Открытая олимпиада Университета Иннополис для школьников по программированию

18-19 апреля 2015 года

Иннополис

Сисамдин

- Идея задачи — Тимур Хисматуллин
- Подготовка тестов — Тимур Хисматуллин
- Разбор задачи — Тимур Хисматуллин

Постановка задачи

Даны компьютеры, соединенные в виде двоичного дерева. Каждый ход соединяют два компьютера.

Необходимо после каждого хода сообщить количество «важных» компьютеров – тех, при отключении которых найдутся 2 компьютера, которые не соединены.

Переформулирование

Дан граф, представляющий собой двоичное дерево. Каждый ход мы добавляем ребро в граф. Необходимо в каждый момент времени сообщить число точек сочленения в графе.

Решение на 20 баллов

Попробуем удалить каждый из компьютеров. Если после его удаления граф распался на > 1 компоненту связности, то компьютер важный.

Если искать количество компонент связности за $O(E)$, то суммарная асимптотика будет $O((n+m)*n*m)$.

Решение на 40 баллов

- Существует алгоритм, который находит все точки сочленения за *O(E),* поэтому нет надобности удалять каждую вершину, чтобы проверить, является ли она точкой сочленения.
- Суммарная асимптотика такого решения – $O((n+m)\cdot m)$.

Идея полного решения

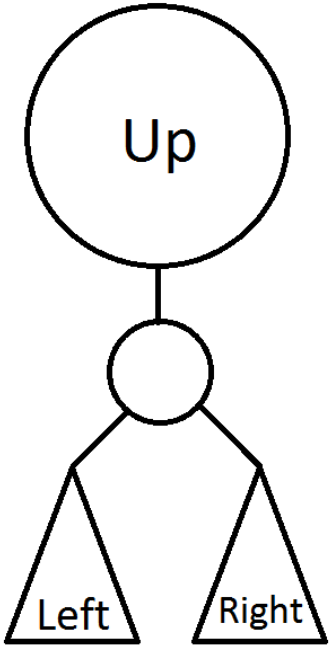
Первое наблюдение: листья не являются точками сочленения. Не листьями являются все вершины с номерами 1..$\left\lfloor \frac{N}{2} \right\rfloor$(кроме случая *N=2,* где вершина *1* тоже является листом.

Второе наблюдение: если точка в какой-то момент перестала быть точкой сочленения, то она никогда ей не будет.

Идея полного решения

Третье наблюдение: у каждой не листовой вершины, кроме *1,* есть левое поддерево, правое поддерево и всё остальное.

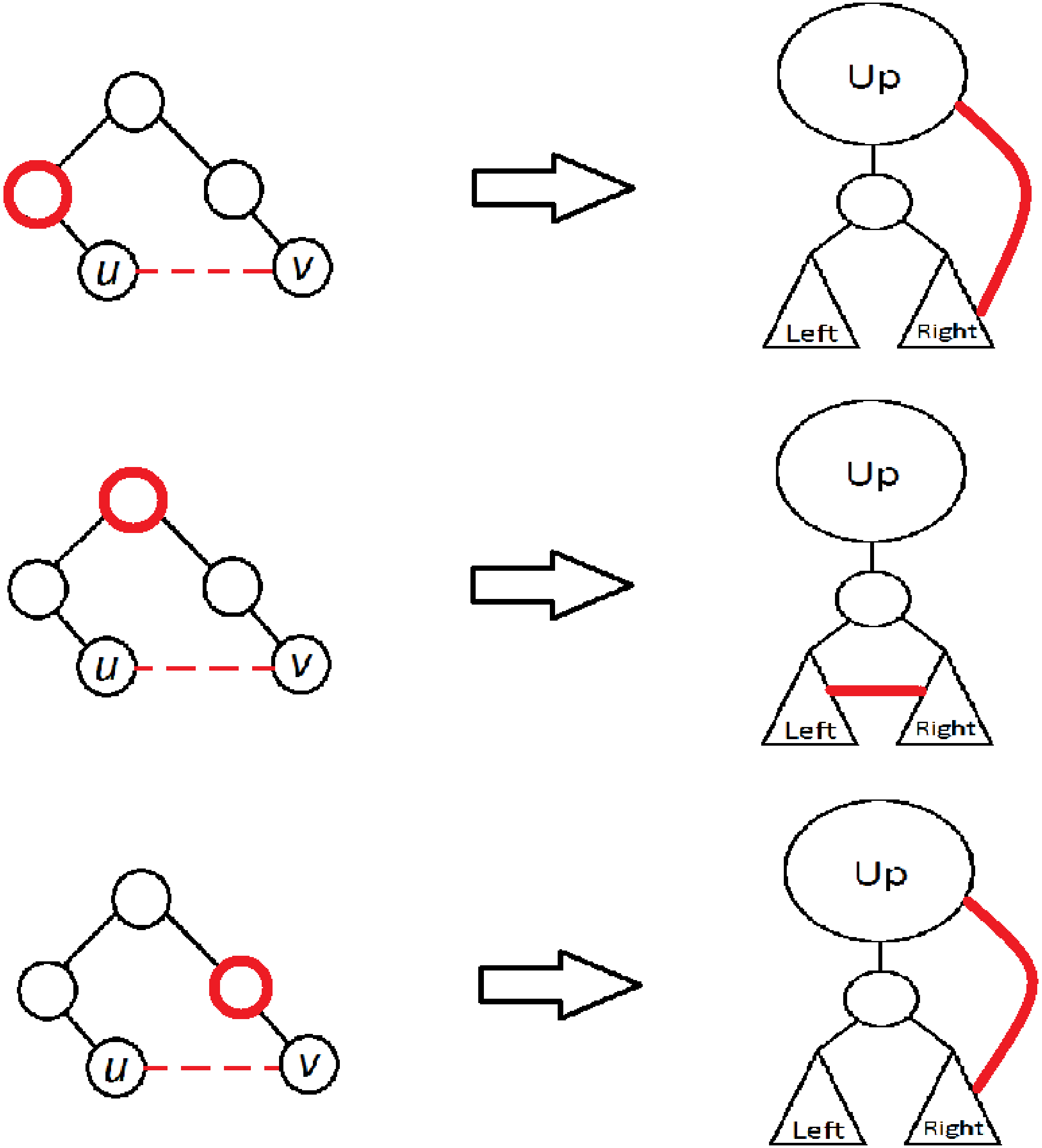
Если эти компоненты соединить между собой, то вершина перестанет быть точкой сочленения.



Когда компоненты соединяются?

Пусть мы соединили вершину *v* с вершиной *u*. Тогда если рассмотреть все вершины, которые лежат на пути от *v* до *u*, то для них соединились две компоненты.

Разбор задач - Сисадмин



Решение на 70 баллов

Каждый путь в дереве состоит из не более $O(\log n)$ вершин. Выписав все пути и поддерживая для каждой вершины какие её компоненты соединены, можно получить решение за $O(m \log n + n)$.

Полное решение

Если рассматривать только вершины, которые когда-то встретятся в каком-то из путей, то можно не хранить информацию для всех вершин.

Если реализовывать обращение к элементу с помощью бинпоиска или map, то решение будет иметь асимптотику $O(m \log^2 n)$. Если же использовать unordered_map, то $O(m \log n)$

Гонка

- Идея задачи — Владислав Белов
- Подготовка тестов — Нияз Нигматуллин, Павел Маврин, Владислав Белов
- Разбор задачи — Владислав Белов

Постановка задачи

Даны два набора чисел *a* и *b,* по *n* элементов в каждом.

Найти максимальное *m*, при котором из каждого набора можно выбрать подотрезки длины *m*, такие что максимумы на этих подотрезках равны.

Решение первой подзадачи

Переберем длину и пару индексов начала.

Проверим выполнения условия равенства максимумов.

Сложность решения - $O(n^4)$.

Решение второй и третьей подзадач

- Переберем пару индексов начала и длину отрезка
- Для каждой длины будем пересчитывать максимум
  I. За O(log(n)), используя структуру данных дерево отрезков
  II. За O(1), используя значение, вычисленные на предыдущем шаге
- Сложность решения
  I. $O(n^3 \log(n))$
  II. $O(n^3)$

Решение четвертой и пятой подзадач

- Для каждой длины отрезка предпосчитаем для каждого набора чисел список максимумов
- Для фиксированного индекса и длины будем пересчитывать максимум
  - За O(log(n)), используя структуру данных дерево отрезков
  - За O(1), используя значение, вычисленные на предыдущем шаге
- Пересечем списки максимумов для соответствующих длин
- Сложность решения
  - $O(n^2 \log(n))$
  - $O(n^2)$

Решение шестой и седьмой подзадач

- Запустим бинарным поиск по ответу
- Для каждой набора будем выписывать максимумы на отрезках соответствующей длины
- Максимум на отрезке можно выписывать
  I. Структурой данных дерево отрезков
  II. Очередью максимумов
- Сложность решения
  I. $O(n \cdot \log^2(n))$
  II. $O(n \cdot \log(n))$

Аттракцион

- Идея задачи — Тимур Хисматуллин
- Подготовка тестов — Тимур Хисматуллин
- Разбор задачи — Тимур Хисматуллин

Постановка задачи

Даны положения пуль барабана револьвера до вращения и после. Неизвестным остается только состояние одного паза барабана.

Требуется определить, если это возможно, есть ли пуля в скрытом пазу барабана после вращения.

Переформулирование

Даны две бинарные циклические строки с одним неизвестным символом (обозначим как ‘?’).

Необходимо для второй строки определить, может ли на месте ‘?’ быть 1, если известно, что это одна и та же строка.

Идея решения

Так как на месте ‘?’ может стоять только 0 или 1, то можно перебрать, что мы будем ставить на место ‘?’ в обеих строчках.

Если во второй строке можно подставить только 1, чтобы получилась первая, то ответ ‘Yes’, если только 0, то ‘No’. Иначе, ‘Random’.

Теперь остается проверить, совпадают ли две циклические строки, если в концы соответствующих строк мы добавим 0 или 1.

Решение на 40 баллов

Чтобы проверить, равны ли две циклические строчки, образованные строками *a* и *b*, можно проверить, совпадает ли один из циклических сдвигов строки *b* со строкой *a*.

Если сравнивать строчки за *n*, то асимптотика такого решения будет $O(n^2)$.

Решение на 80 баллов

- Можно сравнивать строчки за *n/64*, если использовать, что строчки бинарные и сравнивать подстроки длины 64 как числа.
- Также не нужно сравнивать две циклические строки на равенство, если у них различное число единичек.

Полное решение

1. Можно сравнивать строки за *O(1),* если использовать хэши. Тогда асимптотика будет *O(n).*
2. Заметим, что любая подстрока строки *bb*(строка *b*, записанная 2 раза) является циклическим сдвигом строки *b.*

$\underline{b_1, b_2 \ldots b_{n-1}, b_n}$: $\underline{b_1, b_2, b_3 \ldots b_{n-1}, b_n,}$ $b_1, b_2, b_3 \ldots b_{n-1}, b_n$

$\underline{b_2, b_3 \ldots b_n, b_1}$: $b_1,$ $\underline{b_2, b_3 \ldots b_{n-1}, b_n, b_1,}$ $b_2, b_3 \ldots b_{n-1}, b_n$

$\underline{b_3, b_4 \ldots b_n, b_1, b_2}$: $b_1, b_2,$ $\underline{b_3 \ldots b_{n-1}, b_n, b_1, b_2,}$ $b_3 \ldots b_{n-1}, b_n$

…

Полное решение

Причем каждый из циклических сдвигов содержится в *bb*. Остаётся проверить, что *a* является подстрокой *bb*.

Это можно сделать с помощью алгоритма Кнута-Морриса-Пратта за $O(|a| + |bb|) = O(n)$.

Телепортация

- Идея задачи — Нияз Нигматуллин
- Подготовка тестов — Дарья Яковлева, Нияз Нигматуллин
- Разбор задачи — Дарья Яковлева

Постановка задачи

- Дано n телепортов, расположенных на одной линии.
- Координата i-го телепорта x_i. Все координаты различны.
- Из i-го телепорта можно переместиться в телепорт с номером j, если $l_i \le |x_i - x_j| \le r_i$.
- Стоимость перемещения из i-го телепорта равна c_i.
- Посчитать минимальное расстояние из телепорта с номером s до всех остальных.
- Определить, существуют ли хотя бы два кратчайших пути из телепорта s до всех остальных

Решение на 48 баллов

- Кратчайшие пути найдем алгоритмом Дейкстры.
- Как посчитать число кратчайших путей?
  - Пусть мы пришли в вершину to из вершины v. Количество путей в i-ю вершину равно p[i]
  - Если d[v] + w(v, to) = d[to], то p[to] := p[to] + p[v]
  - Если d[v] + w(v, to) < d[to], то p[to] := p[v]
  - Число путей может быть очень большим, а требуется найти хотя бы два, поэтому p[to] := min(p[to] + p[v], 2)
- Время работы $O(n^2)$

Подготовка

- Чтобы написать решение, время работы которого меньше, чем $O(n^2)$, выполним предварительную подготовку
- Отсортируем координаты телепортов, запоминая при этом исходный номер каждого
- Для каждого телепорта все достижимые из него телепорты располагаются на двух отрезках
- Границы отрезков можно найти бинарным поиском

Решение на 31 балл

- Пусть веса ребер равны единице
- Запустим обход в ширину на нашем графе
- Заметим, что на каждом шаге нужно добавлять в очередь те вершины, для которых еще не найдены кратчайшие пути
- Значит, нужно реализовать структуру данных, которая должна уметь находить на отрезке все такие вершины, добавлять их в очередь и удалять

Решение на 31 балл

- Как посчитать число кратчайших путей?
  - Пусть мы пришли в вершину u_1 из вершины v, значит хотя бы один кратчайший путь мы нашли
  - Если мы нашли два пути, то вершину v пометим как посещенную. Удалим ее из нашей структуры данных
  - Если еще не были найдены два кратчайших пути и мы пришли второй раз в вершину v из другой вершины u_2, то если $d[u_2] + w(u_2, v) = d[v]$, значит второй путь мы нашли
  - Если нет, значит двух кратчайших путей нет и вершину v можно больше не рассматривать. Удалим ее из структуры данных
- Время работы $O(n \cdot \log(n))$

Решение на 95-100 баллов

- Пусть в нашем графе веса не единичного веса
- По условию ребра из каждой вершины v одинакового веса w(v)
- Реализуем очередь с приоритетом d[v] + w(v)
- Очередь с приорететом d[v] нам не подойдет (подумайте почему)
- Заметим, что первый путь, по которому мы придем в вершину, будет для нее кратчайшим
- Поиск двух кратчайший путей выполняется аналогично

Решение на 95-100 баллов

- Научимся для телепорта перебирать только те достижимые из него телепорты, для которых еще не найдены два кратчайших пути
- Первый способ
  - Реализуем дерево отрезков с операциями на отрезке: для нового значения m и каждого значения a_i посчитать и запомнить $\min(m, a_i)$, если $a_i = m$, то запомнить, что существуют два кратчайших пути в вершину
  - Соответственно, для каждой вершины мы храним только левую и правую достижимые вершины
  - Обновление на отрезке занимает $O(\log(n))$ времени
  - Время работы $O(n \cdot \log(n))$

Решение на 85 баллов

- Второй способ
  - Научимся для телепорта перебирать только те достижимые из него телепорты, для которых еще не найдены два кратчайших пути
  - Будем поддерживать дерево отрезков с операцией сумма на отрезке
  - Для каждого телепорта будем хранить в дереве отрезков 1, если два кратчайших пути еще не найдены, и 0, в противном случае
  - Пусть сумма элементов с 0 по i-й равна s, бинарным поиском найдем первую позицию, сумма до которой равна s + 1, это позиция и будет являться следующим нужным телепортом
  - Время работы $O(n \cdot \log^2(n))$

Решение на 85 баллов

- Третий способ
  - Будем поддерживать следующую структуру: для каждого телепорта храним первый телепорт справа, для которого мы еще не нашли два кратчайших пути
  - Такую структуру можно реализовать с помощью структуры данных set
  - Время работы $O(n \cdot \log(n))$ с большой константой

Решение на 100 баллов

- Заметим, что вместо структуры данных set можно использовать систему непересекающихся множеств
- Время работы $O(n \cdot \log^{*}(n))$

Клевер

- Идея задачи — Михаил Киндер
- Подготовка тестов — Михаил Киндер
- Разбор задачи — Михаил Киндер

Постановка задачи

На четырёхлистном листе клевера требуется найти точку равновесия, для которой сумма расстояний до центров пластинок наименьшая.

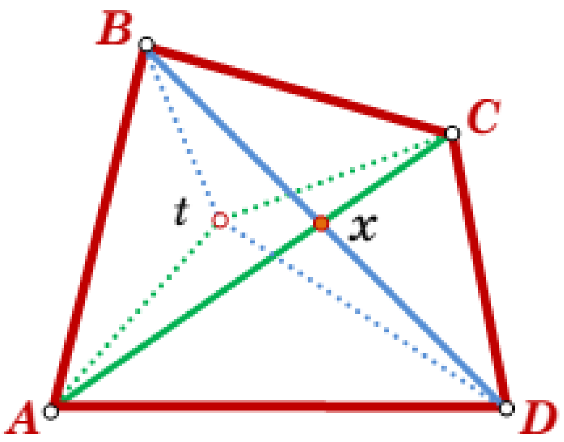
Более формально, необходимо найти точку, для которой сумма расстояний до четырёх заданных точек наименьшая.

Первая подзадача

Точки *A*, *B*, *C*, *D* лежат на одной прямой. Пусть, например, точки *C* и *D* лежат между *A* и *B*. Искомой точкой будет любая точка отрезка *CD*. Тогда требуемое расстояние равно *AB* + *CD*.

Вторая подзадача

Точки *A*, *B*, *C*, *D* образуют выпуклый четырёхугольник. В этом случае искомой точкой является точка пересечения диагоналей *AC* и *BD*. Наименьшее расстояние равно сумме диагоналей *AC* + *BD*. Доказательство — неравенство треугольника.



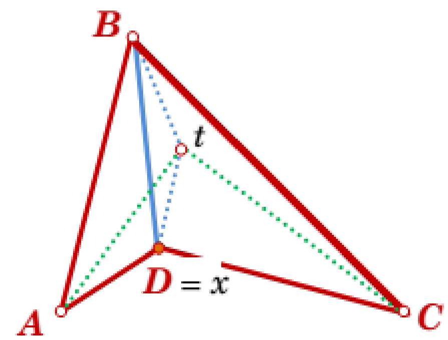
$$d(t,A) + d(t,C) \geq d(A,C)$$

$$d(t,B) + d(t,D) \geq d(B,D)$$

2 случай.

Третья подзадача

Точки *A*, *B*, *C*, *D* не образуют выпуклый четырёхугольник, но не лежат на одной прямой. В этом случае одна из точек (например, точка *D*) лежит *внутри* или *на стороне* треугольника *ABC*. Точка *D* — искомая, наименьшее расстояние равно *AD* + *BD* + *CD*.



$$d(t,A) + d(t,C) \geq d(x,A) + d(x,C)$$

$$d(t,B) + d(t,D) \geq d(x,B) + d(x,D)$$

3 случай.