C – Number Snake

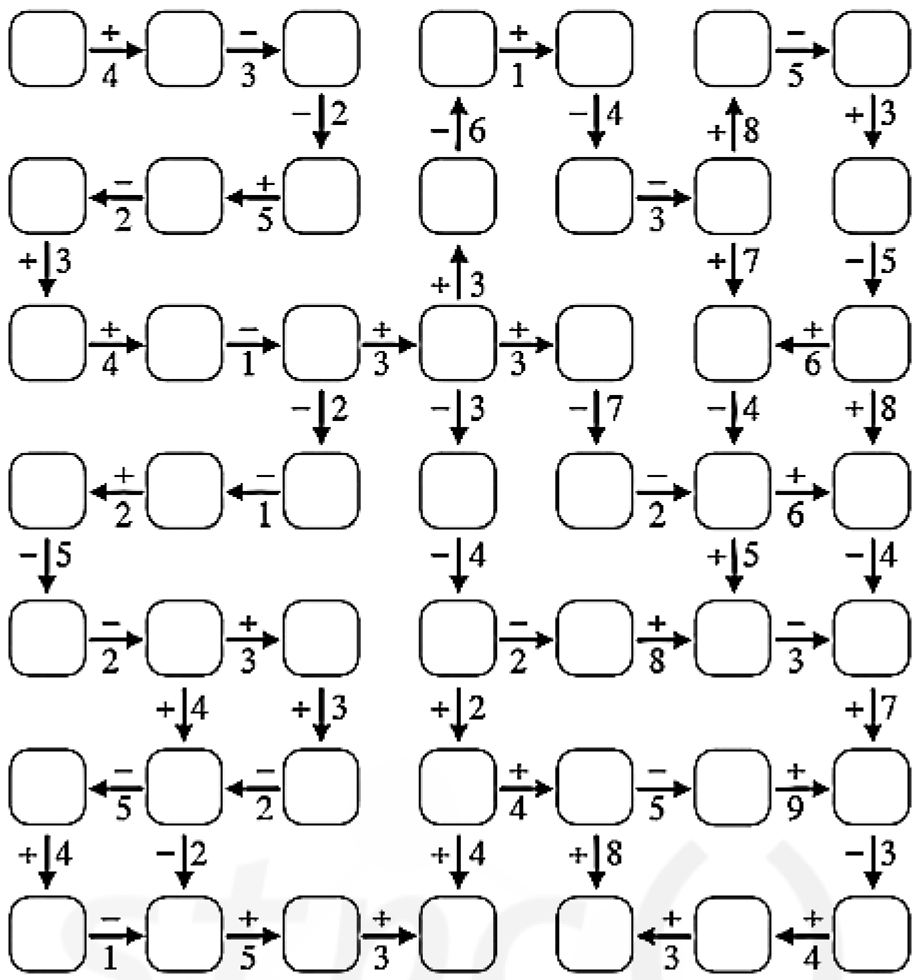
Luo Tsz Fung {pepper1208}

2025-05-16

Background

Problem Idea by `pepper1208`

Preparation by `pepper1208`, `rina__owo`



An example of Number Snake

Problem Restatement

Fill in all squares in number snake with integer within X and Y inclusive.

There may be rules between two squares such that the integers in two squares must satisfy a given relation (addition / subtraction).

If it is impossible to fill in all squares while satisfying all conditions, output `Impossible`.

Otherwise, output `Possible` and the content of the number snake.

Subtasks

- Attempts: 19
- Max: 100
- First solved by **Chan Tsz Hang** at **2h 16m 45s**

Subtasks Constraints

	Score	Constraints
1	14	$M = N - 1$ For the *i*-th rule, $x = i, y = i + 1$
2	11	$M = N$ For the *i*-th rule, $x = i, y = (i \bmod N) + 1$
3	27	$M = N - 1$ For the *i*-th rule, $x = i, y = i + 1$ or $x = i + 1, y = i$
4	48	No additional constraints

Subtask 1 (14%): A chain

- Obviously, we can treat the number snake as a graph.
  - Implementation of graph: adjacency list
- The puzzle forms a chain, with node 1 as the head of the chain.
- Therefore, we can fill in an arbitrary integer in node 1, and fill in the rest of the node based on the given relations.
- However, there is a catch: what should we do if there are some integers lie outside the bound (*X* and *Y* inclusive) ?

Subtask 1 (14%): A chain

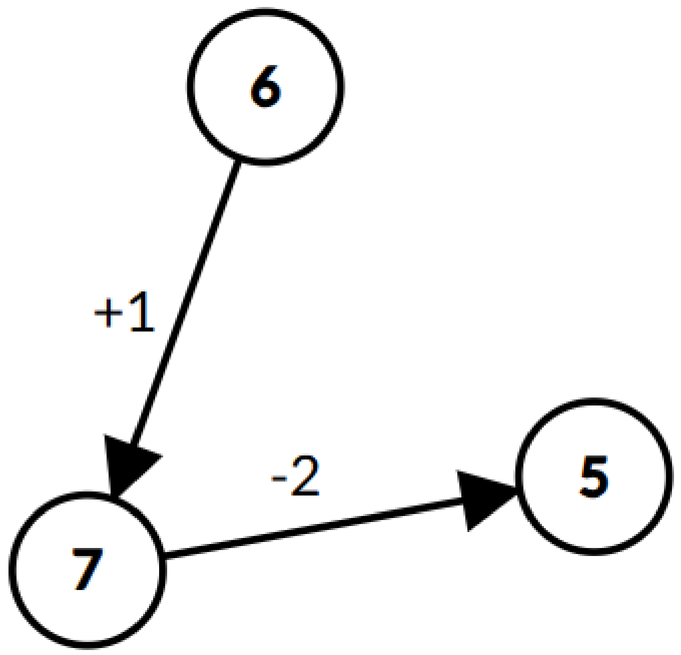
- Observation 1.1: If there exists a possible construction, the range of the integers used must not exceed $Y - X$.
- Proof: The least integer in the construction must not be less than X and the greatest integer in the construction must not be greater than Y.

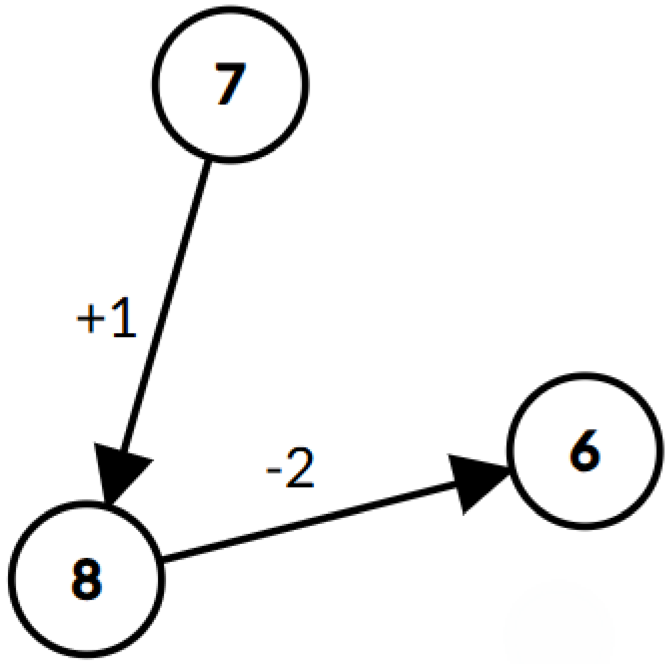
- Therefore, if we determine the range of the integers in the construction, we can determine whether it is possible to generate a construction.

Subtask 1 (14%): A chain

- Observation 1.2: The range of integers filled is fixed for a given chain.
- Proof: Assume an arbitrary integer N is filled in node 1. Then remaining nodes can be determined. Let integer filled in node i is equal to $N + k_i$. Denote the maximum and minimum integers filled be $N + k_{\max}$ and $N + k_{\min}$ respectively, with the range of integers filled be $k_{\max} - k_{\min}$.
- If the integer filled in node 1 become $N + M$, then the maximum and minimum integers filled will be $N + \mathrm{M} + k_{\max}$ and $N + \mathrm{M} + k_{\min}$ respectively, the range of integers filled will still be $k_{\max} - k_{\min}$.

Subtask 1 (14%): A chain





Subtask 1 (14%): A chain

- After calculating k_{max} and k_{min} respectively, we can determine the range of the construction. k_{max} and k_{min} can be calculated by traversing the chain once, with O(N) time complexity.
- If $k_{max} - k_{min}$ exceed $Y - X$, output `Impossible`.
- Otherwise, choose a suitable N which $N + k_{max}$ and $N + k_{min}$ both lie between the range $[X, Y]$.
  - For example, we can choose a $N = X - k_{min}$.
- Expected score: **14** (Culminative Score: **14**)

Subtask 2 (11%): A cycle

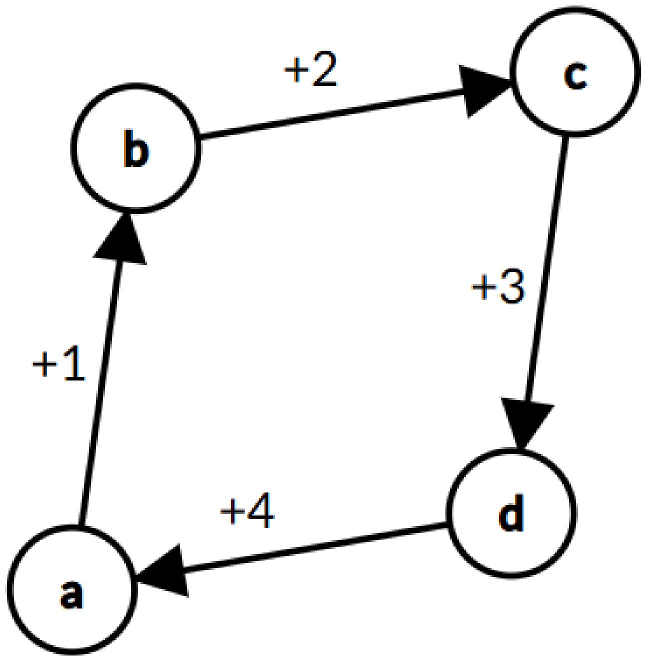
- The puzzle forms a cycle.
- What is the relationship between a cycle and a chain?
- A chain is formed when ignoring exactly one edge in a cycle.
- Therefore, by ignoring one edge, we can fill in all square using the algorithm implemented in subtask 1.
- However, we miss something if we stop here.

Subtask 2 (11%): A cycle

- Denote those edge(s) being ignored as “check edge”. Denote the remaining edge(s) used to help you fill in integers as “fill edge”.
- After using up all the “fill edges” to fill in integers, use all “check edges” to validate the correctness of the construction. If the construction is checked to be invalid by some “check edges”, output `Impossible`.

Subtask 2 (11%): A cycle

- Example for a graph with impossible construction checked by “check edge”:



Subtask 2 (11%): A cycle

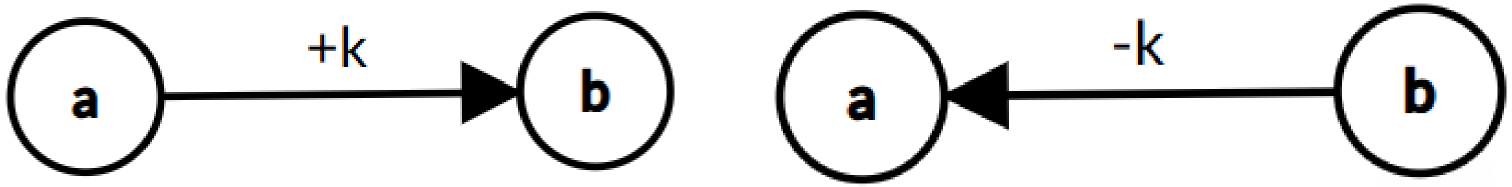
- With additional implementation, it should be easy for you to complete this subtask!
- Expected score: **11** (Culminative Score: **25**)

Subtask 3 (27%): A chain-like graph

- The puzzle forms a chain-like structure.
- It seems like we cannot fill in all integers by traversing once.

Subtask 3 (27%): A chain-like graph

- The puzzle forms a chain-like structure.
- ~~It seems like we cannot fill in all integers by traversing once.~~ We can!
- Observation 2: We can reverse any edge.
- Proof: Trivial.

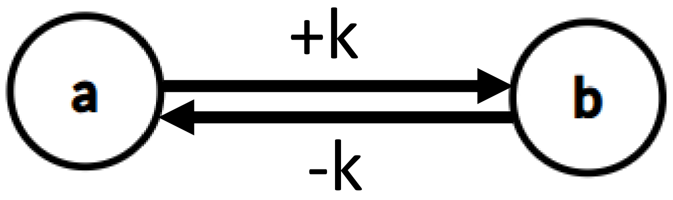


Subtask 3 (27%): A chain-like graph

- The graph is now modified to be a chain again.
- Use the algorithm in subtask 1 again.
- Easy subtask! ☺
- Expected score: **27** (Culminative Score: **52**)

Subtask 4 (48%): No additional constraints

- Now, we can use all of our observations to implement the full solution.
- If there exist a relation between two squares, build the reversed edge between them also.



Subtask 4 (48%): No additional constraints

- Starting from an arbitrary node, we can fill in all the squares as any two squares must exist bidirectional relation if there exist a relation between them before. In other words, there must exist some edges to form a DAG (Direct Acyclic Graph). They are called “fill edges” with the previous definition. Fill in all squares by DFS.
- Use all the remaining “check edges” to validate the correctness of the graph.
- Expected score: **100 AC!**

Subtask 4 (48%): No additional constraints

- Starting from an arbitrary node, we can fill in all the squares as any two squares must exist bidirectional relation if there exist a relation between them before. In other words, there must exist some edges to form a DAG (Direct Acyclic Graph). They are called “fill edges” with the previous definition. Fill in all squares by DFS.
- Use all the remaining “check edges” to validate the correctness of the graph.
- Expected score: ~~100 AC!~~ **52** (Why?)

Subtask 4 (48%): No additional constraints

- It is NOT guaranteed that the whole puzzle is a connected components.
- Therefore, we need to use flood fill to check for all connected components.
- Nevertheless, if one of the connected components does not lead to a correct construction, output `Impossible` and terminate the algorithm.
- Expected score: **100 AC!**
- Time complexity: O(N + M)

Takeaways

- Be aware of different approach to model a graph.
- For a graph problem, you can start to think about some special case (e.g. chain, cycle, DAG, …) to help you get the full picture.
- Flood fill is necessary for several connected components exist.
- Generally, draw a lot to help you understand the question clearly.