النيل 🏞️

نريد نقل N من التحف عبر نهر النيل 🌊.

ترقم التحف من 0 إلى $N-1$.

نرمز إلى وزن كل تحفة i $(0 \leq i < N)$ بالرمز $W[i]$.

لنقل التحف يجب عليك استخدام قوارب ⛵ مخصصة بحيث أن كل قارب يمكن أن ينقل تحفتين **على الأكثر**

* اذا قمت بوضع تحفة واحدة فقط في القارب ⛵ فإنه لا يؤثر وزن هذه التحفة على عملية النقل.

* إذا أردت أن تضع تحفتين معاً في نفس القارب ⛵ ، يجب عليك أن تتأكد من توزع الأوزان بشكل متكافئ على القارب ⛵. أي: يمكنك إرسال التحفتين p و q $(0 \leq p < q < N)$

في نفس القارب ⛵ إذا وفقط إذا كانت القيمة المطلقة للفرق بين وزنيهما هي على الأكثر D,

ذلك يعني أن $|W[p] - W[q]| \leq D$.

لنقل التحف يجب عليك أن تدفع كلفة 💰 النقل والتي تعتمد على عدد التحف التي سيتم حملها على نفس القارب ⛵ . حيث أن كلفة نقل التحفة i $(0 \leq i < N)$ هي:

* $A[i]$, إذا وضعت التحفة في قارب لوحدها أو

* $B[i]$, إذا وضعت التحفة في قارب مع تحفة أخرى.

لاحظ أنه في الحالة الثانية يجب عليك دفع تكلفة 💰 كلا التحفتين اللتين يتم نقلهما في نفس القارب ⛵. بشكل أدق إذا قررت إرسال التحفة p و q $(0 \leq p < q < N)$ في نفس القارب ⛵ يجب عليك أن تدفع $B[p] + B[q]$.

كما أنه سيكون دائماً كلفة 💰 إرسال التحفة في قارب ⛵ لوحدها أعلى من كلفة إرسالها مع تحفة أخرى في نفس القارب، أي $B[i] < A[i]$ من أجل كل i بحيث أن $0 \leq i < N$.

لسوء الحظ فإن وضع النهر 🌊 لا يمكن التنبؤ به لذلك فإن قيمة D تتغير بكثرة.

مهمتك هي الإجابة على Q

سؤالاً مرقمة من 0 إلى $Q-1$.

يتم وصف الأسئلة عن طريق مصفوفة E طولها Q.

الإجابة على السؤال j $(0 \leq j < Q)$ هي

أصغر مجموع تكاليف 💰 نقل كل التحف التي عددها N عندما تكون قيمة D تساوي $E[j]$.

تفاصيل البرمجة

يجب عليك برمجة التابع التالي

```
std::vector<long long> calculate_costs(

std::vector<int> W, std::vector<int> A,

std::vector<int> B, std::vector<int> E)
```

* W, A, B: مصفوفات من الأعداد الصحيحة طولها N, تصف أوزان التحف وكلف نقلها.

* E: مصفوفة من الأعداد الصحيحة طولها Q تصف القيم لـ D.

* يجب على التابع أن يعيد المصفوفة R المكونة من Q عدداً صحيحاً. تحوي القيم الصغرى لمجموع التكاليف الخاصة بنقل التحف حيث $R[j]$ تعطي التكلفة عندما تكون قيمة D هي $E[j]$ (من أجل كل j

حيث أن $0 \leq j < Q$).

* سيتم طلب هذا التابع مرة واحدة تماماً من أجل كل حالة اختبار.

القيود

* $1 \leq N \leq 100\,000$

* $1 \leq Q \leq 100\,000$

* $1 \leq W[i] \leq 10^9$

من أجل كل i حيث أن $0 \leq i < N$

* $1 \leq B[i] < A[i] \leq 10^9$

من أجل كل i حيث أن $0 \leq i < N$

* $1 \leq E[j] \leq 10^9$

من أجل كل j حيث أن $0 \leq j < Q$

المسائل الجزئية

| المسألة الجزئية | العلامة | قيود إضافية |

| :-----: | :----: | --------------------- |

| 1 | 6 | $Q \le 5$; $N \le 2000$; $W[i] = 1$ من أجل كل i حيث أن $0 \le i < N$

| 2 | 13 | $Q \le 5$; $W[i] = i + 1$ من أجل كل i حيث أن $0 \le i < N$

| 3 | 17 | $Q \le 5$; $A[i] = 2$ و $B[i] = 1$ من أجل كل i حيث أن $0 \le i < N$

| 4 | 11 | $Q \le 5$; $N \le 2000$

| 5 | 20 | $Q \le 5$

| 6 | 15 | $A[i] = 2$ و $B[i] = 1$ من أجل كل i حيث أن $0 \le i < N$

| 7 | 18 | بدون قيود إضافية.

مثال

ليكن لدينا الاستدعاء التالي.

```
calculate_costs([15, 12, 2, 10, 21],

[5, 4, 5, 6, 3],

[1, 2, 2, 3, 2],

[5, 9, 1])
```

لدينا في هذا المثال $N = 5$ تحف و $Q = 3$ اسئلة.

في السؤال الأول، $D = 5$.

بإمكانك إرسال التحفتين 0 و 3

في قارب واحد معاً (لأن $|15 - 10| \le 5$) وكل واحدة من التحف المتبقية في قارب منفصل.

هذا يعطي أقل تكلفة لنقل كل التحف والتي هي $1 + 4 + 5 + 3 + 3 = 16$.

في السؤال الثاني يكون $D = 9$.

يمكنك إرسال التحفتين 0 و 1 في قارب واحد (لأن $|15 - 12| \le 9$) وإرسال التحفتين 2 و 3 في قارب واحد معاً (لأن $|2 - 10| \le 9$).

يمكن إرسال التحفة المتبقية في قارب منفصل.

هذا يعطي أقل كلفة لنقل كل التحف والتي هي $1 + 2 + 2 + 3 + 3 = 11$.

في السؤال الأخير يكون $D = 1$.

يجب عليك إرسال كل تحفة في قارب خاص بها وهذا يعطي أقل تكلفة نقل لكل التحف والتي هي $5+4+5+6+3=23$.

وهكذا يجب على التابع أن يعيد $[16, 11, 23]$.

Sample Grader

Input format:

```
N
W[0] A[0] B[0]
W[1] A[1] B[1]
...
W[N-1] A[N-1] B[N-1]
Q
E[0]
E[1]
...
E[Q-1]
```

Output format:

```
R[0]
R[1]
...
R[S-1]
```

Here, S is the length of the array R returned by `calculate_costs`.