Доска улик

Общее замечание: Развернём задачу. Вместо того, чтобы «собирать» дерево с нуля, будем наоборот, удалять рёбра из дерева. А после развернём полученный нами ответ. Далее решаем задачу именно в формате «удалить дерево». Список стикеров в вершине дерева далее будем называть стеком, на вершине стека всегда будет стикер, который должен быть удалён первым.

- **Подгруппа №1**

  Подгруппа решается полным перебором. Есть $(n-1)!$ возможных порядков удаления рёбер. Каждый из порядков тривиально проверяется за $\mathcal{O}(n)$.

  Асимптотика: $\mathcal{O}(n!)$

- **Подгруппа №2**

  Бамбук.

  Динамика по префиксу. $dp[v][i]$ — булевое значение: можно ли удалить бамбук из вершин от 1 до v, но если стек для вершины v содержит единственный элемент $c_{v,i}$ $(i \in \{0,1\})$.

  База динамики: $dp[1][0] = dp[1][1] = true$

  Пересчёт динамики: надо перебрать все четыре варианта с какими элементами стеков удалится ребро между v и $v-1$, если это ребро можно удалить пересчитываемся через $dp[v-1][i]$.

  Ответ существует, если $dp[n][0] = true$, иначе — нет.

  Чтобы восстановить ответ надо для каждого ребра запомнить с какими i_1, i_2 оно удалялось. Сначала удалить все рёбра вида $i_1 = 0, i_2 = 0$, потом $i_1 = 1, i_2 = 0$, $i_1 = 0, i_2 = 1$ и $i_1 = 1, i_2 = 1$.

  Асимптотика: $\mathcal{O}(n)$

- **Подгруппа №3**

  Звезда.

  Пройдёмся по листам от 2 до n. Пусть сейчас рассматриваем лист v. В массиве c_1 найдём минимальное число $\geqslant w_v - c_{v,1}$. Несложно понять, что именно с этим числом оптимально будет удалить ребро ведущее к v. После чего удалим найденное число из c_1. В реализации поддерживаем c_1 в `std::multiset`, и используем метод `lower_bound`.

  Нужный порядок удаления рёбер соответствует исходному порядку чисел в c_1.

  Асимптотика: $\mathcal{O}(n \log n)$

- **Подгруппа №4**

  Две звезды, основания соединены ребром.

  Решаем две звезды аналогично группе 4. После в обоих основаниях останется по одному числу. Если они позволяют удалить ребро между основаниями, ответ Yes, иначе No. Подходящий порядок удаления рёбер: все рёбра первого ежа до вершины основания, все рёбра второго ежа до вершины основания, ребро между основаниями, все рёбра первого ежа после вершины основания, все рёбра второго ежа после вершины основания.

  Асимптотика: $\mathcal{O}(n \log n)$

- **Подгруппа №5**

  Значения чисел в вершинах возрастают.

  Если в какой-то момент какое-то ребро можно удалить, то и далее его всегда можно будет удалить. А значит в любой момент времени допустимо удалять любое возможное ребро. Итого решением будет жадный алгоритм: ищем любое ребро которое можно удалить и удаляем из дерева, повторяем $n-1$ раз. Если в какой-то момент нет готовых к удалению рёбер, ответ No.

  Асимптотика: $\mathcal{O}(n^2)$

- **Подгруппа №6**

  Заметим, что для произвольного теста, верен следующий факт: если развернуть все стеки оригинального теста и назвать это *новым* тестом, то если для оригинального теста ответа не существует, то не существует и для нового. А если ответ существует, то ответом на *новый* тест будет развёрнутый ответ для оригинального теста. Из этого наблюдения решение группы 6 напрямую следует из решения группы 5: нужно развернуть все массивы, а также итоговый ответ.

  Асимптотика: $\mathcal{O}(n^2)$

- **Полное решение**

  Полное решение основано на одной ключевой идее: «Для любого способа назначить каждому ребру пару элементов стеков из смежных вершин (без повторений), найдется порядок удаления рёбер такой, что каждое ребро будет удалено вместе с назначенными ему элементами».

  Доказательство ключевой идеи: Рассмотрим любое назначение. Каждая вершина указывает на одно смежное ребро, которому назначен верхний элемент стека, как «наполовину готовое к удалению». Так как вершин n, а рёбер $n-1$, найдётся ребро на которое будут указывать дважды, обе смежные вершины. Именно это ребро и можно будет удалить. После удаления дерево распадается на два других, и в них применяется то же рассуждение.

  После ключевой идеи задача распалась на две части.

  1. Построить любое корректное назначение рёбер элементам стеков
  2. Восстановить порядок удаления рёбер

  Для построения назначения используем жадный алгоритм, похожий на решение группы 3. Выпишем порядок вершин любого обхода дерева. Будем перебирать вершины в перевёрнутом порядке обхода (с листьев). Пусть сейчас перебираемая вершина v. Тогда единственному оставшемуся числу в стеке вершины v назначим минимальное подходящее число ($\geqslant w_v - c_v$) из вершины родителя.

  Для реализации будем в каждой вершине хранить `std::multiset` чисел. Будем делать `lower_bound` и `erase` в нём.

  Восстановление: для каждого ребра поддерживаем насколько оно готово к удалению (0/1/2). Рёбра готовые к удалению храним в стеке. Итеративно достаём любое ребро из стека, удаляем его и увеличиваем «готовность к удалению» у не более 2 других рёбер. Если какое-то из них становится готовым к удалению, добавляем его в стек.

  Финальная асимптотика: $\mathcal{O}(n \log n)$