**Задача А2. Плейлисти**

Цвети прекарва голяма част от времето си в това да създава плейлисти. След толкова опит в тази сфера, тя има чувството, че може да разпознае всеки свой плейлист само от няколко произволни песни в него. За да тества тази си способност тя играе игра, която протича по следния начин:

Първоначално Цвети получава следните стойности:

* – брой плейлисти, които трябва да създаде. Плейлистите са номерирани с целите числа от 0 до .
* – броя песни, който трябва да има във всеки плейлист.
* *–* броя различни песни, които Цвети може да използва. Песните са номерирани с целите числа от 0 до .
* – дробно число между 0.1 и 0.95, равно на желаната точност в познаването на плейлисти.

След това Цвети създава плейлиста с по песни във всеки, използвайки -те песни, които има. **Една песен може да присъства повече от веднъж в даден плейлист.** След като плейлстите са създадени, протичат множество рундове в следния формат:

1. Избира се произволен плейлист и песните в него се разбъркват на случаен принцип. Цвети не знае нито кой е плейлиста, нито разбърканата подредба на песните в него.
2. Цвети започва да слуша плейлиста, като може да изслуша от 0 до всички песни в него. Във всеки един момент тя може да реши да приключи рунда като даде предположение за кой е плейлистът.

Играта продължава точно 20 000 рунда (Цвети има твърде много свободно време). Счита се, че Цвети е спечелила, ако частта правилно познати плейлисти е поне (т.е. ако са познати поне плейлиста). Това е лесна игра за Цвети, затова тя иска не просто да спечели, а да го направи максимално бързо. **Оценката за една игра е средния брой изслушани песни при рундовете, в които плейлистът е познат правилно.** Рундове, в които предположението на Цвети е грешно, не участват в резултата от играта.

Помогнете на Цвети, като напишете програма playlists.cpp, която играе от нейно име. Тази програма ще се компилира с програма на журито.

**Детайли по имплементацията**

За задачата трябва да имплементирате две функции. Първата Ваша функция makePlaylists трябва да има следния прототип:

std::vector<std::vector<int>>

makePlaylists(int n, int k, int s, double p);

Тя ще бъде извикана точно веднъж преди всякакви извиквания на другата Ви функция. Като аргументи ѝ се подават параметрите от задачата. Функцията трябва да върне избраните плейлисти под формата на списък с дължина , чийто всеки елемент е списък с дължина , и всички стойности във вътрешните списъци са числа от 0 до .

Втората Ваша функция guessPlaylist трябва да има следния прототип:

int guessPlaylist();

Всяко извикване на тази функция от програмата на журито индикира начало на нов рунд от играта. Функцията трябва да върне число от 0 до , предположението за скрития плейлист. Плейлистите са индексирани в реда, в който биват върнати от makePlaylists. **Функцията ще бъде извикана по точно 20 000 пъти за всеки тест.** При всеки рунд Вие имате достъп до функцията nextSong на журито, която има следния прототип:

int nextSong();

Функцията връща следващата песен от разбъркания скрит плейлист, избран за съответния рунд. В рамките на един рунд Вашата програма може да извика тази функция от 0 до пъти. Извикване на функцията повече от пъти в рамките на един рунд се счита за грешка. Сложността на функцията е .

Вашата програма трябва да имплементира функциите makePlaylists и guessPlaylist, но не трябва да съдържа функция main. Освен това, тя не трябва да чете от стандартния вход или да печата на стандартния изход. Програмата Ви също така трябва да включва хедър файла playlists.h чрез указание към препроцесора:

#include "playlists.h"

Стига да спазва тези условия, програмата Ви може да съдържа каквито и да е помощни функции, променливи, константи и прочие.

**Ограничения**

Гарантирано е, че стойностите на , , и винаги позволяват да бъде спечелена.

**Оценяване**

Всеки тест се оценява поотделно. За да получите точки за даден тест, частта правилно познати плейлисти трябва да е поне (т.е. от всичките 20000 рунда, трябва в поне да определите плейлиста правилно). Броят точки, който ще получите, зависи от средния брой изслушани песни **в рундовете, в които плейлистът е правилно определен**. Нека тази бройка е за вашето решение и за авторовото решение. Тогава, ако , ще получите пълния брой точки за теста, а иначе ще получите от точките за теста.

*Забележка: Резултатът Ви за задачата е равен на резултата на най-добрия Ви събмит.*

**Локално тестване**

Предоставени са Ви файловете playlists.h и Lgrader.cpp, които можете да компилирате заедно с Вашата програма, за да я тествате. При стартиране на програмата трябва да се въведат числата . След това решението Ви ще се изпълни и ще се изведе резултатът му или описание на грешка, ако възникне такава.

**Примерна комуникция**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Действия на playlists | Действия и отговори на журито | Обяснение |
| 1. | {{0, 1}, {1, 1}, {2, 2}} | makePlaylists(3, 2, 3, 0.5) |  |
| 2. |  | guessPlaylist() | Журито избира плейлист 0 и го разбърква до {1, 0} |
| 3. | nextSong() | return 1 |  |
| 4. | nextSong() | return 0 |  |
| 5. | return 0 |  | Плейлистът е определен вярно след изслушване на 2 песни |
| 6. |  | guessPlaylist() | Журито избира плейлист 2 и го разбърква до {2, 2} |
| 7. | nextSong() | return 2 |  |
| 8. | return 2 |  | Плейлистът е определен вярно след изслушване на 1 песен |
| 9. |  | guessPlaylist() | Журито избира плейлист 0 и го разбърква до {0, 1} |
| 10. | nextSong() | return 1 |  |
| 11. | return 1 |  |  |

**Оценяване на примерната комуникация**

В примера са разиграни 3 рунда, но във всеки тест те ще бъдат точно 20 000. От тези 3 рунда, 2 определят правилно плейлиста, което е успеваемост от , по-висока от желаната , т.е. тестът е успешно решен. Тъй като броим слушанията само за рундове с познат плейлист, то това решение ще има изслушани средно по песни на успешен рунд. Ако приемем, че авторовото решение решава този тест със средно 1 изслушана песен, то решението би получило от точките за теста.