1회차 문제 해설

관용

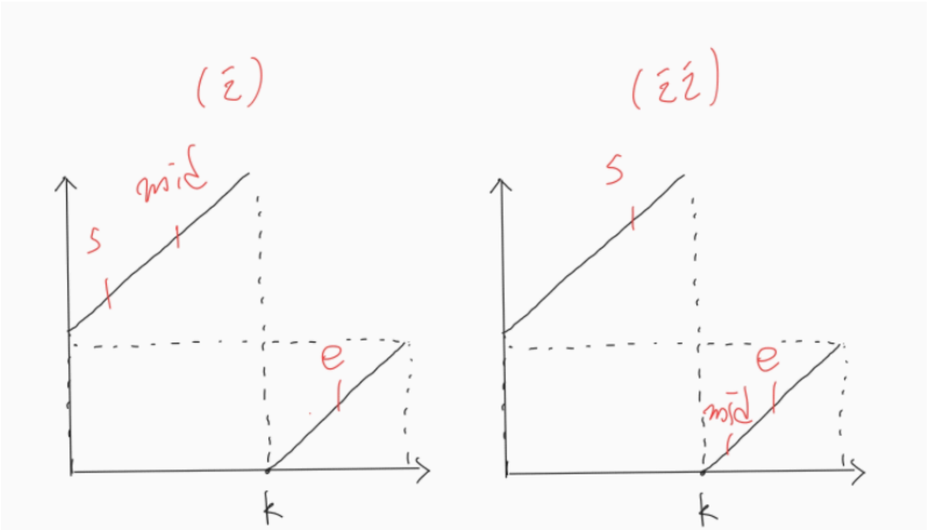
Search in Rotated Sorted Array

개요

- 기본적인 binary search 알고리즘에 rotated sorted array의 특성을 추가하여 해결합니다.
- 알고리즘의 복잡도는 binary search와 같이 시간복잡도 O(logn), 공간복잡도 O(1)을 가집니다.
- 아래와 같이 시작점 s와 중간점 mid의 관계에 따라 탐색 조건을 구분하여 binary search를 수행합니다.

(i) nums[s] <= nums[mid]인 순차 관계

(ii) nums[s] > nums[mid]인 역전 관계

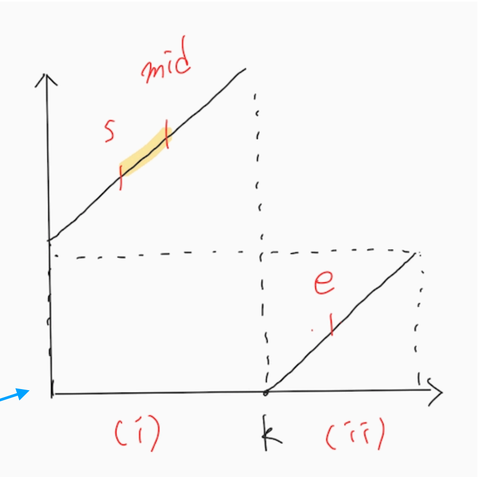


Search in Rotated Sorted Array

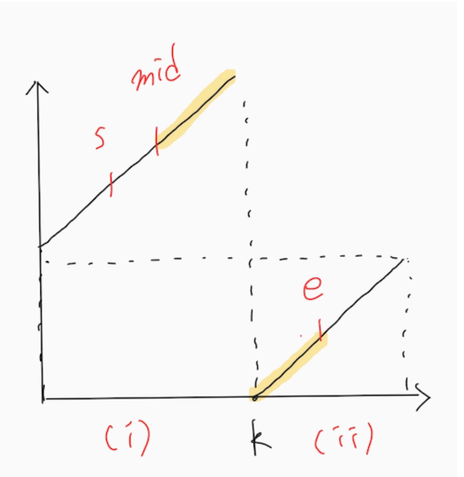
풀이

(1) 순차 관계(i)일 경우 target 추적:

- 형광펜으로 표시된 곳이 target이 존재할 수 있는 범위입니다.




```
if (nums[s] <= nums[mid])
{
    if (target >= nums[s] && target < nums[mid])
    {
        e = mid - 1;
    }
    else
    {
        s = mid + 1;
    }
}
```



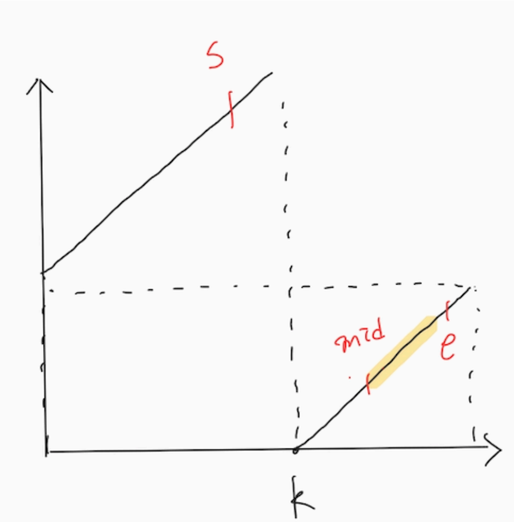
Search in Rotated Sorted Array

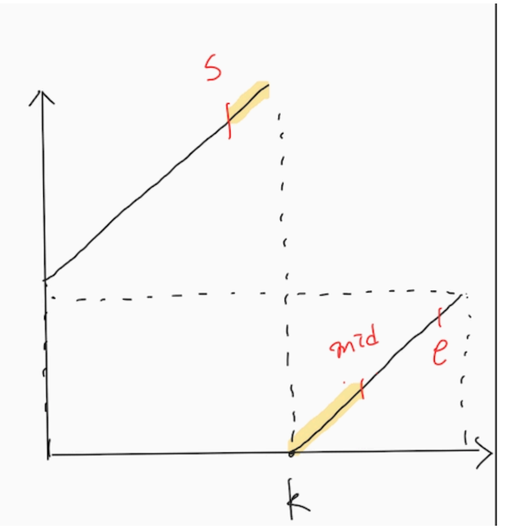
풀이

(2) 역전 관계(ii)에 속할 경우 target 추적:

- 형광펜으로 표시된 곳이 target이 존재할 수 있는 범위입니다.

```
else
{
    if (target > nums[mid] && target <= nums[e])
    {
        s = mid + 1;
    }
    else
    {
        e = mid - 1;
    }
}
```





Course Schedule II

개요

- 다음과 같은 과정을 차례로 거치면 문제를 해결할 수 있습니다.

(1) Direct graph를 Adjacent list를 이용하여 만들기

(2) Topological sort를 수행하며 선수 과목을 모두 들었을때 정답에 추가

(3) 모든 과목에 Topology를 줄 수 있는지 확인

Course Schedule II

풀이

(1) Direct graph를 Adjacent list를 이용하여 만들기

- vertex: 과목, edge: 과목간의 선수관계 표시(a -> b일 경우 a는 b의 선수과목)

- graph[vertex]에는 vertex를 시작점으로 가지는 edge list를 가지고 있습니다.

- indegree[vertex]에는 vertex로 들어오는 edge 의 갯수를 가지고 있습니다.

- 시간 복잡도: O(V + E)

- 공간 복잡도: O(V + E)

```
unordered_map<int, vector<int>> graph;
vector<int> indegree(numCourses, 0);
vector<int> ans;

for (int i = 0; i < prerequisites.size(); i++)
{
    int ai = prerequisites[i][0];
    int bi = prerequisites[i][1];
    graph[bi].push_back(ai);
    indegree[ai]++;
}
```

Course Schedule II

풀이

(2) Topological sort를 수행하며 선수 과목을 모두 들었을때 정답에 추가

- indegree의 값이 0이라는 의미는 선수과목이 없다는 뜻입니다. 이러한 vertex를 큐에 추가합니다. 큐는 선수과목이 없거나 선수과목들을 모두 수강한 과목들을 가지고 있습니다.

```
queue<int> q;
for (int i = 0; i < indegree.size(); i++)
{
    if (indegree[i] == 0)
    {
        q.push(i);
    }
}
```

Course Schedule II

풀이

(2) Topological sort를 수행하며 선수 과목을 모두 들었을때 정답에 추가

- 큐에서 현재 수강할 과목을 뽑습니다. 정답 리스트에 해당 vertex를 추가한 후, 해당 과목은 수강했으므로 이 과목을 선수과목으로 요구하는 vertex들의 indegree 값을 감소시킵니다.

- indegree가 0이 된 과목은 선수과목 조건을 만족했으므로 큐에 추가합니다.

- 시간복잡도: O(V + E)

```
while (q.size() > 0)
{
    int node = q.front();
    ans.push_back(node);
    q.pop();

    vector<int> children = graph[node];
    for (int i = 0; i < children.size(); i++)
    {
        indegree[children[i]]--;
        if (indegree[children[i]] == 0)
        {
            q.push(children[i]);
        }
    }
}
```

Course Schedule II

풀이

(3) 모든 과목에 Topology를 줄 수 있는지 확인

- 정답 리스트의 원소 갯수가 과목 수와 같으면 모든 과목에 수강 순서를 줄 수 있다는 뜻이므로, 완성된 정답 리스트를 리턴합니다. 그 외의 경우 빈 리스트를 리턴합니다.

```
if (ans.size() == numCourses) return ans;
return vector<int>();
```

전체 시간 복잡도: O(V + E)

전체 공간 복잡도: O(V + E)

코드

- https://github.com/tolerance93/LeetCode/tree/master/Search%20in%20Rotated%20Sorted%20Array

- https://github.com/tolerance93/LeetCode/tree/master/Course%20Schedule%20II/Course%20Schedule%20II