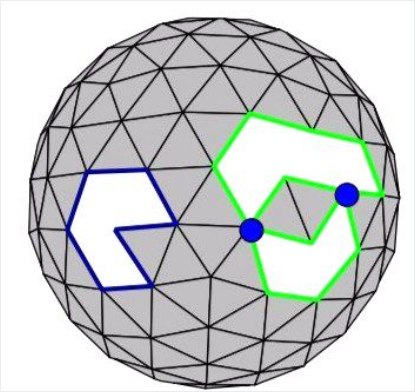
Представление триангуляции в памяти



Содержание

- Формула Эйлера для планарных графов и ее следствия
- Структуры для представления триангуляции
  - Узлы с соседями
  - Узлы, рёбра и треугольники
  - Узлы и треугольники
  - Легкие узлы и треугольники
  - Двойные рёбра
- Домашнее задание

Формула Эйлера

Для произвольного планарного графа G с V вершинами, E ребрами и F гранями справедливо следующее соотношение:

$$V - E + F = 2$$

Формула Эйлера

Для произвольного планарного графа G с V вершинами, E ребрами и F гранями справедливо следующее соотношение:

$$V - E + F = 2$$

Доказательство?

Формула Эйлера

Для произвольного планарного графа G с V вершинами, E ребрами и F гранями справедливо следующее соотношение:

$$V - E + F = 2$$

Доказательство:
Индукцией по количеству граней графа

Формула Эйлера

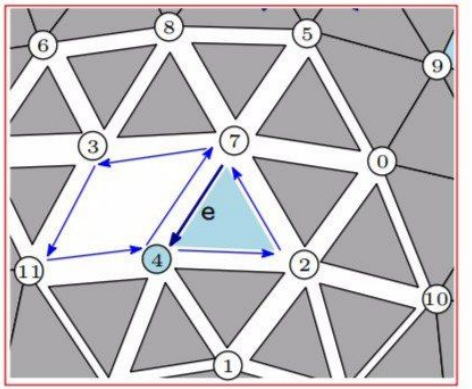
Следствия:

Пусть G связный планарный обыкновенный граф с V вершинами (V >=3), E ребрами и F гранями. Тогда:

- E <= 3V - 6
- F <= 2V - 4
- Средняя степень вершины в планарном графе равна 6, такое же количество и инцидентных треугольников для вершины

Структуры для представления триангуляции

Обозначения:
V - количество вершин
E - количество рёбер
F - количество треугольников



Узлы с соседями

Для каждого узла хранятся координаты и список указателей на соседние узлы (список номеров узлов), с которыми есть общие рёбра.

```
class NodeWithNeighbours:
    def __init__(self, idr: int, p: Point, neigh_nodes: List[int]):
        self.idr = idr
        self.p = p
        self.neigh_nodes = neigh_nodes
```

Узлы с соседями

Для каждого узла хранятся координаты и список указателей на соседние узлы (список номеров узлов), с которыми есть общие рёбра.

```
class NodeWithNeighbours:
    def __init__(self, idr: int, p: Point, neigh_nodes: List[int]):
        self.idr = idr
        self.p = p
        self.neigh_nodes = neigh_nodes
```

Оценка памяти:
Каждое ребро учитывается дважды. На каждую вершину: id, Point(x и y), соседи.

Точная оценка: **2E + 3V**

Узлы с соседями

Для каждого узла хранятся координаты и список указателей на соседние узлы (список номеров узлов), с которыми есть общие рёбра.

```
class NodeWithNeighbours:
    def __init__(self, idr: int, p: Point, neigh_nodes: List[int]):
        self.idr = idr
        self.p = p
        self.neigh_nodes = neigh_nodes
```

Память: 2E + 3V

- Легко реализовать
- Компактно

Узлы с соседями

Для каждого узла хранятся координаты и список указателей на соседние узлы (список номеров узлов), с которыми есть общие рёбра.

```
class NodeWithNeighbours:
    def __init__(self, idr: int, p: Point, neigh_nodes: List[int]):
        self.idr = idr
        self.p = p
        self.neigh_nodes = neigh_nodes
```

Память: 2E + 3V

- Легко реализовать
- Компактно
- Не удобно!

Узлы, рёбра и треугольники

Для каждого ребра хранятся указатели на два концевых узла и два соседних треугольника. Для треугольников хранятся указатели на три образующих треугольник ребра. В каждом узле координаты и инцидентные треугольники.

```
class NodesAndEdgesAndTriangles:
    class Node:
        def __init__(self, idr: int, p: Point, triangles: List[int]):
            self.idr = idr
            self.p = p
            self.triangles = triangles
    class Edge:
        def __init__(self, idr: int, nodes: List[int], triangles: List[int]):
            self.idr = idr
            self.nodes = nodes
            self.triangles = triangles
    class Triangle:
        def __init__(self, idr: int, edges: List[int]):
            self.idr = idr
            self.edges = edges
```

Узлы, рёбра и треугольники

Для каждого ребра хранятся указатели на два концевых узла и два соседних треугольника. Для треугольников хранятся указатели на три образующих треугольник ребра. В каждом узле координаты и инцидентные треугольники.

```
class NodesAndEdgesAndTriangles:
    class Node:
        def __init__(self, idr: int, p: Point, triangles: List[int]):
            self.idr = idr
            self.p = p
            self.triangles = triangles
    class Edge:
        def __init__(self, idr: int, nodes: List[int], triangles: List[int]):
            self.idr = idr
            self.nodes = nodes
            self.triangles = triangles
    class Triangle:
        def __init__(self, idr: int, edges: List[int]):
            self.idr = idr
            self.edges = edges
```

Оценка памяти:
Один узел требует 9 int'ов:
1 (id) + 2 (Point) + 6 (среднее количество инцидентных треугольников).
Одно ребро требует 5 int'ов:
1 (id) + 2 (концы) + 2 (у одного ребра два инцидентных треугольника).
Один треугольник требует 4 int'a:
1 (id) + 3 (ребра треугольника).
Получаем верхнюю оценку:
9V + 5E + 4F

Узлы, рёбра и треугольники

Для каждого ребра хранятся указатели на два концевых узла и два соседних треугольника. Для треугольников хранятся указатели на три образующих треугольник ребра. В каждом узле координаты и инцидентные треугольники.

```
class NodesAndEdgesAndTriangles:
    class Node:
        def __init__(self, idr: int, p: Point, triangles: List[int]):
            self.idr = idr
            self.p = p
            self.triangles = triangles
    class Edge:
        def __init__(self, idr: int, nodes: List[int], triangles: List[int]):
            self.idr = idr
            self.nodes = nodes
            self.triangles = triangles
    class Triangle:
        def __init__(self, idr: int, edges: List[int]):
            self.idr = idr
            self.edges = edges
```

Память:
9V + 5E + 4F

- Легко реализовать
- Удобно
- А не много ли памяти?

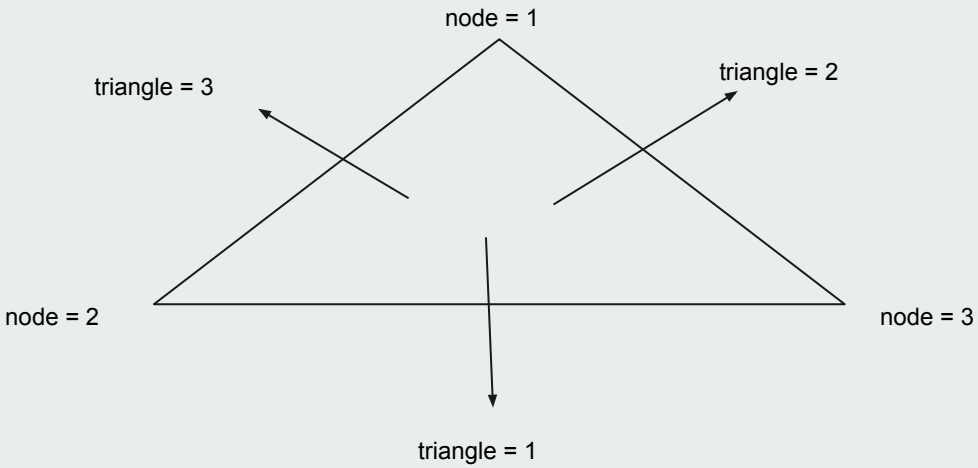
Узлы и треугольники

Для каждого треугольника хранятся три указателя на образующие его узлы и три указателя на смежные треугольники, а каждый узел хранит указатели на инцидентные ему треугольники.

```
class NodesAndTriangles:
    class Node:
        def __init__(self, idr: int, p: Point, triangles: List[int]):
            self.idr = idr
            self.p = p
            self.triangles = triangles
    class Triangle:
        def __init__(self, idr: int, nodes: List[int], triangles: List[int]):
            self.idr = idr
            self.nodes = nodes
            self.triangles = triangles
```

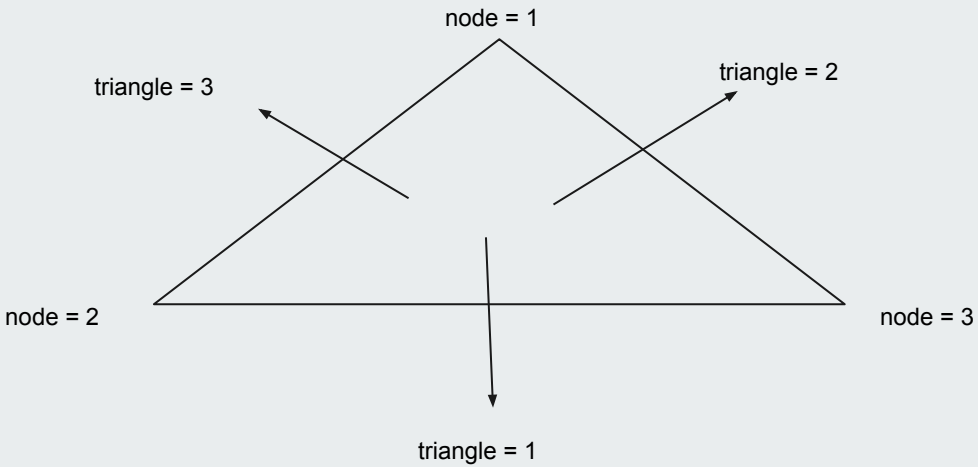
Узлы и треугольники

Нумерация узлов против часовой стрелки, напротив i узла находится i треугольник



Узлы и треугольники

Нумерация узлов против часовой стрелки, напротив i узла находится i треугольник

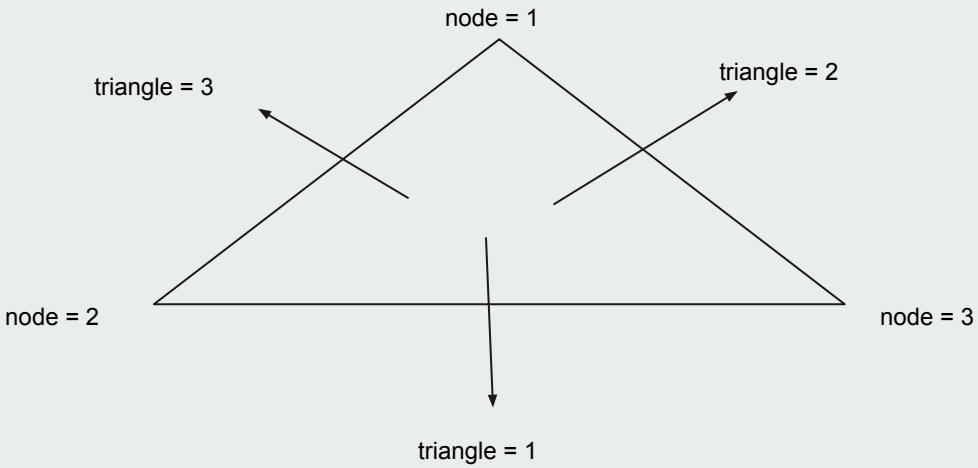


Оценка памяти:
Один узел требует 9 int’ов: 1 (id) + 2 (Point) + 6 (среднее количество инцидентных треугольников).
Один треугольник требует 7 int'ов: 1 (id) + 3 (вершины треугольника) + 3 (соседи).

Получаем верхнюю оценку:
9V + 7F

Узлы и треугольники

Нумерация узлов против часовой стрелки, напротив i узла находится i треугольник



Память:
9V + 7F

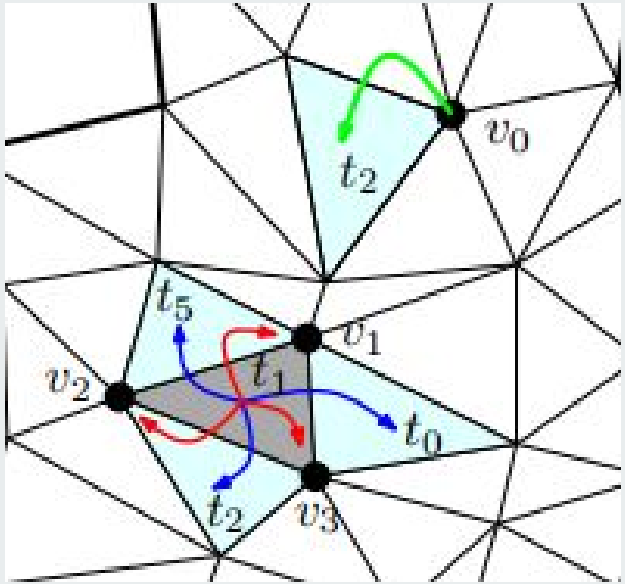
- Легко реализовать
- Компактно
- Удобно
- А можно лучше?

Легкие узлы и треугольники

Для каждого треугольника хранятся три указателя на образующие его узлы и три указателя на смежные треугольники, а каждый узел хранит только один указатель на инцидентный ему треугольник.

```
class LightNodesAndTriangles:
    class Node:
        def __init__(self, idr: int, p: Point, triangle: int):
            self.idr = idr
            self.p = p
            self.triangle = triangle
    class Triangle:
        def __init__(self, idr: int, nodes: List[int], triangles: List[int]):
            self.idr = idr
            self.nodes = nodes
            self.triangles = triangles
```

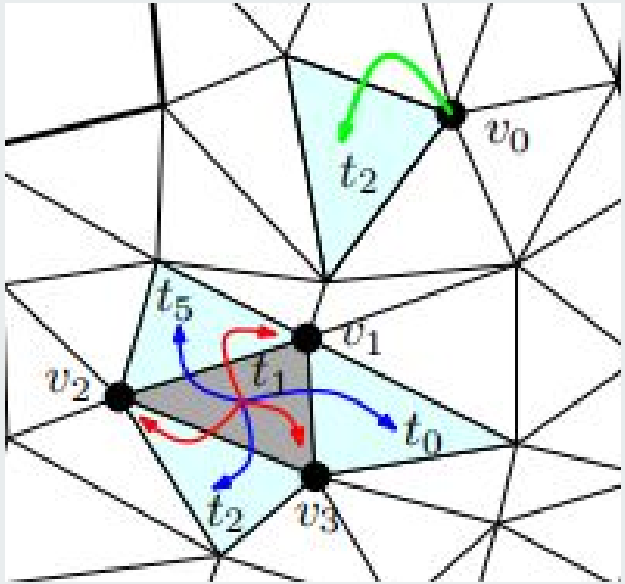
Легкие узлы и треугольники



Оценка памяти:
Один узел требует 4 int'а:
1 (id) + 2 (Point) + 1 (любой инцидентный треугольник).
Один треугольник требует 7 int'ов:
1 (id) + 3 (составляющие треугольник вершины) + 3 (соседние треугольники).

Получаем верхнюю оценку: **4V + 7F**

Легкие узлы и треугольники



Память: 4V + 7F

- Довольно компактно
- Используется на практике (CGAL)

Что еще используется на практике?

Двойные ребра

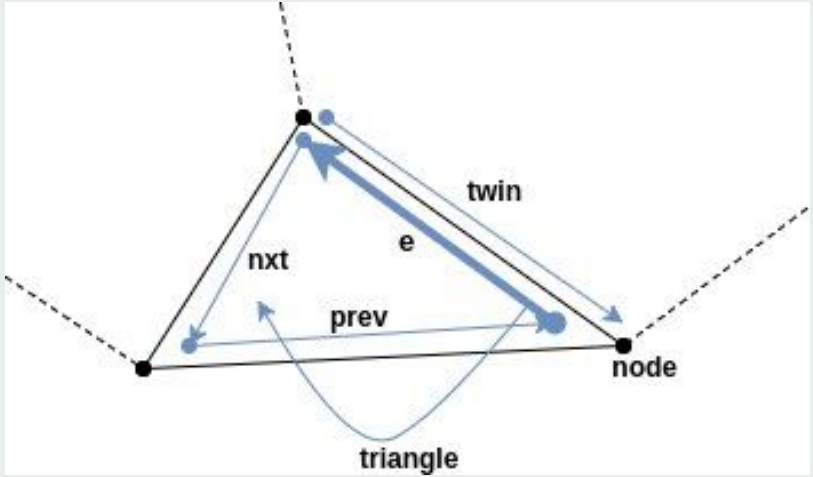
```
class DoubleEdges:
    class Node:
        def __init__(self, idr: int, p: Point, he: int):
            self.idr = idr
            self.p = p
            self.he = he
    class HalfEdge:
        def __init__(self, idr: int, node: int, prev: int, nxt: int,
twin: int, triangle: int):
            self.idr = idr
            self.node = node
            self.prev = prev
            self.nxt = nxt
            self.twin = twin
            self.triangle = triangle
    class Triangle:
        def __init__(self, idr: int, he: int):
            self.idr = idr
            self.he = he
```

Вершина хранит в себе координаты узла и ссылку на любое инцидентное полуребро, выходящее из этого узла.

Полуребро хранит в себе ссылку на выходящий узел, ссылки на обратное, следующее и предыдущее ребра в порядке обхода треугольника, а также ссылку на инцидентный треугольник.

Треугольник хранит в себе любое полуребро, составляющее его границу.

Двойные ребра



Оценка памяти:
Один узел требует 4 int'а:
1 (id) + 2 (Point) + 1 (полурёбро).

Одно полуребро требует 6 int'ов:
1 (id) + 1 (выходящая вершина) + 1 (треугольник) + 3 (prev, next, twin полурёбра), а на каждое ребро есть два полуребра, отсюда x2.

Один треугольник требует 2 int'а:
1 (id) + 1 (любое инцидентное полуребро).

Получаем верхнюю оценку: **4V + 12E + 2F**

Домашнее задание

- Прочитать про структуры в конспекте
- Ознакомиться с реализацией задачи “Walking in triangulation” для рассмотренных структур
- Реализовать алгоритм “Walking in triangulation” для структуры “Легкие узлы и треугольники”