Задатак C11. EAGERS

⏳0.4 sec. 💾 256 MB

Адина су недавно питали да ли би више волео да „гради мишиће или гради eagere". Знао је како би му изградња мишића помогла, али је морао да претражи интернет да би сазнао нешто о изградњи eagera.

Испоставило се да постоји низ од N целих бројева, $A_0, A_1, \ldots, A_{N-1}$, који се називају eager коефицијенти. Адин жели да подели низ A на тачно K узастопних, непразних поднизова[1]. Да би то урадио, бира $K-1$ „места раздвајања" која деле низ.

Формално, поделу можемо дефинисати избором индекса $c_1, c_2, \ldots, c_{K-1}$ тако да важи $0 < c_1 < c_2 < \cdots < c_{K-1} < N$. Тада i-ти подниз чине елементи од индекса c_{i-1} до $c_i - 1$ (где је $c_0 = 0$ и $c_K = N$).

Нека је val_i **битовско ИЛИ**[2] свих елемената у i-том поднизу. Коначна количина eagera рачуна се као **битовско И**[3] ових вредности поднизова:

$$ans = val_1 \,\&\, val_2 \,\&\, \ldots \,\&\, val_K$$

Да би заиста одговорио на питање, Адин жели да пронађе највећу могућу вредност броја ans међу свим (валидним) поделама на K поднизова. Напишите програм **eagers** који за њега израчунава ту вредност.

Улаз

У првом реду стандардног улаза задати су позитивни цели бројеви N и K. У наредном реду задато је N позитивних целих бројева $A_0, A_1, \ldots, A_{N-1}$.

Излаз

На стандардни излаз исписати један број – највећу могућу вредност броја ans међу свим могућим поделама.

Ограничења

- $1 \leq K \leq N \leq 10^6$
- $0 \leq A_i \leq 10^9$

[1] Подниз је узастопни низ елемената оригиналног низа.
На пример, [2,3] је подниз низа [1,2,3], али [1,3] није.

[2] **Битовско ИЛИ** (|) је бинарна операција која пореди два броја у бинарном запису, бит по бит. На конкретној позицији даје 0 ако су оба одговарајућа бита 0, а иначе даје 1.
На пример, $4879 \mid 4772 = 5039$.

[3] **Битовско И** (&) је бинарна операција која пореди два броја у бинарном запису, бит по бит. На конкретној позицији даје 1 ако су оба одговарајућа бита 1, а иначе даје 0.
На пример, $4879 \,\&\, 4772 = 4612$.

Подзадаци

Подзадатак	**Поени**	**Неопходни подзадаци**	N	A_i	**Остала ограничења**
0	0	—	—	—	Примери испод.
1	12	0	≤ 20	$\leq 10^9$	-
2	2	—	$\leq 10^6$	≤ 1	-
3	13	2	$\leq 10^6$	≤ 2	-
4	15	0	≤ 100	≤ 2000	-
5	11	$0, 4$	≤ 300	≤ 2000	-
6	12	$0, 4 - 5$	≤ 2000	≤ 2000	-
7	13	$0 - 1, 4 - 6$	$\leq 10^5$	$\leq 10^9$	-
8	22	$0 - 7$	$\leq 10^6$	$\leq 10^9$	-

Поени за подзадатак добијају се само ако су успешно прошли сви његови тестови и тестови неопходних подзадатака.

Примери

Улаз	**Излаз**	**Објашњење примера**
7 3 2 1 4 3 2 2 5	3	Једно од оптималних решења је да се низ подели на поднизове [2,1,4], [3,2], [2,5]. Њихове одговарајуће вредности (тј. val) су 7, 3, 7. БИТОВСКО И ова три броја јесте 3.
6 2 1 0 0 1 2 2	2	Једино оптимално решење је да се низ подели на поднизове [1,0,0,1,2] и [2]. Њихове одговарајуће вредности (тј. val) су 3 и 2, чије БИТОВСКО И износи 2.

Задатак C12. HAPPY

⏳3 sec. 💾 512 MB

„Happy" је популаран ланац ресторана у Бугарској. Марија је путовала из Бургаса у оближњи град Варну – родно место ланца „Happy". Тамо се састала са својим пријатељима и отишла у један од ресторана да прослави свој недавни успех на IATI-ју. Позната по одличној услузи, конобарица је Марији задала изазов – ако га реши, поруџбина ће бити бесплатна.

Постоји N ресторана ланца Happy у Варни, нумерисаних од 0 до $N-1$. Они су повезани са $N-1$ двосмерних путева, при чему сваки пут има вредност срећне енергије. За два ресторана x и y, *простом рутом* између њих називамо низ путева који почиње у x, завршава се у y и не пролази кроз исти ресторан више пута. Ресторани су повезани тако да између сваког пара ресторана постоји тачно једна *проста рута*.

Из неког непознатог разлога, *срећа* једне *просте руте* дефинише се као **битовски XOR**[1] срећних енергија путева на њој.

Изазов је у томе што Марија треба да пронађе збир *срећа простих рута* између сваког пара отворених Happy ресторана. У почетку су сви ресторани затворени, али имамо Q измена два типа:

- тип 1 – дати су цели бројеви $l[j]$ и $r[j]$ такви да $0 \le l[j] \le r[j] \le N-1$, и сви ресторани са бројевима у $[l[j], r[j]]$ постају отворени (ако су неки ресторани већ отворени, онда такви и остају);
- тип 2 – дати су цели бројеви $l[j]$ и $r[j]$ такви да $0 \le l[j] \le r[j] \le N-1$, и сви ресторани са бројевима у $[l[j], r[j]]$ постају затворени (ако су неки ресторани већ затворени, онда такви и остају);

Помозите Марији да реши изазов!

Детаљи имплементације

Треба да имплементирате функцију `happy`:

```
std::vector<long long int> happy (int N, int Q,
       std::vector<int> u, std::vector<int> v, std::vector<int> h,
       std::vector<int> t, std::vector<int> l, std::vector<int> r)
```

- N: број Happy ресторана;
- Q: број измена;
- u, v и h: низови од $N-1$ целих бројева, који представљају ресторане и срећне енергије путева;
- t, l, r: низови од Q целих бројева, који представљају типове и интервале измена.

Ова функција ће бити позвана једном за сваки тест и треба да врати низ од Q бројева – збир *срећа простих рута* између сваког пара отворених ресторана након сваке измене.

[1] **Битовски XOR** ($\wedge$) је бинарна операција која упоређује два броја записана у бинарном облику, бит по бит. На конкретној позицији резултат је 1 ако су одговарајући битови различити, а 0 ако су једнаки. На пример, $4879 \wedge 4772 = 427$.

Ограничења

- $1 \leq N \leq 300\,000$;
- $1 \leq Q \leq 300\,000$;
- $1 \leq h[i] < 2^{20}$ за свако $0 \leq i < N - 1$;
- Између сваког пара ресторана постоји тачно једна *проста рута*.

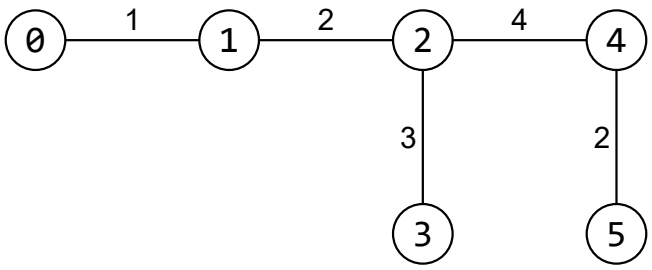
Подзадаци

Подзадатак	**Поени**	**Потребни подзадаци**	N	Q	**Остала ограничења**
0	0	–	–	–	Пример.
1	7	0	≤ 100	≤ 100	-
2	8	$0-1$	$\leq 2\,000$	$\leq 2\,000$	-
3	19	–	$\leq 300\,000$	$\leq 300\,000$	$l[j] = r[j]$ за свако $0 \leq j < Q$ и $u[i] = i$, $v[i] = i + 1$ за свако $0 \leq i < N - 1$.
4	19	–			$l[j] = r[j]$ за свако $0 \leq j < Q$ и $h[i] = 1$ за свако $0 \leq i < N - 1$.
5	10	0	$\leq 300\,000$	≤ 10	-
6	17	–	$\leq 300\,000$	$\leq 300\,000$	Ако је $t[j] = 1$, онда су ресторани у $[l[j], r[j]]$ били затворени пре измене; ако је $t[j] = 2$, онда су ресторани у $[l[j], r[j]]$ били отворени пре измене ($0 \leq j < Q$).
7	20	$0-6$			-

Пример

Размотримо следећи позив функције и илустрацију ресторана, за $N = 6$ и $Q = 5$:

```
happy(6, 5, {0, 1, 2, 2, 4}, {1, 2, 3, 4, 5}, {1, 2, 3, 4, 2},
      {1, 1, 2, 2, 1}, {1, 2, 2, 4, 0}, {2, 3, 2, 5, 5})
```



Тих 5 измена су следеће:

- $1\ 1\ 2$ (ресторани 1 и 2 постају отворени) – након измене имамо само *просту руту* $1 \overset{2}{-} 2$ са срећом 2. Збир је 2;
- $1\ 2\ 3$ (ресторани 2 и 3 постају отворени) – након измене имамо *просте руте* између ресторана 1 и 2 ($1 \overset{2}{-} 2$), 1 и 3 ($1 \overset{2}{-} 2 \overset{3}{-} 3$), и 2 и 3 ($2 \overset{3}{-} 3$), са одговарајућим срећама 2, $2 \wedge 3 = 1$ и 3 ($\wedge$ је C++ оператор за битовски XOR). Збир је $2+1+3=6$;
- $2\ 2\ 2$ (ресторан 2 постаје затворен) – након измене имамо *просту руту* између отворених ресторана 1 и 3. Збир је 1;
- $2\ 4\ 5$ (ресторани 4 и 5 (п)остају затворени) – након измене имамо *просту руту* између отворених ресторана 1 и 3. Збир је 1;
- $1\ 0\ 5$ (ресторани од 0 до 5 постају отворени) – након измене имамо *просте руте* између свих парова ресторана. Збир је 56.

Према томе, позив треба да врати `{2, 6, 1, 1, 56}`.

Пример грејдера

Формат улаза је следећи:

- ред 1: два цела броја – вредности N и Q.
- редови $2+i$ до $2+N-2$: три цела броја $u[i]$, $v[i]$, $h[i]$ – представљају пут између ресторана са бројевима $u[i]$ и $v[i]$, са срећном енергијом $h[i]$.
- редови $N+1+j$ до $N+1+Q-1$: три цела броја $t[j]$, $l[j]$, $r[j]$ – представљају измену типа $t[j]$ за интервал ресторана са бројевима од $l[j]$ до $r[j]$.

Формат излаза је следећи:

- ред i: i-та вредност враћена позивом функције.

Задатак C13. PARTITION

⏳0.1 sec. 💾 1024 MB

Често се каже да браћа и сестре теже да деле ствари што је могуће равноправније. Овде разматрамо ситуацију са две сестре – Мартином и **Д**ени. Мартина је недавно купила N паковања бомбона, при чему i-то паковање садржи позитиван број бомбона A_i за $0 \leq i \leq N-1$, а укупан број бомбона је $S = A_0 + A_1 + \cdots + A_{N-1}$. Сестре треба да поделе бомбоне између себе. Њима није важан само број бомбона, већ и број паковања! После дуге расправе, сестре су се договориле о следећем поступку:

Најпре ће **Д**ени поређати паковања на било који начин који жели. Формално, она ће задати неку њихову пермутацију. Означимо ту пермутацију индексима $i_0, i_1, \ldots, i_{N-1}$, где је $\{i_0, i_1, \ldots, i_{N-1}\} = \{0, 1, \ldots, N-1\}$. Затим Мартина даје паковања $A_{i_0}, A_{i_1}, \ldots$ **Д**ени тим редом све док укупан број датих бомбона не постане $\geq \frac{S}{2}$. Након тога, Мартина узима преостала паковања за себе, па су сва паковања расподељена (тј. свако паковање припада тачно једној од сестара).

Дени воли своју сестру и зато ће покушати да минимизује апсолутну разлику између броја паковања која она добија и броја паковања која добија њена сестра. Дакле, она жели да изабере пермутацију која минимизује ту вредност међу свим могућим пермутацијама паковања. Напишите програм **`partition`** који ће јој помоћи да у T сценарија одреди једну такву пермутацију паковања бомбона. Ако постоји више решења, можете вратити било које од њих.

Детаљи имплементације

Треба да имплементирате функцију `solve`:

```
std::vector<int> solve(std::vector<int> A)
```

- A: вектор од N позитивних бројева, који представљају број бомбона у паковањима које је Мартина купила.

Ова функција се позива T пута – једном за сваки сценарио. За сваки сценарио треба да врати вектор који садржи неку пермутацију индекса $0, 1, \ldots, N-1$ такву да она минимизује апсолутну разлику између броја паковања у насталој подели.

Ограничења

- $1 \leq N \leq 2 \times 10^6$
- $1 \leq \Sigma N \leq 10^7$, где је ΣN збир свих вредности N по сценаријима у једном тест примеру.
- $1 \leq T \leq 5 \times 10^3$
- $1 \leq A_i \leq 10^9$ за свако $0 \leq i \leq N-1$

Подзадаци

Подзадатак	**Поени**	**Потребни подзадаци**	N	ΣN	**Остала ограничења**
0	0	–	–	–	Пример испод.
1	3	–	$\leq 10^5$	$\leq 10^5$	Све вредности $A_0, A_1, \ldots, A_{N-1}$ појављују се паран број пута; $A_i \leq 3 \times 10^7$ за $0 \leq i \leq N-1$; $T = 1$
2	3	–	≤ 25	≤ 250	$A_i = 2^i$ за $0 \leq i \leq N-1$
3	3	2	$\leq 10^6$	$\leq 5 \times 10^6$	$A_i = 2^{s_i}$ за $0 \leq i \leq N-1$, где је $0 \leq s_i \leq 25$
4	5	–	$\leq 2 \times 10^6$	$\leq 10^7$	$A_i = i + 1$ за $0 \leq i \leq N-1$
5	4	0	≤ 7	$\leq 3.5 \times 10^3$	-
6	6	0	≤ 20	≤ 200	
7	19	$0, 2, 5-6$	$\leq 2 \times 10^3$	$\leq 10^4$	
8	28	$0-2, 5-7$	$\leq 10^5$	$\leq 5 \times 10^5$	
9	29	$0-7$	$\leq 2 \times 10^6$	$\leq 10^7$	

Поени за подзадатак добијају се само ако су успешно прошли сви тестови за њега и за све потребне подзадатке.

Пример

Улаз	**Излаз**
2 9 5 6 6 3 1 1 4 4 3 8 2 2 3 2 3 2 2 3	0 3 6 7 4 8 1 2 5 2 4 6 7 0 1 3 5

Дати излаз за први пример представља једно од више могућих решења. Бројеви бомбона у паковањима након преуређивања су: $5, 3, 4, 4, 1, 3, 6, 6, 1$. Дени ће добити паковања са индексима $0, 3, 6, 7, 4$, чиме добија $5 + 3 + 4 + 4 + 1 = 17$ бомбона. Приметимо да је $5 + 3 + 4 + 4 = 16 < \frac{S}{2} = 16.5$, па Мартина не може да стане раније. Мартина ће узети преостала паковања, тако да је апсолутна разлика између броја паковања у подели једнака $|5 - 4| = 1$. Може се доказати да је то оптимално за овај тест.

Пример грејдера

Формат улаза:

- ред 1: један позитиван број T – број сценарија;
- ред 2: један позитиван број N – број паковања;
- ред 3: $A_0\ A_1\ \ldots\ A_{N-1}$ – број бомбона у паковањима.

Формат излаза:

- ред i: бројеви које враћа i-ти позив функције `solve`.

Задатак C21. BALANCE

⏳0.2 sec. 💾 1024 MB

Кирхо има низ од N целих бројева (не нужно ненегативних) — a_0, a_1, ..., a_{N-2}, a_{N-1}. Он може да врши операције над низом. Свака операција је одређена паром (l, r) тако да је $0 \leq l \leq r < N$, и за ту операцију он може да изабере да изврши тачно једну од следеће две радње:

- за свако $x \in \{0, 1, \ldots, r-l-1, r-l\}$, дода $(-1)^x$ на a_{l+x};
- за свако $x \in \{0, 1, \ldots, r-l-1, r-l\}$, одузме $(-1)^x$ од a_{l+x}.

Нека је *баланс* низа дефинисан као најмањи број операција потребан да се сви његови елементи доведу до нуле. Кирхо жели да одреди *баланс* неколико низова, од којих је сваки подниз[1] почетног (главног) низа. Међутим, између упита низ се може мењати ажурирањем појединачних елемената. За више информација о ажурирањима, погледајте одељак *Детаљи имплементације*. Нажалост, Кирхо не може сам да одговори на упите, па вас моли да напишете програм **`balance`** који решава задатак.

Детаљи имплементације

Ово је интерактивни задатак, што значи да се информације између вашег програма и програма жирија неће размењивати преко стандардног улаза и излаза, већ преко посебних функција.

Потребно је да имплементирате следеће функције:

```
void init(const std::vector<int>& A);
```

Ова функција се позива тачно једном на почетку сваког теста. Вектор A садржи N целих бројева – a_0, a_1, ..., a_{N-1}, који представљају почетни низ.

```
long long balance(int u, int v);
```

Ова функција представља упит за *баланс* подниза a_u, a_{u+1}, ..., a_{v-1}, a_v. Функција треба да врати *баланс* тог подниза. Сви позиви функције `balance()` су међусобно независни.

```
void update(int p, int x);
```

Ова функција представља ажурирање низа – поставља a_p на x. Ова промена важи за све наредне позиве функција у оквиру текућег теста.

Потребно је да предате датотеку `balance.cpp`, која садржи функције `init()`, `balance()` и `update()`. Она може садржати и други кôд и помоћне функције неопходне за ваш програм, али **не сме да садржи главну функцију** `main()`. На почетку датотеке мора да стоји:

```
#include "balance.h"
```

Ваш програм ће бити извршен по једном за сваки тест.

[1] Низ p је подниз низа q ако се p може добити из q брисањем неколико (можда ниједног или свих) елемената са почетка и неколико (можда ниједног или свих) елемената са краја. Специјално, сваки низ је подниз самог себе.

Ограничења

Нека је Q укупан број позива функција `balance()` и `update()` у оквиру једног теста.

- $1 \leq N, Q \leq 10^5$
- $-10^9 \leq a_i, x_i \leq 10^9$
- $0 \leq u_i \leq v_i < N$
- $0 \leq p_i < N$
- Гарантовано је да постоји бар један позив функције `balance()`

Подзадаци

Подзадатак	**Поени**	**Потребни подзадаци**	N	Q	**Остала ограничења**
0	0	–	–	–	Пример.
1	11	–	≤ 20	≤ 100	Одговор на сваки упит је највише 3.
2	7	–	–	–	Не постоје ажурирања и важи $a_i \geq 0$ за све i
3	9	–	–	–	Не постоје ажурирања, низ је алтернирајући[2] и $\lvert a_i \rvert \leq \lvert a_j \rvert$ за свако $i \leq j$
4	14	–	–	$= 1$	Не постоје ажурирања, $\lvert a_i \rvert = \lvert a_j \rvert$ за све i, j
5	18	–	–	$= 1$	Не постоје ажурирања, низ је алтернирајући.
6	15	$2 - 5$	–	–	Не постоје ажурирања.
7	26	$0 - 6$	–	–	–

Поени за подзадатак се додељују само ако пролазе сви његови тестови.

[2] Алтернирајући низ је низ у ком су сви бројеви различити од нуле и њихови знаци се наизменично мењају.

Пример интеракције

№	Акције жирија	Одговор вашег програма
1	`init({3,-4,2,-4,3})`	
2	`balance(0,4)`	`6`
3	`balance(1,3)`	`6`
4	`update(0, 2)`	
5	`balance(0,2)`	`4`

У овом примеру, почетни низ је $a = (3, -4, 2, -4, 3)$. За први упит, Кирхо може да изврши следећи низ операција:

$$(3, -4, 2, -4, 3) \to (3, -4, 3, -4, 3) \to (3, -4, 4, -4, 3) \to (3, -3, 3, -3, 3) \to$$
$$(2, -2, 2, -2, 2) \to (1, -1, 1, -1, 1) \to (0, 0, 0, 0, 0).$$

За други упит:

$$(-4, 2, -4) \to (-4, 3, -4) \to (-4, 4, -4) \to (-3, 3, -3) \to$$
$$(-2, 2, -2) \to (-1, 1, -1) \to (0, 0, 0).$$

После ажурирања, низ постаје $a = (2, -4, 2, -4, 3)$, и Кирхо може да изврши:

$$(2, -4, 2) \to (1, -3, 1) \to (0, -2, 0) \to (0, -1, 0) \to (0, 0, 0).$$

Може се доказати да су ови низови операција оптимални.

Локално тестирање

Добијате датотеке `balance.h` и `Lgrader.cpp`, које можете компајлирати заједно са својим програмом да бисте га локално тестирали, као и команде за компилацију.

При покретању локалног грејдера, у првој линији стандардног улаза треба унети позитивне целе бројеве N и Q. У другој линији унесите N целих бројева – a_0, ..., a_{N-1} – који представљају почетни низ. Свака од наредних Q линија описује или упит или ажурирање. У том тренутку грејдер ће позвати `init()` са параметром $A = a_0, \ldots, a_{N-1}$. На почетку сваког упита унесите цео број t_i, где је $1 \leq t_i \leq 2$. Ако је $t_i = 1$, онда унесите два цела броја u_i и v_i, који представљају позив функције `balance()` са параметрима u_i и v_i. Ако је $t_i = 2$, онда унесите два цела броја p_i и x_i, који представљају позив функције `update()` са параметрима p_i и x_i.

За сваки позив функције `balance()`, локални грејдер ће исписати враћену вредност. Редослед позива функција прати улаз.

Задатак C22. EVILUTION

⏳ 3 sec. 💾 1024 MB

Добри момци планирају да победе Лоше момке тако што ће их гађати јонизујућим зрачењем. Да би калибрисали своје оружје, Добри момци морају да знају састав неког интервала ДНК Лоших момака. Лоши момци нису само лоши – они су зли –па се сваког дана развијају тако што свако од слова A, C, G и T у својој ДНК замењују одговарајућим нискама дужине **најмање 2**: $S_{\mathtt{A}}$, $S_{\mathtt{C}}$, $S_{\mathtt{G}}$ и $S_{\mathtt{T}}$. Биће много борби током много дана, па Добри момци морају да обраде Q различитих упита који се састоје од три броја – K_i, L_i и R_i. За сваки упит, потребно је да одредите четири броја, који одговарају броју слова A, C, G и T у затвореном интервалу $[L_i; R_i]$ ДНК Лоших момака K_i-тог дана.

Детаљи имплементације

Треба да имплементирате функцију `solve`:

```
std::vector<std::vector<long long>> solve(
        std::string S_0,
        std::vector<std::string> S_ACGT,
        std::vector<long long> K,
        std::vector<long long> L,
        std::vector<long long> R
)
```

- S_0: ДНК Лоших момака нултог дана.
- $S_{\mathtt{ACGT}}$: 4 ниске $S_{\mathtt{A}}, S_{\mathtt{C}}, S_{\mathtt{G}}, S_{\mathtt{T}}$.
- K: вектор од Q ненегативних целих бројева, чији је i-ти елемент K_i.
- L: вектор од Q ненегативних целих бројева, чији је i-ти елемент L_i.
- R: вектор од Q ненегативних целих бројева, чији је i-ти елемент R_i.

Ова функција се позива тачно једном за сваки тест пример. Треба да врати вектор од Q вектора са по 4 елемента – број слова A, C, G и T у одговарајућим упитима.

Ограничења

- $2 \leq S \leq 10^5$, где је $S = \max(|S_0|, |S_{\mathtt{A}}|, |S_{\mathtt{C}}|, |S_{\mathtt{G}}|, |S_{\mathtt{T}}|)$
- Гарантовано је да су сва слова у нискама из скупа A, C, G, T.
- $1 \leq Q \leq 10^4$
- $0 \leq K_i \leq 10^{18}$ за свако $0 \leq i \leq Q-1$
- $0 \leq L_i \leq R_i \leq 10^{18}$ за свако $0 \leq i \leq Q-1$
- Гарантовано је да за све упите постоје слова са индексима од L_i до R_i.

Подзадаци

Подзадатак	**Поени**	**Потребни подзадаци**	S	Q	K_i	**Остала ограничења**
0	0	—	-	-	-	Пример.
1	7	0	≤ 5	≤ 100	≤ 10	-
2	6	$0-1$	≤ 6	$\leq 10^4$	≤ 10	-
3	13	$0-2$	$\leq 10^3$	$\leq 10^4$	≤ 50	-
4	10	$0-3$	$\leq 10^5$	$\leq 10^4$	≤ 50	-
5	15	$0-4$	$\leq 10^5$	$\leq 10^4$	$\leq 5 \times 10^3$	-
6	7	—	$\leq 10^5$	$\leq 10^4$	$\leq 10^{18}$	$L_i = R_i = 0$.
7	17	$0-3$	$\leq 10^3$	$\leq 10^4$	$\leq 10^{18}$	-
8	25	$0-7$	$\leq 10^5$	$\leq 10^4$	$\leq 10^{18}$	-

Поени за подзадатак добијају се само ако су успешно прошли сви његови тестови и тестови потребних подзадатака.

Пример теста

Улаз	**Излаз**	**Објашњење примера**
TAG TGT CGT CCT CGC 10 1 3 3 2 2 5 0 0 2 0 2 2 0 0 2 1 8 8 0 1 1 1 0 5 0 0 1 1 2 7	0 0 0 1 0 2 0 2 1 0 1 1 0 0 1 0 1 0 1 1 0 0 0 1 1 0 0 0 0 2 2 2 1 0 0 1 0 3 1 2	ДНК Лоших момака нултог, првог и другог дана изгледа овако: • TAG • CGCTGTCCT • CGTCCTCGTCGCCCTCGCCGTCGTCGC

Пример грејдера

Формат улаза:

- редови 1 до 5: $S_0, S_\mathtt{A}, S_\mathtt{C}, S_\mathtt{G}, S_\mathtt{T}$.
- ред 6: Q – број упита.
- редови 7 до $7 + (Q-1)$: K_i, L_i, R_i.

Формат излаза:

- ред i – бројеви које враћа i-ти позив.

Задатак C23. TERRITORY

⏳0.35 sec. 💾 1024 MB

Због своје жеље за моћи, Сашка је одлучила да створи нову државу. Она ће се простирати на територији Бугарске, која се може представити мрежом од N области, нумерисаних од 1 до N, повезаних са $N-1$ путева. Из сваке области могуће је стићи до сваке друге користећи путеве. Сашкина држава ће обухватати неколико (једну или више) области тако да су оне повезане. Области су повезане ако између било које две изабране области постоји пут на ком се не налази ниједна неизабрана област.

Најновији попис је утврдио да i-та област има a_i становника. Зато ће број становника Сашкине државе бити збир броја становника свих изабраних области. Пошто би Сашки било страшно досадно да разматра само једну варијанту своје државе, она ће разматрати све могуће. Она жели да пронађе збир популација за све могуће изборе изгледа државе. Напишите програм **`territory`** који проналази ту вредност. Пошто збир може бити превелик, пронађите његов остатак при дељењу са 998244353.

Улаз

У једном реду стандардног улаза дат је позитиван цео број N. У наредном реду дато је N позитивних целих бројева $a_1, a_2, \ldots, a_N$. У преосталих $N-1$ редова описана је мрежа путева: i-ти од тих редова садржи целе бројеве u_i и v_i, који одређују пут између области u_i и области v_i.

Излаз

На стандардни излаз исписати тражену вредност.

Ограничења

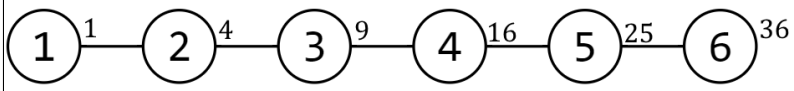
- $1 \leq N \leq 200\,000$
- $1 \leq a_i \leq 10^7$
- $1 \leq u_i, v_i \leq N$

Подзадаци

Подзадатак	Потребни подзадаци	Поени	N	Додатна ограничења
0	–	0	–	Пример.
1	0	15	≤ 15	–
2	–	10	$\leq 200\,000$	$u_i = i,\ v_i = i+1$
3	$0-1$	35	$\leq 2\,000$	–
4	$0-3$	40	$\leq 200\,000$	–

Поени за подзадатак додељују се само ако су успешно прошли сви тестови предвиђени за њега.

Примери

Улаз	**Излаз**	**Објашњење**
6 1 4 9 16 25 36 1 2 2 3 3 4 4 5 5 6	812	Бугарска је приказана на слици. 
7 10 7 5 8 4 6 9 2 7 2 5 7 1 6 5 7 4 3 5	864	Бугарска је приказана на слици.