Zadanie: KON
Konewka

Olimpiada Informatyczna

XX OI, etap II, dzień drugi. Plik źródłowy `kon.*` **Dostępna pamięć: 128 MB.** *14.02.2013*

Zostałeś ogrodnikiem u królowej Bajtoliny. Wspaniale, prawda? Skoro tak uważasz, to chyba jeszcze nie wiesz wszystkiego o tej pracy. Obok zamku królowej znajduje się wielki ogród z n drzewami ustawionymi po kolei jedno za drugim. To jeszcze nic strasznego, ale czy potrafisz o każdej porze dnia i nocy odpowiedzieć swojej władczyni, które z jej drzewek są teraz dojrzałe? Zakładamy, że drzewo jest dojrzałe, gdy ma przynajmniej k bajtymetrów wysokości.

Czasem królowa prosi Cię, abyś niektóre z jej drzewek podlał za pomocą magicznej konewki. Każda taka operacja powoduje, że wszystkie podlane drzewa rosną o dokładnie jeden bajtymetr.

Udowodnij, że nadajesz się do tej pracy i szybko odpowiedz na wszystkie pytania królowej!

Komunikacja

Napisz bibliotekę komunikującą się z programem oceniającym. Powinna ona zawierać przynajmniej następujące trzy funkcje, wywoływane przez program oceniający:

- `inicjuj(n, k, D)` – ta funkcja zostanie wywołana dokładnie raz, na początku sprawdzania. Możesz ją wykorzystać do inicjalizacji swoich struktur danych. Przyjmuje jako parametry liczbę drzew n, dolne ograniczenie wysokości dojrzałego drzewa k oraz tablicę D o długości n zawierającą początkowe wysokości wszystkich drzew. Drzewa są ponumerowane kolejnymi liczbami całkowitymi od 0 do $n-1$.
  - **C/C++:** `void inicjuj(int n, int k, int *D);`
  - **Pascal:** `procedure inicjuj(n, k : LongInt; var D : array of LongInt);`
- `podlej(a, b)` – oznacza, że królowa poprosiła Cię o podlanie wszystkich drzew od a-tego do b-tego włącznie ($0 \leq a \leq b \leq n-1$). Wywołanie tej procedury/funkcji oznacza, że każde z tych drzew rośnie o 1 bajtymetr.
  - **C/C++:** `void podlej(int a, int b);`
  - **Pascal:** `procedure podlej(a, b : LongInt);`
- `dojrzale(a, b)` – królowa pyta Cię, ile spośród drzew o numerach od a do b włącznie ($0 \leq a \leq b \leq n-1$) jest już dojrzałych.
  - **C/C++:** `int dojrzale(int a, int b);`
  - **Pascal:** `function dojrzale(a, b : LongInt) : LongInt;`

Twoja biblioteka **nie może** czytać żadnych danych (ani ze standardowego wejścia, ani z plików). **Nie może** również nic wypisywać do plików ani na standardowe wyjście. Może pisać na standardowe wyjście diagnostyczne (`stderr`) – pamiętaj jednak, że zużywa to cenny czas.

Jeśli piszesz w C/C++, Twoja biblioteka **nie może** zawierać funkcji `main`. Jeśli piszesz w Pascalu, powinieneś dostarczyć moduł (patrz przykładowe programy na Twoim dysku).

Twoje rozwiązanie uzyska punkty za dany test tylko wtedy, gdy spełni określone powyżej wymogi techniczne oraz odpowie poprawnie na wszystkie pytania królowej.

Ograniczenia

We wszystkich testach zachodzą następujące warunki:

- $1 \leq n \leq 300\,000$
- $1 \leq k \leq 10^9$
- łączna liczba wywołań procedury/funkcji `podlej` i funkcji `dojrzale` nie przekracza 300 000
- początkowe wysokości wszystkich drzew to dodatnie liczby całkowite nieprzekraczające 10^9.

W testach wartych łącznie 20% punktów zachodzą dodatkowe warunki:

- $n \leq 2\,000$
- łączna liczba wywołań procedury/funkcji `podlej` i funkcji `dojrzale` nie przekracza 10 000

W testach wartych łącznie 50% punktów wszystkie wywołania procedury/funkcji `podlej` następują przed wszystkimi wywołaniami funkcji `dojrzale`.

Kompilacja

Twoja biblioteka – `kon.c`, `kon.cpp` lub `kon.pas` – zostanie skompilowana z programem oceniającym przy użyciu następujących instrukcji:

- **C:** `gcc -O2 -static -lm kon.c kongrader.c -o kon`
- **C++:** `g++ -O2 -static -lm kon.cpp kongrader.cpp -o kon`
- **Pascal:**
  ```
  ppc386 -O2 -XS -Xt kon.pas
  ppc386 -O2 -XS -Xt kongrader.pas
  mv kongrader kon
  ```

Przykładowe wykonanie

Poniższa tabela zawiera przykładowy ciąg wywołań funkcji oraz poprawne wyniki funkcji `dojrzale`.

Wywołanie	Wynik	Wyjaśnienie
`inicjuj(4, 6, D)` (`D[0]=5`, `D[1]=4`, `D[2]=3`, `D[3]=7`)		Mamy $n = 4$ drzew o wysokościach 5, 4, 3 i 7, a $k = 6$.
`dojrzale(2, 3)`	1	Ile dojrzałych drzew jest na przedziale $[2, 3]$?
`podlej(0, 2)`		Podlej drzewa 0, 1 i 2.
`dojrzale(1, 2)`	0	Ile dojrzałych drzew jest na przedziale $[1, 2]$?
`podlej(2, 3)`		Podlej drzewa 2 i 3.
`podlej(0, 1)`		Podlej drzewa 0 i 1.
`dojrzale(0, 3)`	3	Ile dojrzałych drzew jest na przedziale $[0, 3]$?

Testowanie

W katalogu **/home/zawodnik/kon/** na dysku Twojego komputera możesz znaleźć przykładowy program oceniający (`kongrader.c`, `kongrader.cpp` i `kongrader.pas`). Żeby uruchomić program oceniający z Twoją biblioteką, powinieneś umieścić ją w pliku `kon.c`, `kon.cpp` lub `kon.pas` w odpowiednim katalogu (`c`, `cpp` lub `pas`). Na początku zawodów znajdziesz tam przykładowe, błędne rozwiązania tego zadania. Program oceniający wraz z Twoją biblioteką możesz skompilować za pomocą polecenia:

```
make kon
```

które działa dokładnie tak, jak opisano w sekcji *Kompilacja*. Kompilacja rozwiązania w C/C++ wymaga pliku `koninc.h`, który również znajduje się w odpowiednich katalogach.

Tak skompilowany program oceniający wczytuje ze standardowego wejścia specjalnie przygotowany opis testu, wywołuje odpowiednie funkcje Twojej biblioteki, a wyniki wypisuje na standardowe wyjście.

Opis testu powinien być w następującym formacie. Pierwszy wiersz testu składa się z dwóch liczb całkowitych n i k. Drugi wiersz zawiera n liczb oznaczających wysokości początkowe kolejnych drzew. Trzeci wiersz zawiera liczbę q. Dalej następuje q wierszy, z których każdy zawiera pojedynczy znak `p` lub `d` oraz dwie nieujemne liczby całkowite. Znak określa, która funkcja powinna być wywołana: `p` dla `podlej` i `d` dla `dojrzale`, zaś liczby – z jakimi argumentami należy tę funkcję wywołać. Funkcja `inicjuj` zostanie wywołana od razu po wczytaniu pierwszych dwóch wierszy. Pamiętaj jednak, że przykładowy program oceniający nie sprawdza, czy dane są sformatowane poprawnie ani czy spełnione są ograniczenia podane w sekcji *Ograniczenia*.

W katalogu `/home/zawodnik/kon/` znajdziesz plik `kon0.in`, który odpowiada przykładowemu wykonaniu programu opisanemu powyżej. Aby uruchomić program oceniający na podanym przykładzie, użyj następującego polecenia:

```
./kon < kon0.in
```

Wyniki wywołań funkcji `dojrzale` zostaną wypisane na standardowe wyjście. Poprawny wynik dla powyższego przykładowego wykonania znajdziesz na dysku w pliku `kon0.out`.

Aby sprawdzić, czy wynik wypisany przez Twoje rozwiązanie jest prawidłowy, możesz również wysłać rozwiązanie (Twoją bibliotekę) do SIO.

Testy „ocen”:

`1ocen:` $n = 2\,000$, łączna liczba wywołań procedur/funkcji `podlej` i `dojrzale` wynosi 10 000, wszystkie te wywołania dotyczą pojedynczych drzew, początkowe wysokości drzew losowe z przedziału $[1; 100]$, $k = 100$;

`2ocen:` $n = 100\,000$, najpierw 100 000 wywołań procedury/funkcji `podlej`, dotyczące wszystkich drzew, potem 100 000 wywołań funkcji `dojrzale` o pojedyncze drzewa, początkowe wysokości drzew losowe z przedziału $[1; 1\,000]$, $k = 100\,500$.