전 되어서 끝나긴 한다



크거나 같을 때 end를 조정하는거로 해야 find가 하나 작은 게 나오는 알고리즘으로 되는거지. 만약에 ~보다 작은 최솟값을 구하라고 했으면 그냥 바로 리턴하면 되는거고.

그게 아니라 작거나 같을 때 start 값을 조정해서 구해보고 싶다? 그러면 최종

**✅ 2. mid >= target, mid < target 방식 (최솟값 찾기)**

cpp

복사편집

int result = -1;

while (start <= end) {

int mid = (start + end) / 2;

if (mid >= target) { // target 이상인 값 중 가장 왼쪽 값 찾기

result = mid; // 일단 기록 (최솟값 갱신 가능성)

end = mid - 1; // 왼쪽 부분 탐색

} else {

start = mid + 1; // 오른쪽 탐색

}

}

return result;

**특징**:

* **target 이상이 되는 값** 중 **가장 작은 값**을 찾을 때 사용.
* 마지막에 result는 **target 이상의 최솟값**을 담고 있음.
* **예시**: 1 3 3 3 5 7 9에서 target = 3일 경우, **첫 번째 3**의 위치를 찾음.

**⚡ 사용처**:

* **최솟값 경계 찾기** (lower bound 개념)
* **이분 탐색 최적화 문제** (ex. 조건을 만족하는 최소 값 찾기)

**✅ 3. mid > target, mid <= target 방식 (최댓값 찾기)**

cpp

복사편집

int result = -1;

while (start <= end) {

int mid = (start + end) / 2;

if (mid > target) { // target 초과 값만 찾기

end = mid - 1; // 왼쪽 탐색

} else { // target 이하 최댓값 찾기

result = mid; // 일단 기록 (최댓값 갱신 가능성)

start = mid + 1; // 오른쪽 탐색

}

}

return result;

**특징**:

* **target 이하 값 중 가장 큰 값**을 찾을 때 사용.
* 마지막에 result는 **target 이하의 최댓값**을 담고 있음.
* **예시**: 1 3 3 3 5 7 9에서 target = 3일 경우, **마지막 3**의 위치를 찾음.

**⚡ 사용처**:

* **최댓값 경계 찾기** (upper bound 개념)
* **조건을 만족하는 최대 값 찾기**

최솟값을 구할 때는 원래 하던대로 하고 mid<=value로 묶어

최댓값을 구할때는 mid>value ; end=mid; 그리고 나머지는 그대로 그냥 있는거 구할때는 3개로 나눠서 원래대로 원래대로라 함은 end=mid-1; start=mid +1 break

~보다 작은 수중 최댓값(상한)을 구할 때는 mid>=val인 경우에 end를 mid보다 한칸 내려 그리고 start는 mid+1로 올려 그러면 정해진다. //이렇게 하면 중복된 값들 중에서도 맨 먼저 있는 것을 구할 수 있다.

~보다 큰 수중 최솟값(하한)을 구할 때는 mid가 val보다 크면 end를 mid까지만 갖다놓고 같거나 mid가 val보다 작을 때 start를 mid+1로 올려🡪 언젠가 start(mid),바로 다음에 end가 있을거고 이때의 end는 val보다 큰 경우다. 근데도 start가 mid+1로 간다는 것은 현재 mid가 val보다 같거나 작다는 이야기이고, 그럼 start,end,mid가 모두 같아지는 경우가 온다 그때의 mid 우리가 구하고 싶은 값이다. 이때 mid는 val보다 조금 더 크게 되고 이 경우에 end는 다시 mid까지만 가게 되고 그렇게 되면 런타임 에러가 난다.//

그래서 이렇게 하면 안되고

~보다 큰 수중 최솟값을 구할 때는

이걸 end를 그대로 놔두고 변경해보려고 한다면,

최솟값을 구할 때는 위에는 동일하지

최댓값을 구할 때 end=mid-1로 놓고 mid>value 이렇게만 하고, 나머지에서 start=mid+1하면 최댓값이 구해지네? 그러면 mid>=val이랑 mid>val이랑 이 차이로 최댓값을 구할 수 있나? 근데 여러 개 겹치면 어렵겠지