Задача Barcelona. Медианный горный хребет

Имя входного файла:	`input.txt` или стандартный поток ввода
Имя выходного файла:	`output.txt` или стандартный поток вывода
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Берляндия — огромная страна с разнообразной географией. Одной из самых знаменитых природных достопримечательностей Берляндии является «Медианный горный хребет». Этот горный хребет представляет из себя n подряд идущих горных вершин, расположенных на одной прямой, пронумерованных в порядке следования от 1 до n. Высота i-й горной вершины равна a_i.

«Медианный горный хребет» знаменит тем, что с ним ежедневно происходит так называемое *выравнивание горных вершин*. В момент выравнивания одновременно для каждой вершины от 2 до $n-1$ её высота становится равна медианной высоте среди неё и двух соседних гор. А именно, если до выравнивания высоты были равны b_i, то новые высоты a_i устроены следующим образом: $a_1 = b_1$, $a_n = b_n$, а для всех i от 2 до $n-1$ $a_i = \texttt{median}(b_{i-1}, b_i, b_{i+1})$. Медианой трёх чисел называется второе по счёту число, если отсортировать эти три числа по возрастанию. Например, $\texttt{median}(5,1,2) = 2$, а $\texttt{median}(4,2,4) = 4$.

Недавно Берляндские учёные доказали, что какими бы ни были высоты гор, процесс выравнивания рано или поздно стабилизируется, то есть в какой-то момент высоты гор перестанут изменяться после выравнивания. Правительство Берляндии хочет понять через сколько лет это произойдёт, то есть, найти величину c — сколько произойдет выравниваний, при которых у хотя бы одной горы изменится её высота. Помогите ученым решить эту важную задачу!

Обратите внимание, что в некоторых группах тестов помимо значения c вам необходимо определить также и высоты гор после c выравниваний, то есть узнать, какими высоты гор останутся навсегда.

Формат входных данных

Первая строка содержит целые числа n и t ($1 \leqslant n \leqslant 500\,000$, $0 \leqslant t \leqslant 1$) — количество гор и параметр, который определяет, необходимо ли определить итоговые высоты гор.

Вторая строка содержит целые числа $a_1, a_2, a_3, \ldots, a_n$ ($1 \leqslant a_i \leqslant 10^9$) — текущие высоты гор.

Формат выходных данных

В первой строке выведите c — число выравниваний вершин, при которых высота хотя бы одной горы изменится.

Если $t = 1$, то во второй строке выведите n чисел — итоговые высоты гор после c выравниваний.

Примеры

ввод	вывод
5 1 1 2 1 2 1	2 1 1 1 1 1
6 1 1 3 2 5 4 6	1 1 2 3 4 5 6
6 0 1 1 2 2 1 1	0

Пояснение

В первом примере высоты на позициях 1 и 5 не меняются. Так как медиана чисел 1, 2, 1 это 1, то на позициях 2 и 4 после первого выравнивания оказываются числа 1, и так как медиана чисел 2, 1, 2 это 2, то на позиции 3 после первого выравнивания оказывается число 2. Итого после первого выравнивания горных вершин горы имеют высоты 1, 1, 2, 1, 1. После второго выравнивания высоты становятся 1, 1, 1, 1, 1 и дальше они меняться не будут, соответственно всего было 2 меняющих высоты выравнивания.

В третем примере после выравнивания ни у одной горы её высота не изменится и число выравниваний, при которых высоты изменятся, равно 0. Так как $t = 0$, то выводить итоговые высоты гор не нужно.

Система оценки

Тесты к этой задаче состоят из шести групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов **необходимых** групп. **Offline-проверка** означает, что результаты тестирования вашего решения на данной группе станут доступны только после окончания соревнования.

Группа	Баллы	Дополнительные ограничения			Необх. группы	Комментарий
		n	a_i	t		
0	0	–	–	–	–	Тесты из условия.
1	19	$n \leqslant 1000$	–	–	0	Гарантируется, что $c \leqslant 10\,000$.
2	24	–	$a_i \leqslant 2$	–	–	
3	14	–	$a_i \leqslant 100$	$t = 0$	–	
4	14	–	$a_i \leqslant 100$	–	0, 2, 3	
5	14	–	–	$t = 0$	3	
6	15	–	–	–	0, 1, 2, 3, 4, 5	**Offline-проверка.**

Задача Wimbledon. Интересные конкурсы

Имя входного файла:	`input.txt` или стандартный поток ввода
Имя выходного файла:	`output.txt` или стандартный поток вывода
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Учительница дала классу, в котором учится Дмитрий, очень странное задание — она попросила каждого из учеников придумать последовательность произвольной длины, состоящую только из открывающих и закрывающих скобок. После этого все ученики по очереди называли придуманные ими последовательности. Когда очередь дошла до Димы, он внезапно понял, что у всех его одноклассников получились правильные скобочные последовательности, а получилась ли у него правильная скобочная последовательность, он не знал.

Дима подозревает, что он просто прослушал слово «правильная» в постановке задания, поэтому хочет срочно исправить ситуацию — для этого он решил немного изменить свою последовательность. А именно, он решил произвольное число раз (возможно, нулевое) выполнить операцию *перемешивания*. Для выполнения одной операции перемешивания Дима выбирает произвольный подотрезок последовательности и произвольным образом переставляет все символы на нём. Такая операция занимает ровно l наносекунд, где l — длина подотрезка, который Дима выбрал для перемешивания. Легко заметить, что при этом число открывающих скобок не меняется, равно как и число закрывающих.

Уже совсем скоро подойдёт очередь Димы называть свою последовательность, поэтому он обратился за помощью к вам. Помогите ему определить минимальное время, за которое он может сделать свою последовательность правильной, или определите, что сделать это невозможно.

Формат входных данных

В первой строке содержится одно целое число n ($1 \leqslant n \leqslant 10^6$) — длина последовательности Димы.

Вторая строка содержит строку длины n, состоящую только из символов «`(`» и «`)`».

Формат выходных данных

Выведите единственное число — минимальное количество наносекунд, необходимое, чтобы сделать последовательность правильной, или «`-1`», если сделать последовательность правильной с помощью операций перемешивания невозможно.

Примеры

ввод	вывод
8))((()((	6
3 (()	-1

Пояснение

Напомним, что скобочная последовательность называется правильной, если путем вставки в неё символов «+» и «1» можно получить из неё корректное математическое выражение. Например, последовательности «`(()()`», «`()`» и «`((()(()))`» — правильные, в то время как «`)(`», «`(()`» и «`(()))(`» — нет.

В первом примере можно сначала перемешать подотрезок с первого по четвёртый символ, заменив его на «`()()`» — получится последовательность «`()()())(`», а затем перемешать подотрезок с седьмого по восьмой символ, заменив его на «`()`». В итоге получится правильная скобочная последовательность «`()()()()`», такие действия займут суммарно $4 + 2 = 6$ наносекунд.

Система оценки

Тесты к этой задаче состоят из двух групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов **предыдущих** групп.

Группа	Баллы	Дополнительные ограничения	Комментарий
		n	
0	0	–	Тесты из условия.
1	50	$n \leqslant 1000$	
2	50	–	

Задача Zermatt. Разработка тарифов

Имя входного файла:	`input.txt` или стандартный поток ввода
Имя выходного файла:	`output.txt` или стандартный поток вывода
Ограничение по времени:	6 секунд
Ограничение по памяти:	512 мегабайт

Мэр города М. решил в 2020 году запустить несколько новых веток метро. Поскольку бюджет города сильно ограничен, он принял решение не копать новые туннели, а воспользоваться уже существующей подземной сетью.

Туннельная система города М. состоит из n станций метро. Станции соединены между собой двунаправленными туннелями, которых всего $n - 1$. Между любыми станциями v и u есть ровно один простой путь. Каждая ветка метро, которую хочет создать мэр, является простым путём от станции a_i до станции b_i. Ветки могут пересекаться, то есть иметь общие станции или туннели. Однако, ещё не решено, в какую из двух сторон будут следовать поезда на каждой из веток. А именно, на пути между станциями a_i и b_i поезда будут следовать либо от станции a_i к станции b_i, либо от станции b_i к станции a_i, но только в одну из сторон.

В городе M действует особая система тарифов. Каждой станции присваивается целое положительное число c_i — тарифная зона станции, а стоимость переезда от станции v до станции u определяется как $c_u - c_v$ бурлей. Разумеется, такой переезд разрешен, только если есть ветка метро, по которой поезда идут из v в u. Мэр города не хочет, чтобы между какими-то двумя станциями на одной ветке стоимость проезда была отрицательной, поэтому было принято решение выбрать направления движения поездов по веткам и изменить тарифные зоны всех станций города таким образом, чтобы на каждой ветке тарифные зоны станций строго возрастали, если рассмотреть её в сторону движения поездов.

Мэр сначала хочет присвоить каждой станции тарифную зону, а потом выбрать направления всех веток метро, чтобы вдоль каждой ветки тарифные зоны строго возрастали. В связи со скорым наступлением дня города, из всех возможных вариантов назначения тарифных зон мэр хочет выбрать то, где максимальное c_i будет как можно меньше. Помогите мэру составить новое распределение, или скажите, что это невозможно. Обратите внимание, от вас требуется только назначить тарифные зоны оптимальным образом, а направления для веток выводить не требуется. Таким образом, ваше решение считается верным, если существует способ выбрать направление следования поездов на каждой ветке так, чтобы вдоль всех веток тарифные зоны строго возрастали.

Обратите внимание, что в некоторых группах тестов минимизировать ответ не нужно, а требуется лишь определить, возможно ли назначить тарифные зоны нужным образом.

Формат входных данных

Первая строка содержит целое число n, m, t ($2 \leqslant n, \leqslant 500\,000,\ 1 \leqslant m \leqslant 500\,000,\ 0 \leqslant t \leqslant 1$) — количество станций в городе, количество веток метро, а также параметр теста t. Если $t = 0$, то минимизировать ответ не обязательно. Если $t = 1$, то от вас требуется минимизировать номер тарифной зоны самой дорогой станции.

Каждая из следующих $n - 1$ строк описывает очередной туннель метро. Каждый туннель задаётся целыми числами v_i, u_i ($1 \leqslant v_i,\ u_i \leqslant n$, $v_i \neq u_i$). Гарантируется, что между любыми двумя станциями есть ровно один простой путь.

Каждая из следующих m строк описывает очередную ветку метро. Каждая ветка задаётся целыми числами a_i, b_i ($1 \leqslant a_i,\ b_i \leqslant n$, $a_i \neq b_i$).

Формат выходных данных

В первой строке выведите целое число k — максимальный номер тарифной зоны Если параметр теста t равен 0, то от вас не требуется минимизировать k. Если $t = 1$, то k должно быть минимально возможным.

В следующей строке выведите числа $c_1, c_2, \ldots, c_n$ ($1 \leqslant c_i \leqslant k$) — тарифные зоны станций.

Если существует несколько ответов, выведите любой из них. Если же не существует ни одного способа назначить тарифные зоны, выведите «`-1`».

Примеры

ввод	вывод
3 1 1 1 2 2 3 1 3	3 1 2 3
4 3 0 1 2 1 3 1 4 2 3 2 4 3 4	-1

Пояснение

В первом примере ветка $1 \to 3$ проходит по станциям в порядке 1, 2, 3. При таком порядке посещения станций их тарифные зоны возрастают. Поскольку на этой ветке 3 станции, то нам потребуется как минимум 3 тарифные зоны. Таким образом, ответ 1, 2, 3 оптимален.

Во втором примере ни одно распределение тарифных зон не согласуется с ветками метро.

Система оценки

Тесты к этой задаче состоят из десяти групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов **необходимых** групп. **Offline-проверка** означает, что результаты тестирования вашего решения на данной группе станут доступны только после окончания соревнования.

В группах 4 и 5 не существует туннеля, принадлежащего двум веткам метро.

Длиной ветки назовем количество туннелей, которые ей принадлежат.

Группа	Баллы	Дополнительные ограничения		Необх. группы	Комментарий
		n, m	t		
0	0	–	–	–	Тесты из условия.
1	6	$n,\ m \leqslant 8$	–	0	
2	10	$n,\ m \leqslant 15$	–	0, 1	
3	15	$n,\ m \leqslant 100$	–	0, 1, 2	
4	11	$n,\ m \leqslant 100\,000$	–	–	Нет общих туннелей.
5	10	–	–	4	Нет общих туннелей.
6	8	–	–	–	У всех туннелей есть общий конец.
7	10	$n,\ m \leqslant 100\,000$	$t = 0$	–	Суммарная длина веток до 100 000.
8	6	–	$t = 0$	7	
9	14	$n,\ m \leqslant 100\,000$	–	0, 1, 2, 3, 4, 7	
10	10	–	–	0 – 9	**Offline-проверка.**

Задача Dakar. Двойной палиндром

Имя входного файла:	`input.txt` или стандартный поток ввода
Имя выходного файла:	`output.txt` или стандартный поток вывода
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Ваня работает на заводе по изготовлению палиндромов. На заводе есть заготовка — строка s длины n, состоящая из строчных букв английского алфавита, из которой Ваня может вырезать любую подстроку на продажу. Напомним, что *палиндром* — строка, читающаяся одинаковым образом как слева направо, так и справа налево.

Обычные палиндромы всем надоели и их никто не покупает, поэтому он решил производить двойные палиндромы. *Двойной палиндром* — это строка, которая является конкатенацией двух палиндромов **равной** длины. Например, строки «`aabb`», «`aaaa`» являются двойными палиндромами, а строки «`abba`» и «`aaaabb`» нет.

Ваня задался вопросом, сколько существует способов вырезать из строки s двойной палиндром, а именно, сколько существует пар (l, r), что подстрока $s_l s_{l+1} \dots s_r$ является двойным палиндромом. Помогите Ване найти ответ на этот важный вопрос!

Формат входных данных

В первой строке задано целое число n ($1 \leqslant n \leqslant 500\,000$) — длина строки s. Во второй строке задана строка s, состоящая из строчных букв английского алфавита.

Формат выходных данных

Выведите единственное число — искомое число подстрок, которые являются двойными палиндромами.

Примеры

ввод	вывод
6 abacac	6
5 aaaaa	6

Пояснение

В первом примере двойными палиндромами являются 5 подстрок длины 2 («`ab`», «`ba`», «`ac`», «`ca`» и «`ac`»), а так же вся строка («`abacac`»).

Система оценки

Тесты к этой задаче состоят из трёх групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов **предыдущих** групп.

Группа	Баллы	Дополнительные ограничения	Комментарий
		n	
0	0	–	Тесты из условия.
1	19	$n \leqslant 500$	
2	33	$n \leqslant 5000$	
3	48	–	