Это набор задач, решение которых будет обсуждаться на спецкурсе «Потоковые алгоритмы». Часть из них может показаться вам знакомой, это нормально. Вы можете заранее попробовать самостоятельно порешать эти задачи. Желающие смогут рассказать свои варианты решений во время спецкурса.

Важное замечание: совершенно не обязательно решать их заранее для того, чтобы прийти на спецкурс. Даже если вы ни разу не взглянете на них, вы все равно можете прийти. Не стоит жертвовать ради этих задач кучей своего времени, в ЛКШ и без этого происходит много интересного :)

Задачи выложены на всеобщее обозрение для повышения интерактивности спецкурса и просто для интереса, не более того.

Кроме того, обсуждение решений с друзьями и даже преподавателями не осуждается и даже вполне приветствуется.

Некоторые слова могут быть вам непонятны и незнакомы – это нормально. Можно спросить у кого-нибудь, что они значат, а можно и не спрашивать, во время спецкурса будет объяснено, что это все такое.

Наконец, решать задачи по порядку не имеет смысла, их сложность может повышаться внутри каждой из шести серий, последние две-три задачи каждой серии могут оказаться очень сложными. Можно пробовать сначала решать только первые-вторые задачи всех серий, и только потом остальные. А вообще надо идти в том порядке, в котором комфортнее и интереснее.

Good luck and have fun!

Задачи по потоковым алгоритмам

Во всех задачах по умолчанию подразумевается, что у нас мало памяти (в частности, мы не можем сохранить все входные данные в массив). Считается, что мы можем использовать $\mathcal{O}(1)$ переменных, если не оговорено иное. При этом мы полагаем, что все входные числа и счетчики циклов помещаются в какой-нибудь стандартный целочисленный тип данных.

Далее, мы можем сделать только один проход по входым данным, то есть каждое число на входе можно считать, как-то обработать, а дальше вернуться к нему будет уже нельзя.

Будем считать, что во всех задачах нам дано m чисел, каждое из промежутка $[1 \dots n]$.

I. **Поиск пропущенных чисел.**

1. Из набора $[1 \dots n]$ выкинули одно число, остальные перемешали. Найти выкинутое число.
2. Теперь пусть выкинули числа $[l \dots r]$, найти l и r.
3. Выкинули два каких-то числа. Найти их, используя $\mathcal{O}(\log n)$ переменных.
4. Выкинули два числа, найти, используя $\mathcal{O}(1)$ переменных.
5. Выкинули k чисел, найти их, используя $\mathcal{O}((k \log n)^5)$ переменных (в реальности, конечно, будет меньше, чем пятая степень, но лишь бы не больше).
6. Выкинули k чисел, найти, используя $\mathcal{O}(k)$ переменных. Решить, полагая, что число n^{5k} влезает в одну переменную (снова пятерка тут ничего не значит, просто с запасом).
7. Если считать, что в переменную помещаются только числа порядка $\mathcal{O}(n)$, то число n^k займет $\mathcal{O}(k)$ переменных. Если использовать в решении прошлого пункта большие числа, то на самом деле памяти может быть потрачено $\mathcal{O}(k^2)$ (хотя может и не быть). Придумать, как сделать так, чтобы было все-таки $\mathcal{O}(k)$ памяти.

II. **Поиск чисел, встречающихся нечетное количество раз.**
1. Число x встречается во входном наборе ровно один раз, все остальные ровно два раза (либо ни разу). Найти x. Например, для $[4, 2, 8, 4, 1, 1, 2]$ ответ 8.
2. x встречается нечетное количество раз, все остальные – четное. Найти x.
3. Два каких-то числа встречаются нечетное количество раз, остальные – четное. Найти те, которые встречаются нечетное количество раз. $\mathcal{O}(\log n)$ памяти.
4. k чисел вместо двух, найти их, используя $\mathcal{O}((k \log n)^5)$ памяти.

III. **Эйлеровы пути.**
1. Неориентированный нраф задан списком ребер. Известно, что он содержит эйлеров путь. Найти его концы. $\mathcal{O}(\log n)$ памяти.
2. То же для ориентированного графа, $\mathcal{O}(1)$ памяти.
3. Задан списком ребер ориентированный граф, который можно разбить на k путей, которые не пересекаются по ребрам, но могут пересекаться по вершинам. Найти начала и концы путей, используя $\mathcal{O}((k \log n)^5)$ памяти.
4. То же, $\mathcal{O}(k)$ памяти в предположении, что n^{5k} вмещается в одну переменную.
5. То же, в переменную вмещаются только числа порядка $\mathcal{O}(n)$ (то есть тут те же варианты ограничений, что и в серии I).

IV. **Поиск частых элементов.**
1. Некоторый элемент повторяется строго больше, чем $m/2$ раз. Найти его.
2. Есть $k - 1$ элемент, каждый из которых встречается строго больше m/k раз, найти их, $\mathcal{O}(k)$ памяти.
3. Доказать, что требуется не менее n битов памяти, чтобы уметь проверять за один проход, существует ли элемент, встречающийся строго больше, чем $m/2$ раз.

V. **Графовые задачи.** В этих задачах требуется найти решение, использующее $\mathcal{O}(n)$ памяти, где n – число вершин. Заметим, что оно может быть существенно меньше, чем m – число ребер.
1. Проверить граф на связность.
2. Проверить граф на двудольность.
3. Найти минимальное остовное дерево.
4. Найти все мосты (мост – ребро, удаление которого разрывает одну из компонент связности графа).

VI. Найти k-ю порядковую статистику, то есть число, которое стояло бы на k позиции, если бы мы отсортировали входной набор чисел. Использовать $\mathcal{O}(1)$ переменных, но можно делать несколько проходов (несколько раз считать все входные данные, но сохранить их, как всегда, нельзя) и использовать случайные числа. При этом нужно, чтобы алгоритм делал в среднем $\mathcal{O}(\log m)$ проходов.