TFT 2017 Solution
Q3 Secret Meeting

Alex Tung

6 May 2017

Brief Description

- N people, position = $(x_i + t * u_i, y_i + t * v_i)$ on day t (coordinates are even)

- Q queries of the form (C_i, D_i)
- They want to meet on one of days 1...T at (X, Y)
- Target: minimize maximum cost to reach (X, Y)
- Cost from (x_1, y_1) to (x_2, y_2) is:

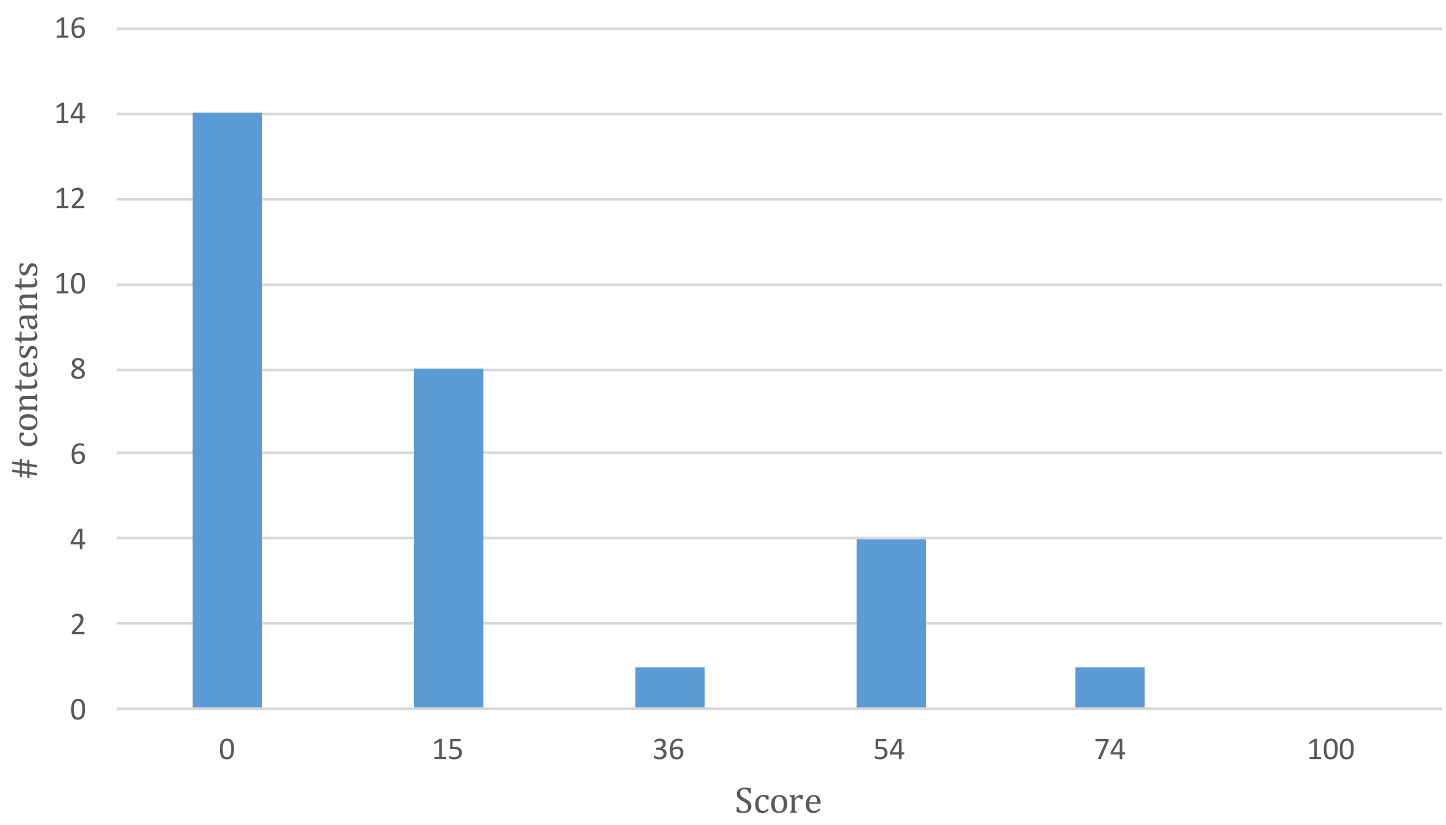
$$\max(C_i * |x_1 - x_2|, D_i * |y_1 - y_2|)$$

SUBTASKS

For all cases: $1 \le N \le 4 \times 10^4$, $1 \le T \le 5 \times 10^5$, $-10^9 \le x_i, y_i \le 10^9$, $-2000 \le u_i, v_i \le 2000$, $1 \le Q \le 10^5$, $1 \le C_i, D_i \le 10^9$.
In addition, x_i, y_i, u_i, v_i are all even numbers.
Note that the constraints guarantee that they will not go beyond the boundary of the city during the T days.

	Points	Constraints
1	12	$N \le 50$ $T = 1$ $Q \le 10$ $-200 \le x_i + u_i \le 200$, $-200 \le y_i + v_i \le 200$
2	22	$1 \le N \le 50$ $1 \le T \le 200$ $1 \le Q \le 2000$
3	21	$1 \le N \le 50$ $1 \le Q \le 2000$
4	18	$1 \le T \le 1000$ $1 \le Q \le 2000$
5	17	$1 \le T \le 30000$ $1 \le Q \le 2000$
6	10	No additional constraints

T173 Statistics



Task	Attempts	Max	Mean	Std Dev
T173 - Secret Meeting	28	74	15.928	22.053

Subtask 1

- The possible meeting places are (X, Y), where X and Y are in [-200, 200]
- Try all positions, calculate maximum cost as described

- Time complexity: $O(QNTR^2)$, where R = 401 and T = 1

Subtask 2

- Observation 1: for a fixed day t, if:
  - The smallest x-coordinate is l_t
  - The biggest x-coordinate is r_t
  - The smallest y-coordinate is d_t
  - The biggest y-coordinate is up_t
- Then ans[t] = $\max(C_i * (r_t - l_t)/2, D_i * (up_t - d_t)/2)$

- In other words, the minimal cost on a day is determined solely by the <u>bounding rectangle</u> of the N points (people)

Subtask 2

- So the algorithm is: for each query, for each day, calculate the bounding rectangle in O(N) time

- Time complexity: O(QNT)

Subtask 3

- Observation 2: The answer is convex in t
- That is, ans[t + 1] – ans[t] >= ans[t] – ans[t – 1]

- Why? Because:
  - dx[i][j][t] := $|(x_i + t * u_i) - (x_j + t * u_j)|$ is convex in t
  - dy[i][j][t] := $|(y_i + t * v_i) - (y_j + t * v_j)|$ is convex in t
  - ans[t] = $\max_{i,j}$(C * dx[i][j][t] / 2, D * dy[i][j][t] / 2)
- So, ans[t] is also convex in t, as a maximum of convex functions

Subtask 3

- We can apply ternary search to find the optimal day
- Alternatively, we can use a binary search:

```
int l = 0, r = T;
while(r – l > 1){
    int mid = (l + r) / 2;
    if(ans[mid] > ans[mid + 1]) l = mid; //O(N) calculation
    else r = mid;
}
```

- Time complexity: O(QN log T)

Subtask 4

- Observation 3: Based on observation 1, the answer depends on the bounding rectangle only. So for each day, we only need O(1) information!
- Precisely, we <u>only need to know (l_t, r_t, d_t, up_t)</u> for each day t
- This can be precomputed in O(NT)

Subtask 4

- Then, for each query, for each t in [1, T], we can calculate ans[t] in O(1) time

- If you get observation 2 (ans[t] is convex), time complexity = O(NT + Q log T)
- Otherwise, time complexity = O(NT + QT)

Subtasks 5, 6

- The issue is that O(NT) precomputation is too slow
- We can do better, using O(N log N + T) to precompute (l_t, r_t, d_t, up_t) for each day t in [1, T]

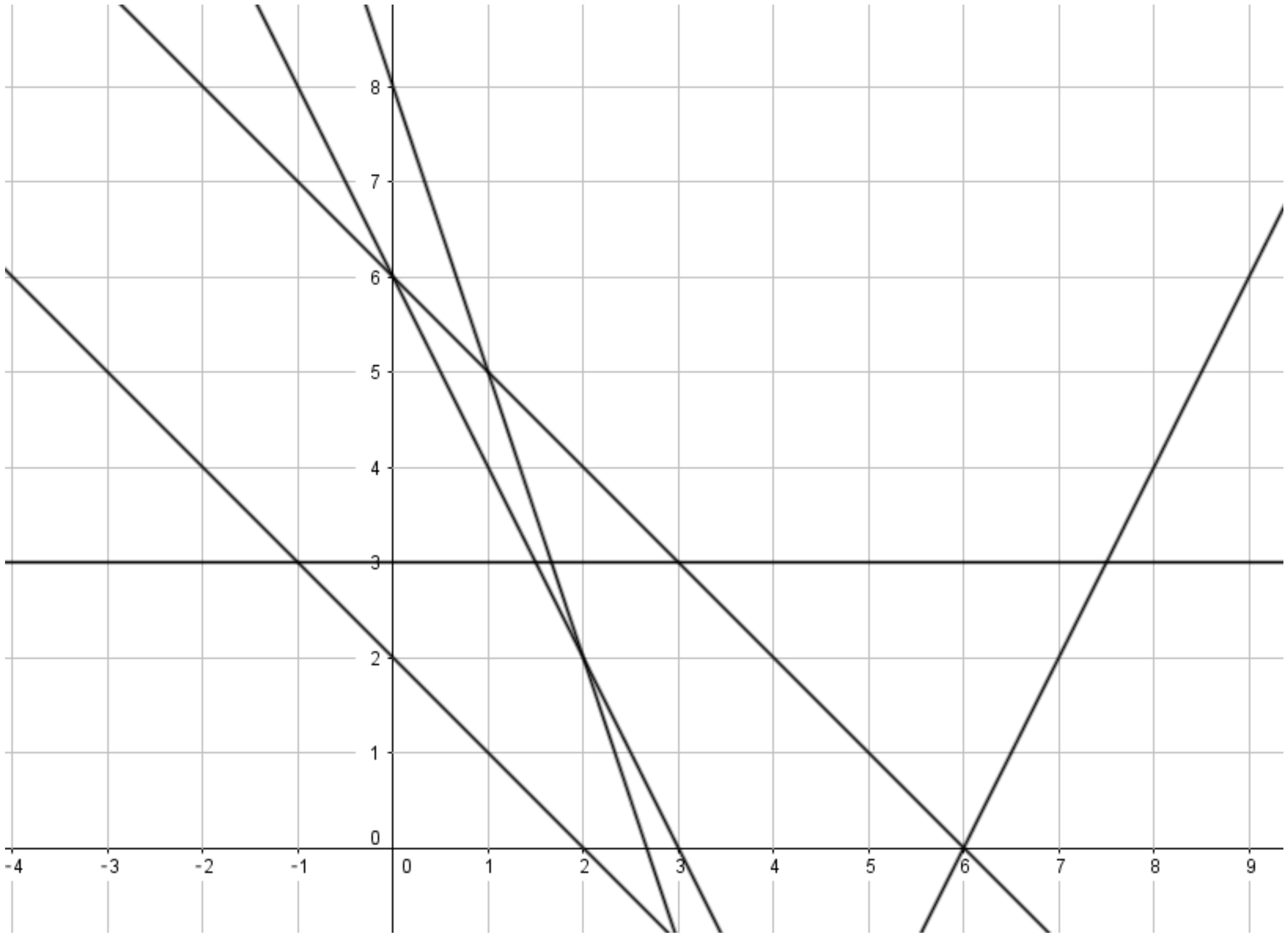
- Suppose this is done
- Overall time complexity = O(N log N + T + QT)
- With observation 2, we have a solution with time complexity = O(N log N + T + Q log T), which gets 100 points :)

Precomputation in $O(N \log N + T)$

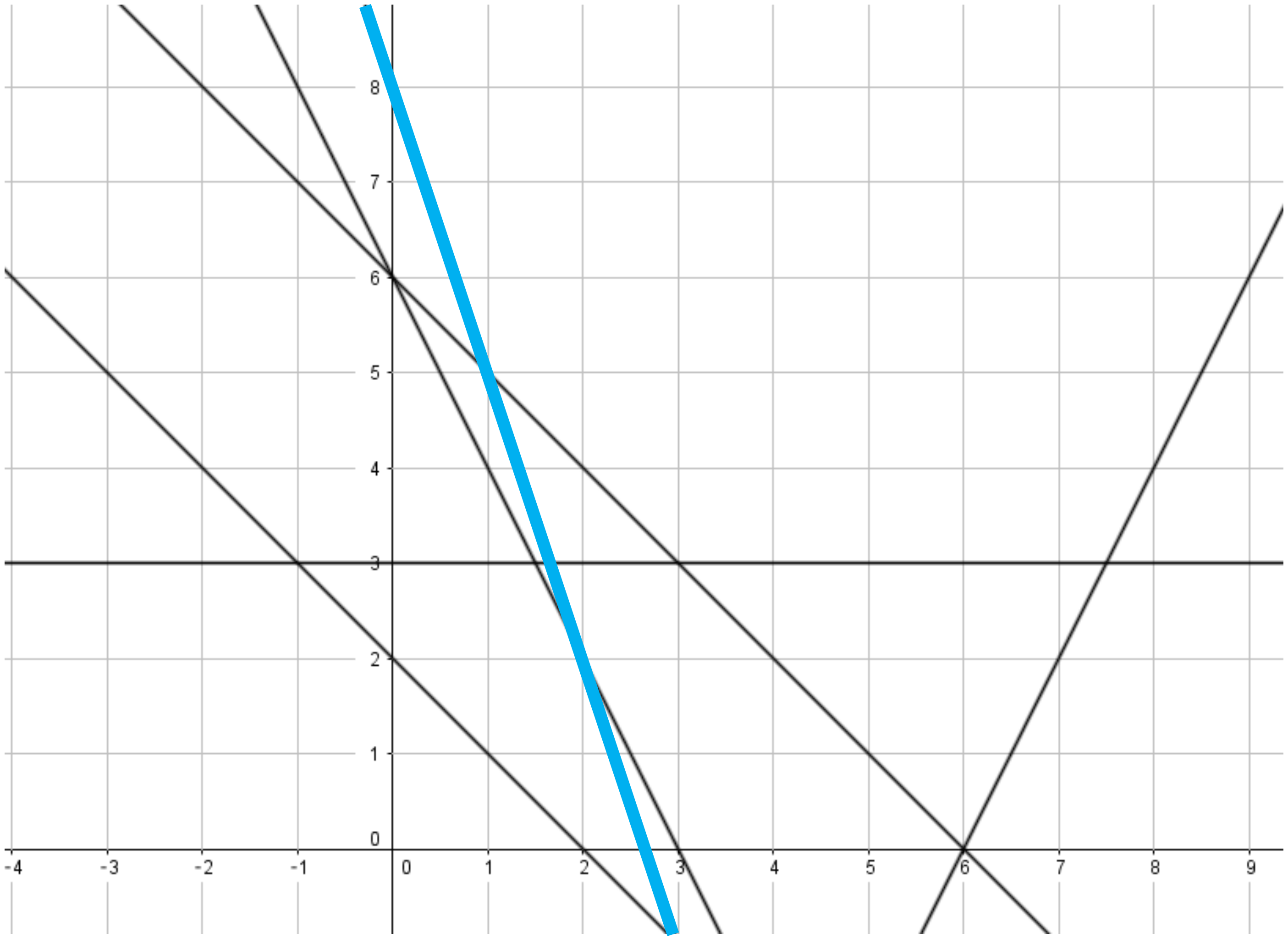
- Let us see how to calculate r_t, the largest x-coordinate on day t (others are similar)
- $r_t = \max_i(x_i + t * u_i)$
- Each $(x_i + t * u_i)$ describes a line in x-t graph
- This is just finding a "convex hull" in x-t graph!

- Sort the N lines by increasing slope, then use a stack to maintain the convex hull
- Then, scan the hull once to get r_t for each t

Finding the convex hull

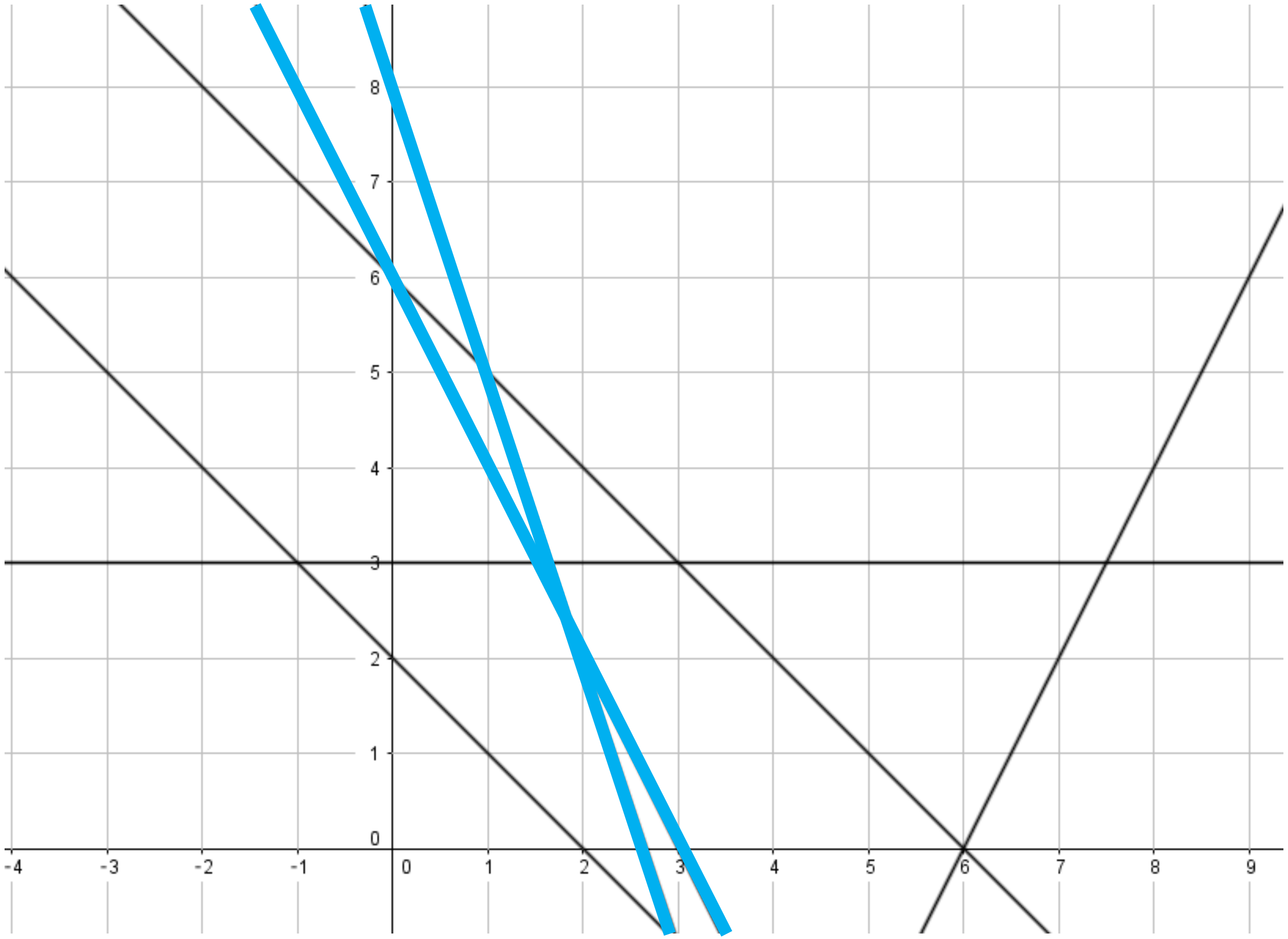


Finding the convex hull



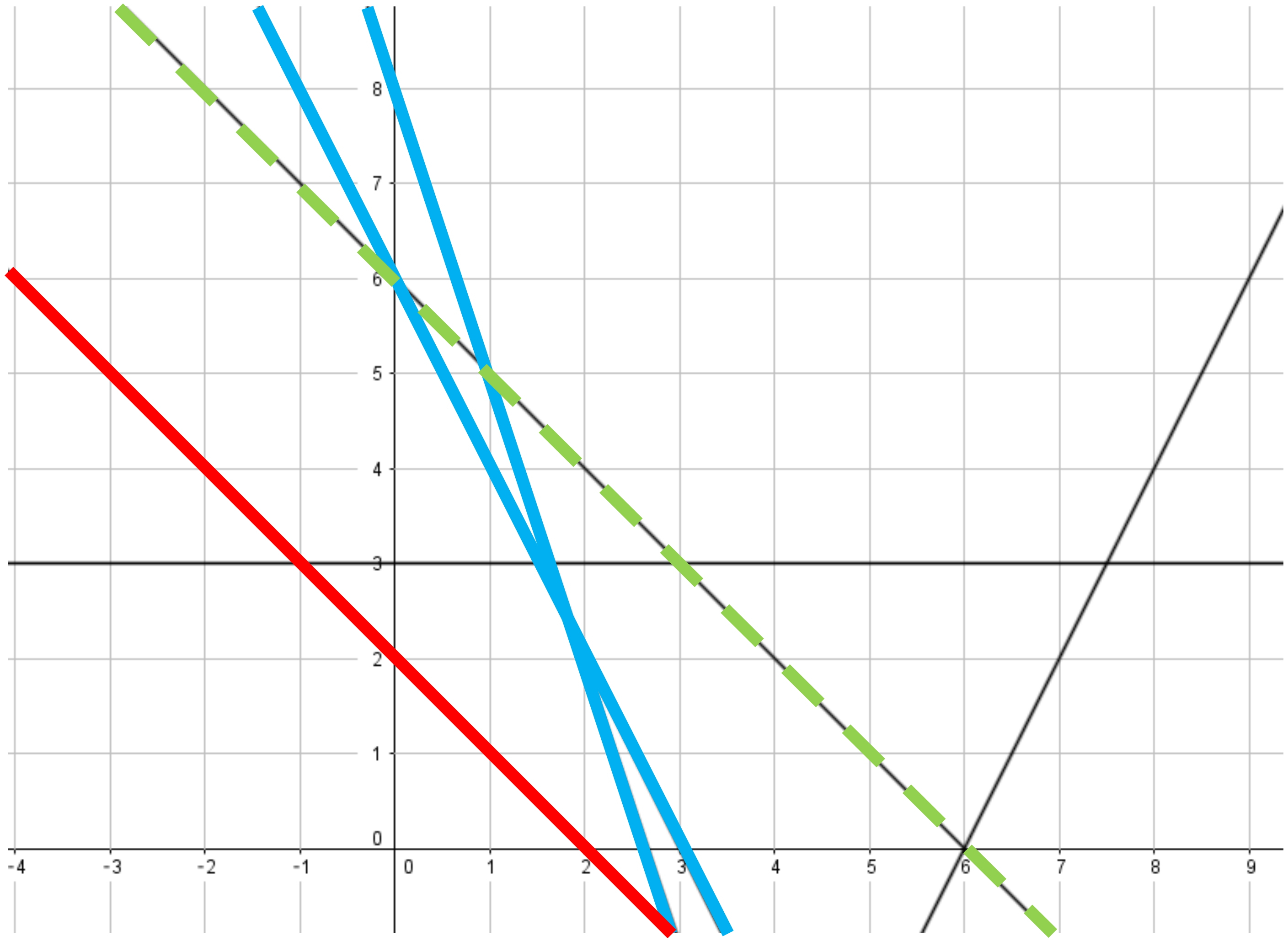
- add a line

Finding the convex hull



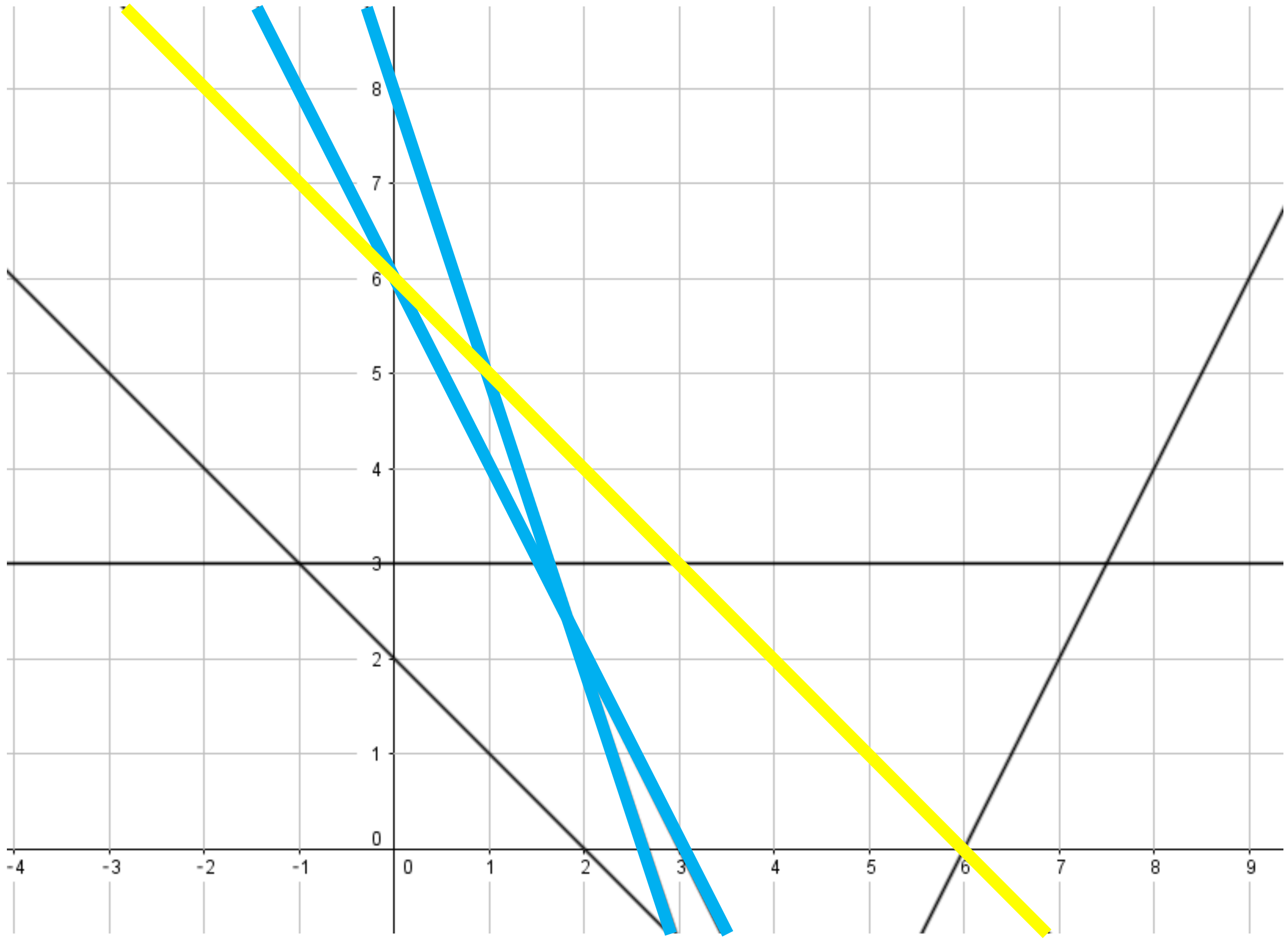
- add a line

Finding the convex hull



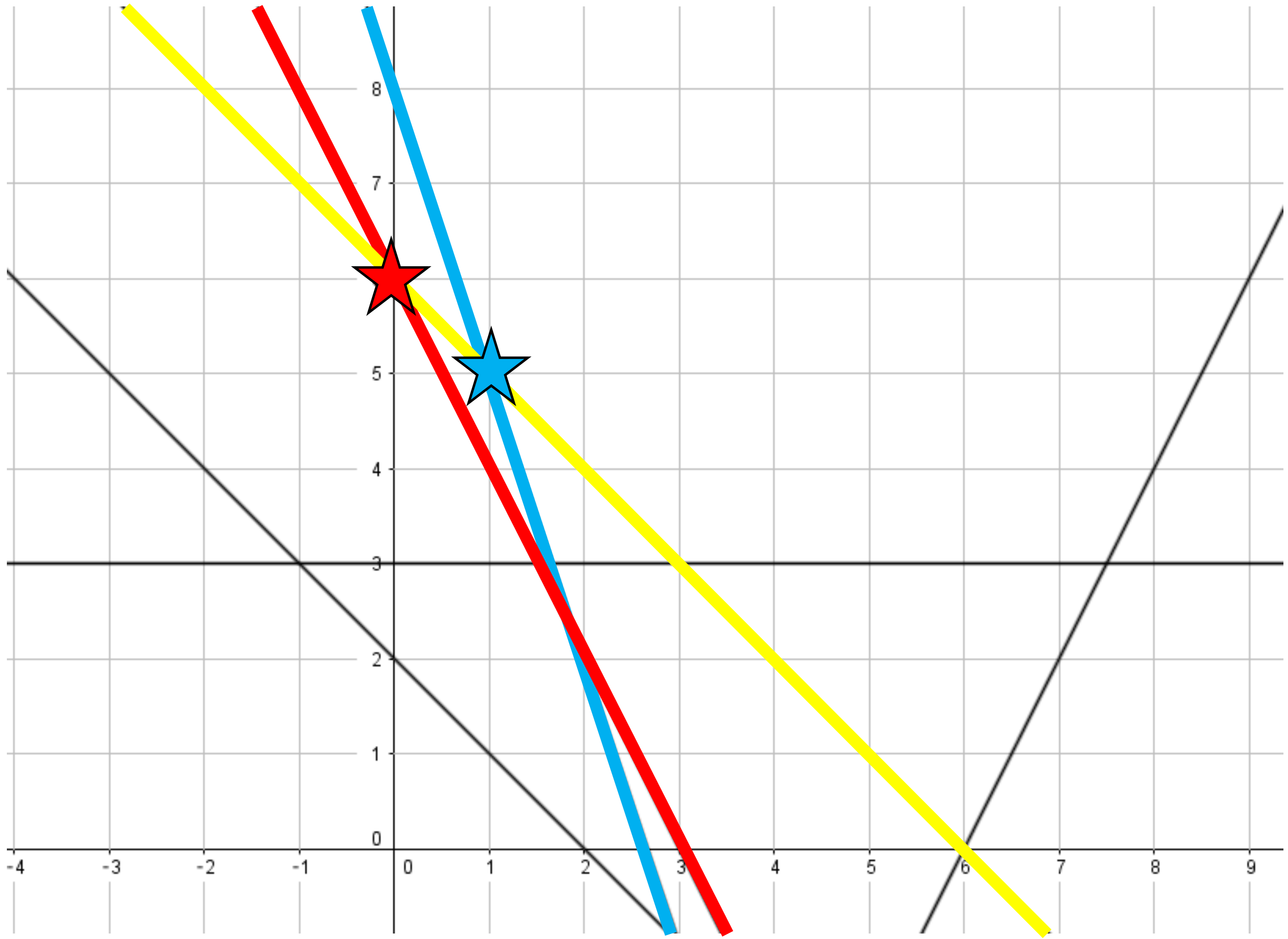
- Ignore red line because there is a green line which "dominates" it (i.e. same slope, larger y-coordinates)

Finding the convex hull



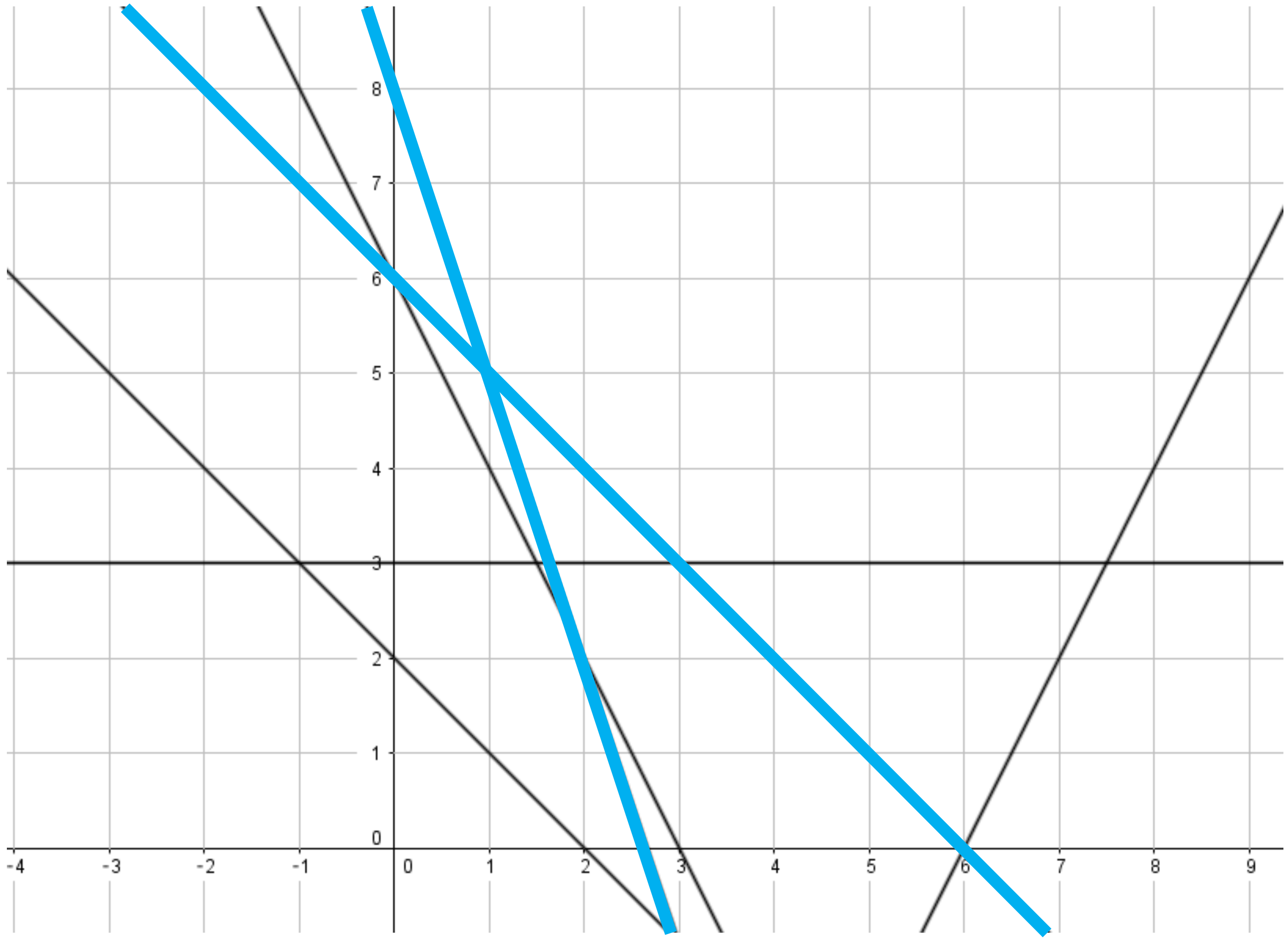
- add a line

Finding the convex hull

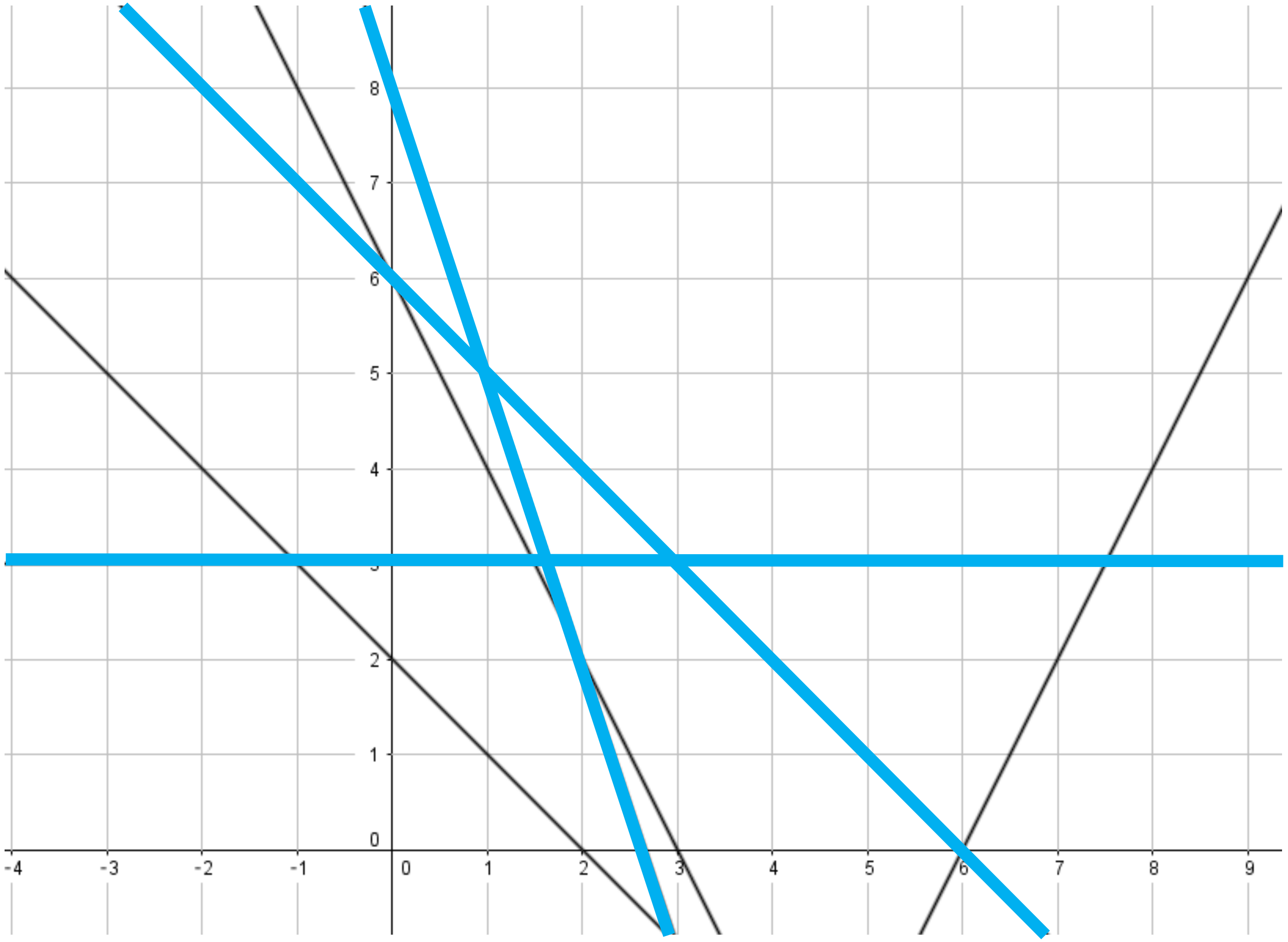


- Red line does not belong to the convex hull; pop from stack
- One way to check: compare positions of intersections

Finding the convex hull

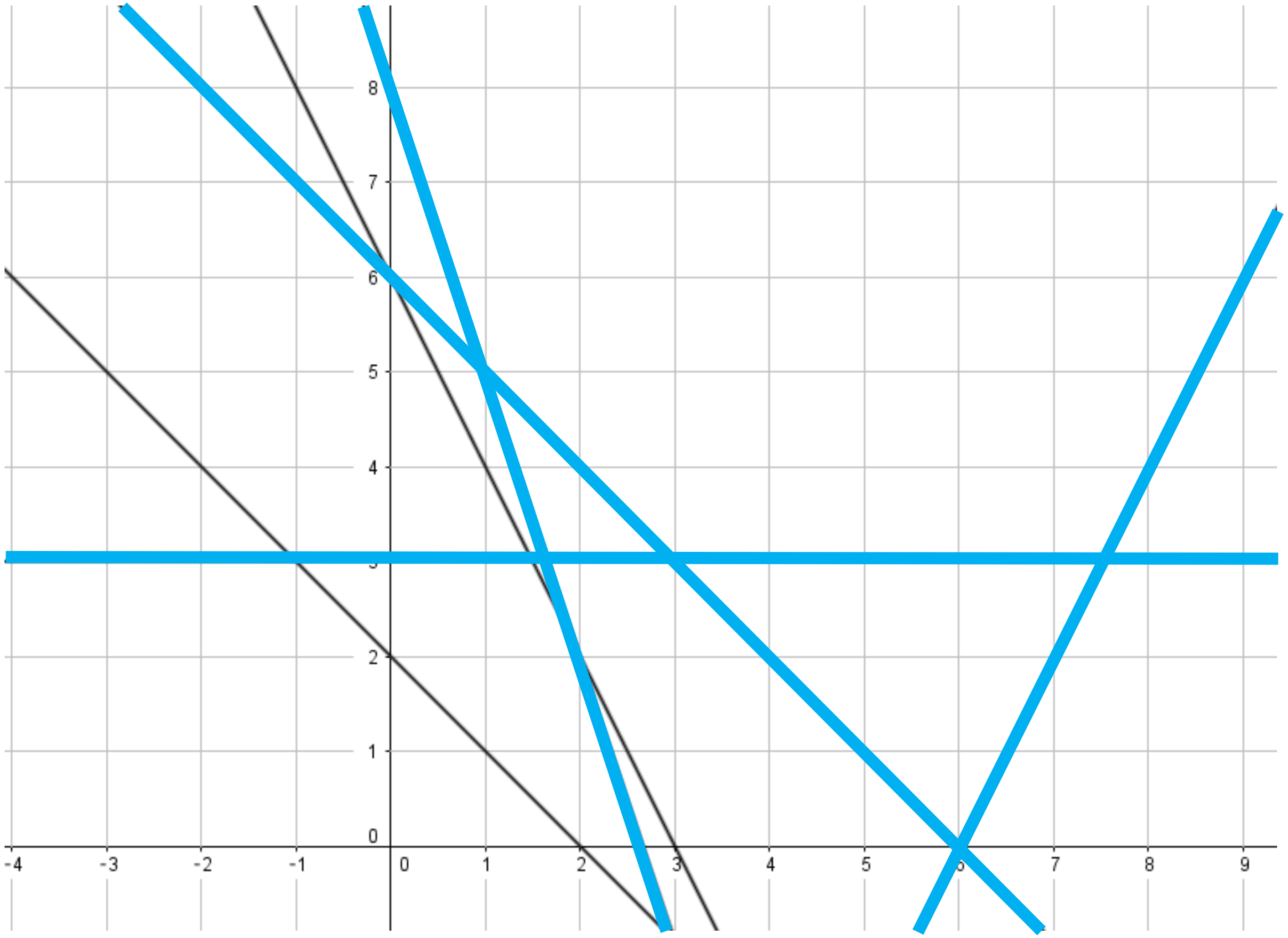


Finding the convex hull



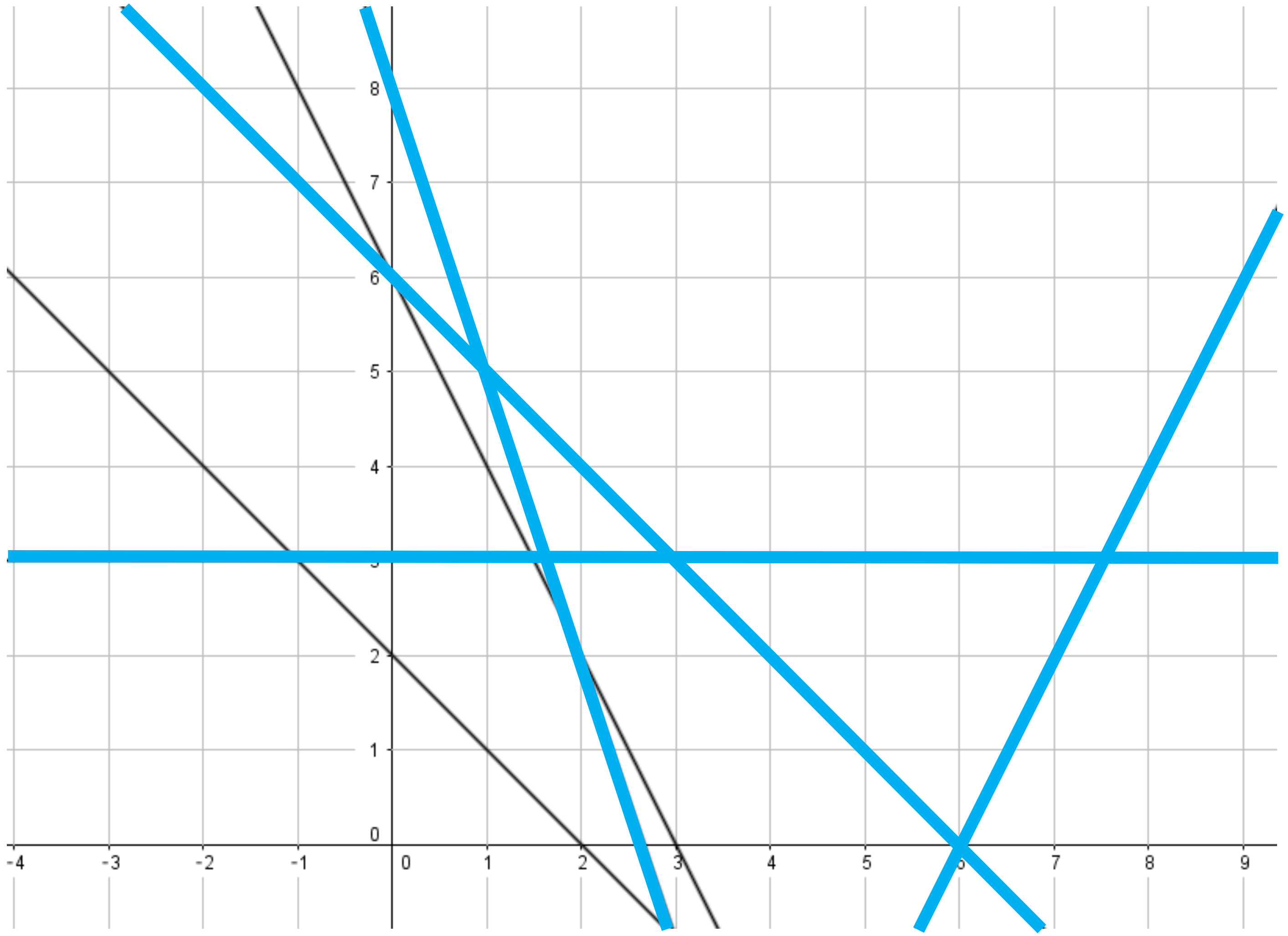
- add a line

Finding the convex hull

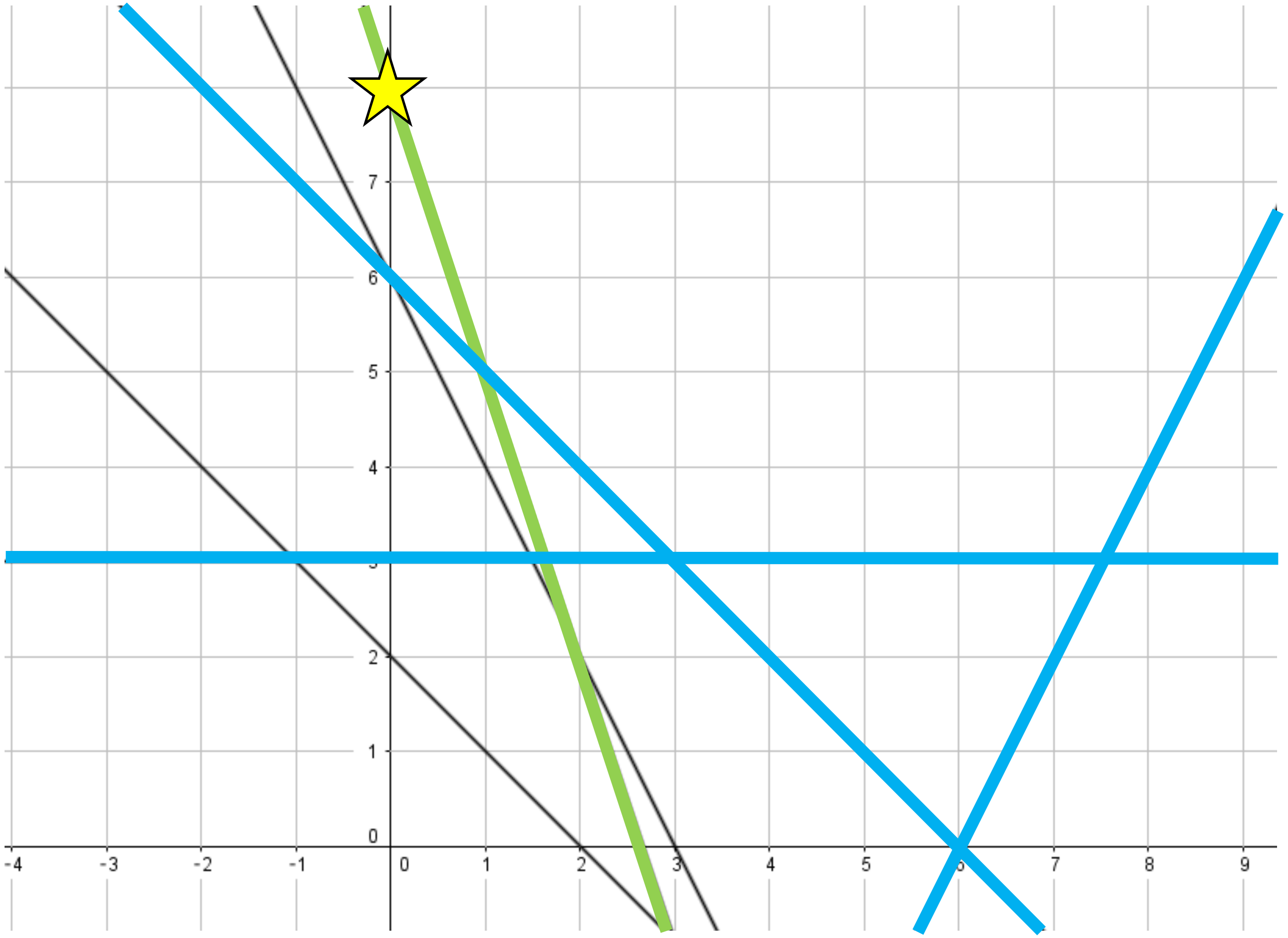


- add a line

Retrieving r_t



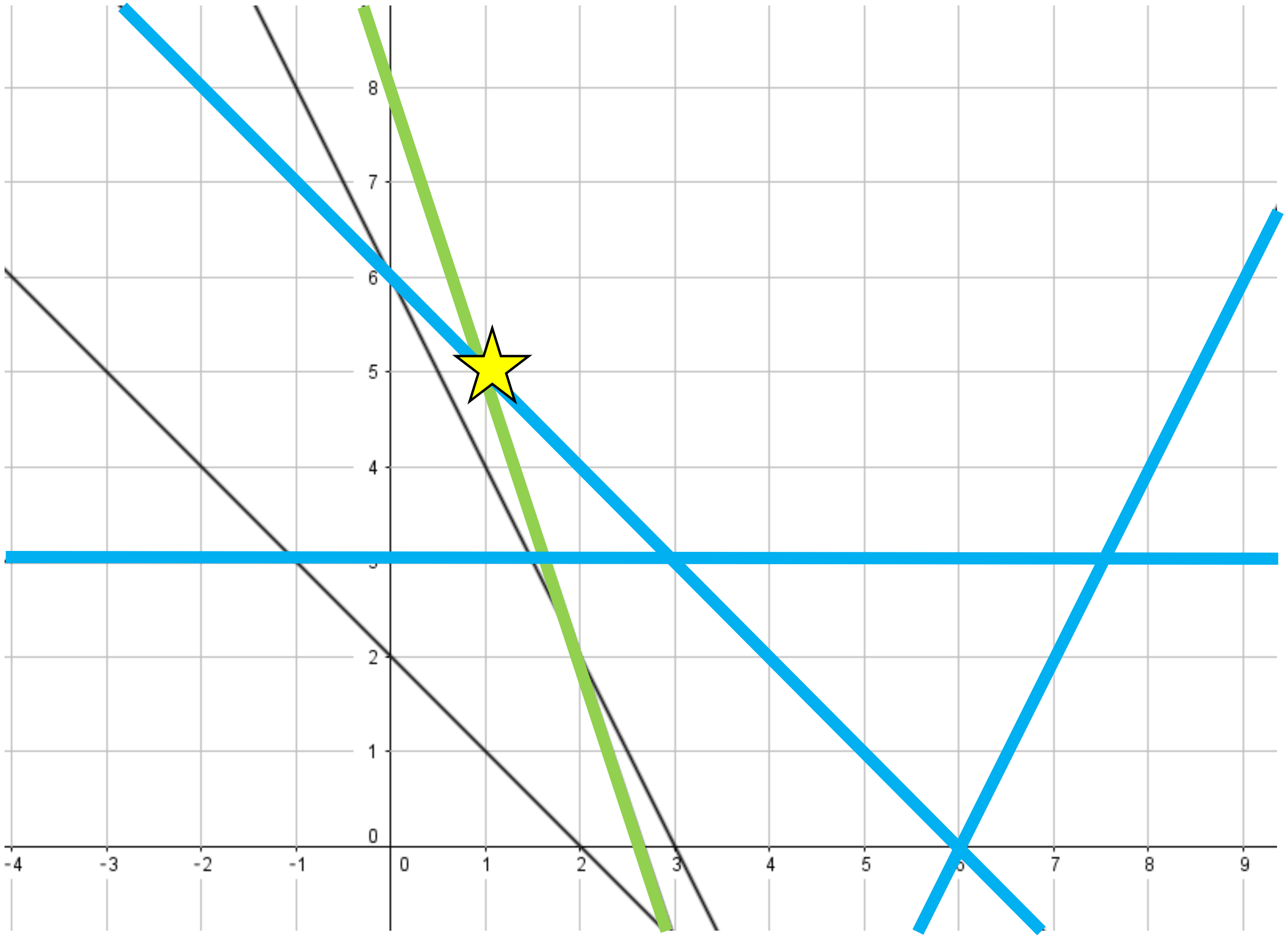
Retrieving r_t



- $t = 0$, $r_0 = 8$

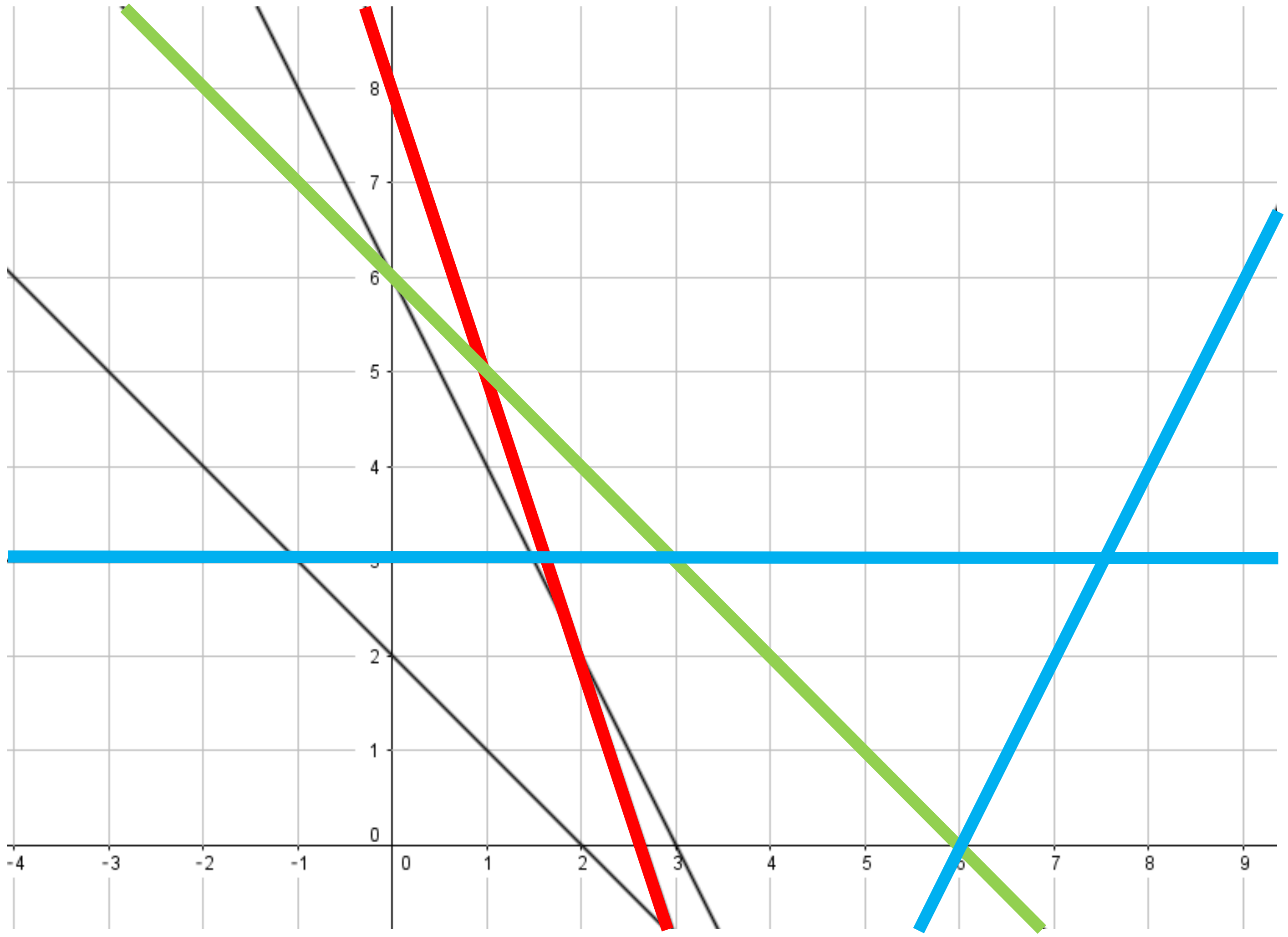
(green: active)

Retrieving r_t



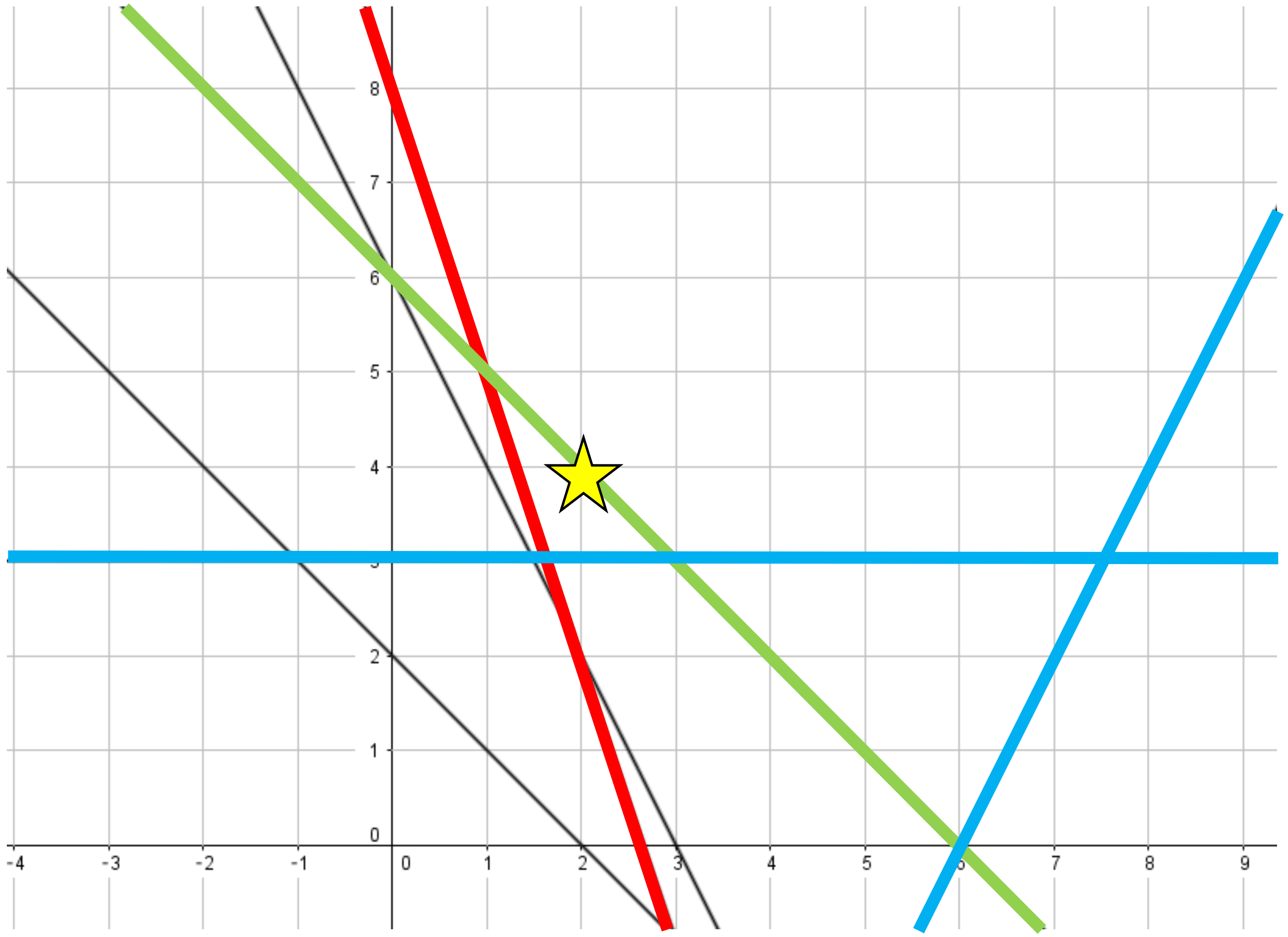
- $t = 1,\ r_1 = 5$

Retrieving r_t



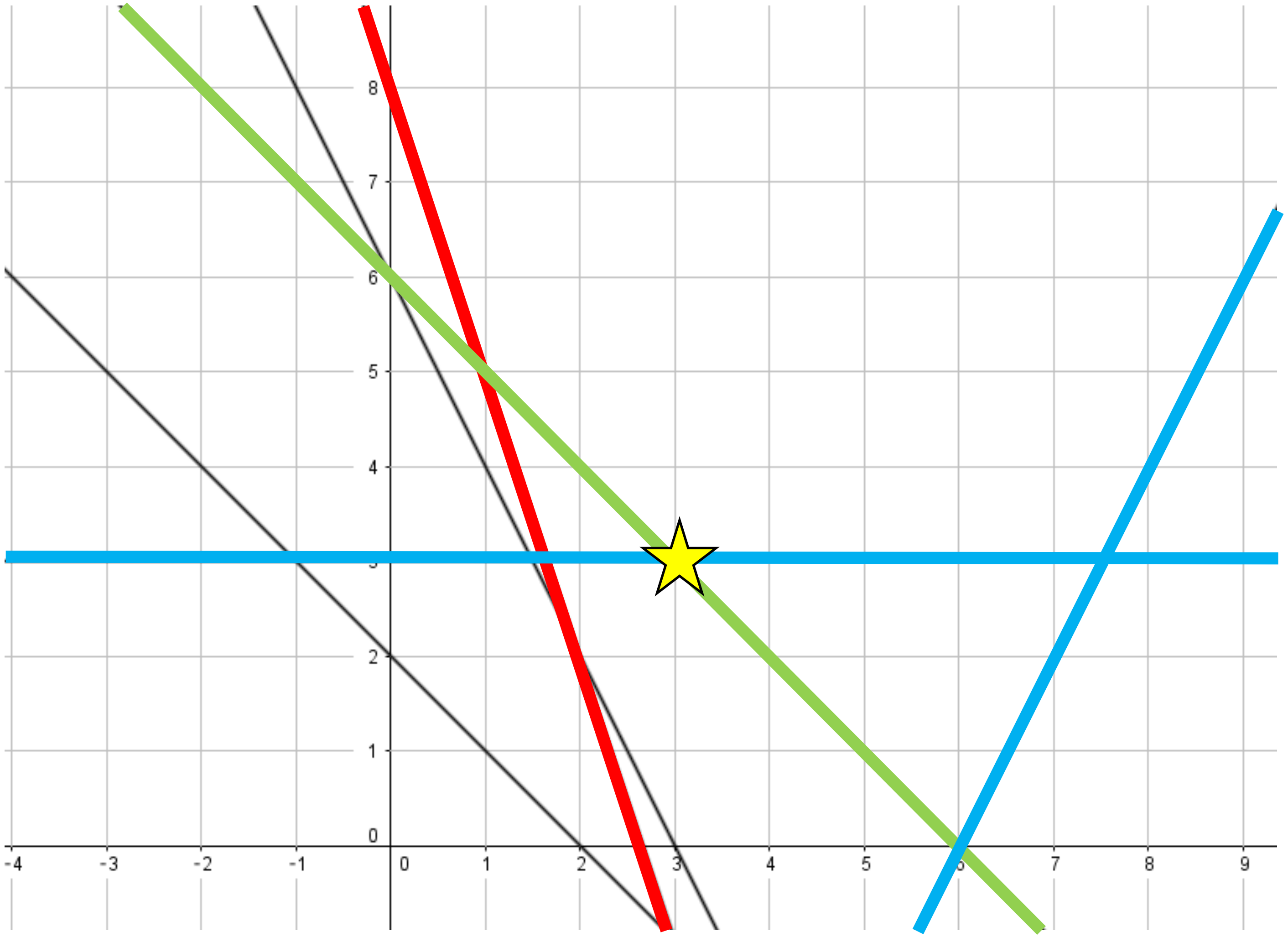
- t = 2, next line better!

Retrieving r_t



- $t = 2$, $r_2 = 4$

Retrieving r_t

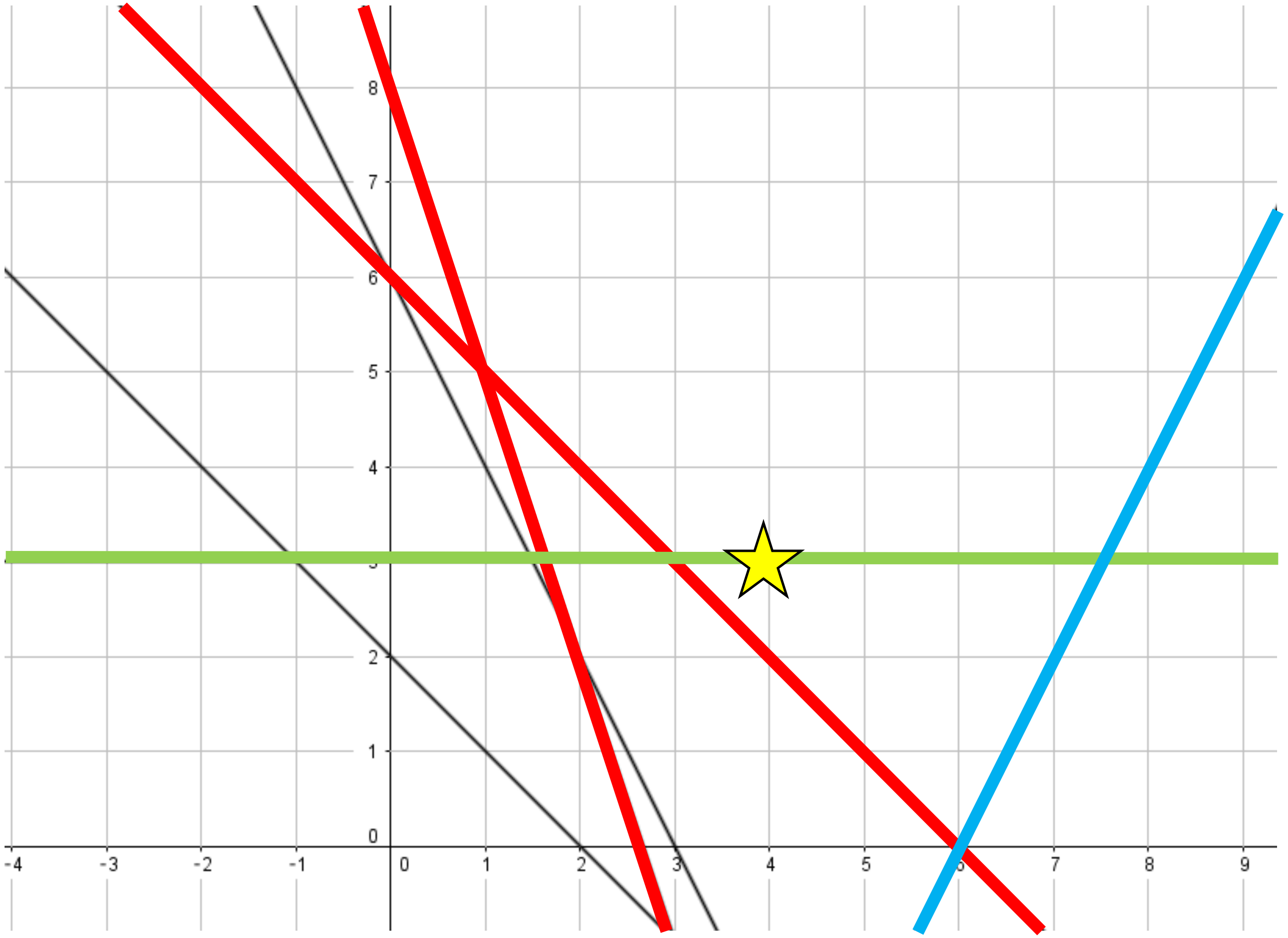


- $t = 3$, $r_3 = 3$

Retrieving r_t

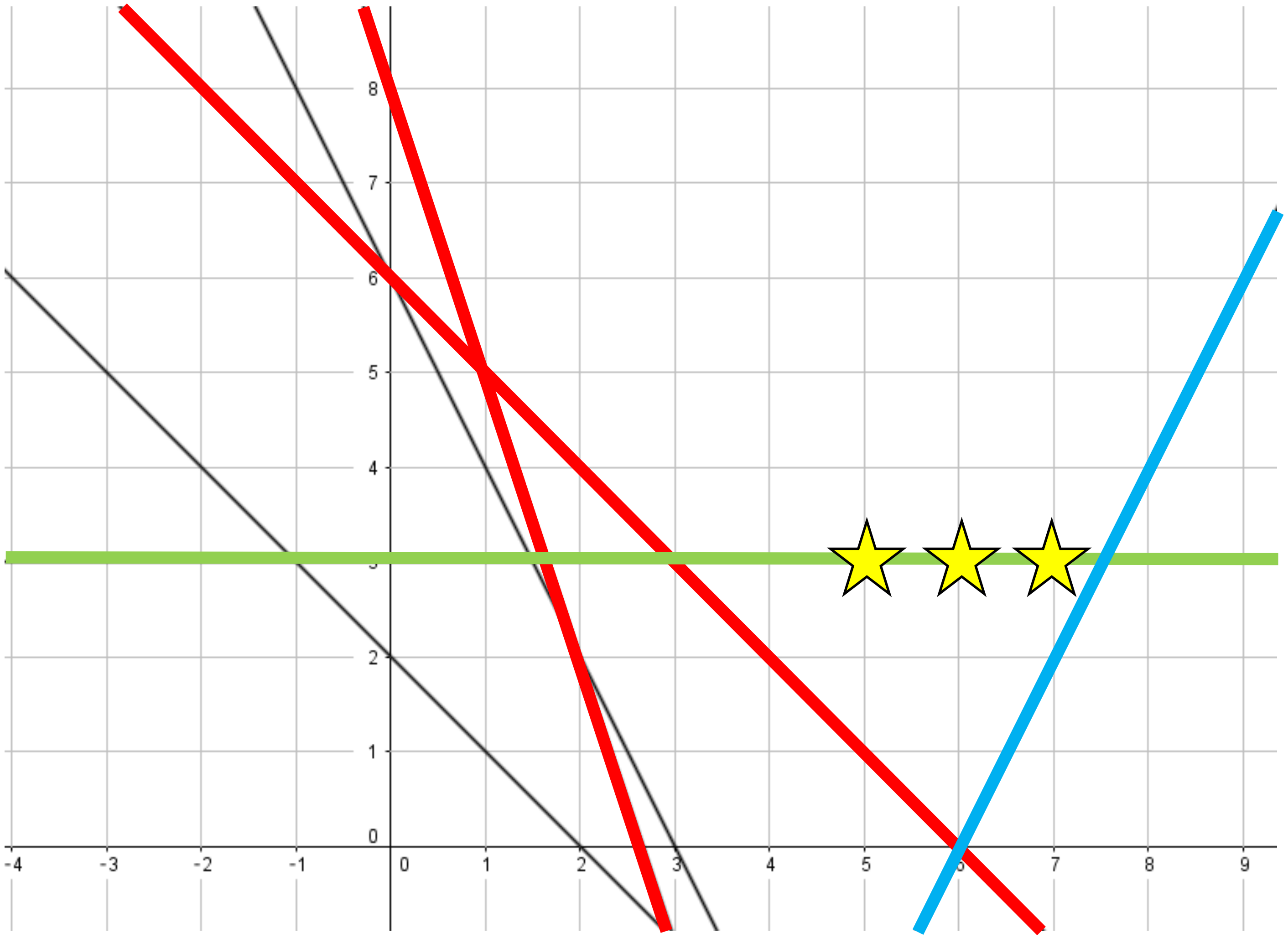
- t = 4, next line better!

Retrieving r_t



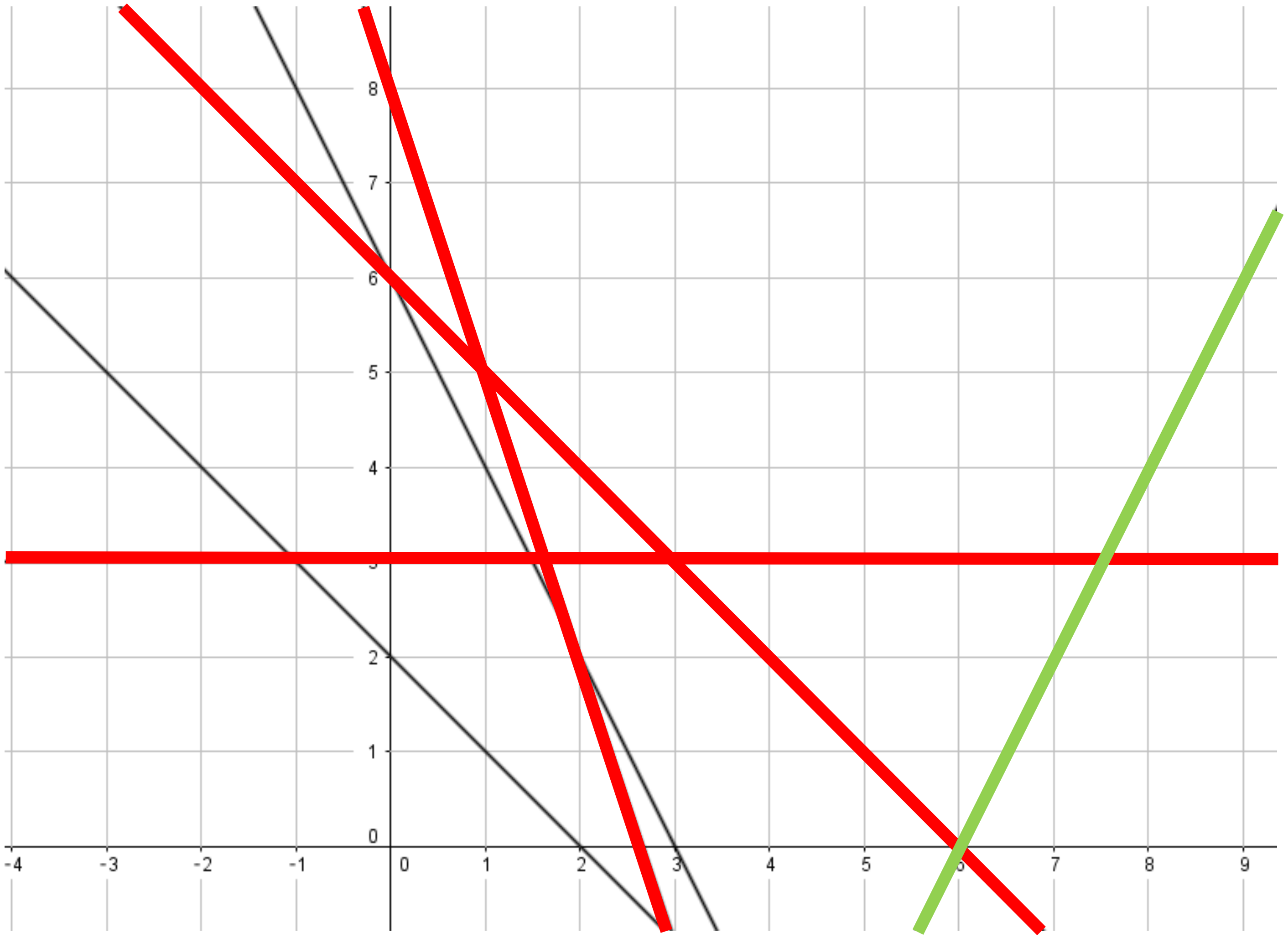
- $t = 4,\ r_4 = 3$

Retrieving r_t



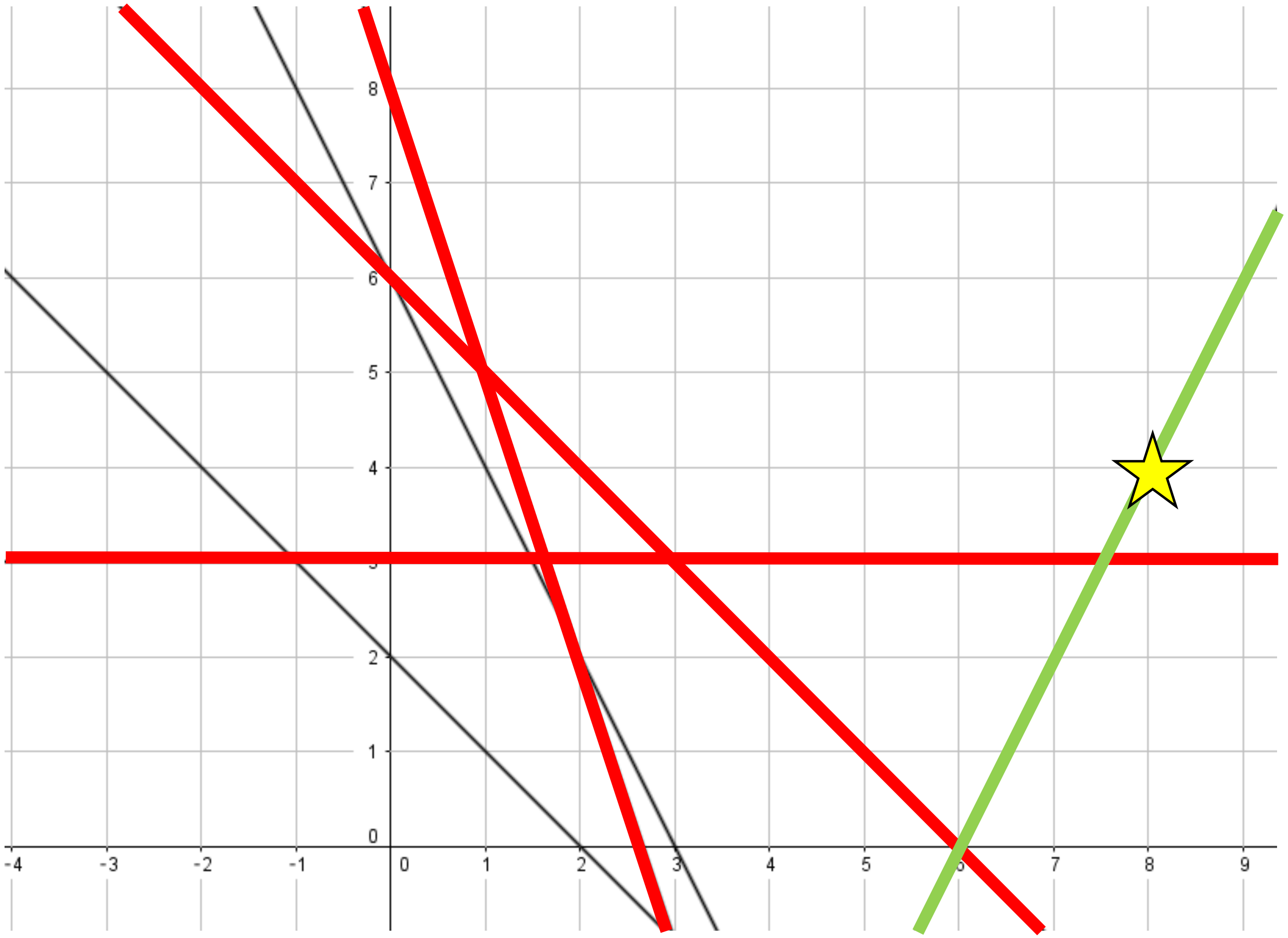
- $t = 5..7,\ r_{5..7} = 3$

Retrieving r_t



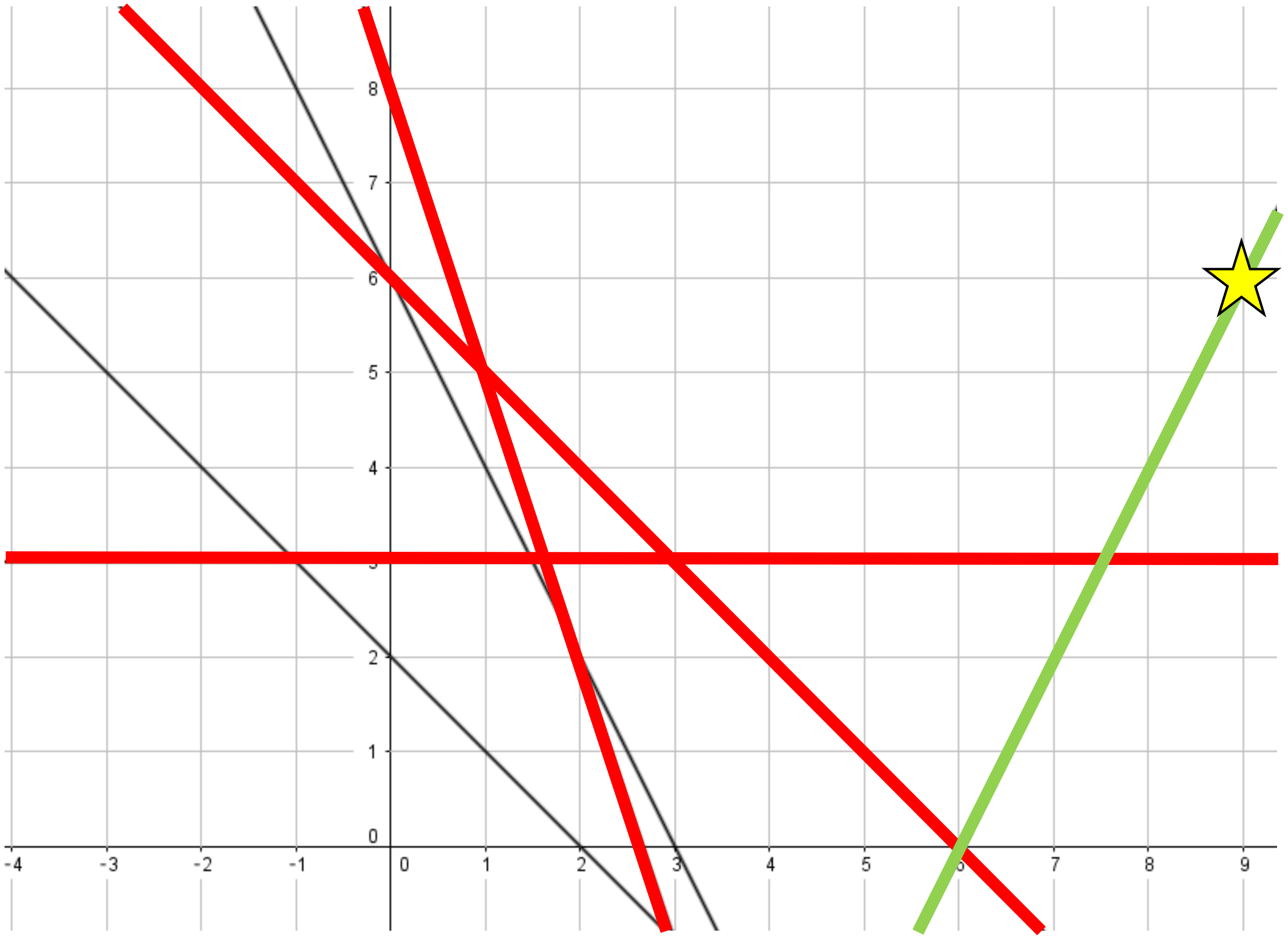
- t = 8, next line better!

Retrieving r_t



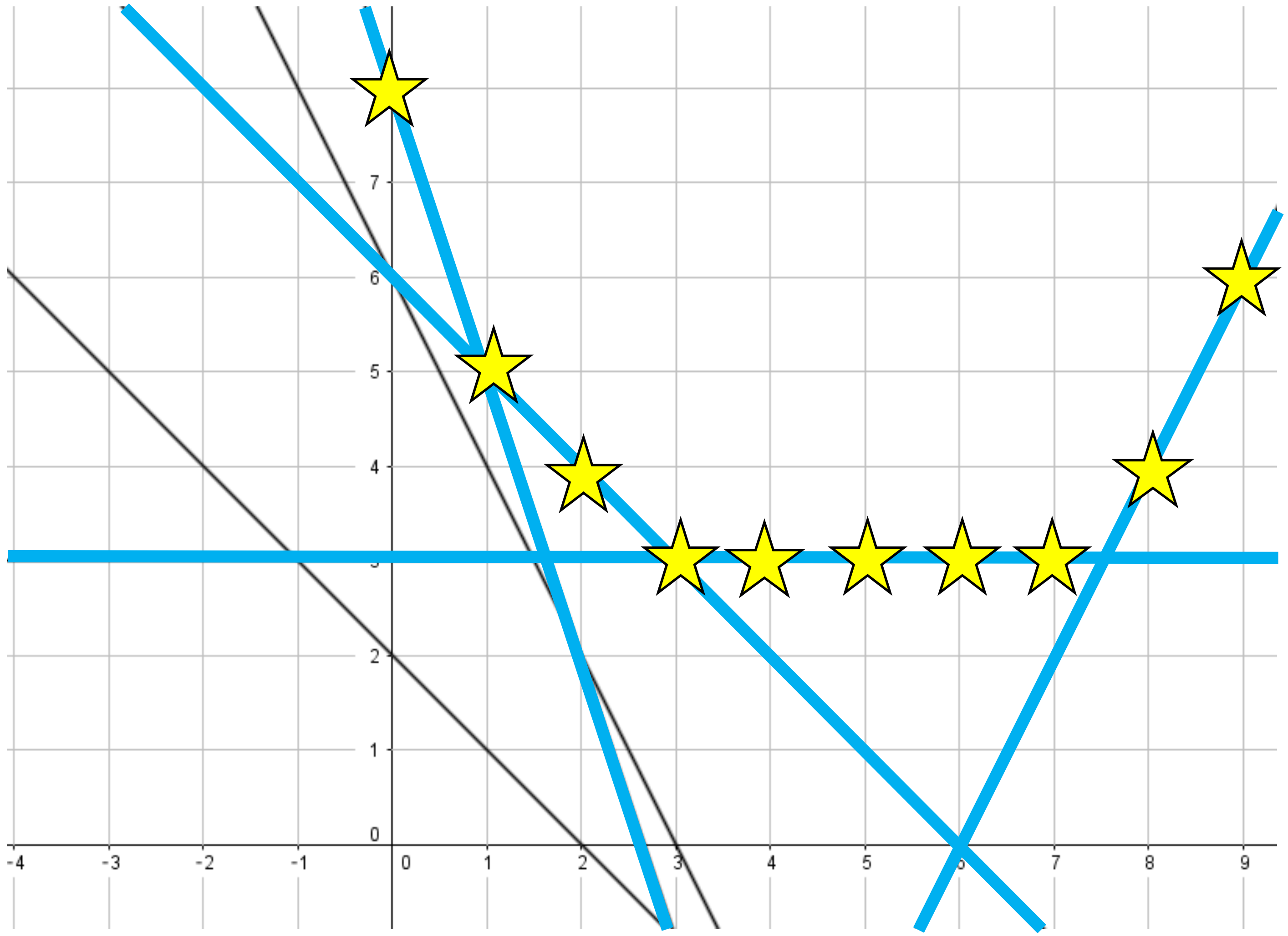
- $t = 8$, $r_8 = 4$

Retrieving r_t



- $t = 9$, $r_9 = 6$

Retrieving r_t



Remarks

- O(N log N + Q log T log N), or other solutions with two or three log terms, do exist
- Depending on implementation, they may get 90 or 100 points (for 90%, Q <= 2000)

- Challenge: What if x_i, y_i, u_i, v_i are <u>not necessarily even</u>? [It is still doable with the same time complexity of O(N log N + T + Q log T)!]

The End