,

Задача A. Финальный RegExp

В данной задаче требовалось сделать regexp, который выделял бы отдельно последовательность букв, отдельно последовательность цифр, проверял бы тег, в который помещена группа, и делал бы замену группы.

Можно использовать regexp «\textit\{((\w+)|(\d+))\}»

Задача B. Долгий путь домой

Заметим, что запросы добавления в конец массива и удаления из конца массива реализуются обычным стеком. Проблема возникает только при удалении произвольного элемента.

Для этого заведем отдельный set, в который будем складывать все удаленные элементы. Чистить стек можно амортизированно: заведем функцию fix_stack, которая будет удалять верхние элементы стека, если эти элементы уже лежат в set-е удаленных элементов. Данный метод можно будет вызывать после каждого запроса — поскольку каждый элемент из стека удаляется всего один раз, то амортизированное время работы $O(n)$.

```
def fix_stack(stack, removed):
    while len(stack) > 0 and stack[-1] in removed:
        stack.pop()
```

Задача C. K-я строка

В задаче предлагается создать бор и поддерживать в каждой вершине количество терминальных вершин в поддереве при добавлении новых строк. При добавлении s количество вершин в поддереве увеличивается только в вершинах, которые лежат на пути добавления строки.

Для нахождения k-й строки используется спуск по бору — на каждом шаге перебирается символ по возрастанию, и следующим добавляется тот символ, при переходе по которому все еще доступна k-я строка.

```
  while k > 1:
    if node.is_term:
      print(res)
      break
    for c in range(26):
      if node.child[c] is None:
        continue
      if node.child[c].size < k:
        k -= curr->child[i]->size;
      res += chr(ord('a') + c)
      node = node.child[c]
      break
```

Задача D. Игрушечный лабиринт

Сначала с помощью прохода по строкам и столбцам найдем для каждой клетки ближайшую стенку (или финишную клетку) слева, справа, сверху и снизу. С помощью этих данных можно построить граф — у каждой клетки будет иметься 4 ребра, соответствующие наклону доски в разные стороны.

В получившемся графе достаточно запустить bfs из стартовой клетки с поиском одной из финишных клеток.

Задача E. Рецепт

Данная задача, на самом деле, решается практически так же, как задача 9C.

В задаче задан набор вершин с заданными стоимостями — базовые ингредиенты. Также с помощью рецептов заданы наборы ребер.

Таким образом, для каждой вершины верна следующая формула пересчета:

,

$$value_v = min(start\_value_v, \sum_{v \to u} value_u)$$

Поскольку в задаче гарантируется отсутствие циклов, то такую динамику на DAG нужно пересчитать: либо с помощью dfs, либо с помощью topsort.