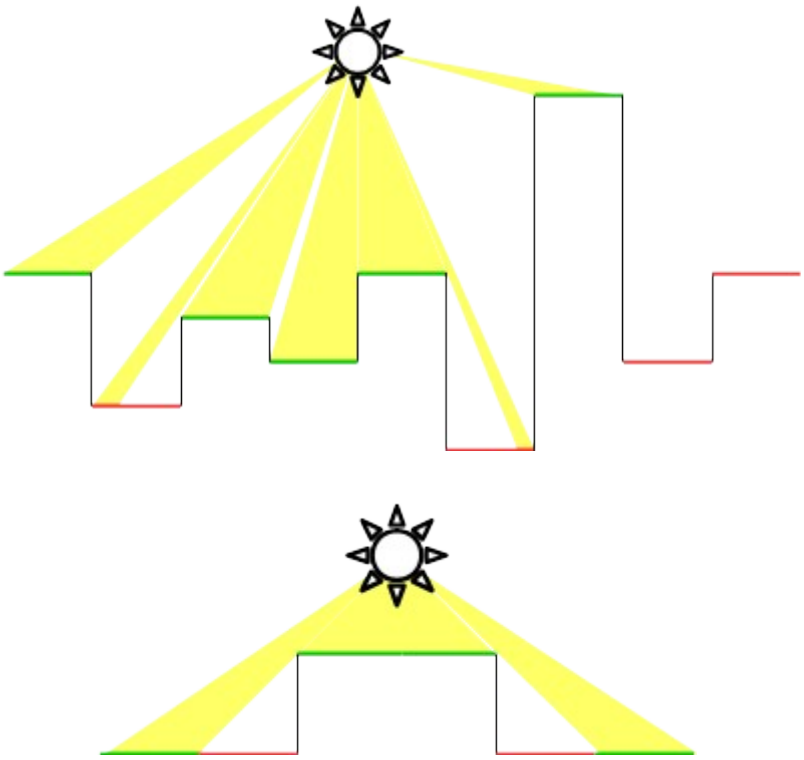
Hodowla roślin

Liczba punktów: 70
Limit czasu: 1s
Limit pamięci: 256MB

Jaś zajął się amatorsko hodowlą roślin. Zagospodarował w tym celu szklarnię. Światło zapewnia jedna żarówka. Każda roślina wymaga miejsca (jednego metra) na podłodze oraz aby całe to miejsce było oświetlone. Niestety, podłoga nie jest równa, i światło nie wszędzie dociera. Pomóż Jasiowi zdecydować, ile maksymalnie roślin może posadzić w szklarni.
Każdy kolejny metr podłogi jest idealnie równy, ale może być na innej wysokości niż pozostałe. Poniższy przykład przedstawia przykładową szklarnie oraz jej oświetlenie. Zieloną linią zaznaczono miejsca, w których rośliny mogą rosnąć, czerwoną miejsca, gdzie nie dociera dostatecznie dużo światła.



Wejście

W pierwszej linii pojawią się trzy liczby: długość hodowli (n), odległość żarówki od lewego brzegu hodowli (x) oraz wysokość żarówki nad podłogą (y). W następnych n wierszach pojawi się n liczb – wysokości kolejnych fragmentów nad poziomem zerowym. Żadna z tych wartości nie przekroczy 60000. Żarówka zawsze wisi wyżej, niż najwyższy punkt podłogi.

Wyjście

Na wyjściu należy wypisać maksymalną liczbę roślin, które Jaś może posadzić.

Przykład

Wejście

```
9 4 9
4
1
3
2
4
0
8
2
4
```

Wyjście

```
5
```

Wejście

```
6 3 2
0
0
1
1
0
0
```

Wyjście

```
4
```

Podział rafy

Liczba punktów: 90
Limit czasu: 1-9s
Limit pamięci: 256MB

Na dnie pewnego oceanu korale zaczęły budować rafę. Wszystko zaczęło się od jednego koralowca, do którego dołączył inny, do nich dołączył kolejny itd. Za każdym razem gdy nowy koralowiec chciał dołączyć do rafy, wybierał sobie sąsiada (jednego koralowca z rafy) i doklejał się do niego. Upłynęło wiele czasu, rafa się rozrosła i koralowcom zrobiło się zbyt tłoczono. Zdecydowały więc o podziale rafy na dwie części. Podział nastąpi poprzez rozdzielenie dwóch wybranych sąsiadów. Aby jedna z części nie czuła się odrzucona przez większość, podział należy zrobić tak, aby po podziale różnica wag dwóch nowopowstałych raf była jak najmniejsza. Waga każdej części jest sumą wag każdego koralowca należącego do tej części, z kolei waga pojedynczego koralowca jest jego prywatną sprawą.

Wejście

Koralowce numerowane są od zera.
Pierwszą liczbą w pierwszym wierszu na wejściu będzie liczba koralowców w rafie (n). Kolejna liczba, w kolejnym wierszu to waga koralowca, od którego zaczęła się rafa (w_0). Następnie pojawi się n-1 wierszy, każdy zawierający parę liczb
w_j s_j
dla $j = 1, ..., n$-1 oznaczających, że do rafy dołączył koralowec o wadze w_j doklejając się do koralowca s_j ($s_j < j$). Waga żadnego koralowca nie przekroczy 1000, liczba koralowców w rafie nie będzie większa niż 1000000.

Wyjście

Na wyjściu należy wypisać: w pierwszej linii minimalną różnicę wag raf po podziale oraz, po spacji, liczbę możliwych podziałów, które doprowadzą do minimalnej różnicy (k). W kolejnych k wierszach należy wypisać wszystkie te podziały. Każdy podział powinien być opisany przez dwie liczby u v – koralowce które powinny się rozdzielić ($u < v$). Podziały powinny być uporządkowane rosnąco według koralowca u (w przypadku remisów, rosnąco według koralowca v).

Przykład

Wejście

```
8
1
1 0
1 1
1 2
1 3
1 1
1 1
1 4
```

Wyjście
```
0 1
1 2
```

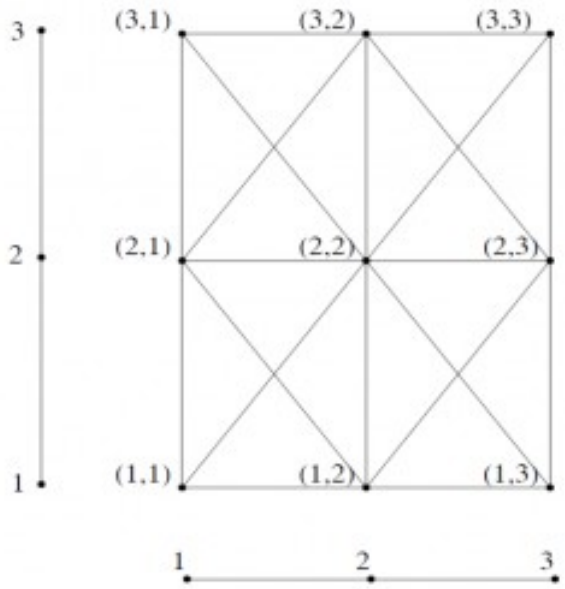
Wejście
```
5
2
2 0
2 1
2 2
2 3
```

Wyjście
```
2 2
1 2
2 3
```

Sieć

Liczba punktów: 100
Limit czasu: 1s
Limit pamięci: 256MB

Rzeczywiste sieci to wielkie grafy, które można modelować za pomocą tzw. silnego iloczynu. Dla danych grafów G1 oraz G2, silny iloczyn oznaczamy przez G1*G2 . Wierzchołkami grafu G1*G2 są wszystkie pary wierzchołków (v1 ,v2), gdzie v1 jest wierzchołkiem grafu G1, a v2 jest wierzchołkiem grafu G2. Uwaga, możemy przyjąć, że wierzchołkami grafu są liczby.
Pomiędzy dwoma wierzchołkami (v1,v2) oraz (u1,u2) grafu G1*G2 istnieje krawędź (połączenie) wtedy i tylko wtedy, gdy {v1,u1} jest krawędzią w grafie G1 oraz {v2,u2} jest krawędzią w grafie G2 lub
v1 = u1 oraz {v2 , u2} jest krawędzią w grafie G2 lub
v2 = u2 oraz {v1 , u1} jest krawędzią w grafie G1.
Przez G^{*p} oznaczamy p-tą silną potęgę grafu G, tj. G*G*G* ... *G, gdzie G występuje p razy (p>0).
Według najnowszych badań wielkie sieci ewoluują tak jak silna potęga grafu G.
Twoim zadaniem będzie określenie liczby krawędzi silnej potęgi grafowej (połączeń w sieci) dla zadanego grafu G oraz potęgi p.



Wejście

W pierwszej linii ilość grafów w teście (nie więcej niż 50).
W kolejnych liniach liczba wierzchołków grafu. Następnie po spacji potęga p (nie większa niż 100). Następnie po spacji macierz sąsiedztwa grafu G zapisana w jednej linii po spacjach (od lewej do prawej oraz z góry na dół).

Wyjście

W kolejnych liniach liczba krawędzi grafu G^{*p}.

Przykład

Wejście

```
4
6 1 0 1 0 0 0 1 1 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1 1 0 0 0 1 0
3 2 0 1 0 1 0 1 0 1 0
5 2 0 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0
3 3 0 1 0 1 0 0 0 0 0
```

Wyjście

```
6
20
100
49
```

Pierwsze cukierki

Liczba punktów: 80
Limit czasu: 1s
Limit pamięci: 256MB

Twój wujek ma fabrykę cukierków. Obiecał, że jeśli nauczysz się własności liczb pierwszych, to dostaniesz łakocie.
Wujek poustawiał w linii kosze słodyczy i powiedział, że możesz wybrać kosze o ile najbliższe wybrane kosze są oddalone o nieparzystą liczbę pierwszą (z obu stron). Wybierz kosze tak, aby spełnić warunki wujka i zebrać jak najwięcej cukierków.
Przypomnienie: liczba pierwsza p ma dokładnie dwa dzielniki: 1 oraz p

Wejście

W pierwszej linii znajduje się liczba koszyków *n* (*n*<1000000). W następnej linii *n* liczb informujących o liczbie cukierków w koszykach. Liczba cukierków każdym koszu jest nie większa niż 100000000.

Wyjście

Liczba zebranych cukierków.

Przykład

Wejście
```
6
5 1 1 1 1 5
```

Wyjście
```
10
```

Wejście
```
10
2 3 4 2 9 3 4 2 5 7
```

Wyjście
```
19
```

wybieramy drugi, piąty, dziesiąty

BrainIgETI

Liczba punktów: 60
Limit czasu: 1s
Limit pamięci: 256MB

Należy napisać interpreter języka BrainIgETI. Język ten jest modyfikacją języka Brainf*** stworzonego w latach 90.
Na interpreter składają się:

- maszyna wirtualna
- kod

Maszyna składa się z *n* elementowej tablicy bajtów bez znaku zainicjalizowanych zerami oraz głowicy (wskaźnik) ustawionej na pierwszy element tablicy.
Język BrainIgETI składa się z operacji +-<>[], które oznaczają:

<	przesuń głowicę o 1 element w lewo
>	przesuń głowicę o 1 element w prawo
+	zwiększ wartość o 1 elementu tablicy wskazywanego przez głowicę
-	zmniejsz wartość o 1 elementu tablicy wskazywanego przez głowicę
[	jeśli wartość elementu tablicy wskazywanej przez głowicę jest równa 0, wtedy przechodzimy do instrukcji za sparowaną]
]	skacze do sparowanej instrukcji [

Nawiasy [] muszą być sparowane - odpowiadają pętli 'while'.
Jeśli głowica jest na początku tablicy, to operacja < nic nie wykonuje.
Jeśli głowica jest na końcu tablicy, to operacja > nic nie wykonuje.
Operacja - na wartości 0 daje 255 natomiast + na 255 daje 0.

Program kończy się, gdy nie ma więcej instrukcji do wykonania.
Interpreter zlicza wykonane operacje.

Wejście

W pierwszej linii znajdują się 2 liczby. Pierwsza to liczba bajtów w tablicy maszyny wirtualnej(<=256), druga to ogranicznik na liczbę operacji (<100000).
W kolejnej linii znajduje się kod programu.

Wyjście

Jeśli kod jest nieprawidłowy (nie sparowane nawiasy[]) należy wypisać 'BrainFail'. W przeciwnym wypadku w pierwszej linii należy wypisać wartości tablicy maszyny wirtualnej, zaś w drugiej linii należy wypisać dwie liczby: pozycję głowicy oraz liczbę wykonanych operacji.

Przykład

Wejście
```
4 100
++
```

Wyjście
2 0 0 0
1 2

Wejście
4 100
++[>++<-]

Wyjście
0 4 0 0
1 17

wytłumaczenie - + wykonały się 6 razy, - 2 razy < - 2 razy > 2 razy [3 razy] 2 razy
kolejność wykonywania ++[>++<-][>++<-][

Wejście
3 12
++[>++<-]

Wyjście
1 3 0
2 12
kolejność wykonywania ++[>++<-][>+

Wejście
3 100
+++[>++[>++<-]<-]

Wyjście
0 0 12
1 70