Всероссийская олимпиада школьников по информатике
Региональный этап, I тур
Разбор задач

Андрианов И. А.
Стрекаловский О. А.

Вологодский государственный университет
Факультет прикладной математики,
компьютерных технологий и физики

Вологда
2015 г.

«Выбор зала»

Пусть длина наименьшей стороны x, а большей — y.
Тогда заметим, что должны выполняться следующие ограничения:
$y \geq x$
$x * y \geq A \Rightarrow y \geq A/x$
$2 * (x + y) \geq C \Rightarrow y \geq C/2 - x$
Таким образом, минимально подходящее y равно:
$y_{min} = \max\{x, \lceil A/x \rceil, \lceil C/2 - x \rceil\}$

«Выбор зала »

$x * y \geq B \Rightarrow y \geq B/x$
$2 * (x + y) \geq D \Rightarrow y \geq D/2 - x$
Таким образом, максимально подходящее y равно:
$y_{max} = \min\{\lfloor B/x \rfloor, \lfloor D/2 - x \rfloor\}$

«Выбор зала»

- Переберём меньшую сторону зала до $\sqrt{B}$
- Для каждого x вычислим y_{min} и y_{max} и прибавим к ответу $\max\{0, y_{max} - y_{min} + 1\}$

«Выбор зала»

Типичные ошибки

- Перебор стороны зала до B, вместо перебора до $\sqrt{B}$
- Использование 32-х разрядного типа данных
- Округления не в ту сторону
- Не обработали случай $y_{min} > y_{max}$

Вопросы по задаче?

«Призы»

Воспользуемся префиксными суммами и префиксными максимумами.

- Научимся за $O(1)$ находить сумму значений ценности для любого отрезка номеров
  - Создадим массив префиксных сумм s.
    $s[i] := (a_1 + a_2 + \ldots + a_i)$ для всех i от 0 до n
  - Это можно сделать за один проход по всем a_i
- Теперь сумма значений a_i на отрезке от L до R вычисляется как $s[R] - s[L-1]$

«Призы»

Для всех i от 1 до n вычислим массивы префиксных и суффиксных максимумов

- $pref[i] =$
  $\max\{s[k] - s[0], s[k+1] - s[1], \ldots, s[i] - s[i-k]\}$
- $suff[i] =$
  $\max\{s[i+k-1] - s[i-1], s[i+k] - s[i], \ldots, s[n] - s[n-k]\}$
- $pref$ и $suff$ можно вычислить за линейный проход по s

«Призы»

Теперь для решения задачи достаточно перебрать ход Алисы

- Если Алиса выбрала отрезок призов $[A, A + k - 1]$, то максимальное значение, которое может достаться Бобу будет
  $\max\{pref[A - 1], suff[A + k]\}$

Выбрать минимум из всех возможных максимальных выигрышей Боба

«Призы»

Типичные ошибки

- Неэффективное решение.
  Много проходов по исходным данным.
- Выходы за границы массивов
- Использование 32-х разрядного типа данных

Вопросы по задаче?

«Река»

Общий подход к хранению информации о предприятии для частичных решений

- Используем двухсвязный список
- Для предприятия с номером v будем хранить
  $next[v]$ — номер следующего предприятия и
  $prev[v]$ — номер предыдущего предприятия вдоль реки
- При банкротстве предприятия соответствующий ему элемент списка удаляется, а при разделении он удаляется и в соответствующее место списка вставляются номера новых предприятий

«Река»

Подсчет суммы квадратов длин отрезков реки

Будем хранить в ans текущую сумму квадратов длин отрезков реки, а в $len[i]$ — длину отрезка реки i-ого предприятия

- Считаем ans во время заполнения len по входным данным и выводим ans как ответ до обработки запросов

«Река»

Подсчет суммы квадратов длин отрезков реки

Пересчитываем ans и len при событиях

- При банкротстве предприятия с номером v:
  $ans- = (len[v])^2$
  $ans- = (len[next[v]])^2$
  $ans- = (len[prev[v]])^2$
  $len[prev[v]]+ = len[v]/2$
  $len[next[v]]+ = (len[v]+1)/2$
  $ans+ = (len[next[v]])^2$
  $ans+ = (len[prev[v]])^2$
- При разделении предприятия на два с номером v:
  $ans- = (len[v])^2$
  $len[first] = len[v]/2$
  $len[second] = (len[v]+1)/2$
  $ans+ = (len[first])^2$
  $ans+ = (len[second])^2$

«Река»

Алгоритм работы частичного решения для 1–ой и 2–ой подзадачи

- Создаём двунаправленный список по входным данным, заполяем ans и len
- При запросе проходимся по списку и за $O(N)$ для 1–ой подзадачи или за $O(1)$ для 2–ой подзадачи и ищем нужное предприятие
- Обрабатываем добавление/удаление предприятий в списке, пересчитываем ans и len за $O(1)$

Происходят только банкротства $\Rightarrow$новых предприятий не создается.
Для поиска и удаления i–ого предприятия за $O(\ln N)$ можно использовать структуры данных:

- Дерево отрезков[1]
- Дерево Фенвика[2]

[1] http://e-maxx.ru/algo/segment_tree
[2] http://e-maxx.ru/algo/fenwick_tree

Декартово дерево по неявному ключу[3]

- Удаление, вставка и поиск i–ого элемента за $O(\ln N)$

SQRT–декомпозиция[4]

- Удаление, вставка и поиск i–ого элемента за $O(\sqrt{N})$

[3] http://e-maxx.ru/algo/treap
[4] http://e-maxx.ru/algo/sqrt_decomposition

«Река»

Типичные ошибки

- Неэффективное решение.
  Много проходов по исходным данным, копирование массивов
- Отсутствие контроля при поиске предыдущего и следующего предприятия (их может не быть)
- Использование 32-х разрядного типа данных

Вопросы по задаче?

«Чемпионат по поиску в сети Меганет»

Основные идеи решения:

- Заметим, что имя сервера и имя раздела состоят не более чем из 5и частей каждое $\Rightarrow$ для каждого адреса существует лишь небольшое число фильтров, под которые он потенциально может подходить
- Разместив строки с фильтрами в структуре данных, в которой возможен быстрый поиск и добавление за $O(\ln N)$ или $O(1)$, мы можем быстро проверять наличие фильтра
  - Префиксное дерево (Trie)[5]
  - Стандартные структуры данных в Java, C++, Python, C#
  - Самописные хэш–таблицы для быстрого поиска строк
- В решении следует учитывать возможность появления фильтров–дубликатов во входных данных

[5] https://en.wikipedia.org/wiki/Trie

Пусть дан сервер "a.b.c.d.e".
Возможны следующие варианты:

- a.b.c.d.e
- *.a.b.c.d.e
- *.b.c.d.e
- *.c.d.e
- *.d.e
- *.e

Итого 6 вариантов фильтров для сервера

Пусть в сервере задан путь "a/b/c/d/e".
Возможны следующие варианты:

- a/b/c/d/e
- a/b/c/d/e/*
- a/b/c/d/*
- a/b/c/*
- a/b/*
- a/*
- /*

Итого 7 вариантов фильтров для пути в сервере

«Чемпионат по поиску в сети Меганет»

Итоговый алгоритм:

- Сохраняем все фильтры в структуру данных
- Разделяем строку на название сервера и путь
- Для каждого названия сервера и пути строим список подходящих фильтров
- Перебираем все комбинации
  "фильтр для сервера"+"фильтр для пути"
  и ищем эту строку среди фильтров

«Чемпионат по поиску в сети Меганет»

Типичные ошибки

- Неэффективное хэширование при поиске строк
- Неучитывание фильтров–дубликатов

«Чемпионат по поиску в сети Меганет»

Типичные ошибки

- Неэффективное хэширование при поиске строк
- Неучитывание фильтров–дубликатов

Решение на C++

- Неэффективая работа при использовании стандартного контейнера map/unordered_map для типа string
- Медленный ввод-вывод данных через cin и cout

«Чемпионат по поиску в сети Меганет»

Типичные ошибки

- Неэффективное хэширование при поиске строк
- Неучитывание фильтров–дубликатов

Решение на C++

- Неэффективая работа при использовании стандартного контейнера map/unordered_map для типа string
- Медленный ввод-вывод данных через cin и cout

Решение на Java

- На стандартном HashMap решение проходит без дополнительной оптимизации

Вопросы по задаче?

Спасибо за внимание!