OIS11 – Final Round

Analysis Session

Olimpiadi di Informatica a Squadre

February 21, 2020

Statistics

- 967 submissions

Statistics

- 967 submissions
- 61 in C, 900 in C++ and 6 in Pascal

Statistics

- 967 submissions
- 61 in C, 900 in C++ and 6 in Pascal
- 1,7MB of source code

Statistics

- 967 submissions
- 61 in C, 900 in C++ and 6 in Pascal
- 1,7MB of source code
- 17176 open parentheses, 17134 close parentheses ($\Delta = 42$)

Statistics

- 967 submissions
- 61 in C, 900 in C++ and 6 in Pascal
- 1,7MB of source code
- 17176 open parentheses, 17134 close parentheses ($\Delta = 42$)
- 8490 open brackets, 8436 close brackets ($\Delta = 54$)

Statistics

- 967 submissions
- 61 in C, 900 in C++ and 6 in Pascal
- 1,7MB of source code
- 17176 open parentheses, 17134 close parentheses ($\Delta = 42$)
- 8490 open brackets, 8436 close brackets ($\Delta = 54$)
- Keywords:

Statistics

- 967 submissions
- 61 in C, 900 in C++ and 6 in Pascal
- 1,7MB of source code
- 17176 open parentheses, 17134 close parentheses ($\Delta = 42$)
- 8490 open brackets, 8436 close brackets ($\Delta = 54$)
- Keywords:

  int 8839

Statistics

- 967 submissions
- 61 in C, 900 in C++ and 6 in Pascal
- 1,7MB of source code
- 17176 open parentheses, 17134 close parentheses ($\Delta = 42$)
- 8490 open brackets, 8436 close brackets ($\Delta = 54$)
- Keywords:

int	8839
float	1

Statistics

- 967 submissions
- 61 in C, 900 in C++ and 6 in Pascal
- 1,7MB of source code
- 17176 open parentheses, 17134 close parentheses ($\Delta = 42$)
- 8490 open brackets, 8436 close brackets ($\Delta = 54$)
- Keywords:

int	8839
float	1
short	64

Statistics

- 967 submissions
- 61 in C, 900 in C++ and 6 in Pascal
- 1,7MB of source code
- 17176 open parentheses, 17134 close parentheses ($\Delta = 42$)
- 8490 open brackets, 8436 close brackets ($\Delta = 54$)
- Keywords:

int	8839
float	1
short	64
if	4173

Statistics

- 967 submissions
- 61 in C, 900 in C++ and 6 in Pascal
- 1,7MB of source code
- 17176 open parentheses, 17134 close parentheses ($\Delta = 42$)
- 8490 open brackets, 8436 close brackets ($\Delta = 54$)
- Keywords:

int	8839
float	1
short	64
if	4173
bug	8

Statistics

- 967 submissions
- 61 in C, 900 in C++ and 6 in Pascal
- 1,7MB of source code
- 17176 open parentheses, 17134 close parentheses ($\Delta = 42$)
- 8490 open brackets, 8436 close brackets ($\Delta = 54$)
- Keywords:

int	8839
float	1
short	64
if	4173
bug	8
todo	18

Statistics

- 967 submissions
- 61 in C, 900 in C++ and 6 in Pascal
- 1,7MB of source code
- 17176 open parentheses, 17134 close parentheses ($\Delta = 42$)
- 8490 open brackets, 8436 close brackets ($\Delta = 54$)
- Keywords:

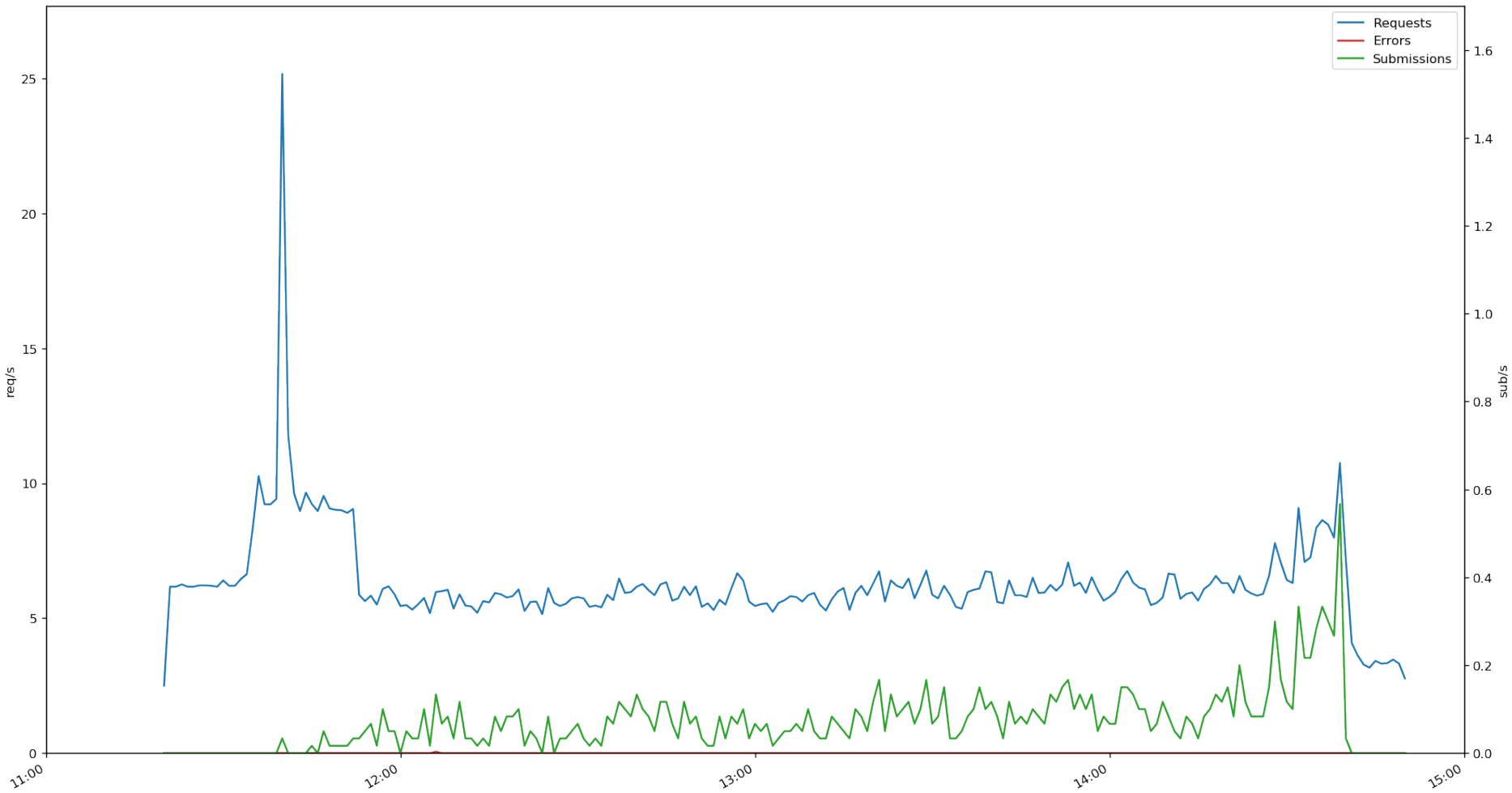
int	8839
float	1
short	64
if	4173
bug	8
todo	18
ois	0

Statistics

- 967 submissions
- 61 in C, 900 in C++ and 6 in Pascal
- 1,7MB of source code
- 17176 open parentheses, 17134 close parentheses ($\Delta = 42$)
- 8490 open brackets, 8436 close brackets ($\Delta = 54$)
- Keywords:

int	8839
float	1
short	64
if	4173
bug	8
todo	18
ois	0
OIS2020	52

Statistics


Requests
Errors
Submissions
req/s
sub/s
0
5
10
15
20
25
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
11:00
12:00
13:00
14:00
15:00

Solutions

Pay That Box Again! (gameshow2)

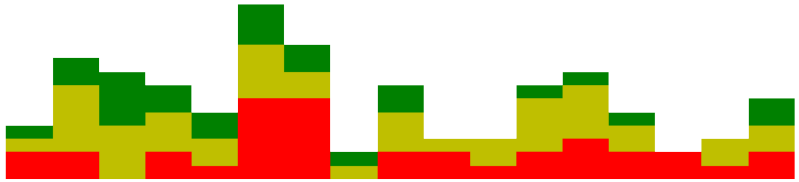


Figure: Submissions of gameshow2

- Author: Giorgio Audrito
- Submissions: 91
- Score average: 83.55

Pay That Box Again! (`gameshow2`)

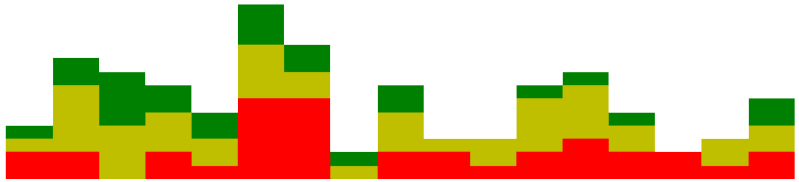


Figure: Submissions of `gameshow2`

- Author: Giorgio Audrito
- Submissions: 91
- Score average: 83.55
- First points: `Banani in fiore` (30/100, 18min)

Pay That Box Again! (gameshow2)

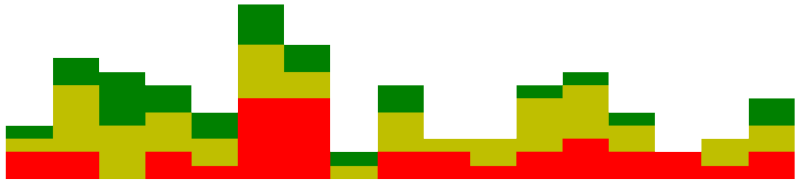
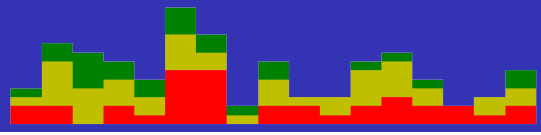


Figure: Submissions of gameshow2

- Author: Giorgio Audrito
- Submissions: 91
- Score average: 83.55
- First points: `Banani in fiore` (30/100, 18min)
- First solution: `GALILEI_01` (18min)

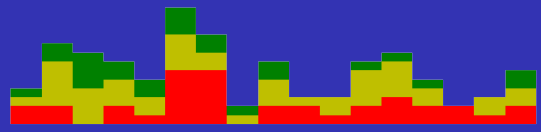
Pay That Box Again! (gameshow2)



Solution

Knowing the compensation C, the price P and the value V of a box you can tell *immediately* whether it is better to buy it or not ($V - P \leqslant C$).

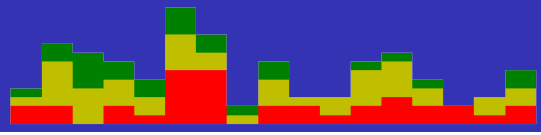
Pay That Box Again! (gameshow2)



Solution

Knowing the compensation C, the price P and the value V of a box you can tell *immediately* whether it is better to buy it or not ($V - P \leqslant C$). Looping over all boxes, you can just accumulate the gains (either $V - P$ or C) and the budget required for obtaining them (respectively P or $-C$).

Pay That Box Again! (gameshow2)



Solution

Knowing the compensation C, the price P and the value V of a box you can tell *immediately* whether it is better to buy it or not ($V - P \leqslant C$). Looping over all boxes, you can just accumulate the gains (either $V - P$ or C) and the budget required for obtaining them (respectively P or $-C$).

Complexity

Just a linear pass over all boxes: $O(N)$.

Graduation Card (`text`)

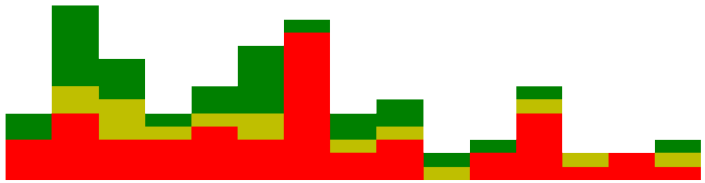


Figure: Submissions of `text`

- Author: Marco Donadoni
- Submissions: 91
- Score average: 91.29

Graduation Card (`text`)

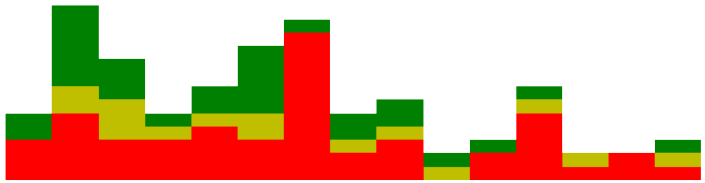


Figure: Submissions of `text`

- Author: Marco Donadoni
- Submissions: 91
- Score average: 91.29
- First points: `GALILEI_01` (100/100, 5min)

Graduation Card (`text`)

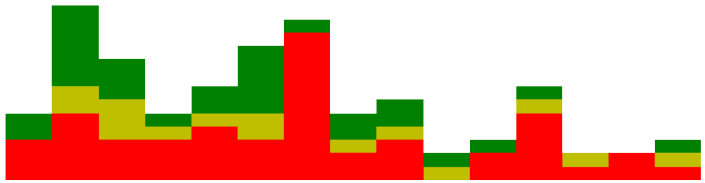
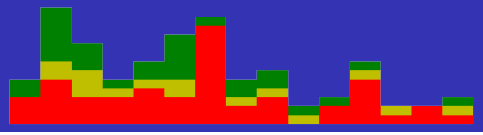


Figure: Submissions of `text`

- Author: Marco Donadoni
- Submissions: 91
- Score average: 91.29
- First points: `GALILEI_01` (100/100, 5min)
- First solution: `GALILEI_01` (5min)

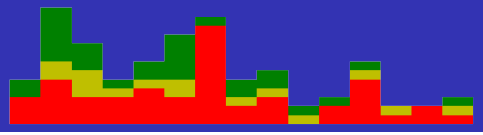
Graduation Card (text)



Solution

The solution of the problem just simulates the process described in the problem statement. A variable, let's say SL, is used to keep the space left in the current line. If the word to be printed fits in the current line (i.e. $|W_i| \leq SL$), it is printed and $SL = SL - |W_i| - 1$; otherwise the word is printed in a new line and $SL = K - |W_i| - 1$. The -1 accounts for the presence of a space between consecutive words.

Graduation Card (text)



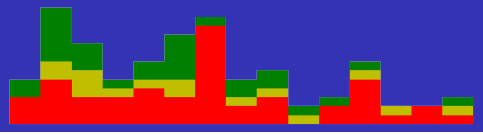
Solution

The solution of the problem just simulates the process described in the problem statement. A variable, let's say SL, is used to keep the space left in the current line. If the word to be printed fits in the current line (i.e. $|W_i| \leq SL$), it is printed and $SL = SL - |W_i| - 1$; otherwise the word is printed in a new line and $SL = K - |W_i| - 1$. The -1 accounts for the presence of a space between consecutive words.

Complexity

This solution has linear complexity and gets full score.

Graduation Card (text)



Example

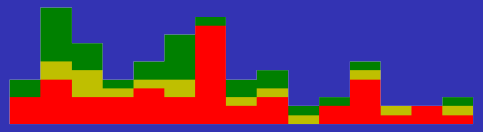
$N = 5$, $K = 8$

`this is a sample message`

1	2	3	4	5	6	7	8
t	h	i	s		.	.	.

$$SL = 3$$

Graduation Card (text)



Example

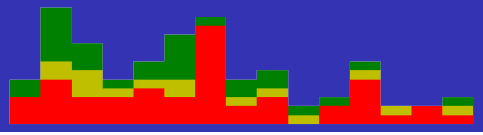
$N = 5$, $K = 8$

`this is a sample message`

1	2	3	4	5	6	7	8
t	h	i	s		i	s	

$SL = 0$

Graduation Card (text)



Example

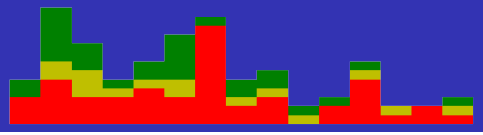
$N = 5$, $K = 8$

`this is a sample message`

1	2	3	4	5	6	7	8
t	h	i	s		i	s	
a		.	.	.	.	.	.

$$SL = 6$$

Graduation Card (text)



Example

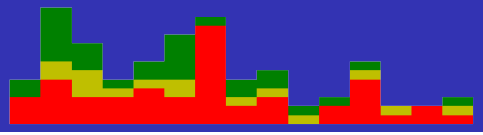
$N = 5$, $K = 8$

`this is a sample message`

1	2	3	4	5	6	7	8
t	h	i	s		i	s	
a		s	a	m	p	l	e

$$SL = -1$$

Graduation Card (text)



Example

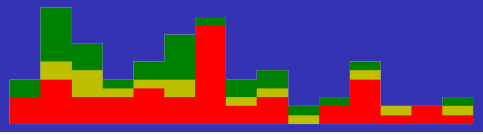
$N = 5$, $K = 8$

`this is a sample message`

1	2	3	4	5	6	7	8
t	h	i	s		i	s	
a		s	a	m	p	l	e
m	e	s	s	a	g	e	

$$SL = 0$$

Graduation Card (text)



Possible common errors

These are possible errors:

- Not counting spaces between words.
- Counting too many spaces in a line (if there are P words in a line, spaces are only $P - 1$).

Splitting the Bill (`money`)

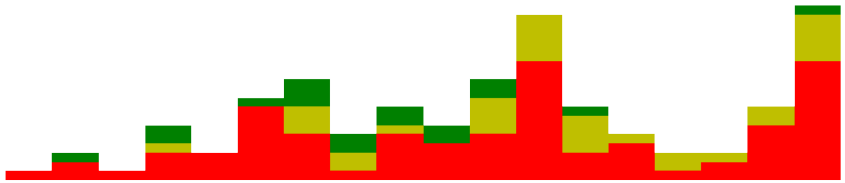


Figure: Submissions of `money`

- Author: Marco Donadoni
- Submissions: 98
- Score average: 55.97

Splitting the Bill (`money`)

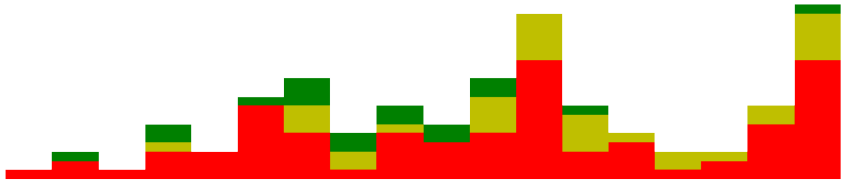


Figure: Submissions of `money`

- Author: Marco Donadoni
- Submissions: 98
- Score average: 55.97
- First points: `BToiS` (100/100, 17min)

Splitting the Bill (`money`)

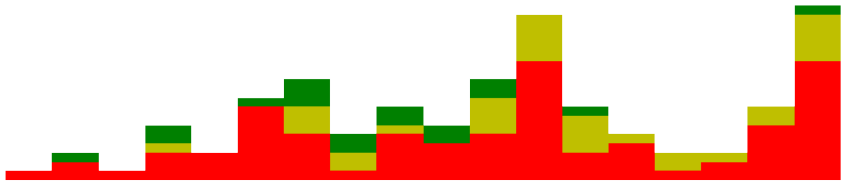
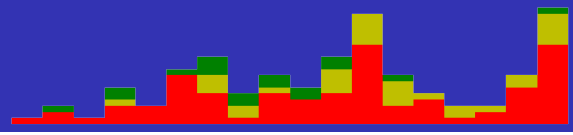


Figure: Submissions of `money`

- Author: Marco Donadoni
- Submissions: 98
- Score average: 55.97
- First points: `BToiS` (100/100, 17min)
- First solution: `BToiS` (17min)

Splitting the Bill (`money`)



First Observation

We only care about the *balance* of the various friends, not about the actual lendings.

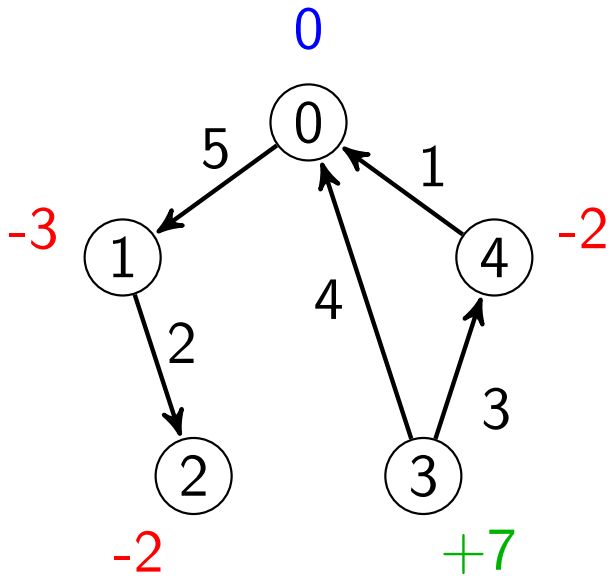
Splitting the Bill (`money`)

First Observation

We only care about the *balance* of the various friends, not about the actual lendings.

Balance

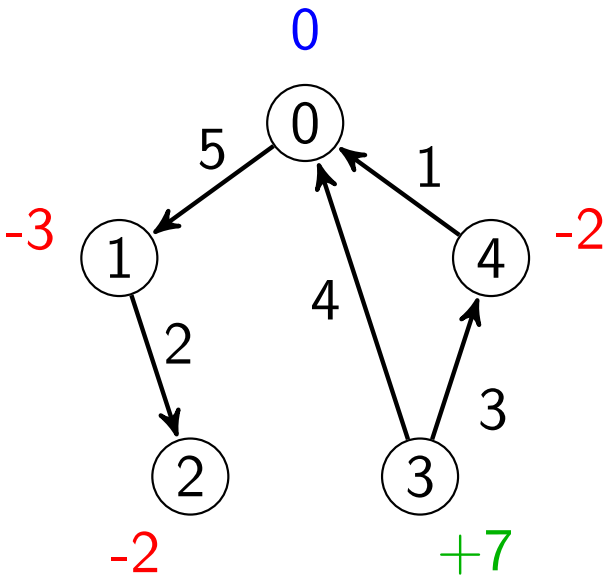
The balance is the (signed) sum of all the money lent or borrowed. Note that the sum of all balances is zero.

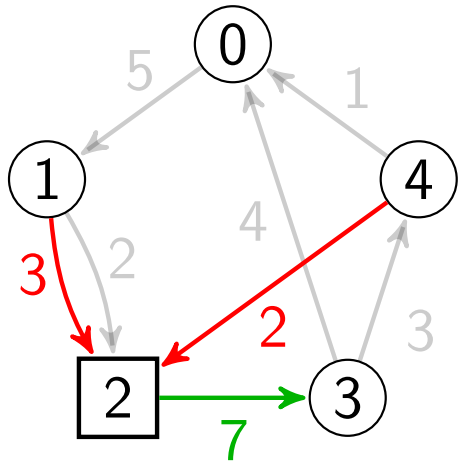


Splitting the Bill (money)

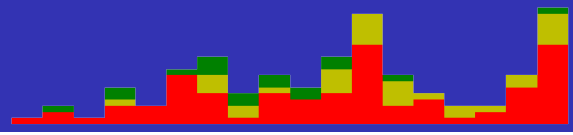
First (easy) Solution

To make things even, we must make all balances equal to zero. To do so, we select one friend to be used as a "bank". All other friends will make their balance zero by making/receiving one money transfer to/from the "bank", with value equal to their balance.





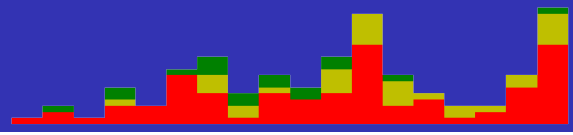
Splitting the Bill (money)



First (easy) Solution - Why it works?

It is easy to see that at most $N - 1$ money transfers are made, one for each friend that is not the "bank". Furthermore every non-"bank" friend will have balance zero (because of how we make the money transfers). Even the "bank" will have balance zero, since we know that the total sum of the balances is zero, and all other friends have zero balance.

Splitting the Bill (money)



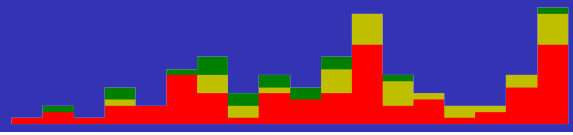
First (easy) Solution - Why it works?

It is easy to see that at most $N - 1$ money transfers are made, one for each friend that is not the "bank". Furthermore every non-"bank" friend will have balance zero (because of how we make the money transfers). Even the "bank" will have balance zero, since we know that the total sum of the balances is zero, and all other friends have zero balance.

Complexity

The computation of the balances can be done in $O(M)$. Finding the money transfers to be made can be done in $O(N)$. The total complexity is therefore $O(M + N)$.

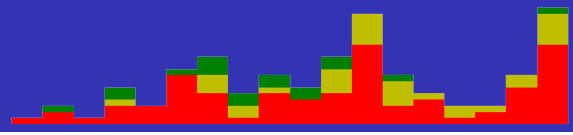
Splitting the Bill (`money`)



Second Observation

If there are multiple lendings between the same pair of friends, they can be considered as one single lending, with value equal to the (signed) sum of the values of the lendings.

Splitting the Bill (money)



Second Observation

If there are multiple lendings between the same pair of friends, they can be considered as one single lending, with value equal to the (signed) sum of the values of the lendings.

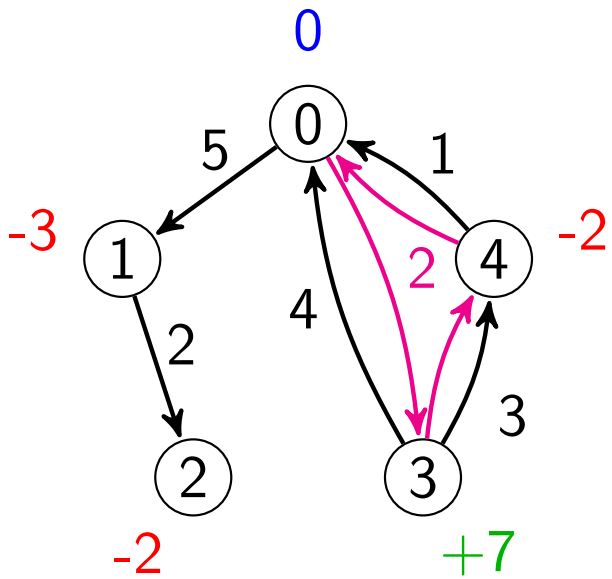
Preprocessing

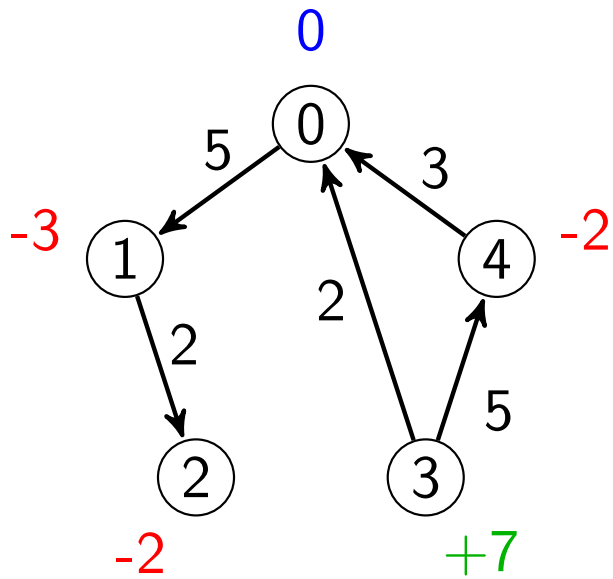
Let's consider the graph of all lendings; using the previous observation we can remove all the multiple arcs present in the graph, so that for each pair of friends we have at most one lending.

Splitting the Bill (money)

Third Observation

Let's consider a cycle of lendings, ignoring the arcs' orientation. If we add an arc for every consecutive pair of friends in the cycle, all of them with the same value and orientation, we get a new lendings graph equivalent to the original one (all friends have the same balances as before).

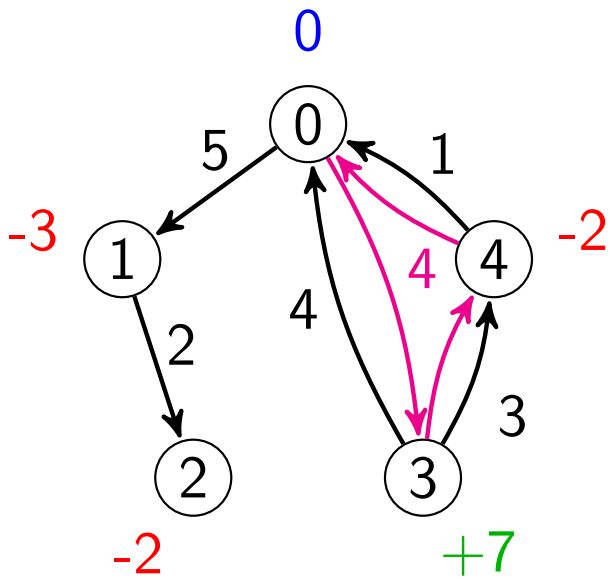


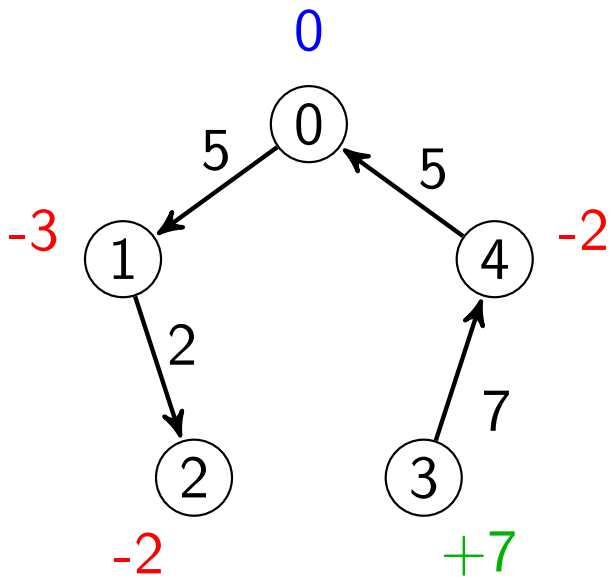


Splitting the Bill (money)

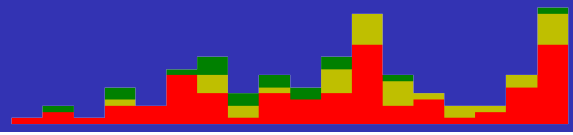
Third Observation (continued)

The value and the orientation of the arcs can be cleverly chosen in order to delete one arc of the cycle, while still having a problem equivalent to the original one.





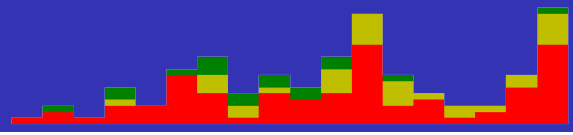
Splitting the Bill (money)



Second (not so easy) Solution

If there is a cycle in the graph, we iteratively apply the third observation to delete one arc of the cycle, until there are no more cycles. This means that, at most, we are left with $N - 1$ arcs. The solution is composed of the remaining arcs, with reverse orientation.

Splitting the Bill (money)



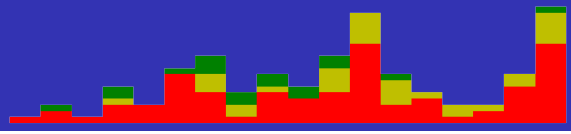
Second (not so easy) Solution

If there is a cycle in the graph, we iteratively apply the third observation to delete one arc of the cycle, until there are no more cycles. This means that, at most, we are left with $N - 1$ arcs. The solution is composed of the remaining arcs, with reverse orientation.

Naive Implementation

The naive implementation needs to make a graph visit for every arc to be removed. The arcs to be removed are $O(M)$ at worst, the cost of the visit is $O(M + N)$ or $O(N^2)$, based on how the graph is stored in memory. The overall complexity is $O(M(M + N))$ or $O(MN^2)$.

Splitting the Bill (money)



Second (not so easy) Solution

If there is a cycle in the graph, we iteratively apply the third observation to delete one arc of the cycle, until there are no more cycles. This means that, at most, we are left with $N - 1$ arcs. The solution is composed of the remaining arcs, with reverse orientation.

Optimal Implementation

The naive solution can be improved by searching (and fixing) all the cycles in only one visit of the graph. If the graph is stored as an adjacency list, the complexity is $O(M + N)$.

Flood Forecasting (`rainstorm`)

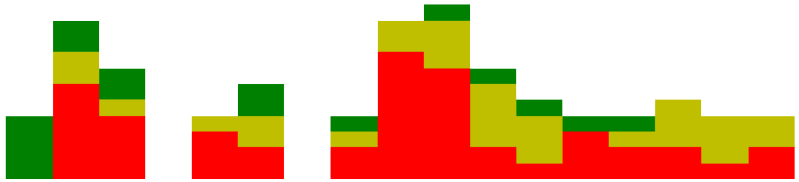


Figure: Submissions of `rainstorm`

- Author: Marco Donadoni
- Submissions: 76
- Score average: 67.74

Flood Forecasting (`rainstorm`)

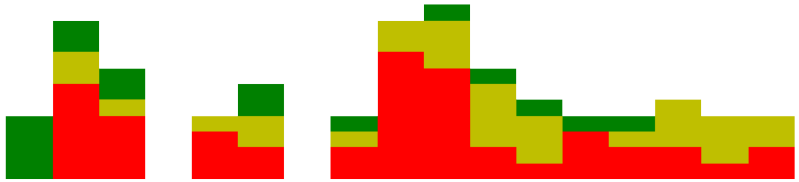


Figure: Submissions of `rainstorm`

- Author: Marco Donadoni
- Submissions: 76
- Score average: 67.74
- First points: `Arachidi Veloci` (100/100, 11min)

Flood Forecasting (`rainstorm`)

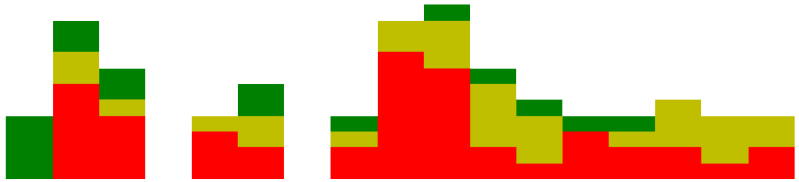
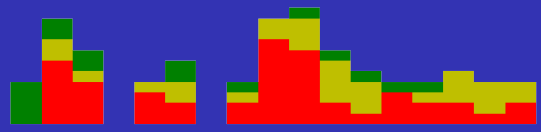


Figure: Submissions of `rainstorm`

- Author: Marco Donadoni
- Submissions: 76
- Score average: 67.74
- First points: `Arachidi Veloci` (100/100, 11min)
- First solution: `Arachidi Veloci` (11min)

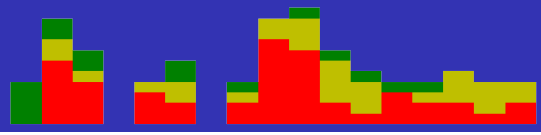
Flood Forecasting (`rainstorm`)



Interesting observation

We must keep the graph connected. One might think to start trying removing some edges to see if it is still connected, but we may also think the other way around. Begin with no edges and repeatedly add edges, starting from the "most robust" ones, until the graph is connected.

Flood Forecasting (`rainstorm`)



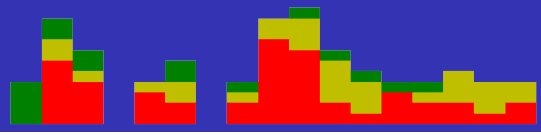
Interesting observation

We must keep the graph connected. One might think to start trying removing some edges to see if it is still connected, but we may also think the other way around. Begin with no edges and repeatedly add edges, starting from the "most robust" ones, until the graph is connected.

MST

This reminds the dual of a classical problem in computer science called *Minimum spanning tree*, which is solved by many algorithms to efficiently implement the above strategy (e.g., *Kruskal*'s algorithm).

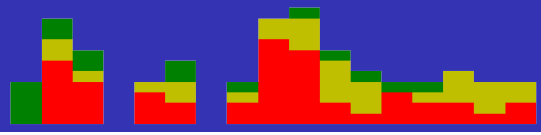
Flood Forecasting (`rainstorm`)



Alternative solution

If we fix the result (e.g., considering only streets that can withstand 10 mm of rain), we can easily check with a standard graph visit whether the graph got disconnected or not. Moreover, if an amount of rain x works, then all values $x' < x$ work too.

Flood Forecasting (`rainstorm`)



Alternative solution

If we fix the result (e.g., considering only streets that can withstand 10 mm of rain), we can easily check with a standard graph visit whether the graph got disconnected or not. Moreover, if an amount of rain x works, then all values $x' < x$ work too.

Binary search

With these facts, we can find the solution using a binary search on the result, reporting the maximum amount of rain that still keeps the graph connected.

Study Plan (`studyplan`)

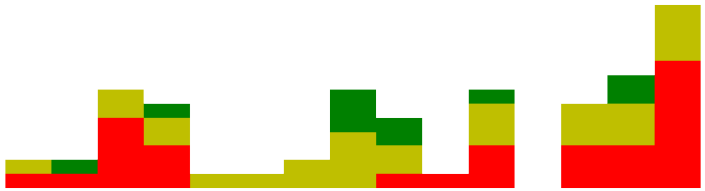


Figure: Submissions of `studyplan`

- Author: Edoardo Morassutto
- Submissions: 41
- Score average: 27.74

Study Plan (`studyplan`)

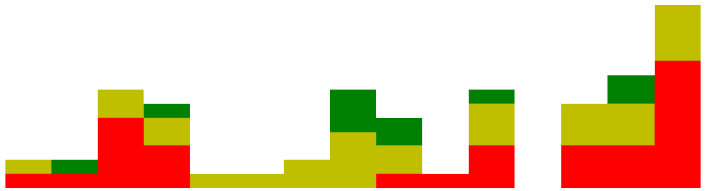


Figure: Submissions of `studyplan`

- Author: Edoardo Morassutto
- Submissions: 41
- Score average: 27.74
- First points: `Arachidi Veloci` (20/100, 30min)

Study Plan (`studyplan`)

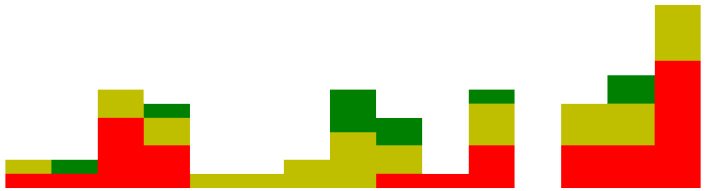
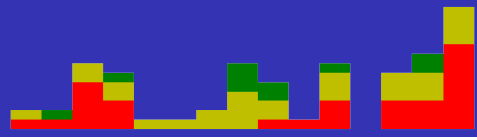


Figure: Submissions of `studyplan`

- Author: Edoardo Morassutto
- Submissions: 41
- Score average: 27.74
- First points: `Arachidi Veloci` (20/100, 30min)
- First solution: `Arachidi Veloci` (45min)

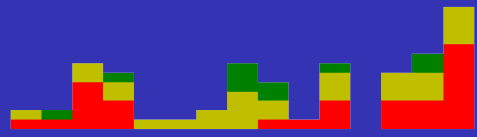
Study Plan (`studyplan`)



First Observation

The prerequisites can be modeled as a Directed Acyclic Graph.

Study Plan (`studyplan`)



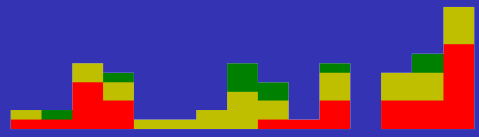
First Observation

The prerequisites can be modeled as a Directed Acyclic Graph.

Compute the total required time

The task implicitly asks you to compute the minimum time to study all the subjects. You can do so with a visit of the DAG.

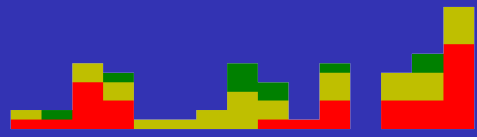
Study Plan (`studyplan`)



Naive solution

Compute the total time with a simple DFS search over all the possible paths, computing the minimum time `Tmin[i]` at which subject i can start. This value is the minimum between `Tmin[j]+H[j]` of all the prerequisites.

Study Plan (`studyplan`)



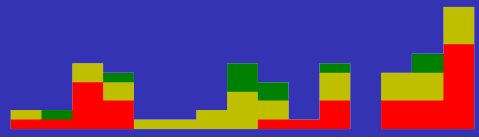
Naive solution

Compute the total time with a simple DFS search over all the possible paths, computing the minimum time `Tmin[i]` at which subject i can start. This value is the minimum between `Tmin[j]+H[j]` of all the prerequisites.

Complexity

In the worst case all the paths in the DAG are visited: $O(2^N)!$

Study Plan (`studyplan`)



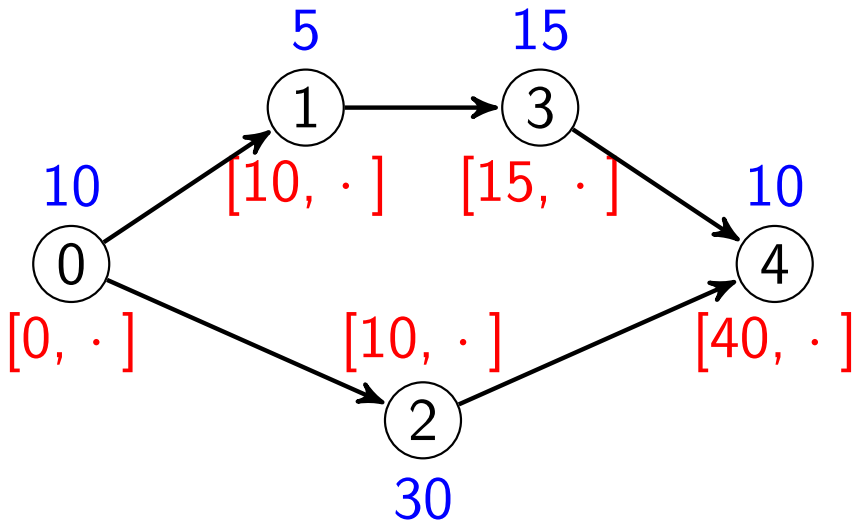
Faster solution

By *topologically sorting* the DAG we can compute `Tmin` in a smarter way: starting from the sources and following the topological order!

Study Plan (studyplan)

Faster solution

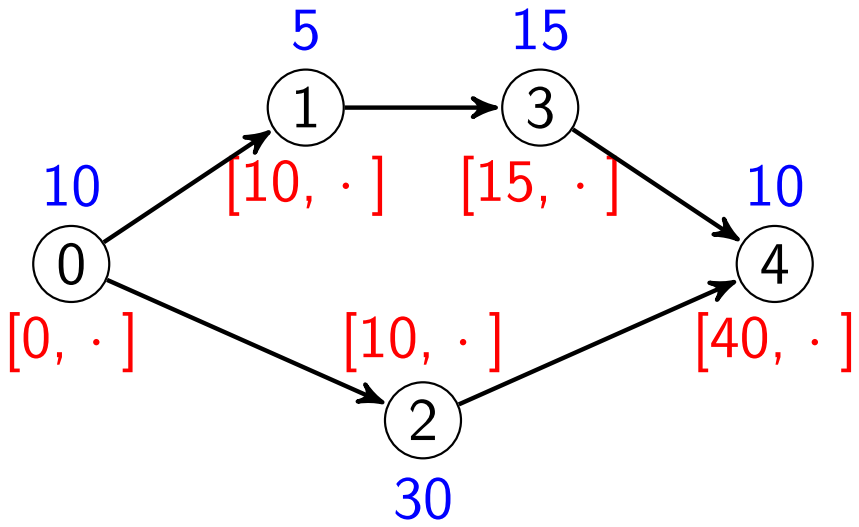
By *topologically sorting* the DAG we can compute `Tmin` in a smarter way: starting from the sources and following the topological order!



Study Plan (studyplan)

Faster solution

By *topologically sorting* the DAG we can compute `Tmin` in a smarter way: starting from the sources and following the topological order!



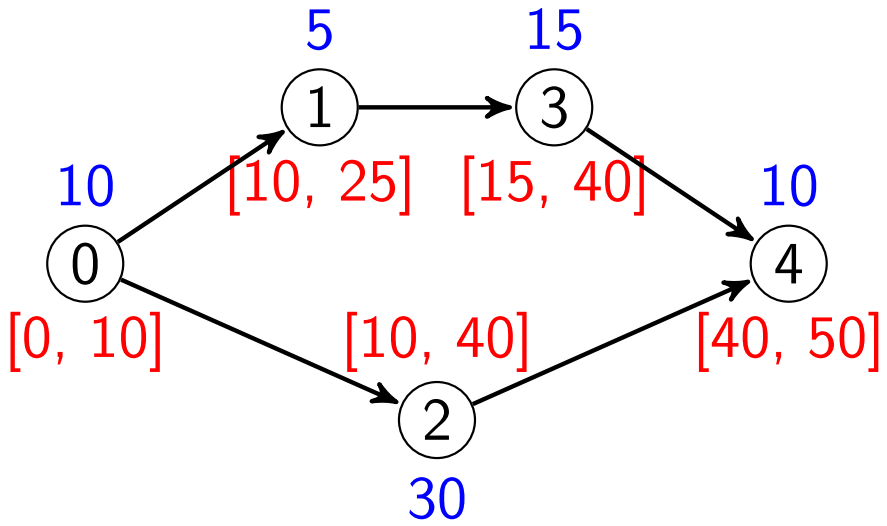
Complexity

The topological sort costs $O(M)$, and the single pass for computing `Tmin` is $O(M)$ as well.

Study Plan (studyplan)

Tmax

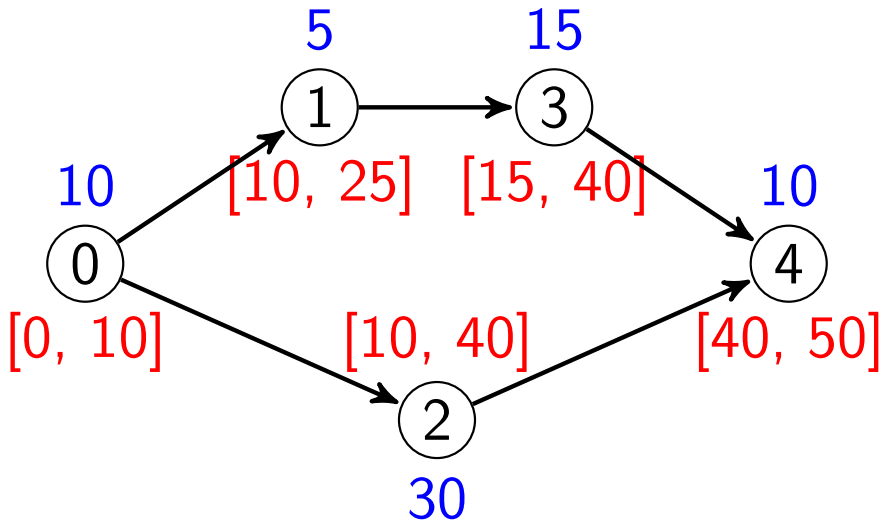
Defining `Tmax[i]` as the latest time subject i must end, we can compute the answer easily (how?). We can compute `Tmax` using:
`Tmax[i]` $= \min($`Tmax[j]` $-$ `H[j]`$)$ over the subjects j that have i as a prerequisite.



Study Plan (studyplan)

Tmax

Defining `Tmax[i]` as the latest time subject i must end, we can compute the answer easily (how?). We can compute `Tmax` using:
`Tmax[i]` $= \min($`Tmax[j]` $-$ `H[j]`$)$ over the subjects j that have i as a prerequisite.



Final complexity

The calculation of `Tmax` requires $O(M)$ operations.

Mysterious Sum (sum)

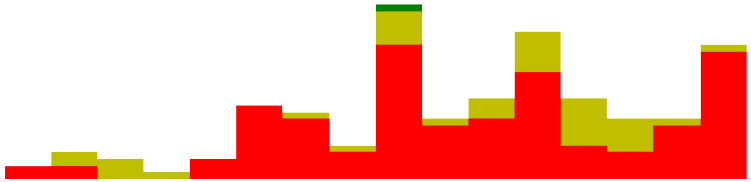


Figure: Submissions of sum

- Author: Edoardo Morassutto
- Submissions: 120
- Score average: 12.90

Mysterious Sum (`sum`)

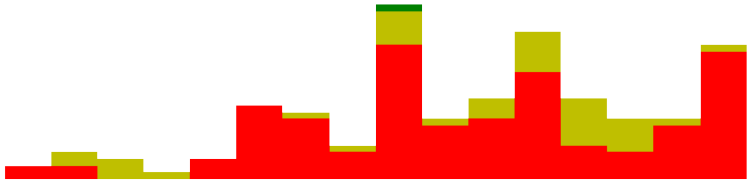


Figure: Submissions of `sum`

- Author: Edoardo Morassutto
- Submissions: 120
- Score average: 12.90
- First points: `GALILEI_01` (50/100, 36min)

Mysterious Sum (`sum`)

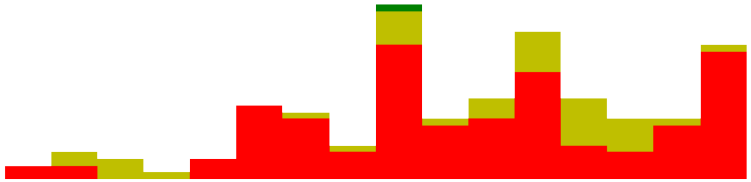
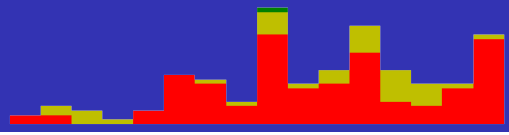


Figure: Submissions of `sum`

- Author: Edoardo Morassutto
- Submissions: 120
- Score average: 12.90
- First points: `GALILEI_01` (50/100, 36min)
- First solution: `Sushi Squad` (1h46min)

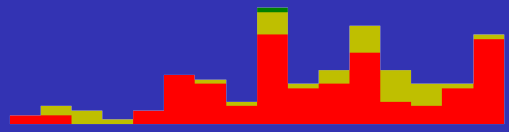
Mysterious Sum (sum)



Summary

The problem asks to find a bidirectional mapping from $A \ldots J$ to $0 \ldots 9$ that, once used for decoding the operands, their sum is the result.

Mysterious Sum (sum)



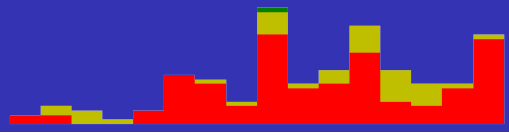
Summary

The problem asks to find a bidirectional mapping from $A \ldots J$ to $0 \ldots 9$ that, once used for decoding the operands, their sum is the result.

First observation

Since the numbers involved can be huge, integers are not a viable option.
The operands and the result should be stored as a string.
The addition operation should be done "by hand".

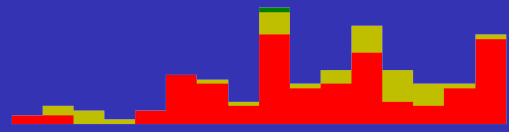
Mysterious Sum (`sum`)



First solution

Try all the possible assignments of the mapping and check whether one of them is valid, for example using `std::next_permutation`.

Mysterious Sum (`sum`)



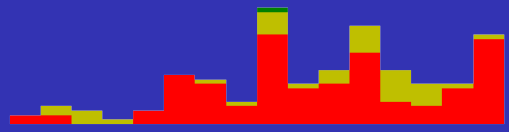
First solution

Try all the possible assignments of the mapping and check whether one of them is valid, for example using `std::next_permutation`.

Complexity

The number of possible mappings is 10!, checking one of them costs $O(N)$, so the total complexity is $O(10!N) = O(N)$.

Mysterious Sum (sum)



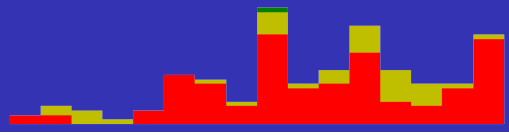
First solution

Try all the possible assignments of the mapping and check whether one of them is valid, for example using `std::next_permutation`.

Complexity

The number of possible mappings is 10!, checking one of them costs $O(N)$, so the total complexity is $O(10!N) = O(N)$.
Notice that $10! = 3\,628\,800$, a very high constant!

Mysterious Sum (sum)



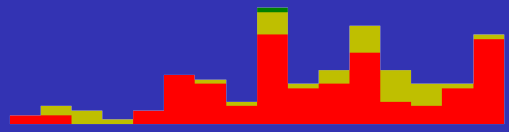
Speeding things up

The first solution doesn't take into account the digits of the 3 numbers!
Many assignments can be excluded taking into consideration the relationship between the operands and the result.

Solution

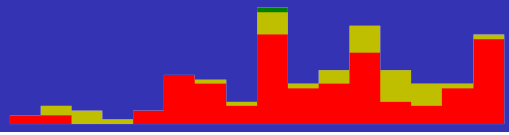
With a recursive function we can enumerate all the *possible* mappings, not visiting the branches with impossible mappings and stopping as soon as a valid solution is found.
This function keeps a set of *used* digits and *known* characters and try as much as possible to deduce new digits.

Mysterious Sum (sum)



Going from the *least significant digits*, when new unknown characters are encountered the following algorithm is followed (naming α the digit of the first operand, β the digit of the second, γ the digit of the result and r the remainder of the previous sum):

Mysterious Sum (sum)

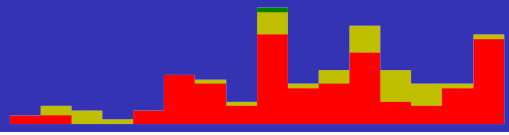


Going from the *least significant digits*, when new unknown characters are encountered the following algorithm is followed (naming α the digit of the first operand, β the digit of the second, γ the digit of the result and r the remainder of the previous sum):

Case #1

$\alpha = \beta$ and they are unknown. $2\alpha + r = \gamma \bmod 10$, the result must be even (or odd if $r = 1$). The only two options are
$\alpha = \beta = (\gamma - r)/2$ [+5] mod 10.

Mysterious Sum (sum)



Going from the *least significant digits*, when new unknown characters are encountered the following algorithm is followed (naming α the digit of the first operand, β the digit of the second, γ the digit of the result and r the remainder of the previous sum):

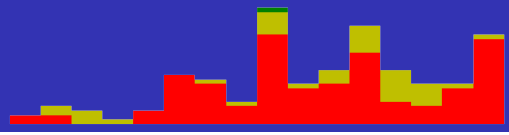
Case #1

$\alpha = \beta$ and they are unknown. $2\alpha + r = \gamma \mod 10$, the result must be even (or odd if $r = 1$). The only two options are
$\alpha = \beta = (\gamma - r)/2$ [+5] mod 10.

Case #2

$\alpha \neq \beta$ and they are both unknown. Try all the possible α and deduce the value of $\beta = \gamma - \alpha - r \mod 10$.

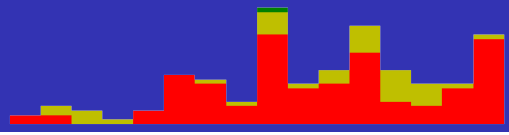
Mysterious Sum (sum)



Case #3 and #4

$\alpha \neq \beta$ and one of them is unknown. Deduce the value of the other, for example $\beta = \gamma - \alpha - r \bmod 10$.

Mysterious Sum (sum)



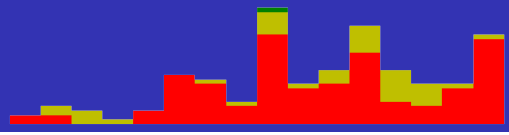
Case #3 and #4

$\alpha \neq \beta$ and one of them is unknown. Deduce the value of the other, for example $\beta = \gamma - \alpha - r \bmod 10$.

Case #5

α and β are known. Just check that the result is valid!

Mysterious Sum (sum)



Case #3 and #4

$\alpha \neq \beta$ and one of them is unknown. Deduce the value of the other, for example $\beta = \gamma - \alpha - r \bmod 10$.

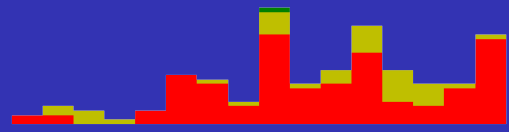
Case #5

α and β are known. Just check that the result is valid!

Reaching the end

Since at the first error the function backtracks, reaching the end means that all the (used) characters are mapped. Remember to check the last remainder (the extra digit of the result!).

Mysterious Sum (sum)



Case #3 and #4

$\alpha \neq \beta$ and one of them is unknown. Deduce the value of the other, for example $\beta = \gamma - \alpha - r \bmod 10$.

Case #5

α and β are known. Just check that the result is valid!

Reaching the end

Since at the first error the function backtracks, reaching the end means that all the (used) characters are mapped. Remember to check the last remainder (the extra digit of the result!).

Complexity

Assigning a digit always fixes *at least* another digit, so at most 5 of them are *bruteforced*: $10!/5! = 30\,240$.

Circus Show (cannons)

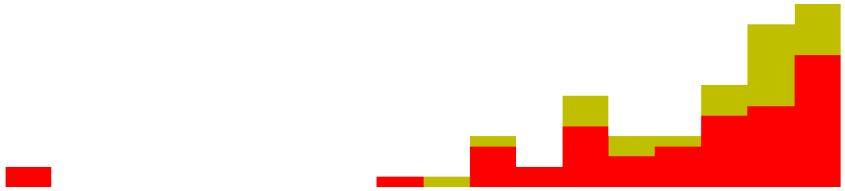


Figure: Submissions of cannons

- Author: Edoardo Morassutto
- Submissions: 29
- Score average: 3.55

Circus Show (`cannons`)

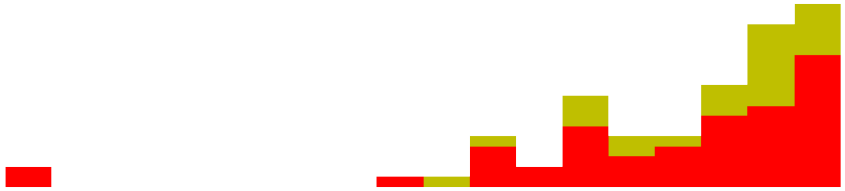


Figure: Submissions of `cannons`

- Author: Edoardo Morassutto
- Submissions: 29
- Score average: 3.55
- First points: `Simpateam` (10/100, 1h30min)

Circus Show (`cannons`)

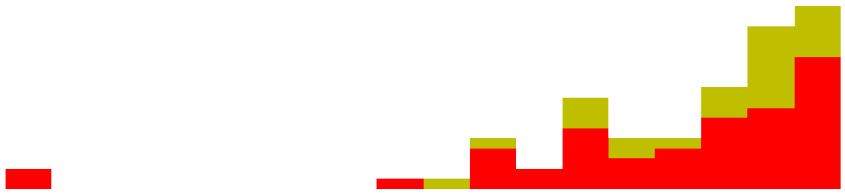


Figure: Submissions of `cannons`

- Author: Edoardo Morassutto
- Submissions: 29
- Score average: 3.55
- First points: `Simpateam` (10/100, 1h30min)
- Best solution: `Sushi Squad` (70/100, 2h12min)

Circus Show (cannons)

Key Idea

Model the problem as the Shortest Path problem on a specific graph: each cannon is a node, there is a directed arc between each pair of cannons with the cost of making that connection (zero for the already present connections).

Circus Show (cannons)

Key Idea

Model the problem as the Shortest Path problem on a specific graph: each cannon is a node, there is a directed arc between each pair of cannons with the cost of making that connection (zero for the already present connections).

First attempt

Since all the weights are non-negative we can use the Dijkstra's algorithm.

Complexity

There are $O(N^2)$ arcs so the total complexity is $O(N^2)$.

Circus Show (cannons)

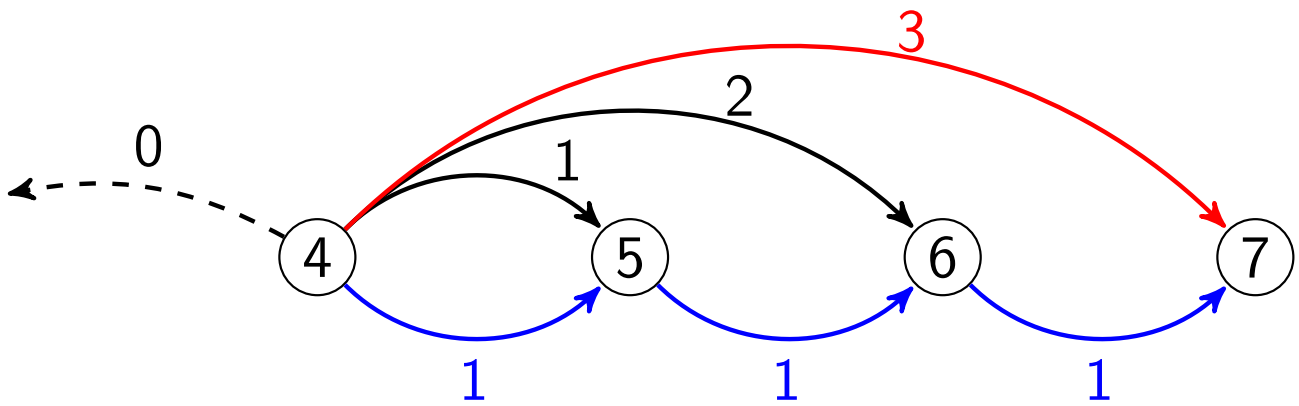
Key Observation

If we decide to change cannon i, we just need to set it to $i+1$ or $i-1$. Most of the arcs are useless, so, for each node at most 3 arcs survive, all of which with cost 0 or 1.

Circus Show (cannons)

Key Observation

If we decide to change cannon i, we just need to set it to $i+1$ or $i-1$. Most of the arcs are useless, so, for each node at most 3 arcs survive, all of which with cost 0 or 1.



Circus Show (cannons)

Solution

Under those conditions we can use the 0-1 BFS algorithm to find the shortest path in this graph.

Complexity

There are $O(N)$ arcs so the total complexity is $O(N)$.

Multi-Layer Dictionary (`dictionary`)

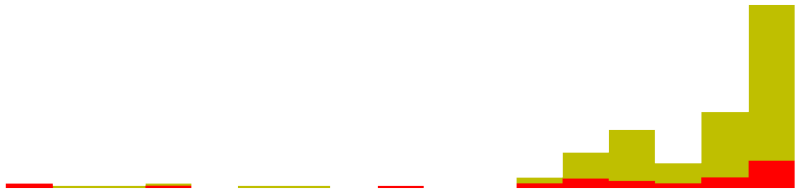


Figure: Submissions of `dictionary`

- Author: William Di Luigi
- Submissions: 56
- Score average: 11.28

Multi-Layer Dictionary (`dictionary`)

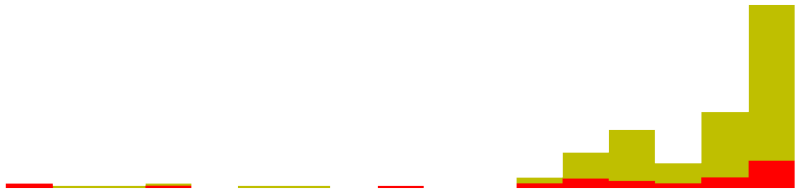


Figure: Submissions of `dictionary`

- Author: William Di Luigi
- Submissions: 56
- Score average: 11.28
- First points: `Sushi Squad` (41.32/100, 28min)

Multi-Layer Dictionary (`dictionary`)

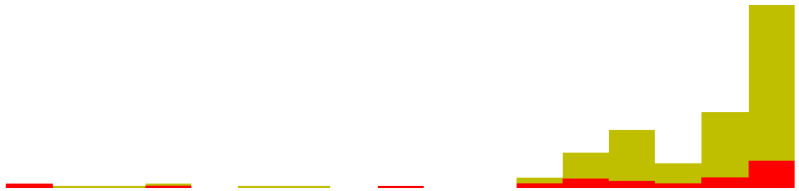


Figure: Submissions of `dictionary`

- Author: William Di Luigi
- Submissions: 56
- Score average: 11.28
- First points: `Sushi Squad` (41.32/100, 28min)
- Best solution: `Sushi Squad` (61.43/100, 2h23min)

Multi-Layer Dictionary (`dictionary`)

Partial scoring

This task is a partial scoring task! To get full score, you need to "beat" our solution and suboptimal solutions can get (some) points, based on how many words they select to be primitive.

Graph modelization

We can model the problem as a graph. The graph has one node for every word, and there is a directed edge from word i to j if j is used in the definition of i.

Multi-Layer Dictionary (`dictionary`)

Words with no definition

In the constructed graph there might be nodes that don't have exiting arcs: these arcs are words with no definition, so they must be part of the primitive words that need to be learned.

Multi-Layer Dictionary (`dictionary`)

Words with no definition

In the constructed graph there might be nodes that don't have exiting arcs: these arcs are words with no definition, so they must be part of the primitive words that need to be learned.

Preprocessing

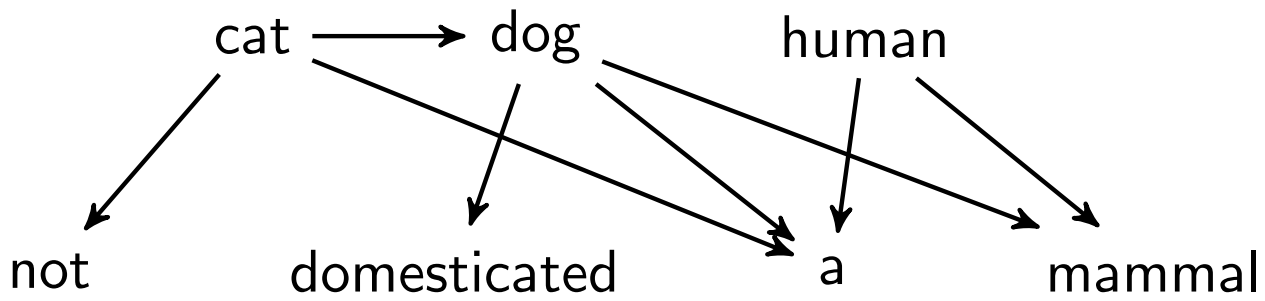
Since words with no definition must be part of the set of primitive words, the graph can be preprocessed by removing all the nodes with no outgoing arcs. When a node is removed, also all arcs incident to it are removed. We are left with D nodes, that is all the words which have a definition.

Multi-Layer Dictionary (`dictionary`)

Example

```
cat = not a dog
dog = a domesticated mammal
human = a mammal
```

Before preprocessing:

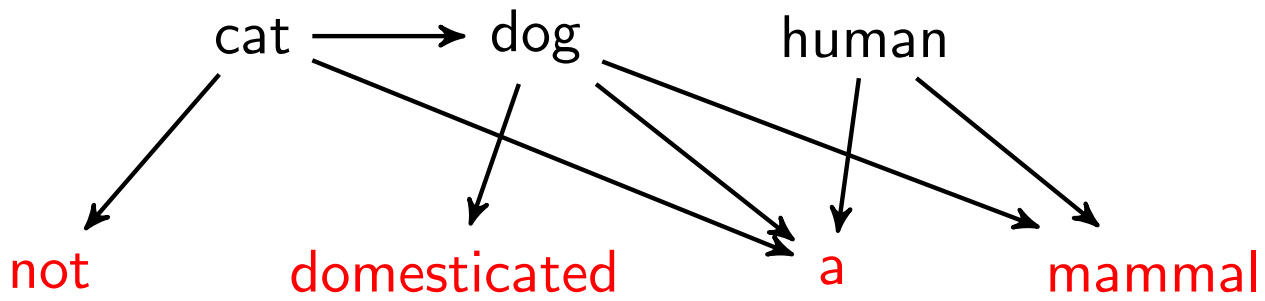


Multi-Layer Dictionary (`dictionary`)

Example

```
cat = not a dog
dog = a domesticated mammal
human = a mammal
```

Before preprocessing:



After preprocessing:

Multi-Layer Dictionary (`dictionary`)

First Observation

In the processed graph, there can still be nodes with no exiting arcs. These nodes are the words that have a definition and all the words of the definition are already learned. This means that those words can be learned without introducing new primitive words, and they can be removed from the graph. This step can be repeated iteratively, until no more nodes can be removed.

Multi-Layer Dictionary (`dictionary`)

First Observation

In the processed graph, there can still be nodes with no exiting arcs. These nodes are the words that have a definition and all the words of the definition are already learned. This means that those words can be learned without introducing new primitive words, and they can be removed from the graph. This step can be repeated iteratively, until no more nodes can be removed.

Second Observation

By the first observation, if the processed graph doesn't contain a cycle, all the words can be learned without adding new words to the primitive set (think about the topological sorting of the nodes!).

Multi-Layer Dictionary (`dictionary`)

Example

```
cat = not a dog
dog = a domesticated mammal
human = a mammal
```

Processed graph:

Multi-Layer Dictionary (`dictionary`)

Example

```
cat = not a dog
dog = a domesticated mammal
human = a mammal
```

Processed graph:

cat ⟶ dog human

The words “dog” and “human” can be learned by using already known words and can be removed:

cat

Multi-Layer Dictionary (`dictionary`)

Example

```
cat = not a dog
dog = a domesticated mammal
human = a mammal
```

Processed graph:

cat ⟶ dog human

The words "dog" and "human" can be learned by using already known words and can be removed:

cat

Also "cat" can be learned without adding new primitive words.

Multi-Layer Dictionary (`dictionary`)

Solution

Let P be the set of primitive words. First of all the solution preprocess G to remove words with no definition and puts them in P.
Let G be the processed graph and G' an initially empty graph. We consider the nodes of G one by one, in a given order. If by adding a node in consideration to G', along with its edges, a cycle is not formed then the node is inserted into G', otherwise the word is added to P.

Multi-Layer Dictionary (`dictionary`)

Solution

Let P be the set of primitive words. First of all the solution preprocess G to remove words with no definition and puts them in P.
Let G be the processed graph and G' an initially empty graph. We consider the nodes of G one by one, in a given order. If by adding a node in consideration to G', along with its edges, a cycle is not formed then the node is inserted into G', otherwise the word is added to P.

Why does this give a correct solution?

The solution is trying to reconstruct the original graph, but if a word causes a cycle then it is considered as a primitive one. As we said in the second observation, if we end up with a graph that doesn't contain a cycle then all the words can be learned without new primitive words!

Multi-Layer Dictionary (`dictionary`)

Strategy to select nodes

We need to devise how to order to nodes that need to be considered in the solution. Possible strategies:

- Random, but not very good.

Multi-Layer Dictionary (`dictionary`)

Strategy to select nodes

We need to devise how to order to nodes that need to be considered in the solution. Possible strategies:

- Random, but not very good.
- Ordered by number of incoming arcs (from smallest to greatest).

Multi-Layer Dictionary (`dictionary`)

Strategy to select nodes

We need to devise how to order to nodes that need to be considered in the solution. Possible strategies:

- Random, but not very good.
- Ordered by number of incoming arcs (from smallest to greatest).
- Randomly, but nodes with more incoming arcs have less probability of being chosen.

Multi-Layer Dictionary (`dictionary`)

Strategy to select nodes

We need to devise how to order to nodes that need to be considered in the solution. Possible strategies:

- Random, but not very good.
- Ordered by number of incoming arcs (from smallest to greatest).
- Randomly, but nodes with more incoming arcs have less probability of being chosen.
- Generate K random solutions. For each word count how many times it is present in the primitive set of the K solutions. Then select randomly starting from the words which appear zero times, up to the words that appear the most.

Questions?

This year...

This year...

Writing the solution of task Polygon (round4)

"ad una certa c'era un array offBy1x e offBy1y per fixare i casi particolari, ma era più buggato che altro ed alla fine la cosa giusta era avere solo zeri lì dentro ahahah"

"quindi off-by-one negli off-by-one ahah"

This year...

Writing the solution of task Polygon (round4)

"ad una certa c'era un array offBy1x e offBy1y per fixare i casi particolari, ma era più buggato che altro ed alla fine la cosa giusta era avere solo zeri lì dentro ahahah"

"quindi off-by-one negli off-by-one ahah"

Bug Digitali 😎

```
unacceptable character #x1f60e:  special characters are not allowed position 12761
```

This year...

Writing the solution of task Polygon (round4)

"ad una certa c'era un array offBy1x e offBy1y per fixare i casi particolari, ma era più buggato che altro ed alla fine la cosa giusta era avere solo zeri lì dentro ahahah"

"quindi off-by-one negli off-by-one ahah"

Bug Digitali 😎

```
unacceptable character #x1f60e:  special characters are not allowed position 12761
```

Way too many bugs...

...caused by Pascal being case insensitive!

This year...

Plagiarism

This year a total of 4850 points were removed because they were obtained by cheating.

This year...

Plagiarism

This year a total of 4850 points were removed because they were obtained by cheating.

Python and its slowness

We spent countless hours adjusting the time limits for letting Python score some points while not letting lame solutions pass due the high limits.

This year...

Plagiarism

This year a total of 4850 points were removed because they were obtained by cheating.

Python and its slowness

We spent countless hours adjusting the time limits for letting Python score some points while not letting lame solutions pass due the high limits.

`cms` and its bugs

Have you experienced some errors at the start of the contests? Well, we have no idea of what causes them. All the attempts to reproduce the problem failed miserably.

Some of the inputs of `polygon`