LTH Challenge 2022

Solutions

Jury

- Måns Magnusson
- Maj Stenmark
- Erik Amirell Eklöf
- Thore Husfeldt
- Björn Magnusson
- Jonatan Nilsson

A. Culture Shock

Count the occurrences of the words “he”, “she”, “her” and “him” in the input text.

Having one of the forbidden words as a substring in a word does not count, as in “heard” or even “himself”.

```python
N = int(input())
words = input().split()
cnt = 0
for w in words:
    if w in ['he', 'she', 'him', 'her']:
        cnt += 1
print(cnt)
```

Author: Maj Stenmark

B. Terraforming

Will the environment of Mars be liveable after the environmental changes?

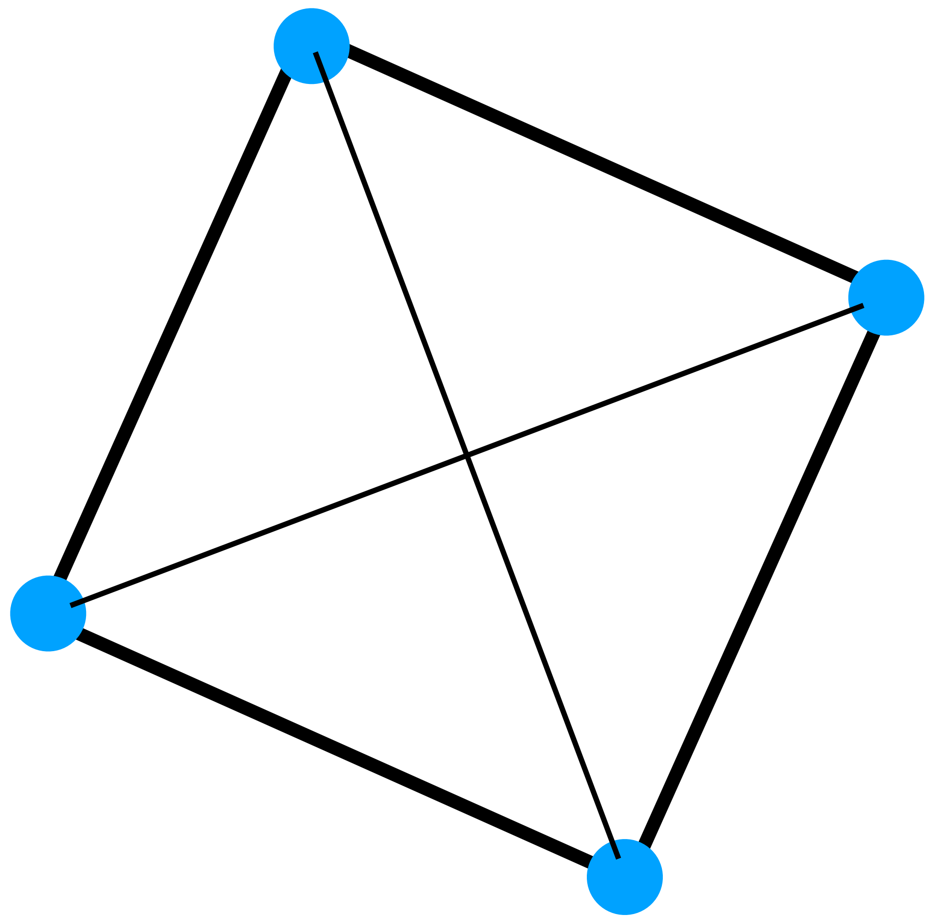
```
state = {"oxygen": 0, "ocean": 0, "temperature": -30}
for _ in range(int(input())):
    variable, change = input().split()
    state[variable] += int(change)

if (state[“oxygen"] >= 14 and
    state["ocean"] >= 9 and
    state["temperature"] >= 8):
    print("liveable")
else:
    print("not liveable")
```

Author: Thore Husfeldt

C. Treehouse

Count the number of squares formed by the points in the input.



Test if 4 points form a square:

- Calculate all 6 pairwise distances.
- The points form a square if the distances are the following:

$$[d, d, d, d, \sqrt{2}d, \sqrt{2}d]$$

20 points if you just test this for all 4-tuples of points in the input.

Author: Måns Magnusson, Maj Stenmark

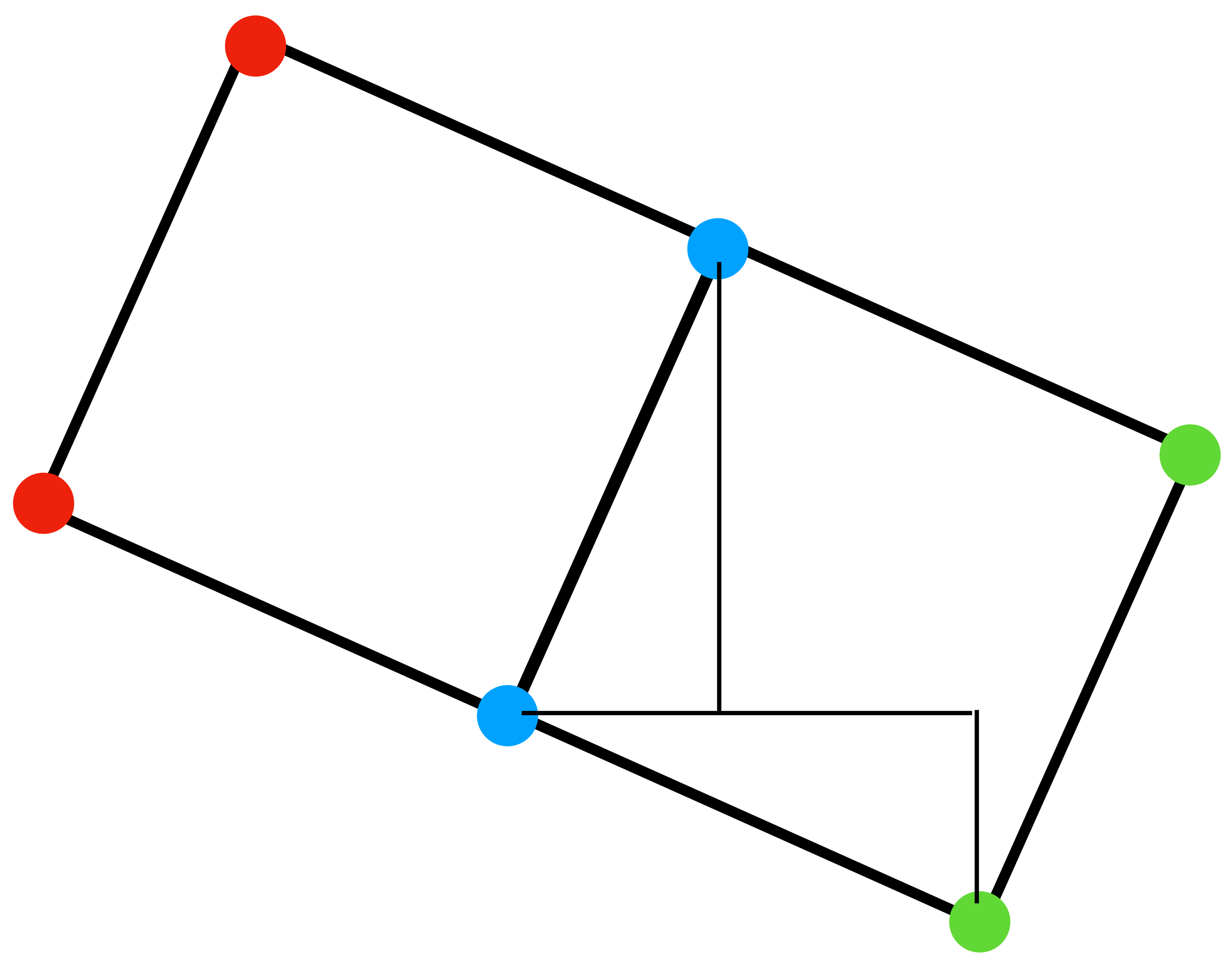
C. Treehouse

Count the number of squares formed by the points in the input.

For 100 points.

For each pair of points test if they form any squares with the remaining points.

- Use a set for quick lookup
- Make sure to not double count any squares



Author: Måns Magnusson, Maj Stenmark

C. Treehouse

Count the number of squares formed by the points in the input.

```python
def findSquares(p1, p2):
    dx = p2[0] - p1[0]
    dy = p2[1] - p1[1]
    #square 1
    p3a = p1[0] - dy, p1[1] + dx
    p4a = p2[0] - dy, p2[1] + dx

    #square 2
    p3b = p1[0] + dy, p1[1] - dx
    p4b = p2[0] + dy, p2[1] - dx
    return (p3a, p4a), (p3b, p4b)
```

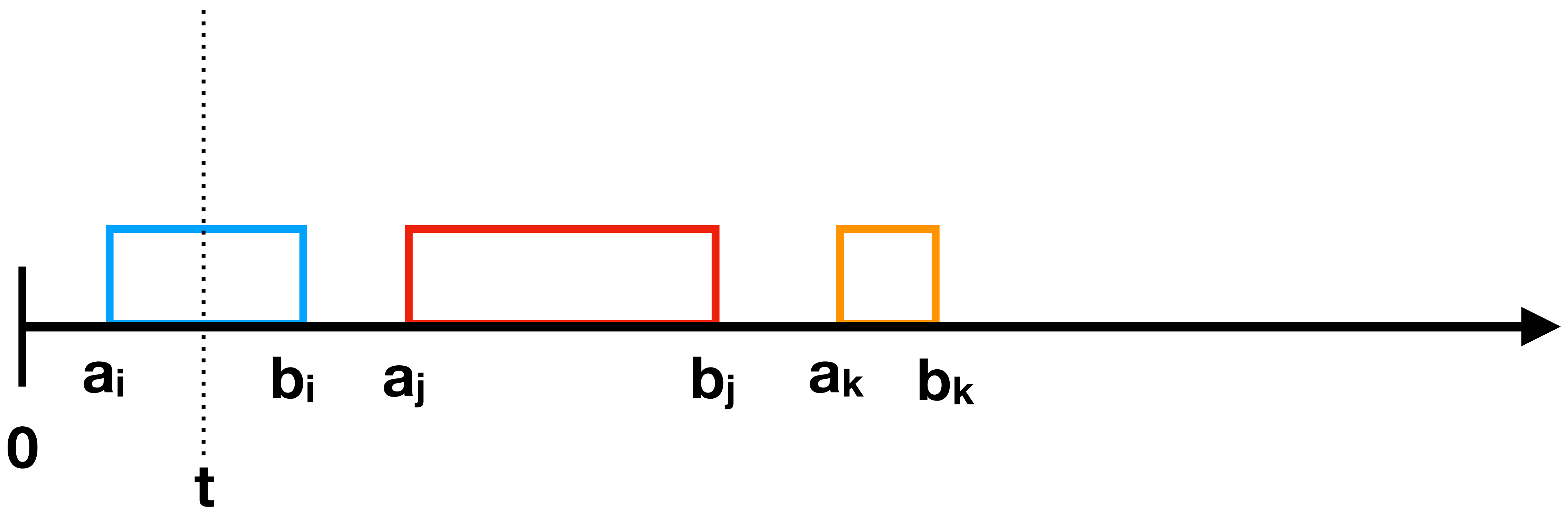
Author: Måns Magnusson, Maj Stenmark

D. Marathon

How fast do Erik have to run to have a 50% chance of winning the race?

For 20 points:

No runner has an overlapping interval.



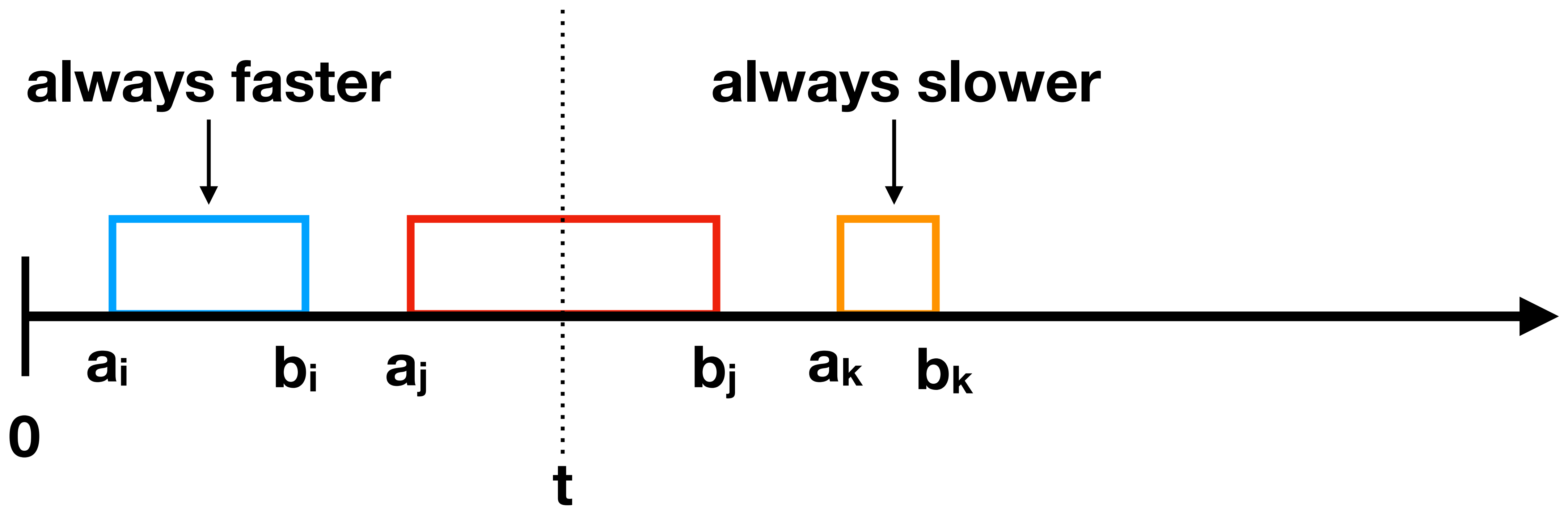
Author: Maj Stenmark

D. Marathon

How fast do Erik have to run to have a 50% chance of winning the race?

For full points:

If an interval is strictly faster than Erik, he always loses. If an interval is strictly slower than Erik, that interval doesn't matter.



Author: Maj Stenmark

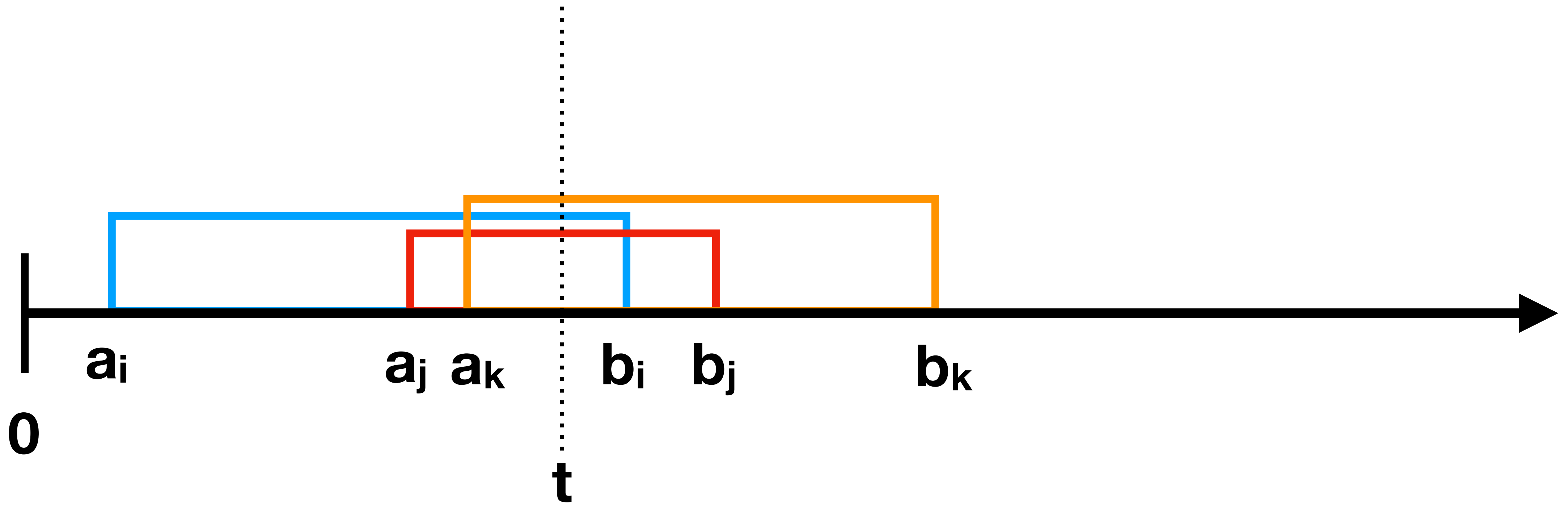
D. Marathon

How fast do Erik have to run to have a 50% chance of winning the race?

Binary search over the answer!

For a given time t, Erik wins with 50% probability if

$$\Pi_i min(1, \frac{b_i - t}{b_i - a_i}) \geq 0.5$$



Given that all $b_i > t$.

Author: Maj Stenmark

E. Bike Party

Find a starting party to remain drunk during your entire night out!

Let A_SUM = sum(alcohol) and D_SUM = sum(distances)

If A_SUM < D_SUM: impossible

If A_SUM == D_SUM: sometimes possible

If A_SUM > D_SUM: always possible

Author: Måns Magnusson, Björn Magnusson, Jonatan Nilsson

E. Bike Party

Find a starting party to remain drunk during your entire night out!

If A_SUM == D_SUM:

Simulate using the first party as starting point:

- If you ever find a party where the alcohol_level becomes negative on arrival, starting at that position instead will increase the alcohol level at all positions.
- Find the most negative position and start from there.
- If the most negative position is shared between many starting points the answer is instead “impossible”, since you will become sober when you reach the other positions.

Author: Måns Magnusson, Björn Magnusson, Jonatan Nilsson

E. Bike Party

Find a starting party to remain drunk during your entire night out!

If A_SUM > D_SUM:

Do the same as for A_SUM == D_SUM, but instead pick the last position which was the most negative, if many are shared.

In this way you will make sure to complete a lap before reaching the other most negative positions, meaning that your alcohol level is A_SUM - D_SUM there.

Author: Måns Magnusson, Björn Magnusson, Jonatan Nilsson

F. Take a break

Minimise the time it takes to complete all tasks.

For 25 points we can test for every number of breaks $X \in \{0, 1, \ldots, N-1\}$ how long time it takes to complete the tasks.

Given X breaks there will be X + 1 tasks with each factor 1, 2, 4, 8, .. until you run out of tasks.

Greedily assign the most difficult tasks to the smallest factors.

Author: Måns Magnusson

F. Take a break

Minimise the time it takes to complete all tasks.

Each break always is an hour = 3600 seconds.

You should never do a task longer than 3600 seconds as a second task.

Similarly, you should never do any task as a 13th task, since 2^12 = 4096:

$$\forall t \geq 1 : t \cdot 4096 > t + 3600$$

Author: Måns Magnusson

F. Take a break

Minimise the time it takes to complete all tasks.

Never do more than 12 tasks in a row:

The number of breaks is always at least $\frac{N}{12}$.

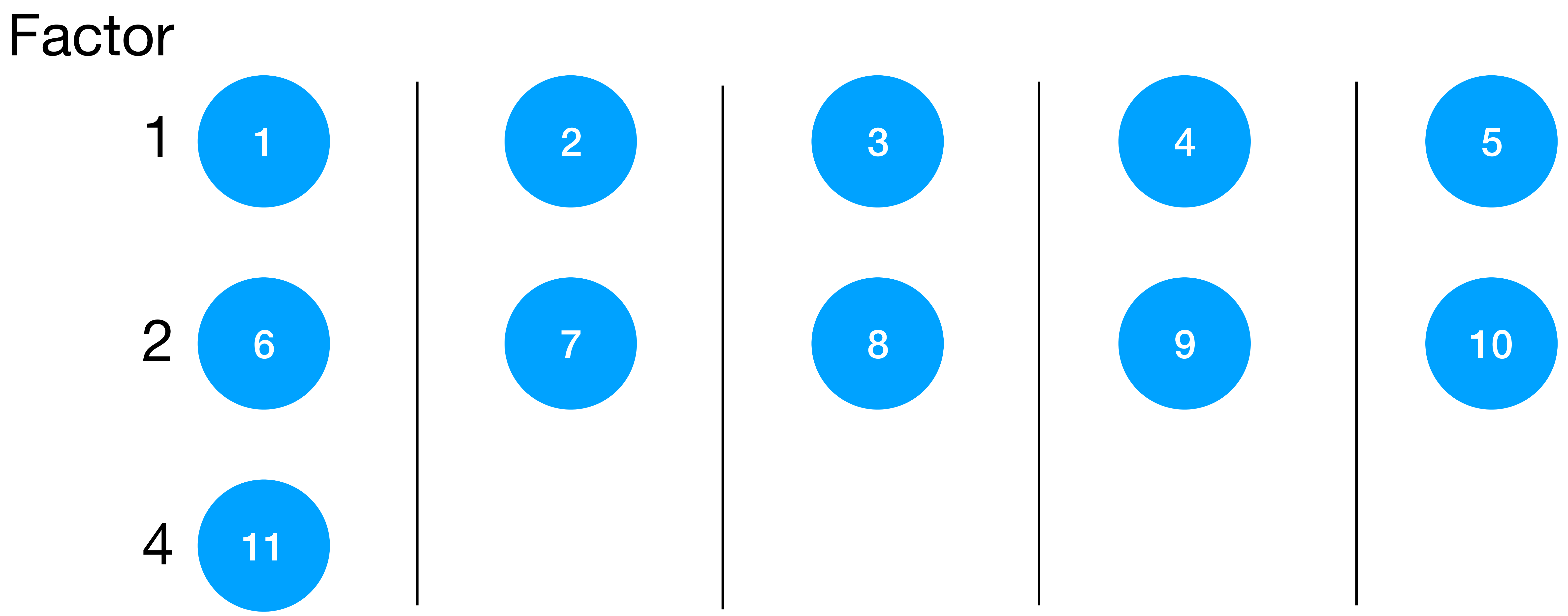
Try for all number of breaks from N-1 to $\frac{N}{12}$.

Keep track of 12 queues, one for the starting tasks, one for the tasks done as second tasks, one for the tasks done as the third tasks, etc.

Author: Måns Magnusson

F. Take a break

Minimise the time it takes to complete all tasks.

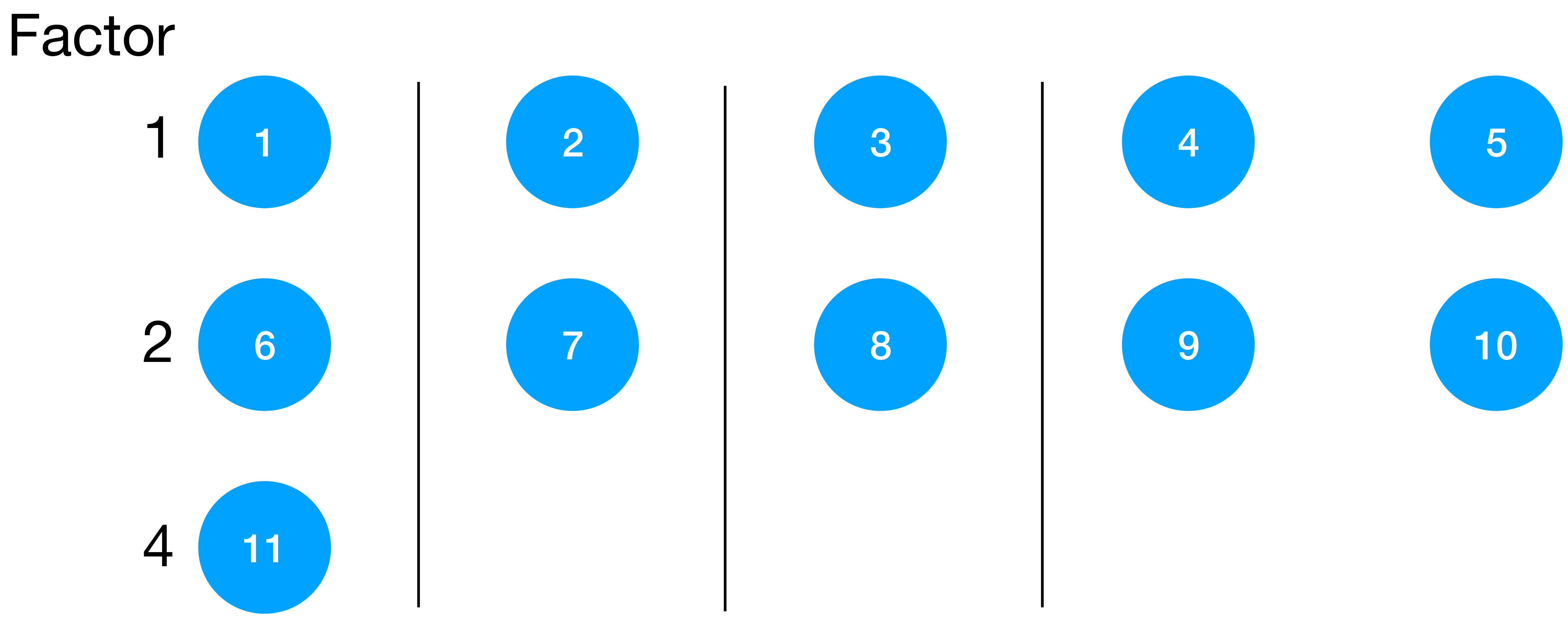


4 breaks, 9 tasks

Author: Måns Magnusson

F. Take a break

Minimise the time it takes to complete all tasks.

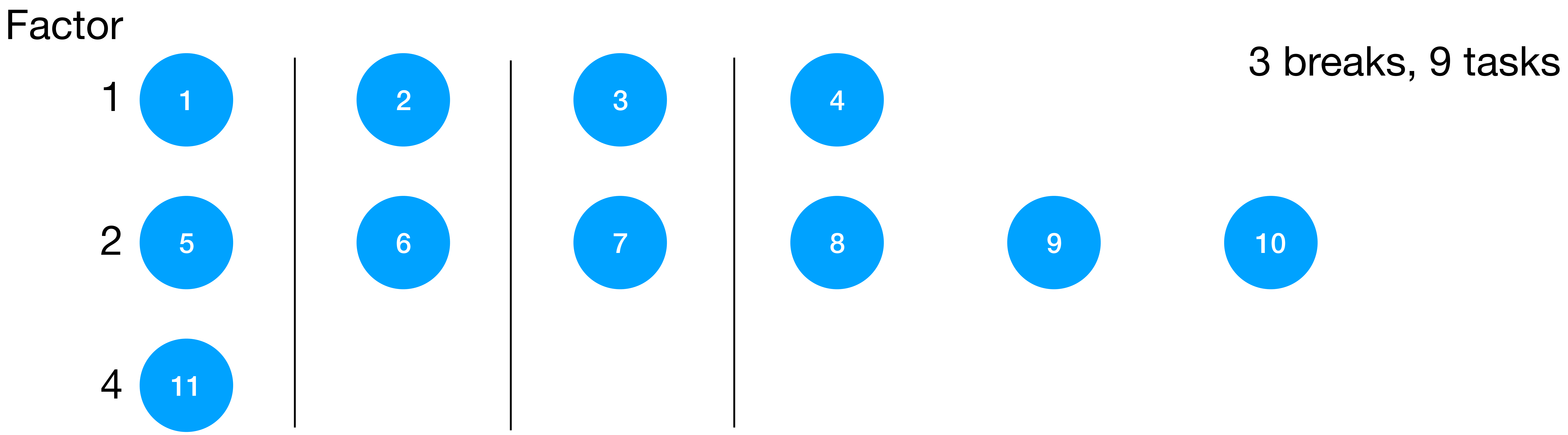


3 breaks, 9 tasks

Author: Måns Magnusson

F. Take a break

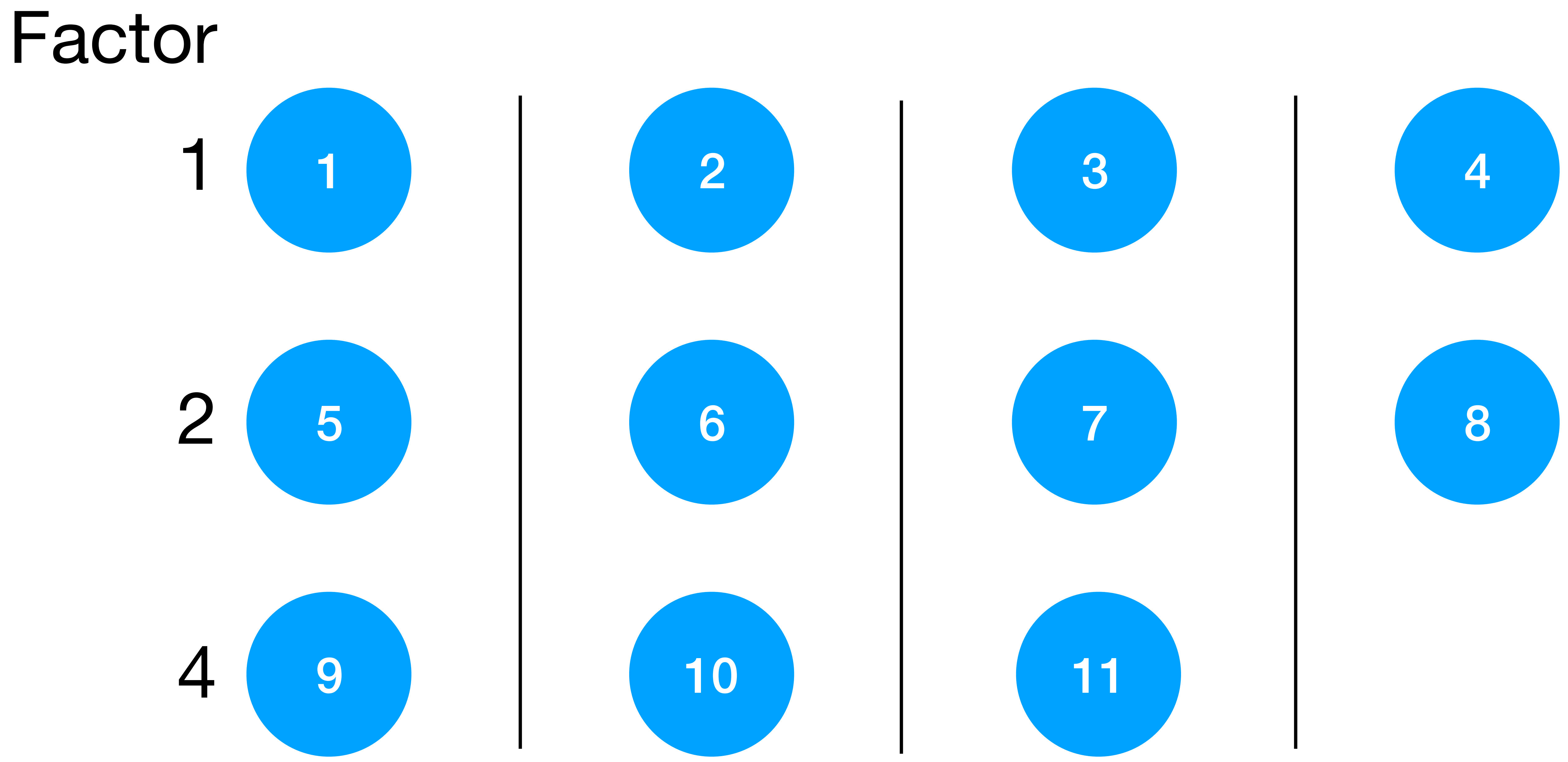
Minimise the time it takes to complete all tasks.



Author: Måns Magnusson

F. Take a break

Minimise the time it takes to complete all tasks.



3 breaks, 9 tasks

Author: Måns Magnusson

F. Take a break

Minimise the time it takes to complete all tasks.

Removing a break moves means moving at most:
1 task from queue 1 to queue 2,
2 tasks from queue 2 to queue 3,
3 tasks from queue 3 to queue 4,
…
11 tasks from queue 11 to queue 12.
= 66 moves at most.

Giving us a time complexity of 66*N = O(N).

Author: Måns Magnusson

G. Fireworks

Maximise the number of Awesome Combinations (R&G fireworks at the same time)

Group 1 (no Xs):

- Test every ignition point
- Count the number of red-green matches in linear time

Group 2

- Test every ignition point
- Count the number R-G, R-X, G-X & X-X matches

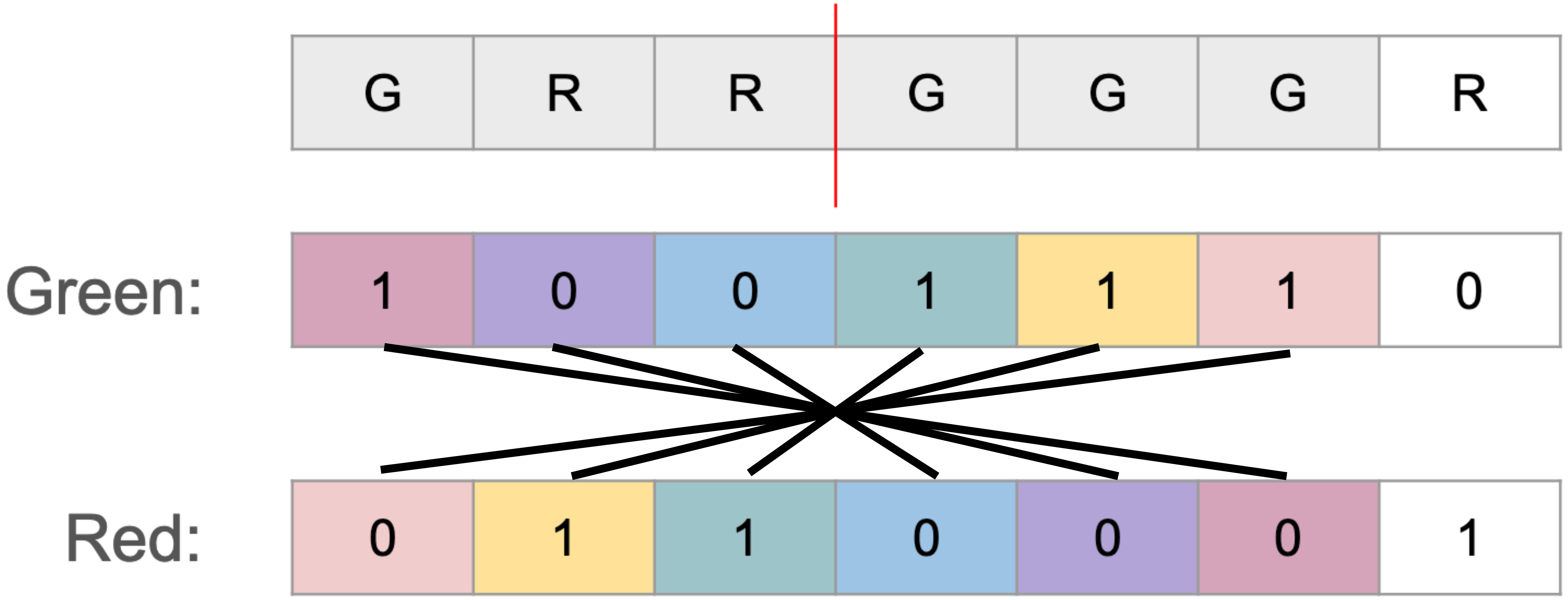
numRG + min(numRX,G) + min(numGX,R) + min(numXX, G-min(numRX,G), R-min(numGX,R))

Author: Erik Amirell Eklöf

G. Fireworks

Maximise the number of Awesome Combinations (R&G fireworks at the same time)

Group 3 (no Xs, but too slow for quadratic)



Number of RG matches is a sum of cross multiplications

Can be computed for every ignition point with FFT

Author: Erik Amirell Eklöf

G. Fireworks

Maximise the number of Awesome Combinations (R&G fireworks at the same time)

Group 4

- Do 4 FFTs to count R-G, R-X, G-X & X-X matches
- Combine in a similar way to before:
  - numRG + min(numRX,G) + min(numGX,R) + min(numXX, G-min(numRX,G), R-min(numGX,R))
- Note that: FFT will double count X-X matches

Author: Erik Amirell Eklöf

H. Sjön Sjön Cleanup

Minimise the number of cells to pour cleaning solution into to clean the entirety of Sjön Sjön.

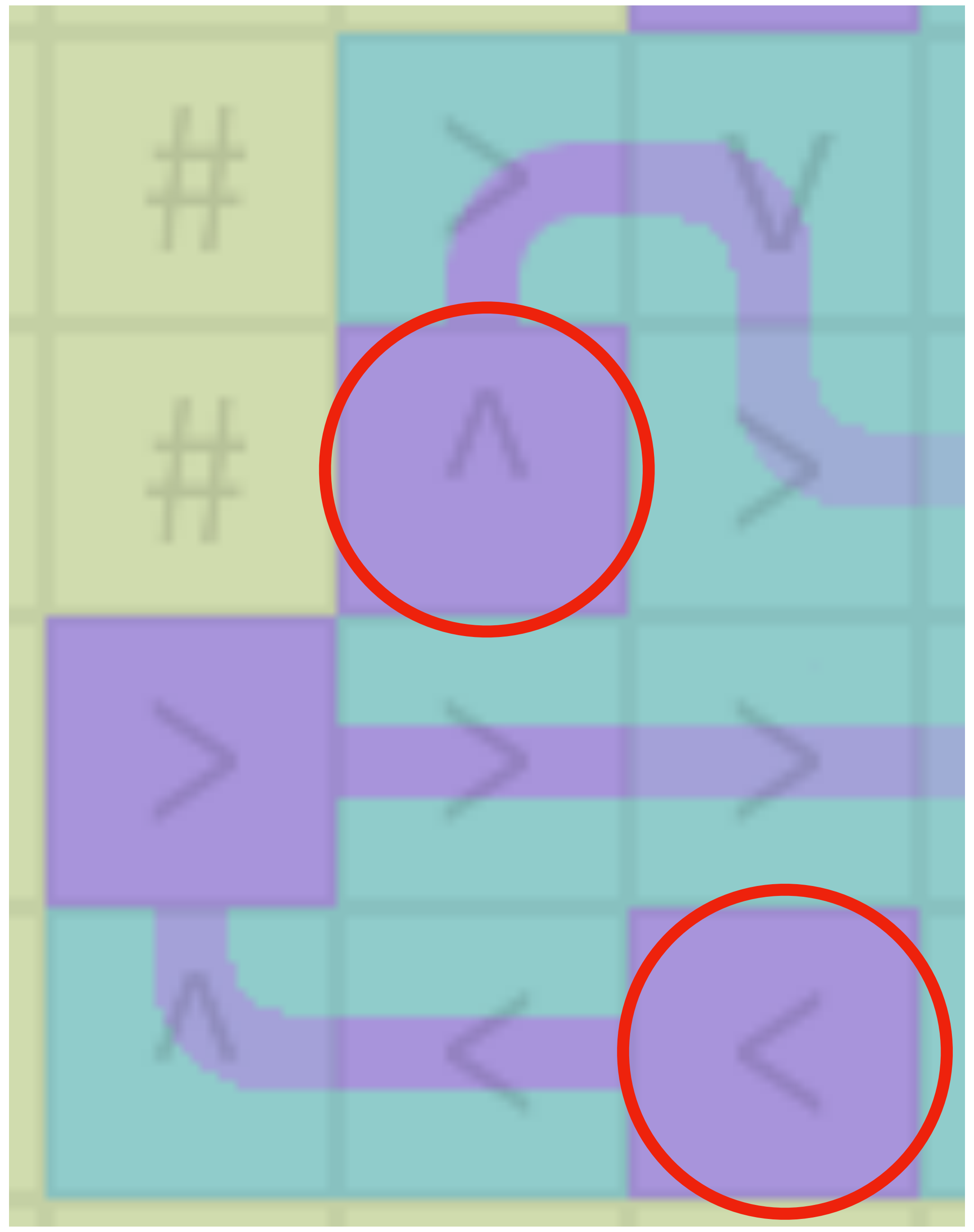
We are forced to pour cleaning solution into some cells (those that have no neighbours point into them)

In test group 2 ($S=10^9$):

1. Pour cleaning solution into cells that are forced.
   If a solution exists, this will cover all cells except cycles.

2. As long as there is an edge cell left uncovered,
   pour solution into that cell and mark all following cells as covered.

3. If there are non-edge cells left uncovered, answer impossible.



Author: Erik Amirell Eklöf

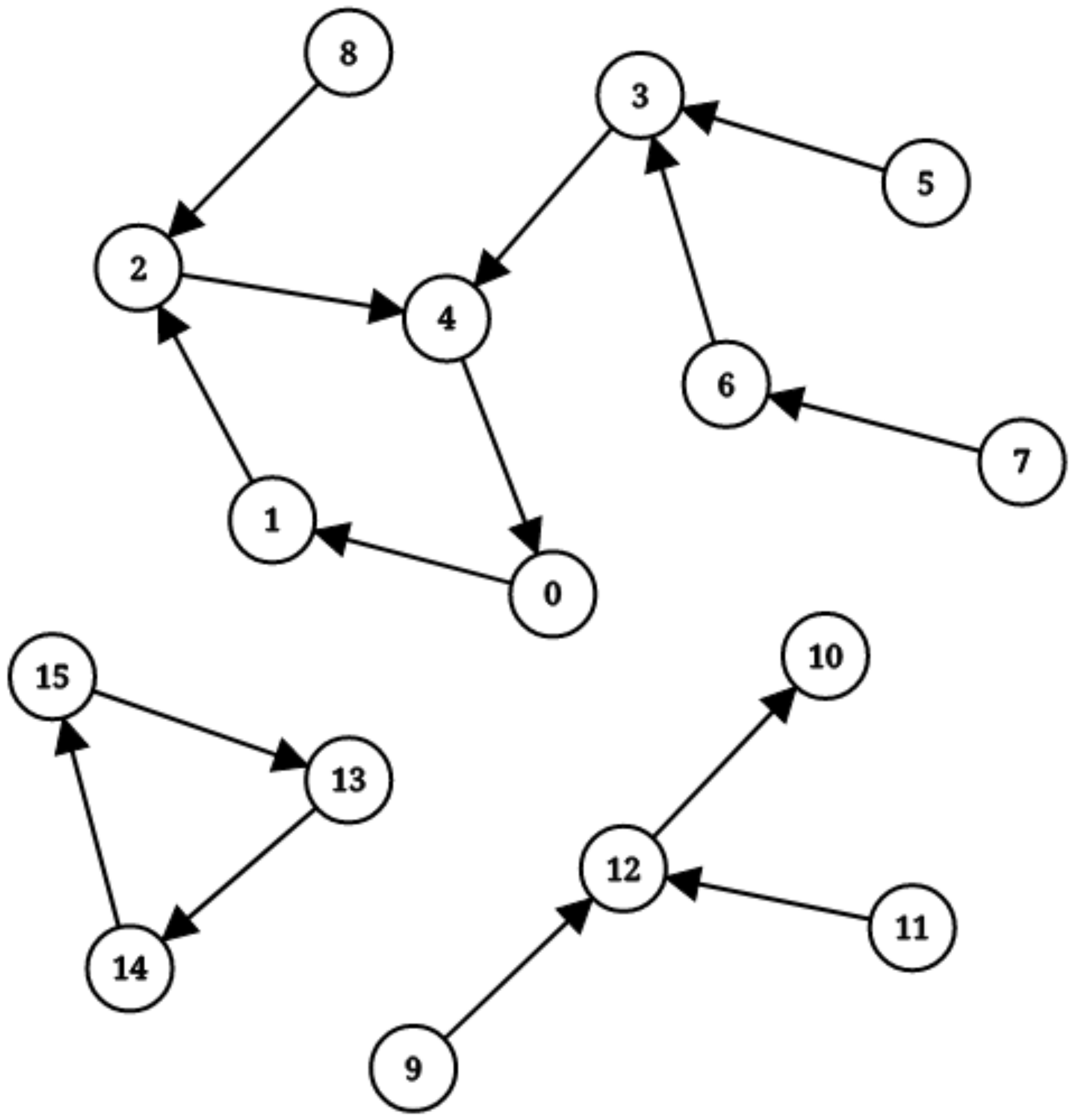
H. Sjön Sjön Cleanup

Minimise the number of cells to pour cleaning solution into to clean the entirety of Sjön Sjön.

The lake is a directed graph with each node having out degree at most 1.

This means that each component is either:

- A tree
- A cycle
- A cycle with trees pointing into it



Author: Erik Amirell Eklöf

H. Sjön Sjön Cleanup

Minimise the number of cells to pour cleaning solution into to clean the entirety of Sjön Sjön.

Trees can be handled with a greedy algorithm

- Pour cleaning solution into leaf nodes.
- This creates new leaf nodes, which also need cleaning solution.
- The new leaf nodes might not be at the edge of the lake.
- Pour cleaning solution into the closest edge-node behind

Author: Erik Amirell Eklöf

H. Sjön Sjön Cleanup

Minimise the number of cells to pour cleaning solution into to clean the entirety of Sjön Sjön.

For the last two subtasks, we need to handle cycles too

Quadratic solution: Test every start point and greedily fill the rest of the cycle.

Linear solution:

- Number the nodes in the cycle with an arbitrary node as node 0
- Compute dp[i] as a pair of two values:
  - Minimum number of cells to pour solution into for the nodes i..N, given that nodes 0..i-1 are already covered.
  - The amount of cleaning solution strength left over after node N.
- Test every starting node, node i is an OK start node if dp[i].second says that every node in the interval 0..i-1 is covered

Author: Erik Amirell Eklöf