



## Задача АВ3. GPUs

Вие, като един модерен предприемач, сте основали generative AI startup. Процесът по генериране на изображения, текстове и т.н. е разбит на  $N$  задачи, всяка от които отнема по една секунда на по едно GPU (графична карта). Предварително знаете кога ще стане налична всяка от задачите – задача  $i$  в секунда  $T_i$ . Имате достъп до външен суперкомпютър с  $M$  на брой GPU-та, но всяко от тях си има различна цена за употреба на секунда – GPU  $j$  струва  $C_j$  на секунда. Трябва да насрочите всяка от задачите  $i$  за конкретно GPU  $j$  в конкретна секунда, така че тази секунда да не е преди  $T_i$  и да няма друга задача насрочена за същото време и GPU. Отново, **едно GPU работи по най-много една задача в дадена секунда.**

Нека финалното време на приключване (т.е. най-късното насрочено време, плюс едно) е  $F$ , а общата платена сума е  $S$ , т.е. ако задача  $i$  е насрочена за GPU  $G_i$ ,  $S = C_{G_1} + C_{G_2} + \dots + C_{G_N}$ . Трябва да намерите минималната възможна стойност на  $F \times S$ . Ще трябва да решите  $Q$  отделни, независими инстанции на задачата.

### Ограничения

- $1 \leq N \leq 10^7$
- $1 \leq M \leq N$
- $0 \leq T_i \leq N$
- $1 \leq C_i \leq 2N$
- $1 \leq Q \leq 5$

### Интеракция

Задачата е интерактивна. Вместо да четете и пишете от стандартните вход и изход, трябва само да напишете функция `solveGpus` със следния тип:

```
__int128 solveGpus(  
    std::vector<int>& gpuCosts,  
    std::vector<int>& reqTimes);
```

Двата вектора от стойности, които функцията получава, ще бъдат сортирани в **ненамаляващ ред**. Тя може да модифицира входните вектори. Функцията връща тип `__int128`, който представлява 128-битово целочислено число. Това е нужно, защото отговорът може да надхвърля лимитите на `long long`. Функцията може да се вика множество пъти. Всяко викане е независима инстанция на задачата.

Във Вашия код не трябва да има `main` функция, но може да има всякакви други помощни функции, класове, променливи и т.н. Кодът Ви трябва да включва `header` файла `gpus.h`, в който, за Ваше удобство, също е дефиниран и оператор за извеждане на стойности от типа `__int128`. Това става със следната директива към препроцесора:

```
#include "gpus.h"
```

Вашият код ще се компилира заедно с `greйдър`, който ще чете от входа и ще пише на изхода. На оценяващата система, единственото време, което се брои към времевия лимит, е времето прекарано в изпълнение на Вашия код, т.е. времето за вход и изход не се брои.



# НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА

Национален кръг

Хасково, 8-10 март 2024 г.

Група АВ, 9 – 12 клас, Ден 1

: 1.4 сек.  
: 1024 MB

За локално тестване са Ви предоставени локален грейдър `Lgrader.cpp` и копие на файла `gpus.h`. Трябва да компилирате Вашия код заедно с локалния грейдър, за да го тествате. Това може да стане като ги сложите в една папка и използвате командата:

```
g++ -O2 -std=c++17 -Wl,--stack,1073741824 -Wall gpus.cpp Lgrader.cpp  
-o gpus.exe
```

## Подзадачи

Подзадача	Точки	$N \leq$	$Q \leq$
1	10	10	1
2	8	800	2
3	13	2200	2
4	14	$10^4$	2
5	11	$10^5$	2
6	15	$10^6$	5
7	29	$10^7$	5

Точките за дадена подзадача се получават само ако се преминат успешно всички тестове, предвидени за нея.

## Пример

Примерна следва входния формат на локалния грейдър ( $Q$ , а след това, за всеки тест:  $N$ ,  $M$ , всички  $C_j$ , всички  $T_i$ ).

Вход	Изход	Обяснение на примера
1 8 4 1 2 2 6 0 0 0 0 1 2 2 2	39	Първите 3 задачи в секунда 0, следващите 2 в секунда 1 и последните 3 в секунда 2. $S = 13$ $F = 3$