


Join the [official Codeforces group](#) in VK. There we will publish announcements of rounds and important news.

[ME000W](#) [BLOG](#) [TEAMS](#) [SUBMISSIONS](#) [GROUPS](#) [CONTESTS](#)

## meoow's blog

# [Tutorial] Convex Hull Trick — Geometry being useful

 By [meoow](#), 17 months ago, 

## The problem

Let us consider the problem where we need to quickly calculate the following over some set  $S$  of  $j$  for some value  $x$ .

$$\max_{j \in S} \{m_j \cdot x + c_j\}$$

Additionally, insertion of new  $j$  into  $S$  must also be efficient.

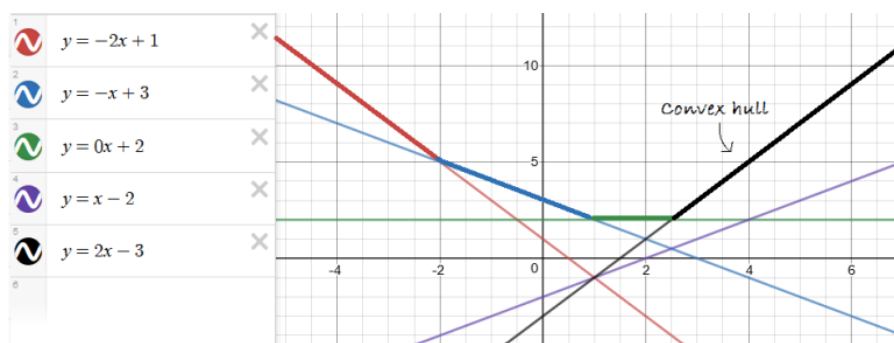
This will most likely be encountered with DP problems. For example, the recent problem [1083E - The Fair Nut and Rectangles](#) from Round #526 has the following DP formulation after sorting the rectangles by  $x$ .

$$f(i) = x_i \cdot y_i - a_i + \max_{1 \leq j < i} \{-x_j \cdot y_i + f(j)\}$$

The problem requires quick calculation of the above define maximum for each index  $i$ . How can this be done?

## The idea

Notice the special form of  $m_j \cdot x + c_j$ . This is identical to the equation of a straight line with slope  $m_j$  and Y-intercept  $c_j$ . So the problem is equivalent to being given a set of lines and asked for the maximum  $y$  value any of those lines can give at a particular  $x$ . If you draw a bunch of straight lines on a plane, you'll notice that the maximum values are along what appears to be a convex hull.



Some observations:

- Every line on the hull provides the maximum value on some contiguous range of  $x$  values. Conversely, every line not on the hull is useless and never provides the maximum.
- All the lines on the hull have different slopes. The order of slopes also determines their position on the hull. A line with lower slope appears on the hull to the left of one with a higher slope.

So, a possible strategy can be to only maintain the convex hull and not keep the useless lines.

## A specific problem

### → Pay attention

#### Before contest

[Codeforces Round #641 \(Div. 1\)](#)  
25:27:37

#### Before contest

[Codeforces Round #641 \(Div. 2\)](#)  
25:27:38

Like 146 people like this. Be the first of your friends.

### → Opcode

Rating: **1330**  
Contribution: **0**

- [Settings](#)
- [Blog](#)
- [Teams](#)
- [Submissions](#)
- [Problemsetting](#)
- [Groups](#)
- [Talks](#)
- [Contests](#)



Opcode

### → Top rated

#	User	Rating
1	<a href="#">MiFaFaOvO</a>	3681
2	<a href="#">tourist</a>	3541
3	<a href="#">Um_nik</a>	3434
4	<a href="#">apiadu</a>	3397
5	<a href="#">maroonrk</a>	3353
6	<a href="#">300iq</a>	3317
7	<a href="#">ecnerwala</a>	3260
8	<a href="#">LHiC</a>	3229
9	<a href="#">TLE</a>	3223
10	<a href="#">Benq</a>	3220

[Countries](#) | [Cities](#) | [Organizations](#)

[View all →](#)

### → Top contributors

#	User	Contrib.
1	<a href="#">Errichto</a>	194
2	<a href="#">antontrygubO_o</a>	193
3	<a href="#">vovuh</a>	174
4	<a href="#">pikmike</a>	173
5	<a href="#">tourist</a>	166
6	<a href="#">Um_nik</a>	165
7	<a href="#">Radewoosh</a>	164
7	<a href="#">McDic</a>	164
7	<a href="#">ko_osaga</a>	164
10	<a href="#">300iq</a>	155

[View all →](#)

Let us further consider the rectangle problem mentioned above.

For clarity, let's substitute  $x$  and  $y$  of the problem statement with  $p$  and  $q$ , and allow  $x$  and  $y$  to only refer to coordinates of the 2D plane where we consider the lines.

$$f(i) = p_i \cdot q_i - a_i + \max_{1 \leq j < i} \{-p_j \cdot q_i + f(j)\}$$

In this problem the slope of the lines  $m_j$  is given by  $-p_j$ . Due to the nature of the constraints (no rectangles are nested), after sorting rectangles by increasing  $p$  we will find they are also sorted by decreasing  $q$ .

The idea:

1. We'll keep the lines of the hull, in sorted order of slope.
2. We iterate over the rectangles from  $i = 1$  to  $n$ , and  $p_i < p_j, q_i > q_j$  for  $i < j$ .
3. When we have to get the maximum at some  $x = q_i$ , all lines that provide the maximum at positions  $> q_i$  are now useless, since further maximum queries are guaranteed to occur at  $< q_i$ .
4. When a new line is inserted, the slope of this line  $-p_i$  is guaranteed to be lesser than all lines in the hull. Therefore, this line is also guaranteed to provide the maximum in some range  $(-\infty, x]$ . However, adding this line may make it so that some lines previously on the hull are no longer on it. These lines need to be removed to maintain the hull.

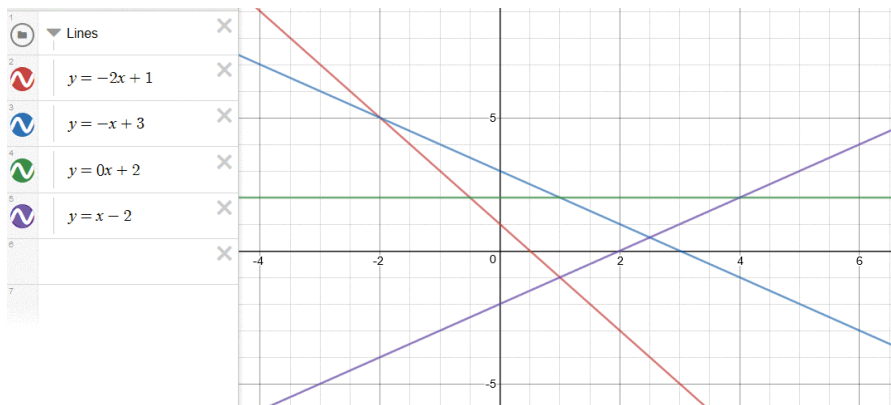
For this particular problem:

- The lines are inserted in sorted order of slope
- The query positions are also in sorted order

Implementing query and insert:

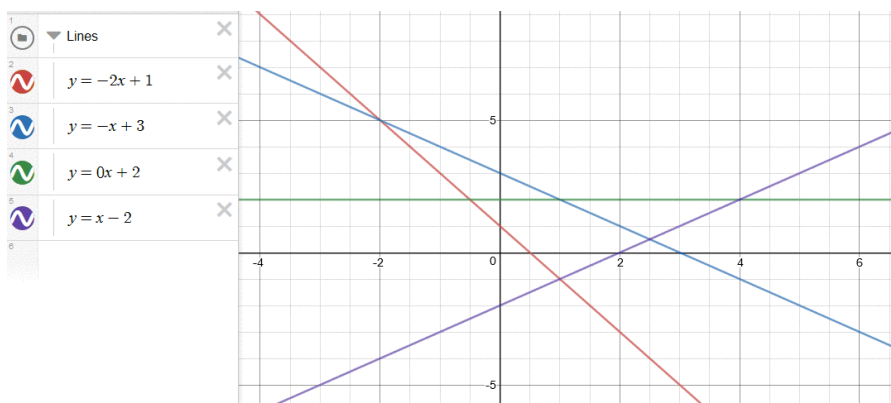
### Query

When querying at  $x = q_i$ , just compare the value at  $x$  of the rightmost line with that of the line next to it. If it is lower, remove it and repeat. When done, get the value at  $x$  of the rightmost line as the answer to the query.



### Insert

When inserting a line, if the intersection point of this line and the leftmost line lies to the right of that of the leftmost line and the line to the right of it, the leftmost line is no longer on the hull. Remove it, and repeat. Parallel lines pose an exception to this since they will never intersect, and must be handled separately if such a situation is possible in the problem.



→ [Find user](#)

Handle:

→ [Recent actions](#)

[ProgSlacking](#) → [Codeforces Round #641](#)



[Monogon](#) → [Codeforces Round #639 Editorial](#)

[de\\_dust2](#) → [Need help in player tournament question](#)

[Tameshk](#) → [Share your template! \(Using Carbon\)](#)

[alt\\_account](#) → [About the crazy rating inflation in the recent Div.4 contest](#)

[chokudai](#) → [AtCoder Beginner Contest 167 Announcement](#)

[decoder\\_1671](#) → [help in cc question](#)

[vovuh](#) → [Codeforces Round #640 \(Div. 4\) Editorial](#)

[kartik8800](#) → [CSES Range Queries section editorial](#)

[CodingKnight](#) → [Why does someone decide to hack his/her own solution?](#)

[npsecmigl](#) → [Practice mate](#)

[vovuh](#) → [Codeforces Round #605 \(Div. 3\)](#)

[pikmike](#) → [Educational Codeforces Round 82 Editorial](#)

[spookywooky](#) → [Unofficial editorial Codeforces Round #640 \(Div. 4\)](#)

[yongwhan](#) → [COVID-19 Virtual Contests \(on vjudge\)!](#)

[FieryPhoenix](#) → [Codeforces Round #638 \(Div. 2\) Editorial](#)

[Tanzir5](#) → [XX Open Cup GP of Bytedance](#)

[neo203](#) → [Div.4 contest rating changes unfair](#)

[pranay.agra](#) → [C++ Debug Flag](#)

[icecuber](#) → [CSES DP section editorial](#)

[MikeMirzayanov](#) → [About Division 4 Rounds](#)

[babu1998](#) → [Phoenix and Memory](#)

[zanoes](#) → [Codeforces Round #185 Editorial \(Div.2 B&Div.1 E\)](#)

[epiphany\\_21](#) → [Coding Interview Parody](#)

[Zeus\\_NoT\\_Zues](#) → [About Polluting Contest Announcements](#)

[Detailed →](#)

And that's it... since we add lines at one end and remove at both ends, the data structure for the job is a deque.

Solution for the rectangle problem:

Complexity is  $\mathcal{O}(N + Q)$  if  $N$  lines are inserted and  $Q$  queries are made.

## A few what ifs

### What if slopes are sorted in increasing order instead?

You can modify the logic accordingly.... or you can observe that negating the slope has the effect of mirroring lines about the Y-axis, so you can use one implementation for both.

### What if slopes are sorted but in reverse order of the query positions?

Both adding and removing will be done at one end, so a stack is required. Of course a deque can also do the job of a stack.

### What if minimum is required instead of maximum?

Again, you can modify the logic... or you can observe that negating both slope and Y-intercept has the effect of mirroring about the X-axis. You can use the same implementation.

## A more general problem

Let us consider a problem where

- The lines are inserted in sorted order of slope
- The query positions are in arbitrary order

To tackle this problem nothing needs to be changed for insertion. However we can no longer remove lines when answering queries. In order to answer queries, notice that each line provides the maximum in some range which is defined by its intersection point with the previous and next line. Given a particular  $x$  we can use binary search to find the line where the value will be maximum.

Solution for the rectangle problem using this method:

Complexity is  $\mathcal{O}(N + Q \log N)$ .

## The original problem

Coming back to the general version,

- The lines are inserted in arbitrary order of slope
- The query positions are in arbitrary order

This is referred to as the "fully dynamic" version of CHT. A convenient way to implement this is using a sorted set, such as `std::set` in C++ or `TreeSet` in Java. The idea is to maintain the set sorted by slope. To query, binary search is used as before. To insert, the position at which the line should be inserted is located. If this line does not appear on the hull, it is not inserted. If it does, useless lines are removed from both the left and right of the inserted line. The complexity using this method is  $\mathcal{O}((N + Q) \log N)$ .

You can find a neat implementation [here](#) (thanks to [Chilli](#) for the link). A couple more can be found [here](#) and [here](#). I do not want to go into further details about this method, because I personally find using Li Chao tree much simpler if the fully dynamic version is required. Li Chao tree is a specialized segment tree that also deals with the convex hull trick, and there exists a nice tutorial for it on [cp-algorithms](#). This page also contains an alternate interpretation of CHT.

## Conclusion

Although this tutorial focuses on the technique of CHT, it is worth mentioning that in contests CHT will almost always be intended as a way to optimize DP. You can refer to link titled "Dynamic Programming Optimizations" below to check out the forms of DP recurrences that can be optimized this way.

**Note about precision:** You may have noticed that the function `intersectX` in the code uses `long double` to find the coordinate. Is this good enough? Since queries are (usually) at integer  $x$ , the lines which provide the maximum in a range completely contained in interval between two consecutive integers are useless since they never provide a maximum at any integer coordinate. So we actually do not even need `long double`, floor/ceil division will do

just fine. Thanks to [tmwilliamlin168](#) for pointing this out to me. Note that integer division is not the same as floor division in C++ for negative numbers.

## Useful links

- [Convex hull trick \(PEGWiki\)](#)
- [Convex hull trick and Li Chao tree \(cp-algorithms\)](#)
- [Algorithms Live — Convex Hull Optimization \(YouTube\)](#)
- [Dynamic Programming Optimizations](#)
- [Fully Persistent Convex Hull Trick](#)

## Problems

Ordered approximately by difficulty:

- [1083E - The Fair Nut and Rectangles](#)
- [319C - Kalila and Dimna in the Logging Industry](#)
- [CYCLRACE – Cyclist Race](#)
- [Covered Walkway](#)
- [Machine Works](#)
- [Squared Ends](#)
- [631E - Product Sum](#)
- [932F - Escape Through Leaf](#)
- [311B - Cats Transport](#)
- [Avoiding Airports](#)
- [JUMP – Jump mission](#)
- [PDELIV – Pizza Delivery](#)
- [YATP](#)
- [POLY – Polynomials](#)
- [WEASELSC – Weasel finds Staircase](#)


That concludes my first tutorial on Codeforces. Thanks for reading and I hope it was useful.

Any suggestions or improvements are welcome.

The nice images above were made with [Desmos](#).

If you want other links/problems on CHT to be added, comment below and I will add them.

 Tutorial of ICPC:Practice Problems

 convex hull optimization, dynamic programming, geometry, tutorial

 +274  

 [meooow](#)

 17 months ago

 29



## Comments (29)

[Write comment?](#)



arknave

17 months ago, <#> | ☆

 +5 

Great tutorial! Another good resource for those who prefer to learn from videos is [Algorithms Live — Convex Hull Optimization](#).

→ [Reply](#)



meooow

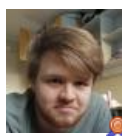
17 months ago, <#> ^ | ☆

← Rev. 2

 +8 

I've added the link. One thing that irked me, in the first part the author says that  $(x - y)^2 + prevCost$  is not really CHT because the functions are parabolic and not straight lines, but the expression can be expanded to  $y^2 - 2xy + x^2 + prevCost$  which needs to be minimized for fixed  $y$  over some  $x$ , so it actually can be solved in the normal way with a convex hull of lines.

→ [Reply](#)



-is-this-fft-

17 months ago, <#> | ☆

 +11 

I think [PDELIV](#) deserves a mention in the problem list. It requires you to use it in a way I personally hadn't considered before.

→ [Reply](#)



17 months ago, # ^ | ☆

▲ 0 ▼

Nice problem, added to the list!

→ Reply

meoooow

17 months ago, # | ☆

← Rev. 2

▲ +23 ▼

I like the implementation created by **simonlindholm**, found in the KTH notebook.I originally saw **ksun48** use it here:<https://codeforces.com/contest/1083/submission/46863810>.

Chilli

You can find it in here: <https://github.com/kth-competitive-programming/kactl/blob/master/content/data-structures/LineContainer.h>

I think it's a lot less magic than the other 2 implementations linked (no mutable member functions/closures), and I believe it's also substantially faster.

→ Reply



17 months ago, # ^ | ☆

▲ +5 ▼

This implementation appears short and neat. Added to the blog. Have you also compared the performance?

→ Reply

meoooow

17 months ago, # ^ | ☆

← Rev. 2

▲ +5 ▼

The primary thing that differentiates this implementation is that it stores the intersection point during insertion. This makes the implementation a lot shorter as well as the queries somewhat faster. Overall, compared to the other 2 implementations linked (called HullDynamic and chtDynamic respectively), it's somewhat slower at insertion than the other two, significantly faster at querying than HullDynamic, and slightly faster at querying than chtDynamic. Overall, it's very competitive in performance.

Insertions (1e7 insertions)

**LineContainer:** 0.588702**HullDynamic:** 0.480461**chtDynamic:** 0.513528

Chilli

Queries (1e5 insertions, 1e7 queries)

**LineContainer:** 0.109056**HullDynamic:** 0.385707**chtDynamic:** 0.146591

Overall (1e7 insertions, 1e7 queries)

**LineContainer:** 0.784574**HullDynamic:** 0.995786**chtDynamic:** 0.768847

And finally, line count :^)

**LineContainer:** 36**HullDynamic:** 44**chtDynamic:** 75

I think the KTH implementation is clearly the winner.

Benchmarks can be found here: <https://ideone.com/caYDdF>

→ Reply



dacin21

17 months ago, # ^ | ☆

▲ +20 ▼

Starting with C++14, `std::less<void>` is transparent, so you don't even need the hack with the global bool Q. Instead, you can use different `operator<` for lines and query points. [submission](#)

→ Reply



Chilli

17 months ago, # ^ | ☆

▲ 0 ▼

Oh, that's nice. Is there any reason you made p mutable?

→ Reply



dacin21

17 months ago, # ^ | ☆

▲ +5 ▼

p is the x-coordinate of the intersection with the next line and you need to update that when inserting new lines. A line inside the set is const, so you need mutable to make p modifiable. (k and m don't need to be changed, so they're not mutable.)

→ Reply



simonlindholm

17 months ago, # ^ | ☆

▲ +10 ▼

Oh, neat! I've made that change to KACTL: <https://github.com/kth-competitive-programming/kactl/commit/165807e28402c9be906f6e6a09452431787bb70d>

→ Reply



low\_kii\_savage

11 months ago, # ^ | ☆

▲ 0 ▼

Is it possible to remove lines from the struct?

→ Reply

11 months ago, # ^ | ☆

▲ +4 ▼



meooow

You can technically remove lines from the structure, but you cannot bring back the lines you previously discarded for the purpose of maintaining only the hull instead of all lines.

One or more of those discarded lines may have the second largest value at some  $x$  where the removed line had the max value, which you cannot recover. So you will be having an incomplete hull.

→ Reply



low\_kii\_savage

11 months ago, # ^ | ☆ 0 ▼

Yeah, that makes sense. So is there any other way which allows remove or update queries on the line parameters while maintaining the complete hull?

→ Reply

9 months ago, # +3 ▼

| ☆



meooow

Not that I know of, assuming you want to keep the same or close enough complexity.

→ Reply

9 months ago, # 0 ▼

| ☆

If queries is offline I



lessmeaning

in queries is similar to think Divide & Conquer  $O(n * \log^2)$  helps like in Dynamic Connectivity (easy google). Online harder, idk maybe some kind of SQRT decomposition on queries. Smth like keep last B queries and proceed in stupid way, for other queries there is built CHT. So number of stupid asks will be  $B * q$ , number of CHT rebuilds will be  $q / B$ . To avoid sorting we can merge, so if  $B = \sqrt{n}$ , and for simplicity  $q = n$ . Complexity is  $O(n * \sqrt{n} + q * \log(n))$

→ [Reply](#)

IF\_IR\_A\_1\_IR

9 months ago, # ^ | ☆

▲ 0 ▼

You can do it using treap

→ [Reply](#)

VLADO

8 months ago, # ^ | ☆

← Rev. 2

▲ 0 ▼

How do I make it query the minimum value instead of the maximum?  
I'm just starting to learn this, so sorry for the dumb question.

Edit: I figured it out, you're supposed to insert the negatives.

→ [Reply](#)

Irvideckis

17 months ago, # | ☆

▲ +5 ▼

Here's another related problem: [YATP](#).→ [Reply](#)

meooow

17 months ago, # ^ | ☆

▲ +5 ▼

Nice problem. If anyone needs hints...

Spoiler→ [Reply](#)

akapros

17 months ago, # | ☆

▲ +5 ▼

Great Tutorial!!!

→ [Reply](#)

its\_ulure

16 months ago, # | ☆

← Rev. 2

▲ +5 ▼

This problem [POLY](#) can also be added here.→ [Reply](#)

meooow

16 months ago, # ^ | ☆

▲ 0 ▼

Thanks, updated.

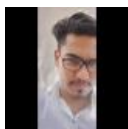
→ [Reply](#)

Sanitator

12 months ago, # | ☆

▲ 0 ▼

Also, [Lena](#) and [Queries](#)→ [Reply](#)



rawnix

10 months ago, # | ☆

▲ 0 ▼

Great Tutorial! I tried solving the problem [1083E - The Fair Nut and Rectangles](#) but for some reason my code is giving WA on test 5. Can someone please help me. [57194241](#)

→ [Reply](#)

meooow

9 months ago, # ^ | ☆

▲ +3 ▼

You're forcibly including the first rectangle always. The optimal solution might leave it out.

Fix is that when in `ll m = get_max(lines, v[i].q);` you find `m < 0` you should not add it to `dp[i]`.

→ [Reply](#)

VLADO

8 months ago, # | ☆

← Rev. 2 ▲ 0 ▼

How do I modify the data structure so it gets the minimum at a point instead of the maximum?

Edit: I figured it out, you should insert the negatives of the slopes and constants.

→ [Reply](#)

JioFell

8 months ago, # | ☆

▲ 0 ▼

which order of the slopes or queries are relevant? decreasing or increasing. Or both? I'll be appreciated if you answer this comment :3

→ [Reply](#)

grey\_rabbit

4 months ago, # | ☆

▲ 0 ▼

The only difference between my AC code [69191641](#) and my WA on test 6 code for problem E — The Fair Nut and Rectangles was the "long double" used for comparing in fuction `check()`, which i put there because I saw that in many other's code. However, I didn't used any division, and the problem statement clearly said that xi, yi, ai are all int, so I'm very confused. Can someone please explain ?

→ [Reply](#)

[Codeforces](#) (c) Copyright 2010-2020 Mike Mirzayanov  
The only programming contests Web 2.0 platform  
Server time: May/11/2020 17:05:05<sup>UTC+6</sup> (h1).  
Desktop version, switch to [mobile version](#).  
[Privacy Policy](#)

Supported by



ITMO UNIVERSITY