Курс «Алгоритмы и алгоритмические языки»
1 семестр 2014/2015

Лекция 15

Топологическая сортировка узлов ациклического ориентированного графа

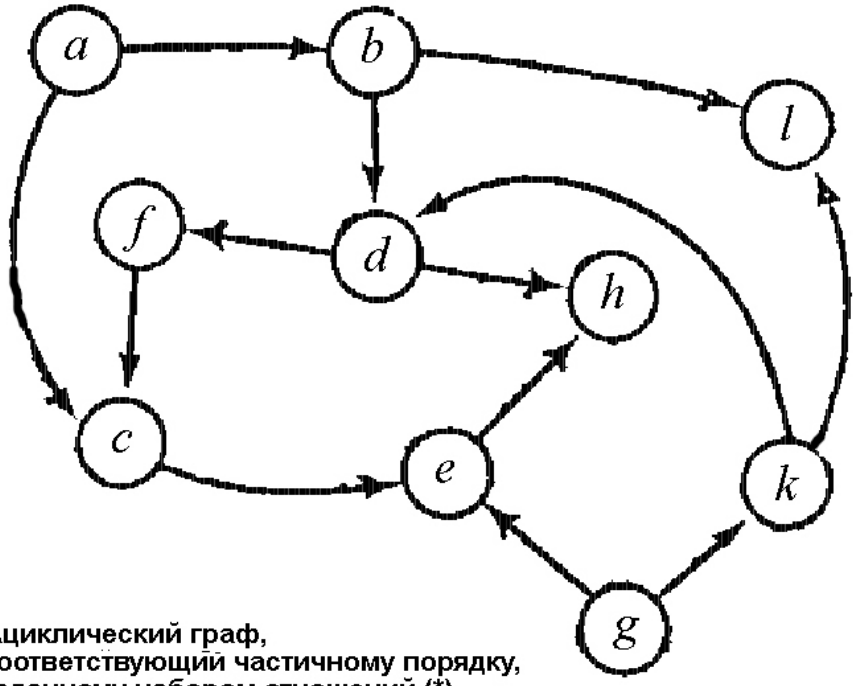
- Ациклический граф можно использовать для графического изображения *частично упорядоченного множества.*
  Цель топологической сортировки:
  преобразовать частичный порядок в линейный.
  Графически это означает, что
  все узлы графа нужно расположить на одной прямой таким образом, чтобы все дуги графа были направлены в одну сторону.

Топологическая сортировка узлов ациклического ориентированного графа

- ***Пример***. Частичный порядок (<) задается следующим набором отношений :

$$a < b,\ b < d,\ d < f,\ b < l,\ d < h,\ f < c,\ a < c, \qquad (*)$$
$$c < e,\ e < h,\ g < e,\ g < k,\ k < d,\ k < l$$

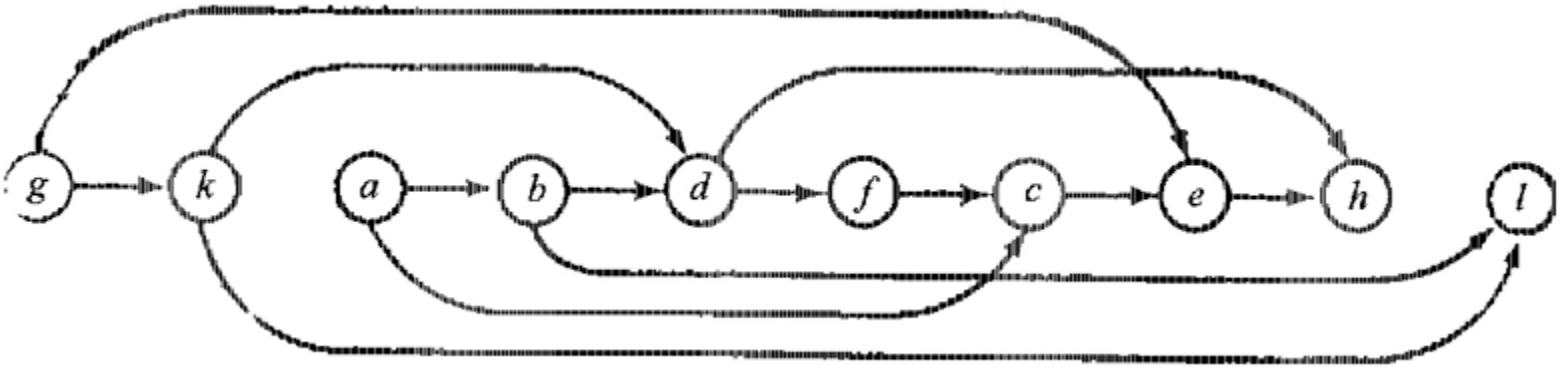
Набор отношений (*) можно представить в виде следующего ациклического графа (см. рисунок):



Ациклический граф, соответствующий частичному порядку, заданному набором отношений (*).

Топологическая сортировка узлов ациклического ориентированного графа

- Требуется привести рассматриваемый граф к линейному графу:



- На этом графе ключи расположены в следующем порядке:

g k a b d f c e h l

(поскольку топологическая сортировка неоднозначна, это один из возможных топологических порядков).

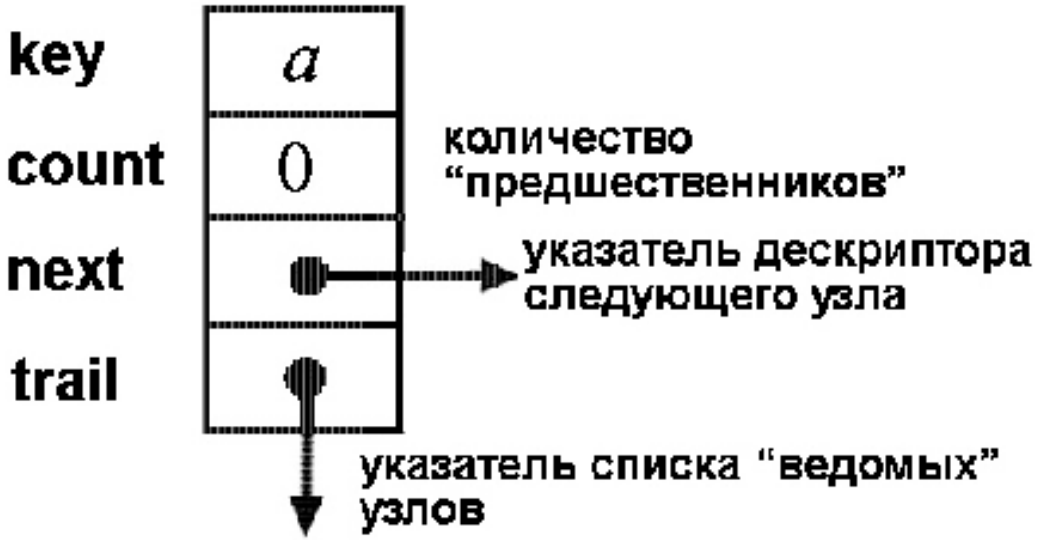
- Последовательная обработка полученного линейного списка узлов графа эквивалентна их обработке в порядке обхода графа.

Топологическая сортировка узлов ациклического ориентированного графа

- Поскольку рассматриваемый граф – ациклический, **существует** хотя бы один узел графа, у которого нет предшествующих узлов. Каждый такой узел будем называть *ведущим* узлом.
  Шаг алгоритма: Выберем один из ведущих узлов и поместим его в начало линейного списка отсортированных узлов, удалив его из исходного графа.
- Поскольку граф, у которого удалили один из ведущих узлов, останется ациклическим, в нем будет хотя бы один ведущий узел. Следовательно, можно повторить шаг алгоритма.
- Легко видеть, что каждый узел графа рано или поздно станет ведущим и попадет в формируемый линейный список, и алгоритм завершится через несколько шагов.

Топологическая сортировка узлов ациклического ориентированного графа

- Структуры данных для представления узлов:
  - Каждый узел исходного графа представляется с помощью дескриптора узла, который имеет вид:

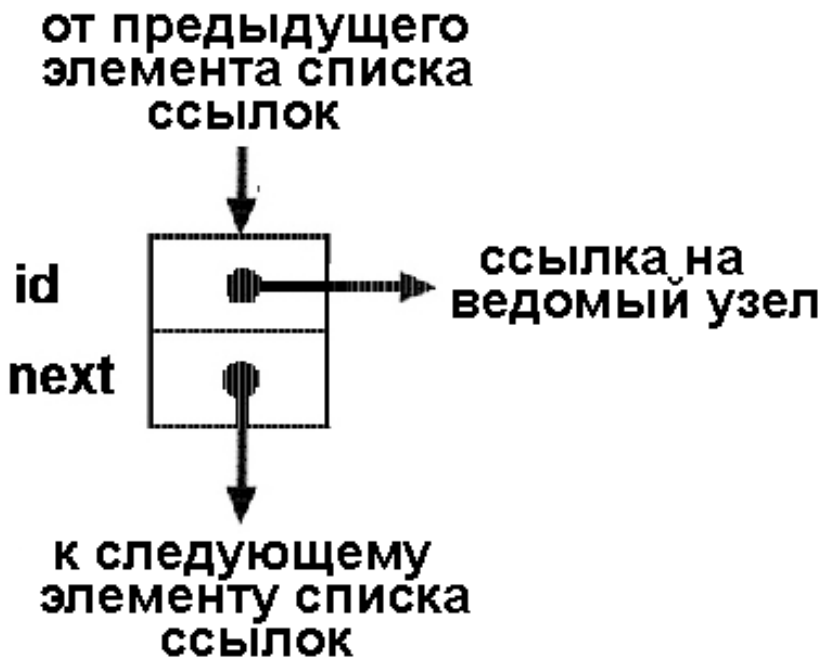


Дескриптор узла.

- *Ведомыми* для узла n будут узлы, для которых n является предшественником. Каждый узел графа (не только ведущий) может иметь один или несколько ведомых узлов.

Топологическая сортировка узлов ациклического ориентированного графа

- Структуры данных для представления узлов:
  - Дескриптор каждого узла содержит ссылки на ведомые узлы. Так как заранее неясно, сколько у узла будет ведомых узлов, эти ссылки помещаются в список. На рисунке представлен элемент списка ссылок.



Топологическая сортировка узлов ациклического ориентированного графа

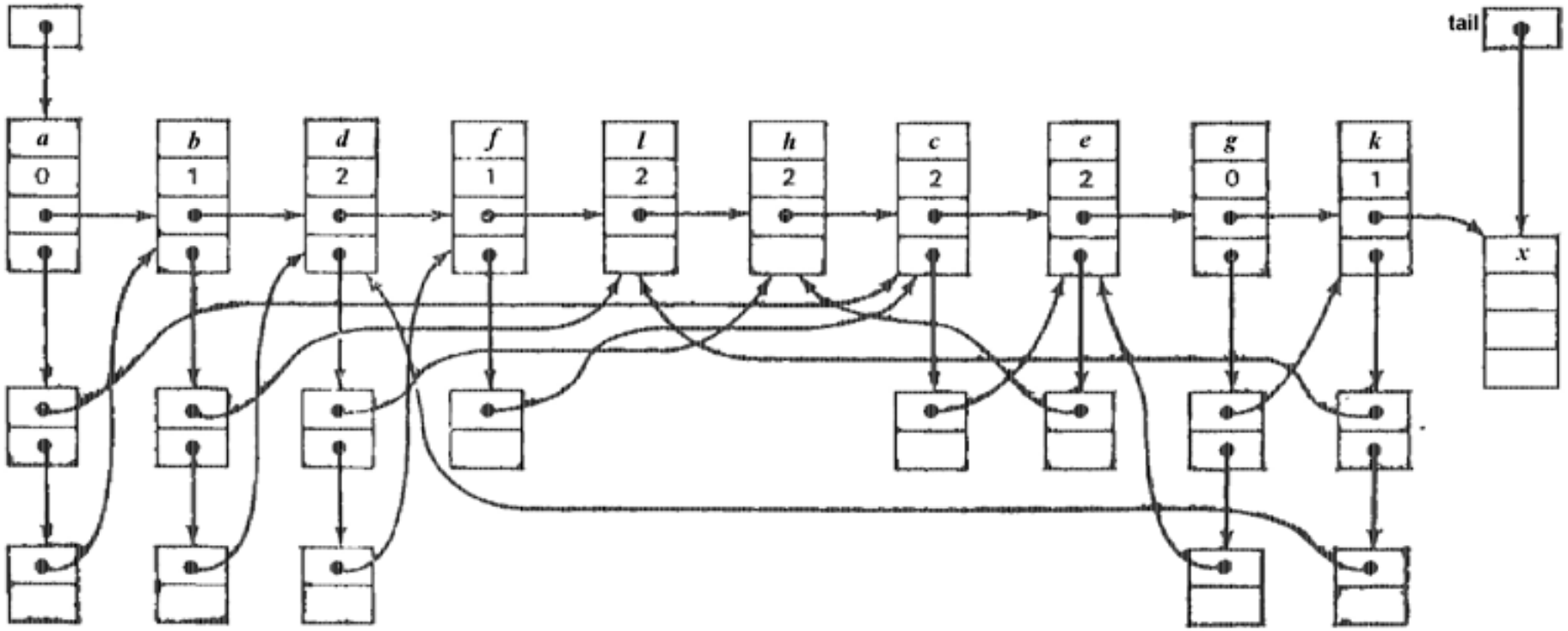
- ***1 фаза алгоритма: ввод исходного графа***

  На этой фазе вводятся пары ключей и из них формируется представление ациклического графа через дескрипторы узлов и списки ведомых узлов.

  - Исходные данные представлены в виде множества пар ключей (*), которые вводятся ***в произвольном порядке***.
  - После ввода очередной пары $x < y$ ключи x и y ищутся в списке «ведущих» и в случае отсутствия добавляются к нему.
  - В список ведомых узлов узла x добавляется ссылка на y, а счетчик предшественников y увеличивается на 1 (начальные значения всех счетчиков равны 0).

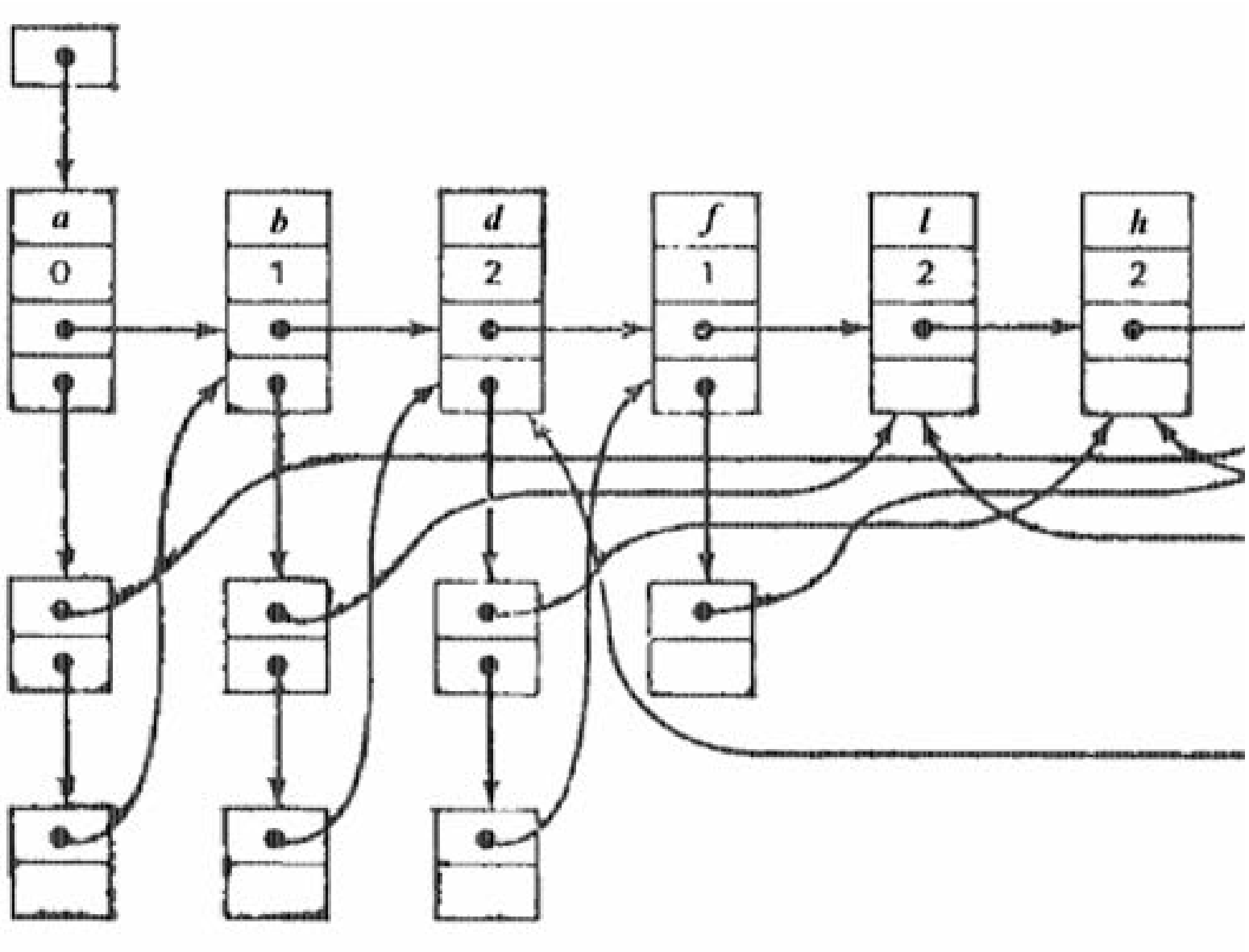
По окончании фазы ввода будет сформирована структура, показанная на следующем слайде (для множества пар ключей (*)).

Топологическая сортировка узлов ациклического ориентированного графа

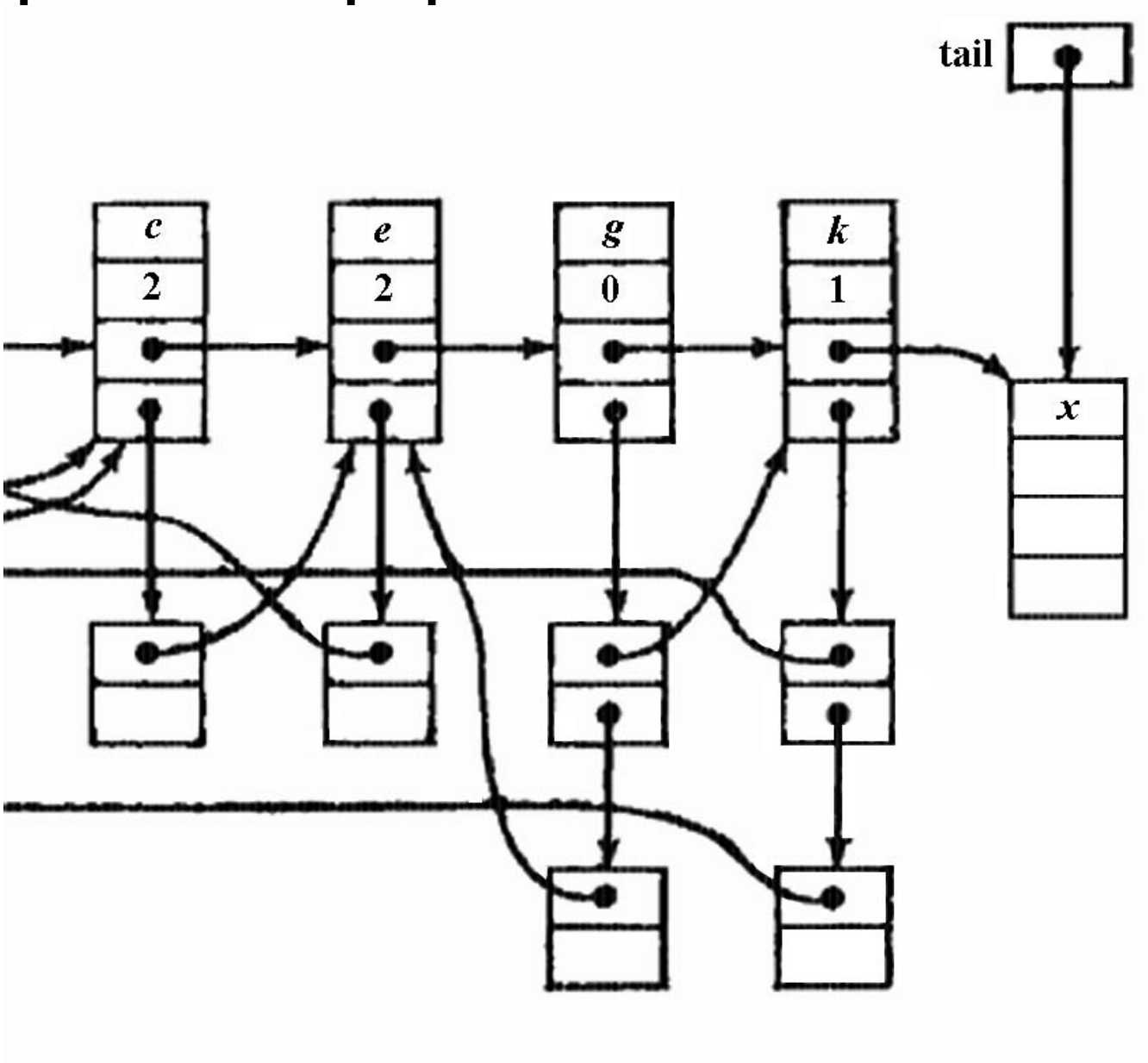


Структура данных, сформированная фазой ввода

Топологическая сортировка узлов ациклического ориентированного графа



Топологическая сортировка узлов ациклического ориентированного графа



Топологическая сортировка узлов ациклического ориентированного графа

- ***2 фаза алгоритма: сортировка***

  (1) В списке «ведущих» находим дескриптор узла ***z***, у которого значение поля **`count`** равно 0.

  (2) Включаем узел ***z*** в результирующую цепочку.

  (3) Если у узла ***z*** есть «ведомые» узлы (значение поля **`trail`** не **`NULL`**)

  - (a) просматриваем очередной элемент списка «ведомых» узлов
  - (b) корректируем поле **`count`** дескриптора соответствующего «ведомого» узла.

  (4) Переходим к шагу (1)

- Так как с каждой коррекцией поля **`count`** его значение уменьшается на 1, постепенно все узлы включаются в результирующую цепочку.