Билет 40. Определение эйлерова цикла. Критерий наличия эйлерова цикла в неориентированном графе.

Def. Вершина называется изолированной, если ее степень равна 0. В случае неориентированного графа это равносильно тому, что из нее не выходит ни одного ребра. В случае ориентированного графа это равносильно тому, что количество входящих ребер и количество исходящих ребер равно 0.

Def. Эйлеров путь — путь, проходящий по всем рёбрам графа и притом только по одному разу.

Def. Эйлеров цикл — цикл, проходящий через каждое ребро графа ровно по одному разу.

Def. Граф эйлеров, если в нем есть эйлеров цикл.

Теорема (Критерий эйлеровости неориентированного графа)**.** *Неориентированный граф, который становится связным после удаления всех изолированных вершин, является эйлеровым тогда и только тогда, когда все его вершины имеют четную степень.*

Доказательство. $\Rightarrow$

Рассмотрим эйлеров обход графа. Заметим, что при попадании в вершину и при выходе из нее мы уменьшаем ее степень на два (помечаем уже пройденые ребра). Кроме того, для стартовой вершины мы уменьшаем ее степень на один в начале обхода эйлерова цикла, и на один при завершении. Следовательно вершин с нечетной степенью быть не может.

$\Leftarrow$

Необходимость мы доказали ранее. Докажем достаточность, используя индукцию по числу вершин n.

База индукции: $n = 0$ цикл существует.

Предположим что граф, имеющий менее n вершин, степени вершин которого четны, содержит эйлеров цикл.

Рассмотрим граф G с $n > 0$ вершинами, степени которых четны. Удалим изолированные вершины. Если такие были, воспользуемся утверждением идукции. В противном случае у нас все еще n вершины, однако теперь мы можем гарантировать, что граф связен.

Пусть v_1 — вершина графа. Начнем идти из этой вершины по ребрам графа до тех пор, пока можем (проходим по каждому ребру не более 1 раза). Если в какой-то момент мы не можем пойти дальше, выведем нашу вершину.

```
void euler(int v) {
  while из( v есть хотя бы одно неиспользованное ребро){
    пусть (v;u) - ребро, пометим его использованным
    euler(u)
  }
  print(v)
}
```

Заметим, что у вершин, которые встречаются в процессе обхода, степень каждый раз уменьшается на 2. Ведь если мы вошли в вершину, и она не стартовая, то в этот момент из нее ведет нечетное число непосещенных ребер. Тогда из нее можно выйти.

Поэтому первая выведенная нами вершина будет стартовая, то есть v_1. Кроме того, мы получим замкнутый путь (цикл), который начинается и заканчивается в вершине v_1.

Назовем этот цикл C_1. Если C_1 является эйлеровым циклом для G, тогда доказательство закончено. Если нет, то пусть G_2 — подграф графа G, полученный удалением всех рёбер, принадлежащих C_1. Поскольку C_1 содержит чётное число рёбер, инцидентных каждой вершине, то каждая вершина подграфа G_2 имеет чётную степень. Этот граф разбивается на некоторое количество компонент связности.

Рассмотрим какую-либо компоненту связности G_2 (не состоящую из изолированной вершины). Поскольку рассматриваемая компонента связности G_2 имеет менее, чем n вершин (как минимум, туда не входит v_1, ведь все ее ребра были удалены), а у каждой вершины графа G_2 чётная степень, то у каждой компоненты связности G_2 существует эйлеров цикл. Пусть для рассматриваемой компоненты связноти это цикл C_2. У C_1 и C_2 имеется общая вершина a, так как G связен. Давайте объединим C_1 и C_2 в новый цикл. Для этого нужно, начиная с вершины a, обойти C_1, вернуться в a, затем пройти по C_2 и вернуться в a. Если новый эйлеров цикл не является эйлеровым циклом для G, продолжаем использовать этот процесс, расширяя наш цикл, пока, в конце концов, не получим эйлеров цикл для G.

□

Билет 41. Реализация алгоритма поиска эйлерова цикла.

```
struct edge{
  int from, to;
}

vector <edge> edges;
\\edges[2k] и edges[2k + 1] -- две половинки одного неориентированного ребра
\\ удобно получать вторую половинку ребра, беря ind^1
vector <bool> used \\used[e] -- использовано ли ребро
vector <int> ptr; \\"указатель" на следующее неиспользованное ребро
vector <vector <int> > g; наш\\ граф в виде списков смежности, но уже с ориентированными ребрами

void euler(int v) {
  while(ptr[v] != g[v].size()) {
    int e = g[v][ptr[v]]; \\g[v] -- список номеров ребер
    if (used[e]){++ptr[v]; continue;}
    int u = edges[e].to;
    used[e] = used[e^1] = true;
    ++ptr[v];
    euler(u);
  }
  cout << v << " ";
}
```

Асимптотика $O(n+m)$

Билет 42. Критерий наличия эйлерова пути в неориентированном графе.

Теорема. *Граф G (связный, если убрать изолированные вершины) содержит эйлеров путь тогда и только тогда, когда количество вершин с нечетной степенью меньше или равно двум.*

Доказательство. В одну сторону, очевидно, верно. Если у нас есть путь, то у вершин, отличных от концов, степень четна, а у начальной и конечной вершины степень нечетна.

В другую сторону. Если выполняется условие на степени вершин, то добавим ребро между двумя вершинами с нечетной степенью. В полученном графе есть эйлеров цикл, удаление из которого добавленного ребра даст эйлеров путь.

В этом доказательстве мы не рассмотрели случай, когда есть ровно одна вершина с нечетной степенью. Однако такой случай невозможен. По лемме о рукопожатиях в любом графе четное число вершин с нечетной степенью. А один – нечетное число. □

Теорема (Лемма о рукопожатиях)**.** *В любом неориентированном графе число вершин с нечетной степенью четно.*

Доказательство. Действительно. Сложим степени всех вершин. Получим некоторое число. Заметим, что оно равно удвоенному числу ребер, ведь каждое ребро в нашей сумме учитывалось два раза – по одному от каждого из его концов. Отсюда следует, что количество вершин с нечетной степенью четно. □

Билет 43. Критерий наличия эйлерова цикла в ориентированном графе.

Теорема (Критерий эйлеровости ориентированного графа)**.** *Ориентированный граф G (который становится сильно связным при удалении изолированных вершин) эйлеров тогда и только тогда, когда для каждой вершины верно, что входная степень равна выходной.*

Доказательство. Абсолютно аналогично доказательству для неориентированного графа. □