**­­Задание**

Это задание — итоговый проект пятого спринта. Вы будете сдавать его на проверку через репозиторий на GitHub. Не забудьте сохранить верное решение.

Расширьте поисковый сервер, добавив в него дополнительные методы.

1. Откажитесь от метода GetDocumentId(int index) и вместо него определите методы begin и end. Они вернут итераторы. Итератор даст доступ к id всех документов, хранящихся в поисковом сервере. Вы можете не разрабатывать собственный итератор, а применить готовый константный итератор удобного контейнера.

Если begin и end определены корректно, появится возможность использовать упрощённую форму for с поисковым сервером:

Скопировать кодCPP

SearchServer search\_server;

for (const int document\_id : search\_server) {

*// ...*

}

1. Разработайте метод получения частот слов по id документа:

Скопировать кодCPP

const map<string, double>& GetWordFrequencies(int document\_id) const;

Если документа не существует, возвратите ссылку на пустой map.

1. Разработайте метод удаления документов из поискового сервера

Скопировать кодCPP

void RemoveDocument(int document\_id);

it must be removed from 3 collections: word\_to\_document\_freqs\_, documents\_, document\_ids\_;

1. word\_to\_document\_freqs\_

iterate through all words, get a value for each key word – a map

check if that map contains the doc\_id with lower\_bound

if yes, delete the such map entry for the with doc\_id key

1. documents\_

find a doc\_id in the map and erase it, lower bound

1. document\_ids\_

find a doc\_id in vector and erase it, find

1. Вне класса сервера разработайте функцию поиска и удаления дубликатов:

Скопировать кодCPP

void RemoveDuplicates(SearchServer& search\_server);

1. Iterate through doc\_ids with begin-end
2. Arrange a vector<int> for doc\_ids that need to be removed
3. Arrange 2 cycles:

* Outer cycle for the smaller ID and Inner cycle for bigger ID
* In outer cycle we check that small doc\_id is in the (b) vector
* Inner cycle starts with iterator of outer cycle’ next()
* In inner cycle we check if big doc\_id is in the (b) vector
* In inner cycle, use GetWordFrequencies for both iterators
* Compare to be equal, if yes, add big doc\_id to (b), else continue

1. When 2 cycles are finished, go through the (b) vector and use RemoveDocument for each doc\_id.

Дубликатами считаются документы, у которых наборы встречающихся слов совпадают. Совпадение частот необязательно. Порядок слов неважен, а стоп-слова игнорируются. Функция должна использовать только доступные к этому моменту методы поискового сервера.

При обнаружении дублирующихся документов функция должна удалить документ с большим id из поискового сервера, и при этом сообщить id удалённого документа в соответствии с форматом выходных данных, приведённым ниже.

Будьте аккуратны, если функция RemoveDuplicates проходит циклом по поисковому серверу так:

Скопировать кодCPP

void RemoveDuplicates(SearchServer& search\_server) {

...

for (const int document\_id : search\_server) {

...

}

...

}

В подобном случае удалять документы внутри цикла нельзя — это может привести к невалидности внутреннего итератора.

Все реализации должны быть эффективными. Если N*N* — общее количество документов, а W*W* — количество слов во всех документах, то:

* сложность GetWordFrequencies должна быть O(\log N)*O*(log*N*);
* сложность RemoveDocument должна быть O(w(\log N+\log W))*O*(*w*(log*N*+log*W*)), где w*w* — количество слов в удаляемом документе;
* сложность begin и end — O(1)*O*(1);
* сложность RemoveDuplicates должна быть O(wN(\log N+\log W))*O*(*wN*(log*N*+log*W*)), где w*w* — максимальное количество слов в документе.

В этом задании может потребоваться рефакторинг вашего кода. Например, замена одного вида контейнера на другой или введение нового индекса.

Вам предстоит оценить сложность разрабатываемых алгоритмов, чтобы знать, что они достаточно быстрые.

**Формат выходных данных**

Функция RemoveDuplicates должна для каждого удаляемого документа вывести в cout сообщение в формате Found duplicate document id N, где вместо N следует подставить id удаляемого документа.

**Ограничения**

Сохраните корректную и быструю работу всех методов кроме GetDocumentId.

**Что отправлять на проверку**

Загрузите полный код поискового сервера и вспомогательных функций. Код нужно разбить на файлы. Функция main при проверке будет игнорироваться.

**Как будет тестироваться ваш код**

Будет проверено, что:

* вы не изменили работу методов и функций, которые не описаны в условии,
* вы реализовали требуемые методы и функцию RemoveDuplicates,
* реализованные методы работают достаточно быстро и их сложность соответствует условию.

**Пример использования**

Скопировать кодCPP

int main() {

SearchServer search\_server("and with"s);

AddDocument(search\_server, 1, "funny pet and nasty rat"s, DocumentStatus::ACTUAL, {7, 2, 7});

AddDocument(search\_server, 2, "funny pet with curly hair"s, DocumentStatus::ACTUAL, {1, 2});

*// дубликат документа 2, будет удалён*

AddDocument(search\_server, 3, "funny pet with curly hair"s, DocumentStatus::ACTUAL, {1, 2});

*// отличие только в стоп-словах, считаем дубликатом*

AddDocument(search\_server, 4, "funny pet and curly hair"s, DocumentStatus::ACTUAL, {1, 2});

*// множество слов такое же, считаем дубликатом документа 1*

AddDocument(search\_server, 5, "funny funny pet and nasty nasty rat"s, DocumentStatus::ACTUAL, {1, 2});

*// добавились новые слова, дубликатом не является*

AddDocument(search\_server, 6, "funny pet and not very nasty rat"s, DocumentStatus::ACTUAL, {1, 2});

*// множество слов такое же, как в id 6, несмотря на другой порядок, считаем дубликатом*

AddDocument(search\_server, 7, "very nasty rat and not very funny pet"s, DocumentStatus::ACTUAL, {1, 2});

*// есть не все слова, не является дубликатом*

AddDocument(search\_server, 8, "pet with rat and rat and rat"s, DocumentStatus::ACTUAL, {1, 2});

*// слова из разных документов, не является дубликатом*

AddDocument(search\_server, 9, "nasty rat with curly hair"s, DocumentStatus::ACTUAL, {1, 2});

cout << "Before duplicates removed: "s << search\_server.GetDocumentCount() << endl;

RemoveDuplicates(search\_server);

cout << "After duplicates removed: "s << search\_server.GetDocumentCount() << endl;

}

Ожидаемый вывод этой программы:

Скопировать кодCPP

Before duplicates removed: 9

Found duplicate document id 3

Found duplicate document id 4

Found duplicate document id 5

Found duplicate document id 7

After duplicates removed: 5

Подсказка: