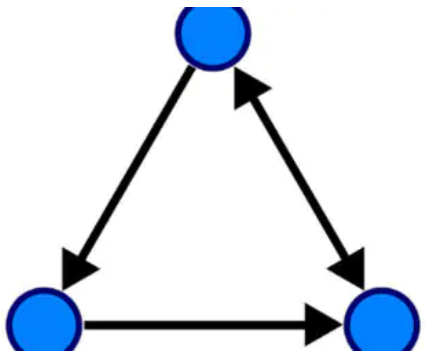
Всё в руках человека! Поэтому их нужно мыть чаще (Ежи Лец).

Алгоритм топологической сортировки

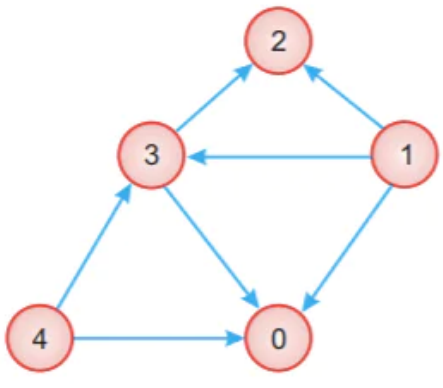
19.03.2022

Алгоритм топологической сортировки работает с DAG (прямой ациклический граф). Смысл топологической сортировки в том, что если какой-либо узел указывает на другой узел, то после него будет идти узел, указывающий на другой узел. Таким образом, в этом случае, если у нас есть циклический граф, мы не можем предсказать, какой узел после какого узла. Вот почему алгоритм топологической сортировки работает только с ациклическим графом, а не с циклическим графом.



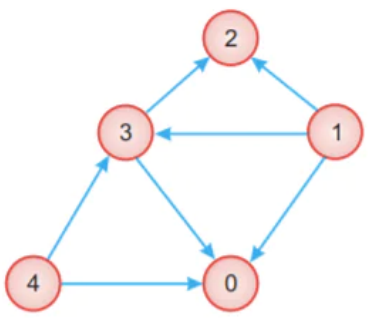
Каждый граф имеет более одной топологической последовательности сортировки, поскольку она зависит от степени вхождения ребер узлов. Первый начальный узел, который мы выбираем с числом узлов в степени, равен 0.

Давайте разберемся с алгоритмом топологической сортировки на примере.

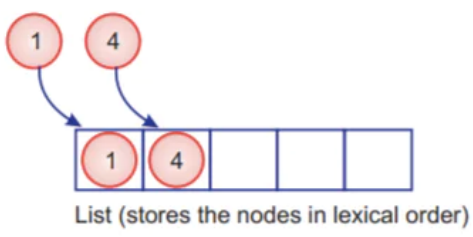


Node	Incoming edges Count
0	3
1	0
2	2
3	2
4	0

Шаг 1: мы вставляем те узлы, количество входящих ребер которых равно 0. Таким образом, эти узлы являются узлами 1 и 4.



Node	Incoming edges Count
0	3
1	0
2	2
3	2
4	0



Шаг 2:

a. Начнем с узла 1. Мы можем выбрать любой узел между узлами 1 и 4.

РЕЛЕВАНТНЫЕ СТАТЬИ

- Как создать уникальное изображение в Midjourney: подробный гайд с личным опытом
- Использование нейросетей для создания контента: баланс между возможностями и ограничениями
- В WordPress 6.7 дебютируют тема Twenty Twenty-Five и функция уменьшения масштаба
- Как принимать платежи на WordPress?
- Основные графовые алгоритмы встречаются на собеседованиях по программированию

Здесь может быть размещена ваша ссылка

САМЫЕ ПОПУЛЯРНЫЕ СТАТЬИ

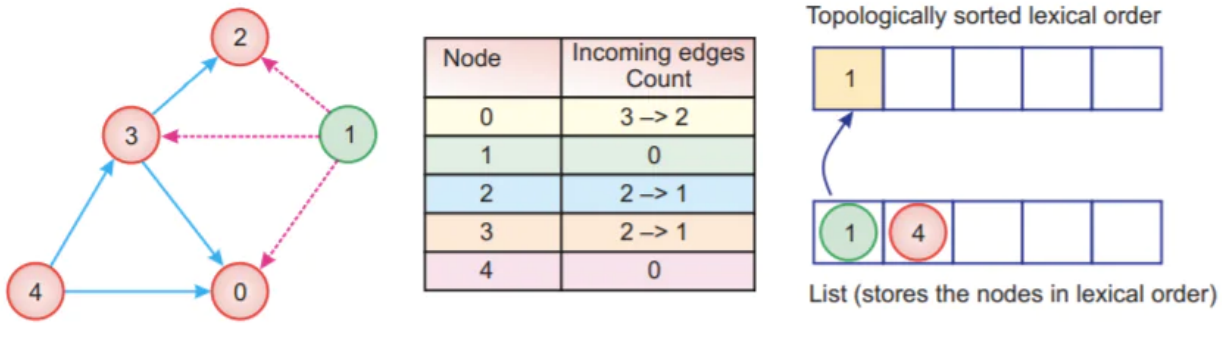
- 4 способа создать файл в терминале Linux
- Как вывести список баз данных и таблиц PostgreSQL с помощью psql
- Как найти IP шлюза по умолчанию в Linux
- Использование команды Sleep в скриптах Bash в Linux
- Команда Sed для удаления строки
- Как увеличить размер swap в Ubuntu
- Разница между постами и статьями в блоге
- 4 способа найти количество ядер CPU в Linux
- Команда Ping не найдена? Установка Ping в Ubuntu

ПАРТНЕРКА

FIRSTVDS
Самый популярный VDS-хостинг в России

б. Мы уменьшаем каждое ребро узла на 1, которое связано с узлом 1. Мы уменьшаем ребро узлов (0, 2 и 3).

в. Мы перемещаем узел 1 из списка в топологически отсортированный список, как показано ниже.



Шаг 3:

а. Теперь мы исходим из списка, который является Node 4.

б. Мы уменьшаем все исходящие ребра узлов, соединенных с узлом 4, которые являются узлами (0 и 3).

в. Теперь узел 3 не имеет входящих ребер, поэтому мы помещаем его в список, а узел 4 переходит в топологически отсортированный список.

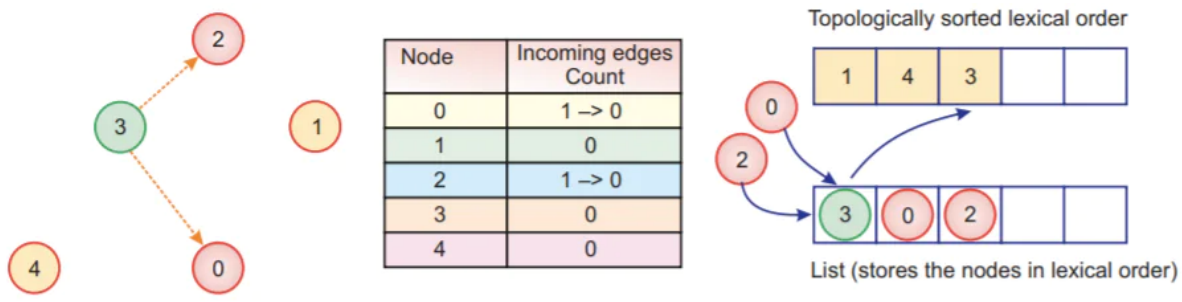
Читать **Типы анализа алгоритмов**

Шаг 4:

а. Теперь мы исходим из списка, который является узлом 3.

б. Мы уменьшаем все исходящие ребра узлов, соединенных с узлом 3, которые являются узлами (0 и 2).

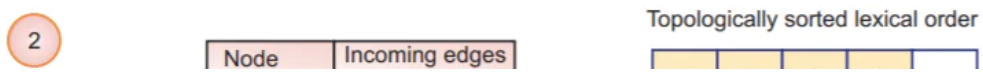
в. Теперь узлы 0 и 2 не имеют входящих ребер, поэтому мы помещаем их в список, а узел 3 переходит в топологически отсортированный список.



Шаг 5:

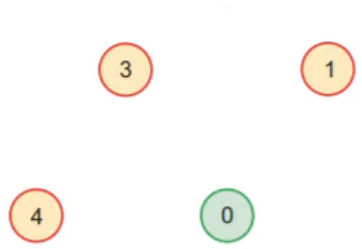
а. Теперь мы исходим из списка, который является Node 0.

б. Поскольку исходящих ребер из Node 0 нет, мы просто добавляем их в список топологической сортировки.

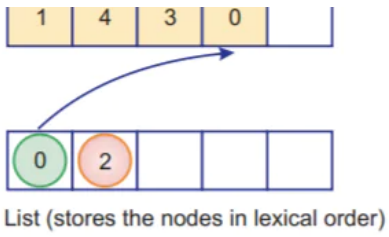


ПОСЛЕДНИЕ НОВОСТИ

- Lutris 0.5.18 добавляет тёмную тему по умолчанию и отображение обложек
- Linux 6.13 достиг «переломного момента», и вскоре ожидается появление новых драйверов Rust
- Обнаружено вредоносное ПО, которое может удалённо отключать индикатор веб-камеры на ThinkPad X230
- Red Hat и Microsoft внедряют RHEL в WSL
- Arch Linux использует лицензию 0BSD для исходных текстов пакетов
- Студент думал, что освоил Unix за несколько недель. Потом он обнаружил rm -rf



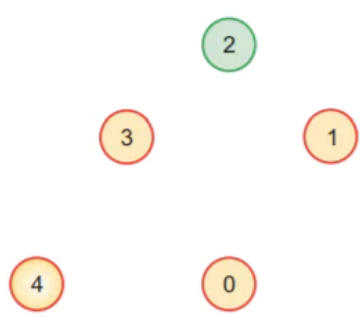
	Count
0	0
1	0
2	0
3	0
4	0



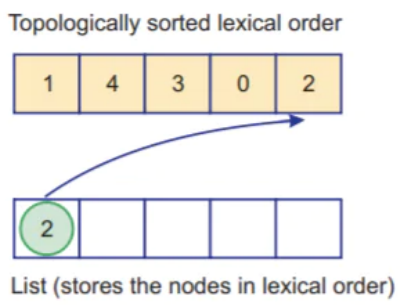
Шаг 6:

а. Теперь мы исходим из списка, который является узлом 2.

б. Поскольку исходящих ребер из узла 2 нет, мы просто добавляем их в список топологической сортировки.

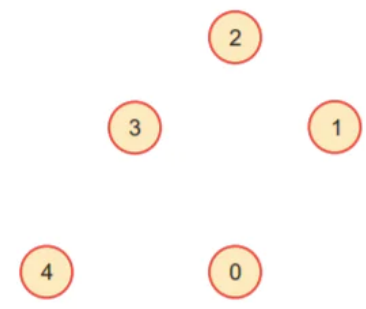


Node	Incoming edges Count
0	0
1	0
2	0
3	0
4	0

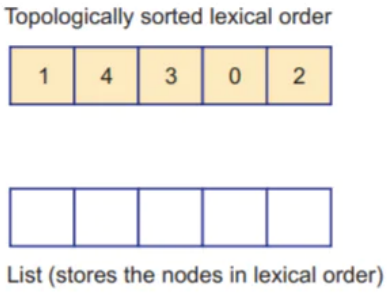


Шаг 7:

Наконец, мы отсортировали список здесь.



Node	Incoming edges Count
0	0
1	0
2	0
3	0
4	0



Алгоритм топологической сортировки

Ниже приведены шаги алгоритма топологической сортировки, которым мы должны следовать.

Шаг 0: Рассчитайте степень вхождения каждого узла графа.

Шаг 1: Сначала нам нужно найти узел, у которого входящие ребра равны нулю.

Шаг 2: мы удаляем этот узел из графа и добавляем его в список топологических порядков сортировки.

Шаг 3: Удалите те узлы, у которых есть исходящие ребра.

Шаг 4: Уменьшите степень вхождения на количество связанных ребер, которые были удалены.

Шаг 5: Повторяйте шаги 1–4, пока не останется узлов с нулевой степенью вхождения.

Шаг 6: Убедитесь, что все элементы расположены в правильной последовательности.

Шаг 7: Теперь мы отсортировали заказ из шага 6.

Читать Алгоритм Прима

Шаг 8: Положите конец алгоритму.

Код Python : ниже приведена реализация приведенного выше примера на Python.

```
fromcollectionsimportdefaultdict

classbuildGraph :

def__init__(self, nodes : int) :
self.nodes= nodes

# Теперь мы сохраняем граф в формате смежного списка
self.adjListDetails=defaultdict(list)

# Он будет хранить информацию о входящих
ребрах определенного узла

# в самом
self.count_numbers_of_incoming_edge_of_a_node= []

# Мы сохраняем отсортированные узлы в топологическом порядке
self.topological_sorted_order= []

# Мы храним информацию обо всех тех узлах, которые
# не имеют входящих ребер в графе
self.nodes_have_zero_incoming_edges= []

# Теперь мы создаем смежный список всех графов для сортировки
defAddGraphEdge (self, source :int, destination : int) :
self.adjListDetails[source].append(destination)
self.count_numbers_of_incoming_edge_of_a_node[destination] +=1

defTopologicalSortAlgorithm (self) :

for node inrange(self.nodes) :
ifself.count_numbers_of_incoming_edge_of_a_node[node] ==0 :
self.nodes_have_zero_incoming_edges.append(node)

whileself.nodes_have_zero_incoming_edges :
self.nodes_have_zero_incoming_edges.sort()
            source =self.nodes_have_zero_incoming_edges.pop(0)

# итерация по соседнему списку
if source inself.adjListDetails :
for node inself.adjListDetails[source] :
self.count_numbers_of_incoming_edge_of_a_node[node] -=1
ifself.count_numbers_of_incoming_edge_of_a_node[node] ==0 :
self.nodes_have_zero_incoming_edges.append(node)

self.topological_sorted_order.append(source)

print("Топологический порядок сортировки:
"+str(self.topological_sorted_order))

defmain() :

number_of_nodes=7
    graph =buildGraph(number_of_nodes)
graph.count_numbers_of_incoming_edge_of_a_node= [0] *number_of_nodes

graph.AddGraphEdge(0,2)
graph.AddGraphEdge(0,5)
graph.AddGraphEdge(1,3)
graph.AddGraphEdge(1,6)
graph.AddGraphEdge(2,4)
graph.AddGraphEdge(3,5)
graph.AddGraphEdge(5,2)
graph.AddGraphEdge(5,4)
graph.AddGraphEdge(6,2)

graph.TopologicalSortAlgorithm()
```

```
if __name__ =="__main__" :
main()
```

Читать **Что такое анализ данных?**

Выход:

```
Топологический порядок сортировки: [0, 1, 3, 5, 6, 2, 4]
```

Временная сложность алгоритма топологической сортировки:

Общее время обработки алгоритма равно O (E + N), где E представляет количество ребер, а N представляет количество узлов в графе. Затем, на следующем шаге, мы должны вычислить степень входа каждого узла, что обычно занимает O(E) раз, а затем поместить все эти узлы в отсортированный список, где их степень входа равна нулю, что занимает O(N) раз. раз. Таким образом, общая временная сложность алгоритма топологической сортировки составляет O (E+N).

Но пространственная сложность алгоритма топологической сортировки составляет O (N), что равно общему количеству узлов в графе.

Применение:

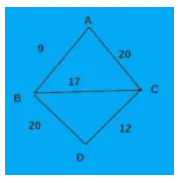
1. Топологическая сортировка очень полезна для нахождения цикла графа.
2. Алгоритм топологической сортировки используется для определения условий взаимоблокировки в операционной системе.
3. Алгоритм топологической сортировки используется для поиска кратчайшего пути во взвешенном ациклическом графе.

Вывод :

В этой статье мы узнали еще об одном важном алгоритме — топологической сортировке. Мы видели, что этот алгоритм работает только с ациклическими графами. Алгоритм топологической сортировки также помогает определить порядок составления задачи. Алгоритм топологической сортировки имеет много преимуществ в реальном времени, например поиск кратчайшего пути. Поскольку топологическая сортировка чрезвычайно полезна, каждый программист и студент должен хорошо понимать этот алгоритм.

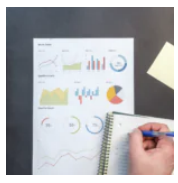
Тэги: , алгоритмы, сортировка

Категории: Программирование, Статьи, Структуры данных и алгоритмы



Алгоритм Крускала

Монотонные отношения



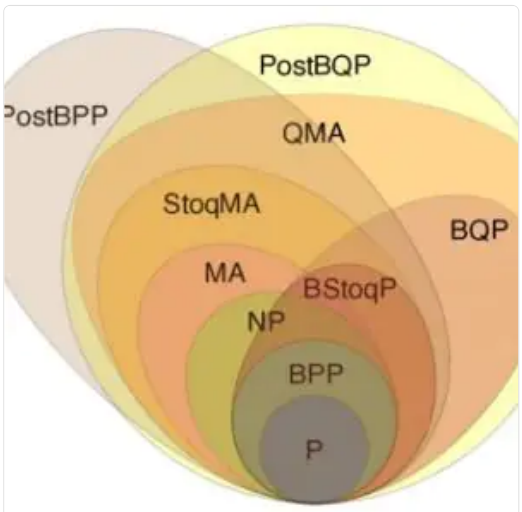
Подписаться авторизуйтесь

Оставьте первый комментарий!

**ссылки nofollow

0 КОММЕНТАРИЕВ

Это может быть вам интересно



Типы классов сложности

8 месяцев назад

Разница между Big Oh, Big Om...

8 месяцев назад

Основы анализа алгоритмов

8 месяцев назад

ОПЕРАЦИОННЫЕ СИСТЕМЫ
- Linux
- Debian
- Centos
- Ubuntu
- OpenSUSE

WORDPRESS
- Wordpress
- Учебное пособие по Wordpress
- Лучшие учебники по Wordpress
- SEO в WordPress для начинающих

БАЗЫ ДАННЫХ
- Базы данных
- Учебное пособие по SQL

BOOTSTRAP
- Учебное пособие по Bootstrap

PYTHON
- Учебное пособие по Python 3

JAVASCRIPT
- Учебное пособие по Javascript

RUBY
- Язык программирования Ruby

ПОСЛЕДНИЕ СТАТЬИ
- В предварительной версии OpenShot 3.3 представлен обновлённый пользовательский интерфейс и повышена производительность
- ScyllaDB переходит на открытую лицензию
- Выпуск CachyOS в декабре 2024 года: оптимизированное ядро, RustiCL и улучшения для беспроводной связи
- Обзор на Multilogin
- SEO vs SMM: Глубокое погружение в мир интернет-маркетинга

ПОСЛЕДНИЕ СТАТЬИ ПО LINUX
- В предварительной версии OpenShot 3.3 представлен обновлённый пользовательский интерфейс и повышена производительность
- Выпуск CachyOS в декабре 2024 года: оптимизированное ядро, RustiCL и улучшения для беспроводной связи
- openSUSE представляет YQPkg — автономный инструмент для управления пакетами с графическим интерфейсом
- DXVK 2.5.2 Улучшает игровой опыт Windows
- Выпущена бета-версия Fish Shell 4.0: полная переработка на Rust с ключевыми улучшениями