Тестовое задание на вакансию Backend разработчик (GoLang)

Для представленного кейса-примера кода нужно провести ревью (review) кода. Ревью должно покрывать вопросы корректности, применимости, эффективности и читаемости кода. Приветствуются все замечания, даже самые мелкие. В результате мы ожидаем получить от вас перечисление мест кода с развернутым описанием проблем и путей их решения. В случае наличия нескольких путей решения, приветствуется их сравнение и объяснение в каких случаях какие варианты лучше.

Ниже для кейса также перечислены некоторые вводные данные.

Результат ревью можете оформить в удобном вам формате.

Также ниже перечислены дополнительные вопросы к кейсу, по которым нам интересно узнать ваши мысли. Включите их в результаты ревью (отдельно оформлять не нужно).

Case 1 - script

Вводные

Мы имеем дело с многоугольниками на плоскости. Каждая точка этих многоугольников имеет целые координаты и вещественный связанный с точкой вес. Весом многоугольника будем называть сумму весов его точек. Нужно разработать скрипт, который получит заданное число многоугольников с удаленного сервера, а затем определит для них:

1. минимальный по площади bounding box, в который они все поместятся;
2. вес самого тяжелого многоугольника;
3. список многоугольников, имеющих вес не меньше 100.

Bounding box - это прямоугольник, стороны которого параллельны осям координат.

Все параметры скрипт должен получить с помощью командной строки.

Получение многоугольника для обработки происходит с помощью HTTP GET запроса по заданному адресу. Ответ запроса имеет следующий JSON формат:

```
{
  "points": [ # список точек многоугольника
    {"x": 0, "y": 10, "weight": 100}, # точка многоугольника
    {"x": 10, "y": 10, "weight": 2},
    {"x": 10, "y": 0, "weight": 3},
    {"x": 0, "y": 0, "weight": 4}
  ]
}
```

Результаты скрипт должен вывести в стандартный поток вывода в JSON

формате:

```
{
  "bbox": { # минимальный по площади bounding box
    "x1": 0,
    "y1": 0,
    "x2": 10,
    "y2": 10
  },
  "max_weight": 109, # вес самого тяжелого многоугольника
  "heavy_polygons": [ # список тяжелый многоугольников
      "points": [ # список точек многоугольника
        {"x": 0, "y": 10, "weight": 100}, # точка многоугольника
        {"x": 10, "y": 10, "weight": 2},
        {"x": 10, "y": 0, "weight": 3},
        {"x": 0, "y": 0, "weight": 4}
      ]
  ]
}
```

Время работы скрипта должно быть ограничено. Если обработка не завершилась за заданное время, она должна быть прервана. Максимальное время работы задается через аргументы командной строки.

Дополнительные вопросы

1. Если ли бы вы делали скрипт с нуля, как бы вы организовали процесс обработки многоугольников? Если вам на ум приходит несколько вариантов, можете описать все, которые кажутся достойными рассмотрения, и сравнить их плюсы и минусы.