고급 5회차
Convex Hull Trick

서강대학교 전해성(seastar105)

Target DP Recurrence Equation

$$dp(i) = \max_{j<i}(m(j)a(i) + b(j)) + C(i)$$

위와 같은 dp식을 생각합시다. 필요한 값이 dp(N)이라면 이를 계산하는 시간복잡도는 $O(N^2)$이 됩니다.

Convex Hull Trick은 이 시간복잡도를 O(NlogN), 특수한 상황에서는 O(N)까지 줄여줍니다.

Maximum Query on Lines

다음과 같은 상황을 생각합시다.

N개의 일차식(y=mx+b), 2차원 좌표계 상에서 N개의 직선들이 주어진다.
다음과 같은 쿼리를 Q개 해결해야 한다.

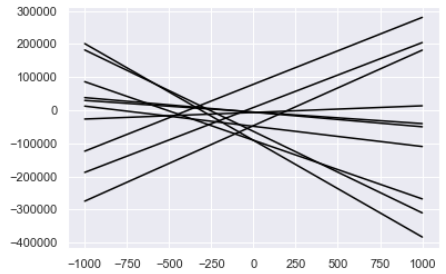
X_i가 주어졌을 때, N개의 직선 중에서 최댓값을 구해라.

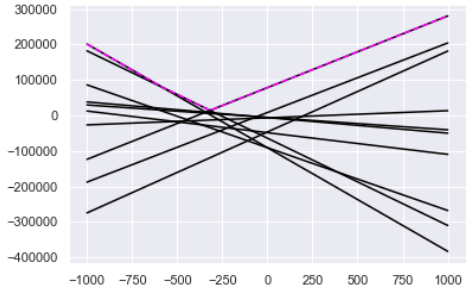
나이브하게 모든 직선에 대해서 값을 구하고 최댓값을 구하면 쿼리 하나당 O(N)입니다.
따라서, 총 시간복잡도는 O(NQ)가 됩니다.

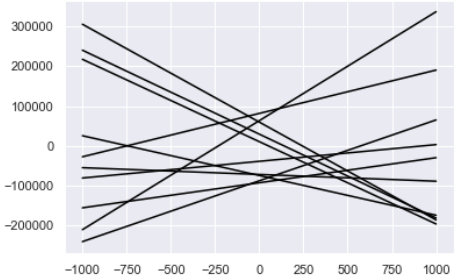
이 문제를 더 빠르게 해결해보는 방법을 알아봅시다.

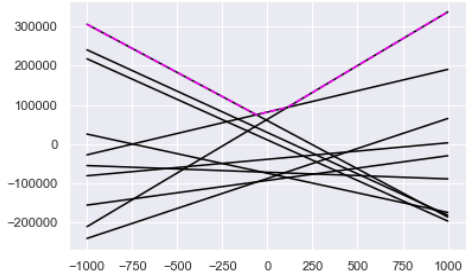
Maximum Query on Lines

직선이 여러 개 있을 때, 최댓값만 그리면 어떤 모습일까요?

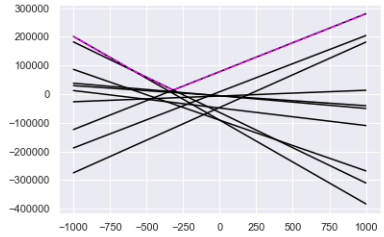


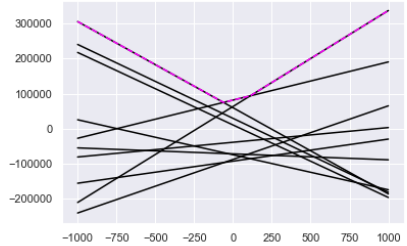






Maximum Points lie on Upper Convex Hull





직선들이 주어져 있을 때, 최댓값만을 따라 그리게 되면 직선들이 이루는 upper convex hull이 됩니다.

따라서, 쿼리에 답하기 위해서는 쿼리로 들어온 x값이 위치하는 곳의 upper convex hull이 어느 직선인지만 알면 됩니다.

convex hull을 잘 관리하면 쿼리를 빠르게 해결이 가능할 거 같습니다.

How to Represent Convex Hull

Convex Hull의 꼭짓점들은 어떤 두 직선의 교차점입니다.

어떤 두 직선이 교차하면 그 점을 기준으로 대소관계가 바뀝니다.

Convex Hull에 포함되었던 직선이 빠졌다가 다시 포함되기 위해선 어떤 구간에서 최댓값이었다가 그 구간을 지나 최댓값이 아니게 되고, 다른 구간에서 다시 최댓값이 되었다는 뜻입니다.이는 불가능합니다.

따라서, 한 직선은 Convex Hull에서 최대 하나의 구간에만 포함될 수 있습니다.

How to Represent Convex Hull

1. Convex Hull은 직선들의 부분부분으로 이루어져있다.
2. 각 직선들은 최대 1회 Convex Hull에 포함되며, 포함된 직선과 어떤 직선의 교차점이 있다면 그 점을 꼭짓점으로 갖게 된다.

위 두 가지 사실을 이용하면 Convex Hull을 나타내도록 합시다.
Convex Hull은 (-∞, ∞)에 걸쳐서 있으며, 몇 개의 구간으로 나뉩니다. 그리고 각 구간마다 담당하게 되는 직선이 존재합니다.

따라서, 구간의 시작점과 구간을 담당하는 직선의 정보만 갖고 있으면 Convex Hull을 나타낼 수 있습니다.

Now, We can answer query Faster

저장된 Convex Hull이 있을 때 쿼리를 어떻게 빠르게 처리할 수 있을까요?

Convex Hull을 이루고 있는 직선이 N개라고 하면, 구간도 N개 있습니다. 구간의 시작점들을 기준으로 오름차순으로 구간들을 가지고 있다면, 이분탐색을 통해 쿼리로 들어온 x에 해당하는 직선을 빠르게 찾을 수 있습니다.

즉, Convex Hull의 구간들을 오름차순으로 구성할 수 있으면 쿼리의 시간복잡도를 O(N)에서 O(logN)으로 줄이는 것이 가능합니다.

이제, Convex Hull을 만드는 방법을 알아봅시다.

How to Construct Convex Hull? (Off-line)

Convex Hull을 구성할 직선 N개가 전부 주어진 오프라인 상황을 먼저 살펴봅시다.

구간들을 시작점 기준으로 오름차순으로 놓는다면 Convex Hull을 구성하는 직선들의 기울기 또한 오름차순으로 될 것은 자명합니다.

따라서, 직선들을 기울기 순으로 정렬합니다.

직선들을 넣을 스택을 만들고, 기울기가 제일 작은 직선부터 스택에 넣습니다.

How to Construct Convex Hull? (Off-line)

현재 들어갈 직선을 l, 스택의 제일 위 직선을 l_1, 그 아래 직선을 l_2이라고 합시다.

l과 l_1의 교점 P를 구하고 l_2과 l_1의 교점 Q와 비교해서 P가 Q보다 왼쪽에 있다면 l_1를 제거합니다. 이 과정을 반복하며 P가 Q보다 오른쪽에 있다면 l을 추가하고 끝냅니다.

How to Construct Convex Hull? (Off-line)

다른 방식도 있습니다.

제일 오른쪽 직선 l_1과 추가될 직선 l의 교점 P를 구합니다.

그리고 l_1의 시작점과 P를 비교해서, P가 더 왼쪽에 있다면 l_1을 지웁니다.

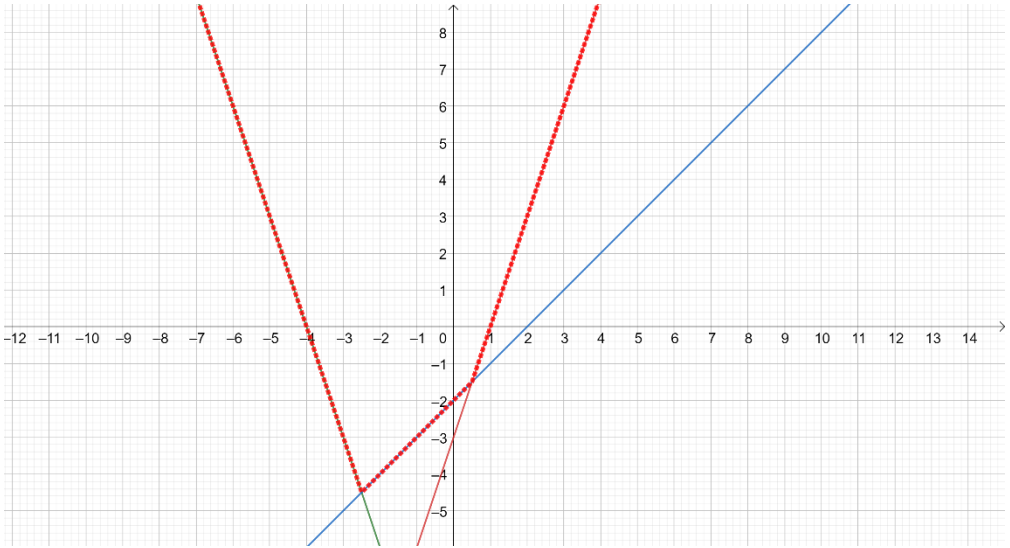
P가 더 오른쪽에 있다면 l을 추가하고 끝냅니다.

How to Construct Convex Hull? (Off-line)

시간복잡도는 정렬에 O(NlogN)이 걸리고 Convex Hull 구성에 O(N)입니다.

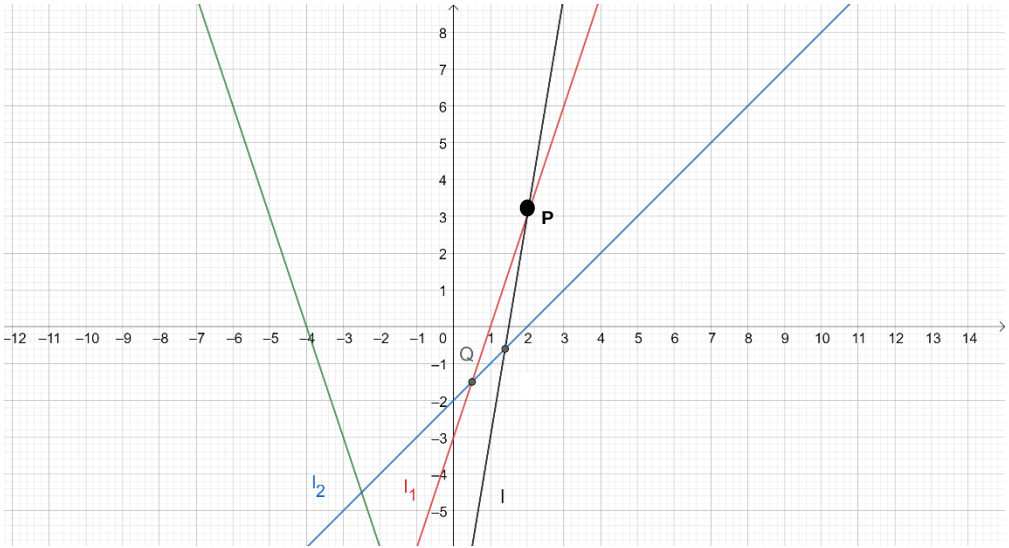
따라서, O(NQ)였던 시간복잡도를 O((N+Q)logN)으로 줄이는 것이 가능해졌습니다.

How to Construct Convex Hull? (Off-line)



왼쪽 그림과 같이 Convex Hull을 이루는 직선이 3개인 상황을 생각해봅시다.

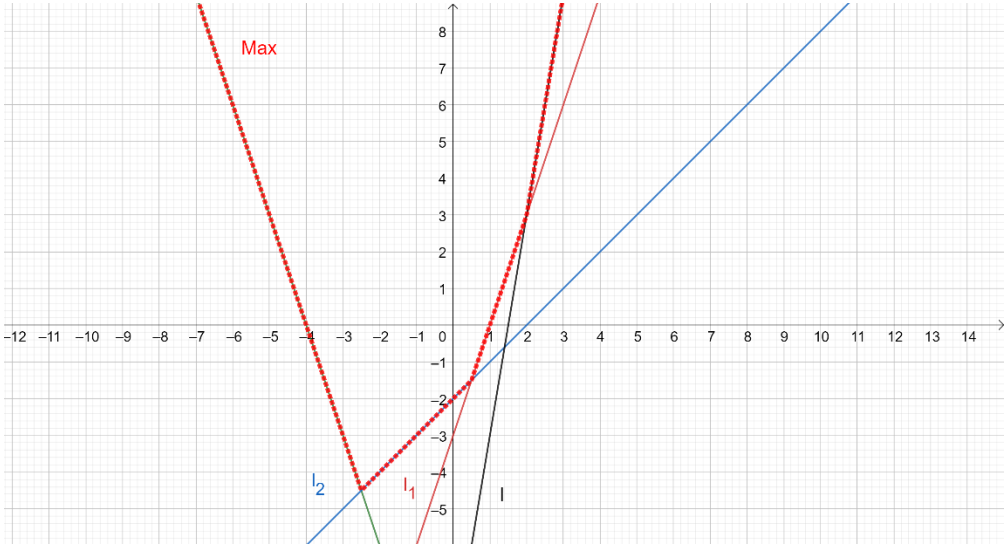
How to Construct Convex Hull? (Off-line)



검은색 직선이 Convex Hull에 추가하고 싶은 직선 l입니다.

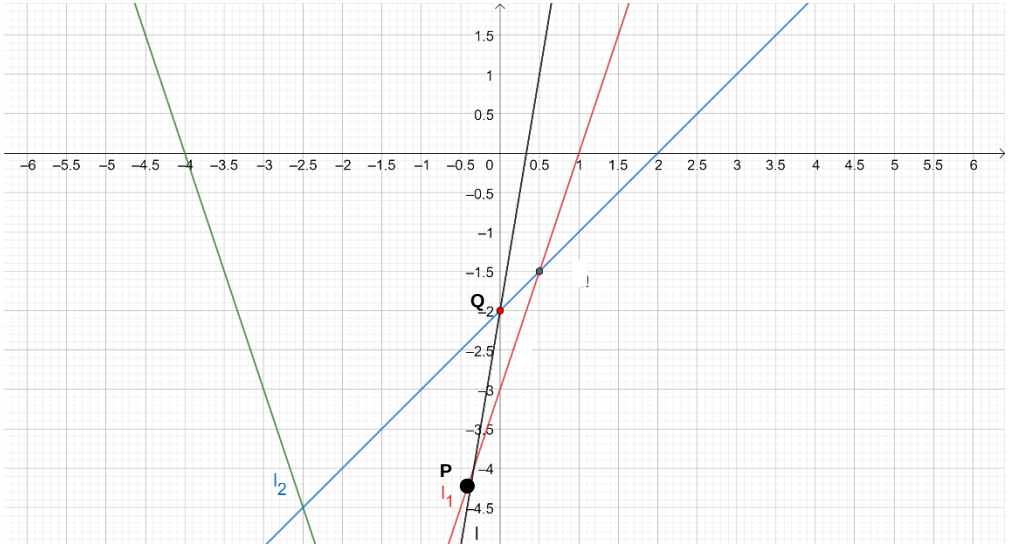
스택의 제일 위, 즉 제일 오른쪽에 있는 두 직선과의 교점 P, Q를 구하고 이를 비교합니다.

How to Construct Convex Hull? (Off-line)



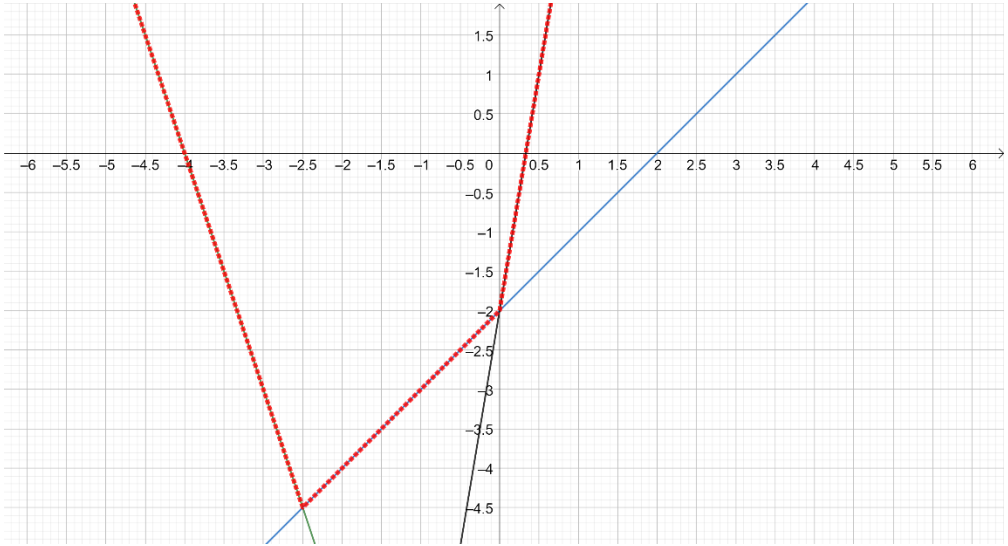
P가 Q보다 오른쪽에 있으므로 l을 추가하고 끝냅니다.

How to Construct Convex Hull? (Off-line)



방금 전과는 다르게 P가 Q보다 왼쪽에 위치합니다.

How to Construct Convex Hull? (Off-line)



제일 위 직선을 삭제합니다.

다시 교점끼리 비교하면 종료조건이니 직선을 그냥 추가하고 끝냅니다.

How to Construct Convex Hull? (On-line)

오프라인이 아니라 이미 Convex Hull이 구성되어 있는 상황에서 임의의 직선을 추가하는 상황을 생각해봅시다. Convex Hull이 N개의 구간으로 나누어졌다고 가정하죠.

How to Construct Convex Hull? (On-line)

Convex Hull의 직선들은 기울기 순으로 정렬되어 있기 때문에 새로 추가할 직선 l이 어디에 위치하는지는 이분탐색으로 O(logN)만에 구할 수 있습니다.

새롭게 추가될 직선의 위치를 i라고 합시다. 이 직선의 왼쪽에 있는 직선 중에서 Convex Hull에서 사라져야 할 직선들을 제거하는 것은 오프라인에서 했던 방식을 쓰면 가능합니다.

How to Construct Convex Hull? (On-line)

오른쪽에 있는 직선들 중에서 삭제할 직선을 삭제하는 것도 오프라인과 비슷합니다.

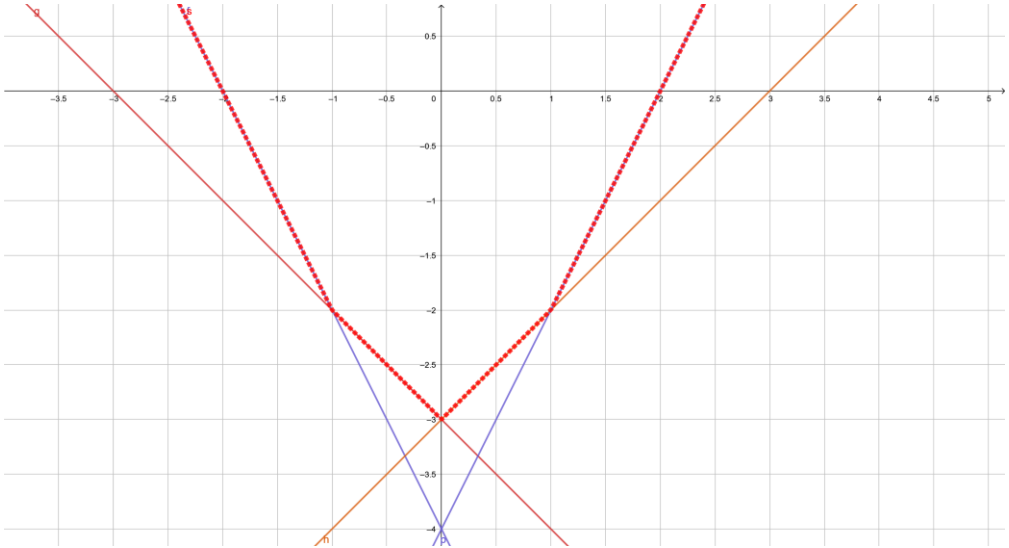
l_i, l_{i+1}, l_{i+2}들을 가지고 l_i, l_{i+1}의 교점 P, l_i, l_{i+2}의 교점 Q를 비교해서 P가 Q보다 오른쪽에 위치하게 되면 l_{i+1}을 제거합니다. 더 제거할 것이 없을 때까지 반복하면 됩니다.

How to Construct Convex Hull? (On-line)

위 방법 말고도 추가한 직선과 바로 오른쪽 직선의 교점을 구하고, 오른쪽 직선의 끝점을 비교해서 교점이 끝점보다 오른쪽에 있으면 오른쪽 직선을 제거합니다. 이 과정도 더 제거할 직선이 없을 때까지 반복하면 됩니다.

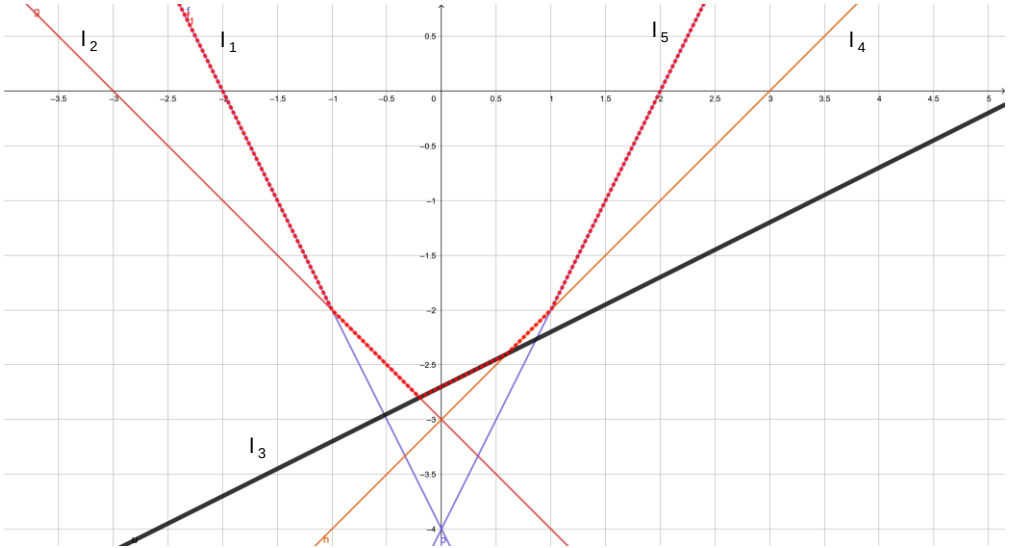
하나의 직선은 최대 한 번 삽입이 일어납니다.
따라서 N개의 직선으로 Convex Hull을 온라인으로 구성해도 O(NlogN)이 가능합니다.

How to Construct Convex Hull? (On-line)



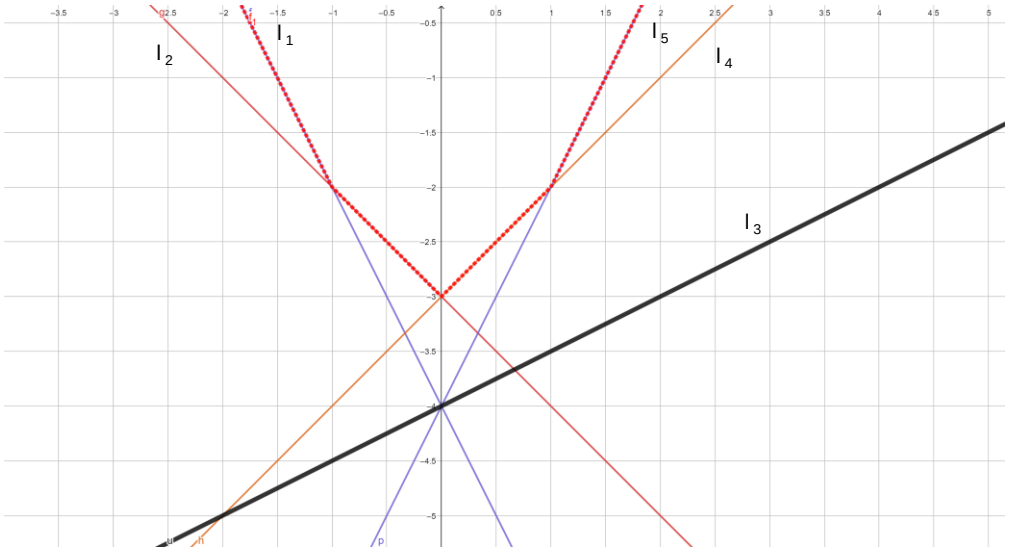
왼쪽 그림과 같이 Convex Hull을 이루는 직선이 4개인 상황을 생각해봅시다.

How to Construct Convex Hull? (On-line)



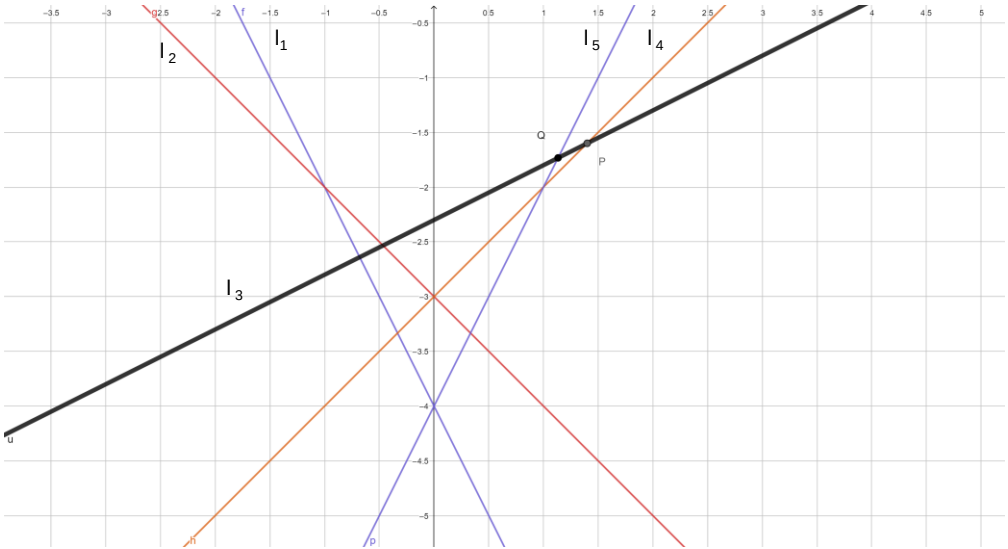
검은색 직선을 Convex Hull에 추가하면, 삭제될 직선은 없습니다. 추가한 직선과 그 다음 직선의 시작점만 조정해주면 삽입과정이 끝납니다.

How to Construct Convex Hull? (On-line)



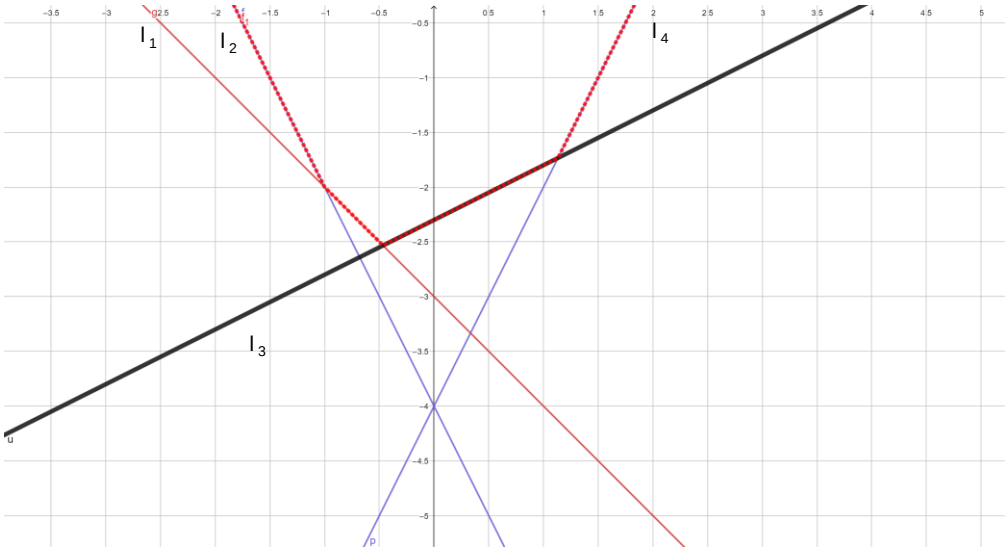
이런 경우 추가한 직선은 기울기 상으로 l_3가 됩니다. 이 직선은 Convex Hull에 포함되지 않습니다. 이런 경우는 l_2, l_3, l_4를 놓고 비교해서 l_3를 지울 수 있습니다.

How to Construct Convex Hull? (On-line)



추가한 직선은 기울기 상으로 l_3가 됩니다. l_3와 l_4의 교점 P가 l_3와 l_5의 교점Q보다 오른쪽에 있으므로 l_4를 제거합니다.

How to Construct Convex Hull? (On-line)



더 제거할 직선이 없으므로 직선의 추가가 끝났습니다.

Implementation Issue

Off-line의 경우 스택을 이용한다고는 했지만 일반적으로 vector를 이용하는 것이 편합니다.

On-line의 경우 std::multiset, std::set을 이용해서 구현을 합니다. 아래 두 구현체를 보시면 좋을 거 같습니다.

https://bit.ly/3hMcOYk - 주로 설명한 방법(직선 3개비교) 의 구현체입니다.
https://bit.ly/38ivnjP - 다른 방법(직선 2개비교)라고 말한 것의 구현체입니다.

On-line 버전을 Dynamic Convex Hull Trick이라고도 부르며 multiset을 이용한 구현 외에도 리차오 트리라는 자료구조를 이용하면 이를 해결할 수 있다고 합니다.

Implementation Issue

최댓값이 아니라 최솟값을 원하는 쿼리라면 직선의 기울기와 y절편의 부호를 바꿔서 저장하고 쿼리 결과의 부호를 바꿔서 사용하면 가능합니다.

혹은, 최솟값의 Convex Hull도 최댓값의 Convex Hull의 구현과 크게 다르지 않기 때문에 그냥 짜시면 됩니다.

팀노트에 LineContainer(두번째 링크)를 넣어가시는 것을 추천 드립니다.

Target DP Recurrence Equation

$$dp(i) = \max_{j<i}(m(j)a(i) + b(j)) + C(i)$$

먼 길을 왔습니다. 제일 처음 말했던 dp식을 다시 봅시다.

max안에 있는 식은 잘 보면, 원하는 값은 기울기가 m(j)고 y절편이 b(j)인 직선들에 x값으로 a(i)를 줬을 때에 최댓값입니다.

따라서, 직선들로 이루어진 Convex Hull을 관리하면 $O(N^2)$이었던 시간복잡도를 O(NlogN)으로 줄이는 것이 가능합니다.

“Any Question?”