Nile

Queres transportar N artefactos pelo rio Nilo. Os artefactos são numerados de 0 a $N-1$. O peso do artefacto i ($0 \leq i < N$) é $W[i]$.

Para transportar os artefactos, vais usar barcos especializados. Cada barco pode levar **no máximo dois** artefactos.

- Se decidires colocar um único artefacto num barco, o peso desse artefacto pode ser arbitrário.
- Se quiseres colocar dois artefactos no mesmo barco, tens de assegurar que o barco fica equilibrado. Especificamente, podes enviar os artefactos p e q ($0 \leq p < q < N$) no mesmo barco só se a diferença absoluta entre os seus pesos é no máximo D, ou seja $|W[p] - W[q]| \leq D$.

Para transportar um artefacto, tens de pagar um custo que depende do número de artefactos transportados no mesmo barco. O custo de transportar o artefacto i ($0 \leq i < N$) é:

- $A[i]$, se puseres o artefacto sozinho num único barco, ou
- $B[i]$, se o puseres num barco juntamente com um qualquer outro artefacto.

Nota que no segundo caso, tens de pagar pelos dois artefactos no barco. Especificamente, se decidires enviar os artefactos p e q ($0 \leq p < q < N$) no mesmo barco, tens de pagar $B[p] + B[q]$.

Enviar um artefacto sozinho num barco é sempre mais caro que enviá-lo com algum outro no mesmo barco, portanto $B[i] < A[i]$ para todo o i tal que $0 \leq i < N$.

Infelizmente, o rio é muito imprevisível e o valor de D está sempre a mudar. A tua tarefa é responder a Q questões numeradas de 0 a $Q-1$. As questões são descritas por um array E de tamanho Q. A resposta à questão j ($0 \leq j < Q$) é o mínimo custo total necessário para transportar todos os N artefactos quando o valor de D é igual a $E[j]$.

Detalhes de implementação

Deves implementar a seguinte função:

```
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: arrays de inteiros de tamanho N, descrevendo os pesos dos artefactos e o custo de os transportar.
- E: um array de inteiros de tamanho Q descrevendo o valor de D para cada questão.
- Esta função deve devolver um array R com Q inteiros contendo o mínimo custo total necessário para transportar os artefactos, onde $R[j]$ é o custo quando o valor de D é $E[j]$ (para cada j tal que $0 \leq j < Q$).
- Esta função é chamada exatamente uma vez para cada caso de teste.

Restrições

- $1 \leq N \leq 100\,000$
- $1 \leq Q \leq 100\,000$
- $1 \leq W[i] \leq 10^9$ para cada i tal que $0 \leq i < N$
- $1 \leq B[i] < A[i] \leq 10^9$ para cada i tal que $0 \leq i < N$
- $1 \leq E[j] \leq 10^9$ para cada j tal que $0 \leq j < Q$

Subtarefas

Subtarefa	Pontos	Restrições Adicionais
1	6	$Q \leq 5$; $N \leq 2000$; $W[i] = 1$ para cada i tal que $0 \leq i < N$
2	13	$Q \leq 5$; $W[i] = i + 1$ para cada i tal que $0 \leq i < N$
3	17	$Q \leq 5$; $A[i] = 2$ e $B[i] = 1$ para cada i tal que $0 \leq i < N$
4	11	$Q \leq 5$; $N \leq 2000$
5	20	$Q \leq 5$
6	15	$A[i] = 2$ and $B[i] = 1$ para cada i tal que $0 \leq i < N$
7	18	Nenhuma restrição adicional.

Exemplo

Considera a seguinte chamada.

```
calculate_costs([15, 12, 2, 10, 21],
                [5, 4, 5, 6, 3],
                [1, 2, 2, 3, 2],
                [5, 9, 1])
```

Neste exemplo temos $N = 5$ artefactos e $Q = 3$ questões.

Na primeira questão, $D = 5$. Podes enviar os artefactos 0 e 3 num único barco (dado que $|15 - 10| \leq 5$) e os restantes artefactos em barcos separados. Isto dá o custo mínimo para

transportar todos o artefactos, que é de $1 + 4 + 5 + 3 + 3 = 16$.

Na segunda questão, $D = 9$. Podes enviar os artefactos 0 and 1 num único barco (dado que $|15 - 12| \leq 9$) e enviar os artefactos 2 e 3 num único barco (dado que $|2 - 10| \leq 9$). O artefacto que sobra pode ser enviado num barco separado. Isto dá o custo mínimo para transportar todos o artefactos, que é de $1 + 2 + 2 + 3 + 3 = 11$.

Na questão final, $D = 1$. Tens de enviar cada artefacto no seu próprio barco. Isto dá o custo mínimo para transportar todos o artefactos, que é de $5 + 4 + 5 + 6 + 3 = 23$.

Portanto, a função deve devolver $[16, 11, 23]$.

Avaliador Exemplo

Formato do input:

```
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]
```

Formato do output:

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

Portanto, S é o tamanho do array R devolvido por `calculate_costs`.