НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА
Национален кръг
София, 10 - 12 март 2023 г.
Група АВ, 9 - 12 клас

Задача АВ2. ПРАВЕЦ

Дени получи доста странен подарък за рождения си ден – нов компютър. Това не изглежда толкова лошо, но когато тя отваря кутията с изненада вижда Правец 386 (стар модел български компютри с 32 бита от преди над 32 години). **Д**ени е безнадежден оптимист и затова решава, че все пак ще успее да намери приложение на новата си придобивка – пробва да напише проста програмка на него (и без това много повече занимания на стария Правец няма). Програмата единствено трябвало да подреди масива $a_0, a_1, \ldots, a_{N-1}$ в нарастващ ред. Но поради ограничените ресурси (RAM памет от само 1 MB), това се оказало почти непосилна задача за **Д**ени и тя се свързва с Вас с молбата да ѝ помогнете.

Задача

Напишете програма **pravetz**, която да помага на **Д**ени да сортира масив, използвайки Правец. Програмата Ви трябва да съдържа функцията `sort_array`, която ще се компилира с програмата на журито.

Можем да представим паметта на компютъра като поредица от клетки с индекси 0, 1, …, 2***N***. Масивът, който програмата въвежда, е записан върху клетките с индекси 0, 1, …, ***N***-1. Останалите клетки – с индекси ***N***, ***N***+1, …, 2***N*** наричаме допълнителна памет. В началото те са празни и тяхната стойност е 0. Компютърът Правец има фабричен дефект и често се рестартира от само себе си. Затова вашата цел е да сортирате масива чрез много отделни пускания на компютъра, всяко от което чете възможно най-малък на брой клетки и по възможност не ползва допълнителната памет. Всяка от тези стъпки ще е реализирана чрез повикване на вашата функция `sort_array`. Тази функция може да чете и записва стойности в паметта на Правец, също да използва локални променливи, но не бива да използва глобална памет.

Детайли по реализацията

Функцията `sort_array` трябва да има следния формат:

```
bool sort_array (int N, bool start);
```

Тя ще бъде извиквана многократно, симулирайки вашия алгоритъм стъпка по стъпка. Първият параметър е размерът на масива, а вторият е със стойност `true` за първото извикване на вашата функция и `false` в останалите случаи. Резултатът, който връща вашата функция трябва да е `true`, ако сте приключили алгоритъма или `false` в противен случай.

Всеки тест ще се състои от един масив за сортиране. В рамките на един тест, вашето решение ще бъде симулирано няколко пъти върху масива, като освен това тези симулации могат да са и на различни изпълнения на вашето решение – възможно е в рамките на една и съща симулация да се извикват `sort_array` от различни изпълнения (разбира се предоставената памет е консистентна за една и съща симулация, независимо от това кое изпълнение се използва).

Във функцията `sort_array` имате достъп до цялата памет (тоест от индекс 0 нататък) чрез функциите `get_memory` и `set_memory` (**не трябва да ги дефинирате сами**):

```
unsigned int get_memory (int index);
void set_memory (int index, unsigned int value);
```

НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА
Национален кръг
София, 10 - 12 март 2023 г.
Група АВ, 9 - 12 клас

Извикването `get_memory(i)`, връща стойността на *i*-тата (броейки от 0) променлива от паметта за текущата симулация. Извикването `set_memory(i, x)` присвоява стойност на *i*-тата променлива от паметта за текущата симулация равна на *x*.

Последното ограничение за алгоритъма ви е, че при изпълнението му при различни симулации над един и същ масив, той трябва да работи по един и същ начин – трябва да прави една и съща поредица от извиквания на `get_memory` и `set_memory` в двете симулации. **Това означава, че всякакво съществено ползване на глобална памет свързано с различни извиквания на `sort_array` би довело до грешно решение и оценка 0 за съответния тест.**

Вие трябва да предадете към системата файл **pravetz.cpp**, който съдържа функцията `sort_array`. Той може да съдържа други функции и код, необходими на програмата Ви, но **не трябва** да съдържа главната функция `main`. Също така, не трябва да четете от стандартния вход или да пишете на стандартния изход! Програмата Ви също така трябва да включва хедър файла **pravetz.h** чрез указание към препроцесора в началото на Вашия файл:

```
#include "pravetz.h"
```

Ограничения

- $20 \leq N \leq 2^{16}$
- $0 \leq a_i < 2^{16}$

Оценяване

- Тестовете са групирани в групи от по 3 теста, като резултатът за дадена група е равен на минималната оценка на тест от нея, умножен по броя точки за групата. Сега ще опишем как се получава оценката за даден тест.
- Ако вашата функция не сортира масива правилно, то ще получите оценка 0 за съответния тест. Освен това ако за приключване на вашето решение, функцията `sort_array` е извикана над 10^7 на брой пъти, то ще получите също оценка 0.
- Ако сте спазили горните изисквания, то решението ви ще се оцени по следния начин.
- Нека *extra* е равно на 1, ако вашата функция не използва допълнителната памет, а ако използва, *extra* е равно на константата *c* за съответната подзадача.
- Нека с *gets* сме означили максималния брой извиквания на `get_memory` от някое извикване на вашата функция `sort_array`, като тази бройка включва и първите N клетки (за ваше улеснение се броят само **различните клетки**, тоест ако извикате два пъти `get_memory(1)` в рамките на едно извикване на `sort_array` това ще допринесе само с 1 към бройката).
- Тогава оценката за един тест се смята по формулата: $extra \times \min\left(1, \frac{num}{gets}\right)$, където *num* е константа за съответната подзадача.

Подзадачи

Подзадача	Точки	N	a_i	c	num	Допълнителни ограничения
1	0	–	–	= 0.7	= 10	Примерът.
2	10	$\leq 2^{10}$	$< N$	= 0.7	= 10	–
3	30	$\leq 2^{14}$	$< N$	= 0.3	= 3	–
4	15	$\leq 2^{16}$	$< N$	= 0.3	= 3	–
5	45	$\leq 2^{14}$	$< 2^{16}$	= 0.3	= 4	–

Групите се оценяват индивидуално.

НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА
Национален кръг
София, 10 - 12 март 2023 г.
Група АВ, 9 - 12 клас

Локално тестване

За локално тестване е предоставен файлът **Lgrader.cpp** и **pravetz.h**. Сложете го в същата папка, в която е Вашият файл **pravetz.cpp**, и компилирайте заедно **Lgrader.cpp** и **pravetz.cpp**. Така ще получите програма, с която ще проверите решението си. Програмата ще изисква от стандартния вход следната последователност от данни:

- на първия ред: едно положително число – големината на масива N
- на следващия ред – числата $a_0, a_1, \ldots, a_{N-1}$ на масива, разделени с интервали.

Примерна комуникация

Действие на вашата програма	Действия и отговори на журито	Обяснение
	`sort_array(5,true)`	Извиква се вашата функция, за да започнете работа. Тук масивът е 2, 1, 1, 0, 4.
`get_memory(0)`	2	
`get_memory(3)`	0	Тук виждаме, че тези числа не са подредени и ще ги разменим със следващите действия.
`set_memory(5,2)`	Клетката с индекс 5 става със стойност 2.	Паметта става: 2, 1, 1, 0, 4, 2, 0, 0, 0, 0, 0.
`set_memory(0,0)`	Клетката с индекс 0 става със стойност 0.	Паметта става: 0, 1, 1, 0, 4, 2, 0, 0, 0, 0, 0.
`return false`		Тук прекратяваме текущото извикване и връщаме стойност, с която указваме, че не сме приключили.
	`sort_array(5,false)`	
`get_memory(5)`	2	
`set_memory(3,2)`	Клетката с индекс 3 става със стойност 2.	Паметта става: 0, 1, 1, 2, 4, 2, 0, 0, 0, 0, 0.
`return true`		Тук прекратяваме текущото извикване и връщаме стойност, с която указваме, че сме приключили. За описаната комуникация максималният брой извикване на `get_memory` е 2. Затова *gets* = 2. Понеже началният масив вече е успешно сортиран, но сме използвали допълнителната памет, то оценката за този тест е: $0.7 * \min\left(1, \frac{10}{2}\right) = 0.7$, където константата *num* е 10.