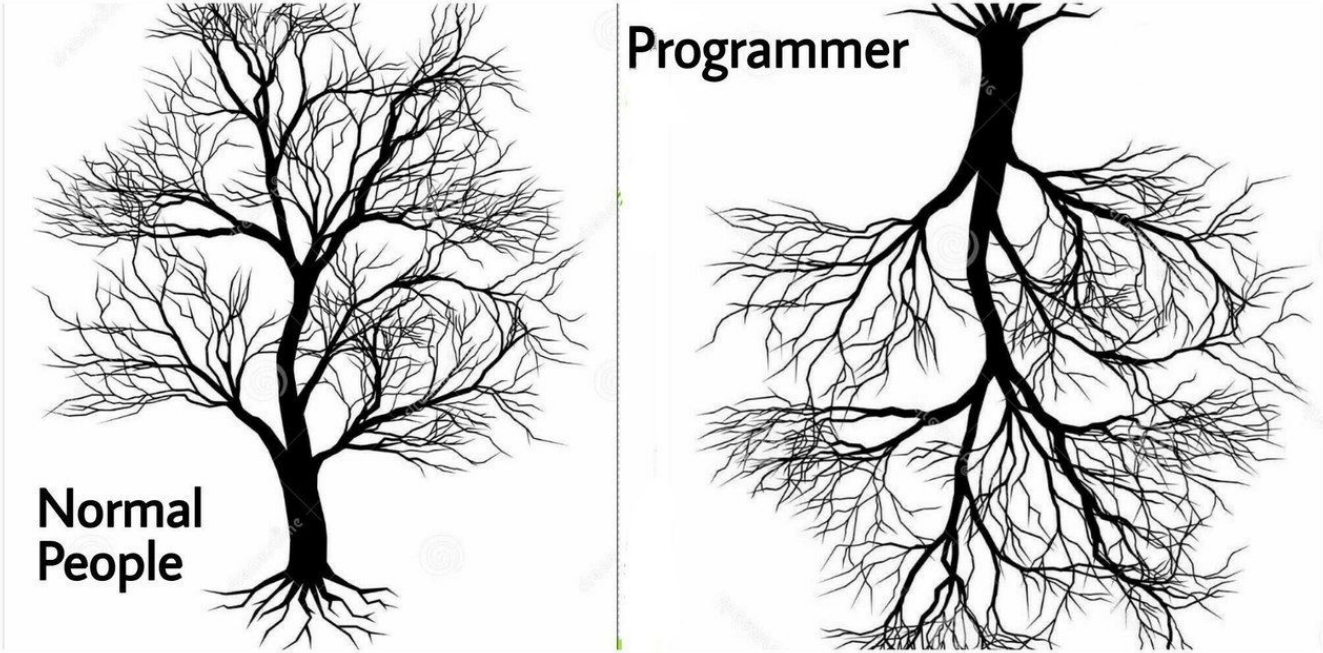
Взращиваем корень дерева отрезков



Мячин Данил, ВШЭ ФКН ПМИ

На прошлом занятии в конце была задача, где мы прибавляли ко всем элементам, и хотели потом находить i-й элемент

На дом на подумать

Задача: n чисел. Есть два вида запросов

1 - сказать чему равно i-е число. O(log(n))

2 - на отрезке [l;r] прибавить ко всем числам число k. O(log(n))

Нам нужно изменить апдейт и гет нашего дерева, и давайте вот какие мысли подумаем

Нам нужно изменить апдейт и гет нашего дерева, и давайте вот какие мысли подумаем

Мы могли бы делать апдейт и спускаться до всех вершин, которые есть в запросе [l;r], но тогда обновление будет работать за n logn, а n обновлений за n^2 logn

Нам нужно изменить апдейт и гет нашего дерева, и давайте вот какие мысли подумаем

Мы могли бы делать апдейт и спускаться до всех вершин, которые есть в запросе [l;r], но тогда обновление будет работать за n logn, а n обновлений за n^2 logn

Когда мы будем делать get(i), мы будем спускаться по всем отрезкам, в которые входит вершина i

Нам нужно изменить апдейт и гет нашего дерева, и давайте вот какие мысли подумаем

Мы могли бы делать апдейт и спускаться до всех вершин, которые есть в запросе [l;r], но тогда обновление будет работать за n logn, а n обновлений за n^2 logn

Когда мы будем делать get(i), мы будем спускаться по всем отрезкам, в которые входит вершина i

Когда мы делали какие-то операции (а конкретно речь про апдейт) мы не спускались, если отрезок дерева и запрос совпадают

Давайте нарисую дерево

Тогда навеивается идея

Тогда навеивается идея

При get мы будем складывать все значения, идущие по пути

Нам осталось совсем немного до того, чтобы апдейтить на отрезке

В нашем апдейте мы не делаем кое-какой вещи, которую раньше делали, причём это было очень важно

Нам осталось совсем немного до того, чтобы апдейтить на отрезке

В нашем апдейте мы не делаем кое-какой вещи, которую раньше делали, причём это было очень важно

Что мы раньше делали, когда обновили поддеревья слева и справа?

Нам осталось совсем немного до того, чтобы апдейтить на отрезке

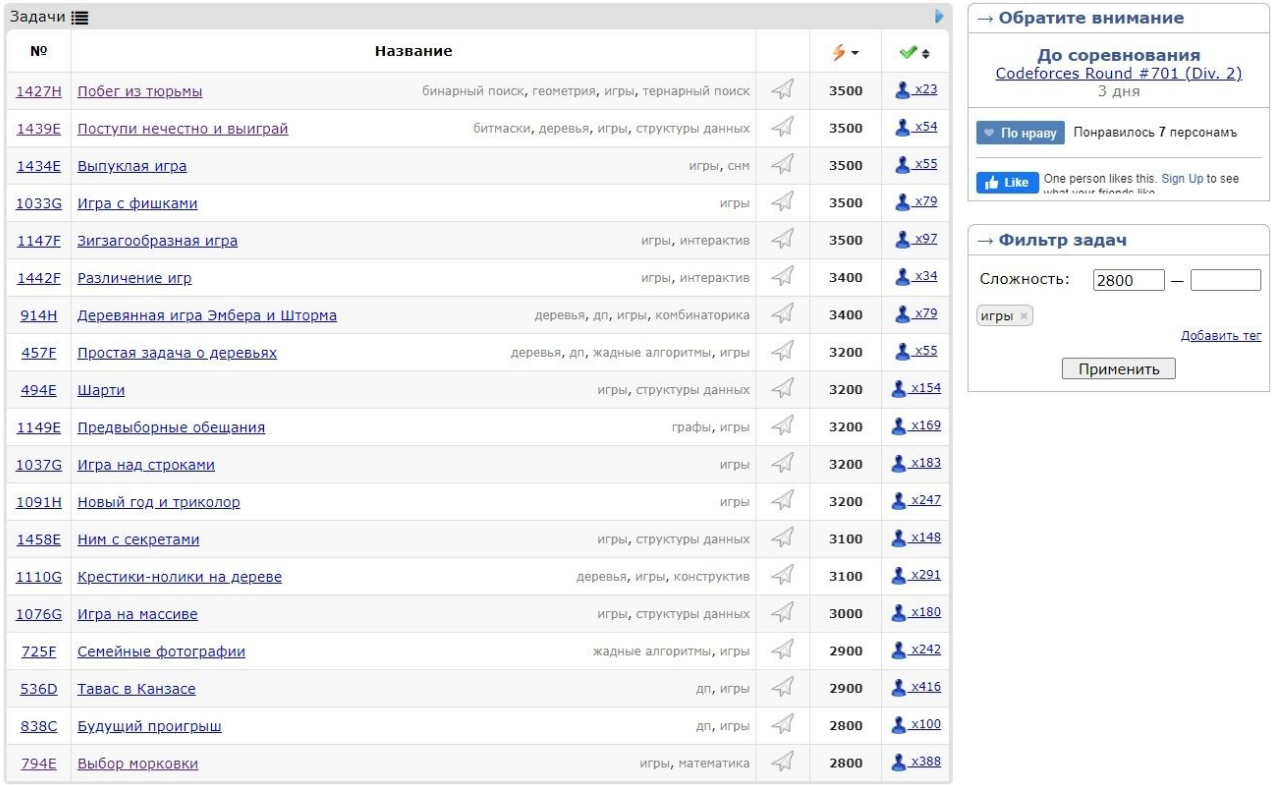
В нашем апдейте мы не делаем кое-какой вещи, которую раньше делали, причём это было очень важно

Что мы раньше делали, когда обновили поддеревья слева и справа?

Мы обновляли результат для их предка, так давайте снова это делать?

Вот мы научились делать базовые операции и увидели базовые задачи на ДОшки

Минута мемов о том, какие игры у программистов в стиме



Задачи

№	Название			
1427H	Побег из тюрьмы	бинарный поиск, геометрия, игры, тернарный поиск	3500	x23
1439E	Поступи нечестно и выиграй	битмаски, деревья, игры, структуры данных	3500	x54
1434E	Выпуклая игра	игры, снм	3500	x55
1033G	Игра с фишками	игры	3500	x79
1147F	Зигзагообразная игра	игры, интерактив	3500	x97
1442F	Различение игр	игры, интерактив	3400	x34
914H	Деревянная игра Эмбера и Шторма	деревья, дп, игры, комбинаторика	3400	x79
457F	Простая задача о деревьях	деревья, дп, жадные алгоритмы, игры	3200	x55
494E	Шарти	игры, структуры данных	3200	x154
1149E	Предвыборные обещания	графы, игры	3200	x169
1037G	Игра над строками	игры	3200	x183
1091H	Новый год и триколор	игры	3200	x247
1458E	Ним с секретами	игры, структуры данных	3100	x148
1110G	Крестики-нолики на дереве	деревья, игры, конструктив	3100	x291
1076G	Игра на массиве	игры, структуры данных	3000	x180
725F	Семейные фотографии	жадные алгоритмы, игры	2900	x242
536D	Тавас в Канзасе	дп, игры	2900	x416
838C	Будущий проигрыш	дп, игры	2800	x100
794E	Выбор морковки	игры, математика	2800	x388

→ Обратите внимание

До соревнования
Codeforces Round #701 (Div. 2)
3 дня

По нраву Понравилось 7 персонамъ

Like One person likes this. Sign Up to see what your friends like

→ Фильтр задач

Сложность: 2800 —

игры ×

Добавить тег

Применить

Тема, которая пригодится нам в будущем в алгоритмах

СНМ - система непересекающихся множеств

CHM - система непересекающихся множеств

Казалось бы, у нас есть set, unordered_set, зачем нам эта штука?

CHM - система непересекающихся множеств

Казалось бы, у нас есть set, unordered_set, зачем нам эта штука?

А затем, что у неё другие интерфейсы

CHM - система непересекающихся множеств

Казалось бы, у нас есть set, unordered_set, зачем нам эта штука?

А затем, что у неё другие интерфейсы

Что нам нужно уметь делать:

СНМ - система непересекающихся множеств

Казалось бы, у нас есть set, unordered_set, зачем нам эта штука?

А затем, что у неё другие интерфейсы

Что нам нужно уметь делать:

- get(a) - узнать, в каком множестве лежит элемент a (какое множество - пока для нас абстрактное понятие, но это нужно в будущем для написания более сложных штук)

СНМ - система непересекающихся множеств

Казалось бы, у нас есть set, unordered_set, зачем нам эта штука?

А затем, что у неё другие интерфейсы

Что нам нужно уметь делать:

- get(a) - узнать, в каком множестве лежит элемент a (какое множество - пока для нас абстрактное понятие, но это нужно в будущем для написания более сложных штук)
- union(a, b) - объединить те множества, в которых содержатся элементы a и b

СНМ - система непересекающихся множеств

Казалось бы, унас есть set, unordered_set, зачем нам эта штука?

А затем, что у неё другие интерфейсы

Что нам нужно уметь делать:

- get(a) - узнать, в каком множестве лежит элемент a (какое множество - пока для нас абстрактное понятие, но это нужно в будущем для написания более сложных штук)
- union(a, b) - объединить те множества, в которых содержатся элементы a и b

И вот уже зачем нам нужно делать это не на set’ах

Если с тем, чтобы определиться, в каком множестве элемент - мы могли бы просто хранить массив (или unordered_map, если элементы это не числа от 0 до n-1) и жизнь была бы сказкой

И вот уже зачем нам нужно делать это не на set’ах

Если с тем, чтобы определиться, в каком множестве элемент - мы могли бы просто хранить массив (или unordered_map, если элементы это не числа от 0 до n-1) и жизнь была бы сказкой

Однако, тогда чтобы объединить два множества, нам потребовалось бы времени O(длины множества)

И вот уже зачем нам нужно делать это не на set’ах

Если с тем, чтобы определиться, в каком множестве элемент - мы могли бы просто хранить массив (или unordered_map, если элементы это не числа от 0 до n-1) и жизнь была бы сказкой

Однако, тогда чтобы объединить два множества, нам потребовалось бы времени O(длины множества)

Небольшой Hint: когда мы сливаем два множества, нам легче меньше присоединить к большему

Тогда вот какая мысль

Тогда вот какая мысль

Давайте в каждом множестве заведём “главный элемент”, и он будет ответом на get

Тогда вот какая мысль

Давайте в каждом множестве заведём “главный элемент”, и он будет ответом на get

Каждый элемент будет указывать на тот, который указывает в направлении главного элемента (давайте нарисую)

Давайте подумаем, как это будет по асимптотике