ИЗУМРУД СТУДЕНТ

3101654160027

Титульный лист

Направление
- ☐ Естественные науки
- ☐ Инженерные науки
- ■ Математика и информатика
- ☐ Социальные и гуманитарные науки
- ☐ Экономика и управление

Вариативный блок ☐ 1 ☐ 2 ☐ 3 ■ 4 ☐ 5

Курс ☐ 1 ☐ 2 ☐ 3 ■ 4 ☐ 5 ☐ отсутствует

Фамилия ЯКОВЛЕНКО

Имя ВЛАДИМИР

Отчество АЛЕКСАНДРОВИЧ

Дата рождения 15 05 2007

Город участия ЕКАТЕРИНБУРГ

Аудитория Д3

Телефон

Дата 03 02 2025 **Подпись**

Пример заполнения

А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф

Х Ц Ч Ш Щ Ъ Ы Ь Э Ю Я 1 2 3 4 5 6 7 8 9 0

ИЗУМРУД СТУДЕНТ

3101654160027

Проверочный лист

Заполняется участниками

Направление
- ☐ Естественные науки
- ☐ Инженерные науки
- ■ Математика и информатика
- ☐ Социальные и гуманитарные науки
- ☐ Экономика и управление

Вариативный блок ☐ 1 ☐ 2 ☐ 3 ■ 4 ☐ 5

Курс ☐ 1 ☐ 2 ☐ 3 ■ 4 ☐ 5 ☐ отсутствует

Город участия Екатеринбург

Заполняется организаторами

Количество доп листов

Количество черновиков к проверке

Время выхода с 12 43 до 12 48

Протокол проверки
Заполняется жюри

Номер задания	1	2	3	4	5	6	7	8	9	10
Балл члена жюри №1	4	25								
Балл члена жюри №2	—	25								

Итоговый балл 25

Подпись члена жюри №1

Подпись члена жюри №2

Пример заполнения

А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф

Х Ц Ч Ш Щ Ъ Ы Ь Э Ю Я 1 2 3 4 5 6 7 8 9 0

Линия отреза

Бланк ответов

Вариативная часть Блок 4 используемая СУБД PostgreSQL

2 Для дифференциации разных типов камней по их форме, необходимо иметь отдельную колонку в таблице, например, shapeType, где в которая будет иметь тип int, а ее значения следующие

Знач	Форма
0	Не определено, значение по умолчанию
1	Треугольная
2	Прямоугольная
3	Трапеция
4	Четырехугольник
5	Пятиугольник

Значения для shapeType будем вносить в отдельную таблицу, т к вполне вероятно, что сама база данных в целом будет заполняться с помощью ORM (object relational Mapping), и тогда целостность данных этой колонки будет поддерживаться тем что соответствующий enum ShapeType в коде будет иметь только нужные значения.

Также для данных о камне должны включать в себя размер длину и ширину. Их положим в отдельные колонки „lengthCm" и „widthCm" в коде это означает, что величина для этих полей - $см^2$

Тип у колонок будет также int4

Размеры будем хранить в JSON-массиве в колонке типа jsonb Поскольку в рамках работы алгоритма потребуются операции с размерами, дополнительно добавим на эту колонку GIN-индекс назовем колонку dimensions

Дополнительно добавим колонку Id для каждого камня, тип - uuid эта колонка будет первичным ключом

Ответственность так за валидацию размеров камней, их типа возложим на геометрическую информационную систему

Таким образом, получаем таблицу вида

Линия отреза

Бланк ответов

1 В условии сказано, что использование растворов должно быть минимальным, следовательно дорожка должна быть максимально возможного размера

Create table stones (
id int4, PK
shapeType int4,
dimensions jsonb, как тип GIN индекс
)

Пример заполнения.

id	shapeType	dimensions
0b83ac74-aabb-cccc-0000000000000	1	[3,4,5]
abca3455-8877-aadd-3456aabb ccddeeaa	2	[2,5]

1. Применим так называемое динамическое программирование. На каждом шаге будем оценивать, на сколько увеличится площадь и сколько в граммах раствора это будет стоить. Для соблюдения условий задачи нам потребуется брать не варианты, где вклад в увеличение площади максимальный, а затраты раствора минимальны.

Каждый последующий шаг будем хранить в узлах графа, сохраняя ссылку на предыдущий. В конце концов с помощью backtracking найдем самый выгодный путь

Структура для узла будет иметь вид

class Node

{

Node prev, – предыдущий узел

~~int SquareDelta – на сколько изменилась по площади~~

int mixtureTotal – расход раствора на данном варианте

OpType opType – тип операции (взять и положить

int StoneSquare – площадь текущего камня взять и обрезать

{} взять и повернуть)

Алгоритм будет выглядеть примерно так curr = new Node(), prev = null

~~StoneCount~~ > 0) // StoneCount – количество камней

p1 = GetForPlacing() // получаем Node оставшихся на участке

для варианта (взять и положить)

p2 = GetForTurning() // и информацию о камне

получаем Node

p3 = GetForCutting(); // получаем Node для варианта (взять, повернуть и положить)

для варианта (взять, обрезать и положить)

Node GetForCurrent() Node, StoneCount) curr Node, StoneCount)

if StoneCount == 0 => return curr Node

p1 = GetForPlacing() ; // CreateNodeForPlacing

p2 = GetForTurning()

p3 = GetForCutting(), → curr = p1

node1 = GetForCurrent StoneCount (p1, curr, StoneCount)

node2 = GetForCurrent (p2) StoneCount) (p2, curr, StoneCount)

node3 = GetForCurrent (p3) StoneCount) (p3, curr, StoneCount)

return resNode = new[] {node1, node2, node3} MinBy(x => x.mixtureTotal)

Линия отреза

Бланк ответов

В конце работы алгоритма получим экземпляр Node, для которого нужно сделать backtrack'у с целью получения всего пути, т.е всех плиток и операций над ними

```
GetResult(Node)
    while (Node prev ≠ null):
        List<Node> result
        while (Node prev ≠ null)
        result add (Node prev)
        Node = Node prev

    return result
```

Тогда в методе main можно написать.

```
var start = new Node
var Node = GetForCurrent(null, storeCountA)
var result = GetResult(Node)
```