НАЦИОНАЛЕН ЛЕТЕН ТУРНИР ПО ИНФОРМАТИКА
Стара Загора, 4 юни 2022 г.
Група А, 11 – 12 клас

Задача А1. Memory

Амнезиакът Филип има една торбичка с N билюри с разнообразни шарки, които често си брои. Важността на дадена шарка е равна на броя билюри с нея, а важността на даден билюр е равна на важността на шарката му. Филип иска (приблизително) да знае сумата от важностите на билюрите, която ще наречем H. Затова той всеки ден t вади билюрите един по един, като разглежда текущия, „обработва“ го (т.е. прави си някакви сметки наум), и продължава със следващия, но никога не се връща да погледне стар билюр. Накрая отново си смята нещо наум и казва приближението си за H, което ще наречем Q_t. Защо приближение? Ами, тъй като Филип амнезиак, може да помни много малко информация (до 32 байта). Това също е и причината да не помни нищо от предните дни. Той цели да разработи алгоритъм, който да му позволи да има относително точна оценка за H, като на първо място той приоритизира средната стойност на приближенията му да е близка до H, а на второ място – всяко отделно приближение да е близко до H.

Нека първо опишем какво точно вижда Филип и какво може да запомня. Той вади билюрите в произволен ред и ги номерира в ред на вадене от 0 до $N-1$. Шарките също имат произволни номера 0 до $N-1$, така че два билюра са от еднаква шарка тогава и само тогава, когато имат еднакви номера на шарките. Редът на вадене на билюри и номерацията на шарките се генерира произволно всеки ден, независимо един от друг и от всички предни дни; също така всички редове и номерации са еднакво вероятни. В началото Филип си избира колко числа (`int`-а) да помни (предпочита да са по-малко, за да не се напряга); нека наречем тази бройка S. След това в ден t той започва помнейки S нули. На всеки билюр Филип знае N, номера на текущия билюр, номера на шарката му и S-те числа, които е запомнил след предния билюр. Мисли малко и решава какви S числа да запомни за следващия билюр. Повтаря това докато не му свършат билюрите. Накрая той знае N и S-те числа, които е запомнил след последния билюр. Мисли малко и избира стойността на Q_t. Обърнете внимание, че Филип може да знае и разни константи, които не се променят, вместо да ги смята всеки път; също така той може да използва допълнителна „работна“ памет по време обработване на даден билюр и по време на смятането на Q_t.

Помогнете на Филип, като напишете програма `memory.h`, която да имплементира такъв алгоритъм. Тази програма ще се компилира с програма на журито.

Детайли по имплементацията

Вашият код ще стои в хедър файл, който ще бъде включен в „средата“ на кода на журито. Над него ще стоят следнтие дефиниции:

```
using uint = unsigned int;
using ull = unsigned long long;
constexpr uint NUM_TESTS = 10000;
constexpr uint MAX_STATE_SIZE = 8;
using State = std::array<uint, MAX_STATE_SIZE>;
```

НАЦИОНАЛЕН ЛЕТЕН ТУРНИР ПО ИНФОРМАТИКА
Стара Загора, 4 юни 2022 г.
Група А, 11 – 12 клас

```
constexpr uint stateSize(uint n);
constexpr void processOne(uint n, uint i, int curr, State& state);
constexpr uint getAnswer(uint n, const State& state);
```

Вие трябва да имплементирате тези три `constexpr` функции. Спецификацията `constexpr` за функция означава, че тя няма (или по-точно може да няма) странични ефекти, т.е. няма да използва глобални не-`constexpr` променливи и фунцкии, няма да въвежда или извежда и т.н. Може да използват локални променливи или (потенциално Ваши) глобални `constexpr` константни, стига те да са инициализирани при създаването си, също може да викат други (потенциално Ваши) `constexpr` функции, да имат `if`-ове и цикли и всички други такива прости конструкции.

Функцията Ви `stateSize` трябва да връща S, число между 1 и 8. Функцията Ви `processOne` ще бъде викана по веднъж за всеки билюр в даден ден; `i` е номерът на билюра (т.е. стойностите му ще бъдат числата от 0 до $N-1$, в този ред); `curr` е номерът на шарката на билюра; `state` е запомненото състояние, което в началото на всеки ден е масив от нули и, с цел да може да ползвате само S `int`-а памет, програмата на журито ще занулява всички елементи на `state` освен първите S след всеки билюр. Функцията Ви `getAnswer` трябва да върне Q_t. Общо има $T = 10^4$ дни.

Тъй като C++ не задължава `constexpr` функциите да са винаги `constexpr`, програмата на журито ще съдържа един вграден тест, на който програмата Ви ще бъде „тествана“ по време на компилация и няма да се компилриа успешно, ако не е `constexpr` на него. Забележете, че е възможно умишлено да напишете програма, която се компилира, но не е винаги `constexpr`, т.е. понякога използва глобални променливи. При откриване на такова решение, то ще се анулира, а при повторни нарушения, ще бъдете дисквалифицирани.

Тъй като програмата на журито, ефективно пуска Вашата програма на пълен тест по време на компилацията е възможно (но много трудно) самата компилация да има `ML` (или `TL`). Това би се случило, ако програмата Ви и без това би имала `TL` или `ML` или има някакви много дълбоки рекурсии.

Ограничения

$N = 10^3$ винаги
$0 \leq \mathrm{i}, \mathrm{curr} < N$

Оценяване

На всеки тест получавате резултат от 0 до 1. Точките Ви за дадена подзадача са равни максималните точки за нея умножени по минималния резултат на тест от подзадачата. Сега нека да видим как се смята резултатът Ви на даден тест.

Първо:

$$E = \sum_{0 \leq t < T} \frac{Q_t - H}{T}$$

$$R = \sqrt{\sum_{0 \leq t < T} \frac{(Q_t - H)^2}{T}}$$

Или иначе казано, E е средната грешка на програмата Ви, а R е корен квадратен от средната ѝ грешка на квадрат. Целта Ви е да имате ниски $|E|$ и R. По-точно и двете се разглеждат спрямо H и също вземаме предвид S:

$$U = \left|\ln\left(\frac{1+H+E}{1+H}\right)\right| \times \sqrt{S}$$

$$V = \ln\frac{\left(1+R\times\sqrt{S}\right)}{\ln(H)}$$

Интуитивно, U мери линейната Ви грешка в лог скала, т.е. $Q_t = \frac{H}{2}$ и $Q_t = 2H$ са еднакво лоши. V пък мери квадратната Ви грешка като H на някаква степен; игнорирайки S и съображения за $-\infty$, V ефективно мери $\log_H(R)$. И двете мерки пенализират допълнително и с $\sqrt{S}$, т.е. да използвате четири пъти повече памет за двойно по ниски грешки не променя резултата Ви. Сега нека дефинираме $B(x) = \min(\max(x, 1), 0)$. Резултатът Ви е:

$$B(3^{0.005-U}) \times \left(0.35 + 0.65 \times B\left(1 - \frac{V-0.85}{0.3}\right)\right)$$

С други думи, за да имате пълен резултат, трябва $U \leq 0.005$ и $V \leq 0.85$. Ще имате резултат 0.35, ако $U \leq 0.005$ и $V \geq 1.15$. Ако $U > 0.005$, резултатът Ви намалява експоненциално с $U - 0.005$.

Локално тестване

Предоставени са Ви файловете `Lgrader.cpp`, `Lgrader2.cpp` и `memory_example.h`. Lgrader.cpp наподобява системния грейдър и може да ползвате, за да тествате програмата си. Това става като компилирате само него, докато Вашето решение `memory.h` е в същата папка. Въвеждат му се броя шарки и сийд, а след това се въвеждат бройките билюри от всяка шарка. Той ще изведе E, R, U, V и резултата на програмата Ви на този тест. `Lgrader2.cpp` е еквивалентен, но не кара функциите Ви да са `constexpr`, т.е. може да го ползвате докато ги разработвате, за да си извеждате неща и т.н. `memory_example.h` е „легална“ примерна програма, която показва какви неща можете да правите с `constexpr`.

Подзадачи

Номер	Точки	Ограничение
1	20	Ще има приблизително равна бройка билюри от всяка шарка (с над нула билюри). Т.е. съществува число X, такова че от всяка шарка има или нула, или X, или $X+1$ билюра.
2	80	Няма

Подзадача 2 включва тестовете от подзадача 1.