**1. Хеш-таблицы. Хеш. Вычисление хеша. Коллизии и разрешение их.  
Хеширование –** это преобразование входного массива данных определенного типа и **произвольной длины** в выходную битовую строку **фиксированной длины.**

Такие преобразования также называются хеш-функциями или функциями свертки, а их результаты называют хешем, хеш-кодом.

Результатом хеш-функции будет в число из определенного диапазона 0 , … , M-1

С точки зрения практического применения, хорошей является такая хэш-функция, которая удовлетворяет следующим условиям:

* функция должна быть **простой** с вычислительной точки зрения;
* функция должна **распределять ключи** в хэш-таблице наиболее **равномерно**;
* функция **не должна** отображать какую-либо связь между значениями ключей в связь между значениями адресов;

функция должна **минимизировать число коллизий**,то есть ситуаций, когда разным ключам соответствует одно значение хэш-функции

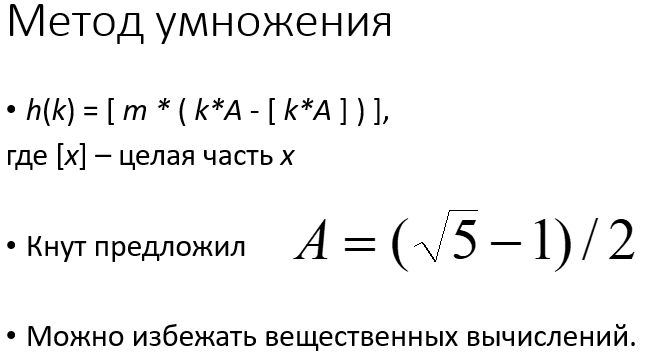
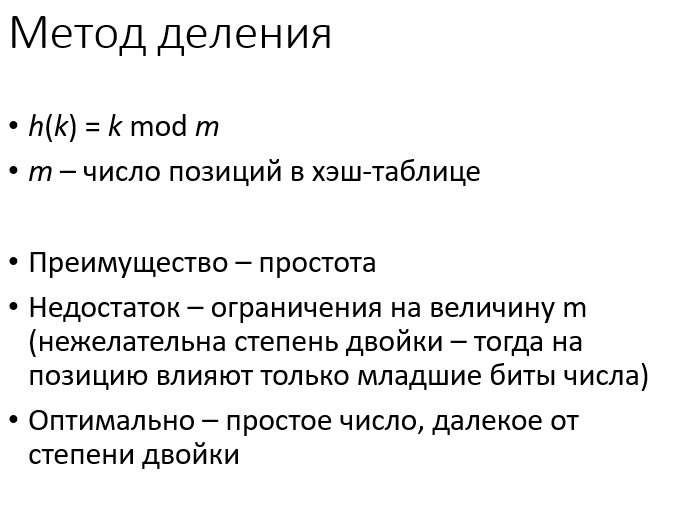
Идеальной хеш-функцией является такая hash-функция, которая для любых двух неодинаковых ключей дает неодинаковые адреса.

Наихудший случай: все ключи хэшируются в один индекс.

Хеширование полезно, когда широкий диапазон возможных значений должен быть сохранен в малом объеме памяти, и нужен способ быстрого, практически произвольного доступа

Варианты хэш-функции

* Метод деления
* Метод умножения
* Универсальное хэширование



**Универсальное хеширование:**

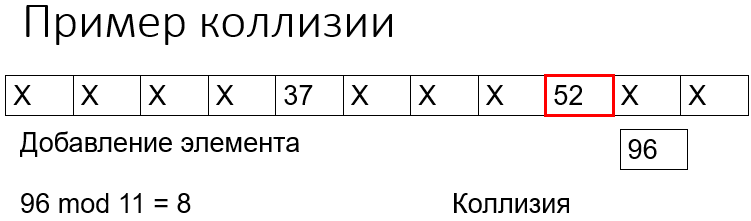
* Ясно, что для любой хеш-функции можно подобрать значения, при которых она работает плохо (коллизии на каждом шаге).
* Злоумышленник может посылать нам такие значения и спровоцировать неработоспособность нашей программы.
* Идея универсального хеширования – случайный выбор хеш-функции так, чтобы для любой сгенерированной злоумышленником последовательности вероятность проблем была мала

**Коллизия** - ситуация, когда разным ключам соответствует одно значение хэш-функции.

Когда?

* Мы не хотим выделять память на каждое возможное значение элемента (реально встретившихся значений обычно много меньше, чем возможных)
* Значит, возможных значений *h*(*x*) меньше, чем возможных значений *x* и существуют такие *x*1, *x*2, что *h*(*x*1)=*h*(*x*2)

Значит, возможна ситуация, когда мы пытаемся добавить элемент, а место занято.



**Разрешение коллизий: открытое хеширование (хранение списков)**

* Будем хранить в каждом элементе массива не значение, а список значений
* Новое значение добавляем в конец списка
* Поиск выполняется по списку

#### Суть этого метода проста: если хеш-функция выделяет один индекс сразу двум элементам, то храниться они будут в одном и том же индексе, но уже с помощью двусвязного списка.  2. Открытая адресация

В отличие от метода цепочек, в открытой адресации несколько элементов в одной ячейке храниться не могут. Суть этого метода заключается в том, что каждая ячейка либо содержит единственный ключ, либо NIL.

Существует несколько видов открытой адресации:

**a) Линейное зондирование**

Линейное зондирование решает проблему коллизий с помощью проверки следующей ячейки.

h(k, i) = (h′(k) + i) mod m,

где

* i = {0, 1, ….}
* h'(k) — новая хеш-функция

Если коллизия происходит в h(k, 0), тогда проверяется h(k, 1). То есть значение i увеличивается линейно.

Проблема линейного зондирования заключается в том, что заполняется кластер соседних ячеек. Это приводит к тому, что при вставке нового элемента в хеш-таблицу необходимо проводить полный обход кластера. В результате время выполнения операций с хеш-таблицами увеличивается.

**b) Квадратичное зондирование**

Работает оно так же, как и линейное — но есть отличие. Оно заключается в том, что расстояние между соседними ячейками больше (больше одного). Это возможно благодаря следующему отношению:

h(k, i) = (h′(k) + c1i + c2i2) mod m,

где

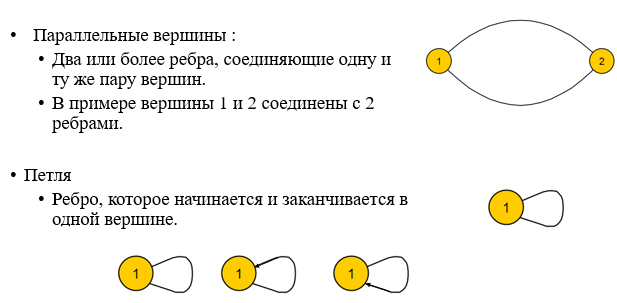
* c1 и c2 — положительные вспомогательные константы,
* i = {0, 1, ….}

**Грокаем алгоритмы стр. 100**

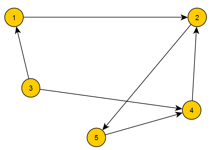
**2. Графы. Представление графа. Матрица смежности. Измерение размера графа.**

**Что такое граф?**

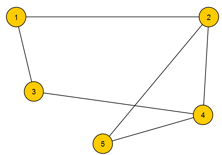
* **Граф** — это структура, представляющая собой набор объектов, в котором некоторые пары объектов в некотором смысле «связаны».
* Объекты, называемые **вершинами** (также называемыми *узлами* или *точками*).
* Каждая из связанных пар вершин называется ребром.



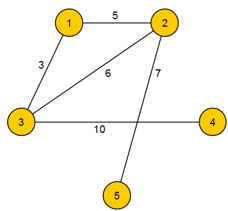
* Если ребра в графе ориентированы, т. е. указывают только в одном направлении, граф **называется ориентированным графом** или **орграфом**.

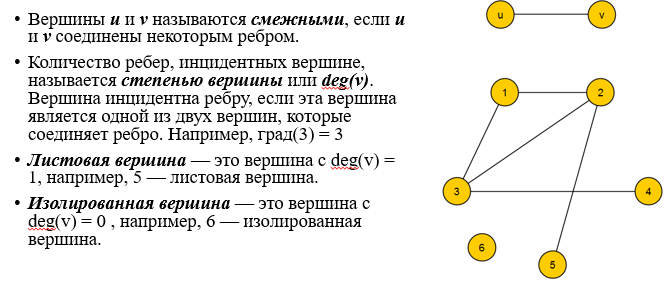
Орграф

* Если ребра в графе не имеют направления, граф называется ***неориентированным графом.***



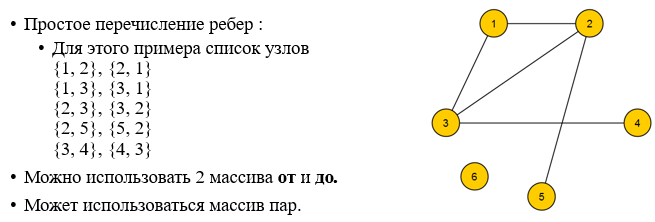
* Граф, в котором каждое ребро имеет числовой «вес», называется ***взвешенным графом***.



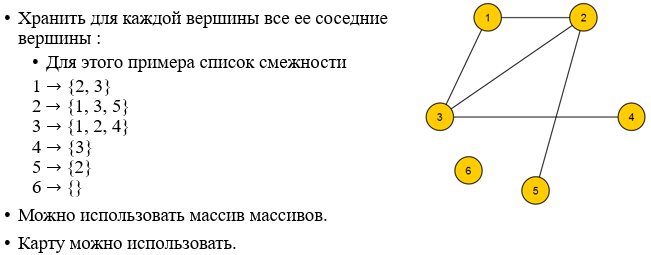


Представление Графа

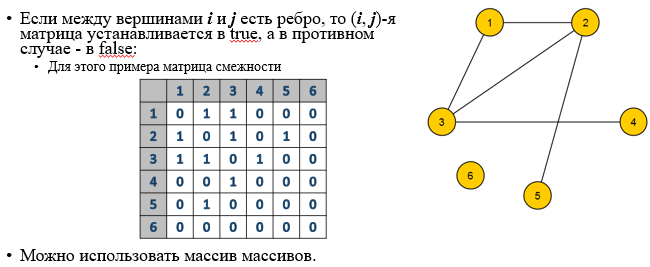
**Список ребер**



**Список смежности**



**Матрица смежности**



**Грокаем алгоритмы стр. 131**

**3. Поиск в графе и его применения. Обобщенный графовый поиск. Поиск в ширину и в глубину.**

 Алгоритмы и структуры данных. Семестр 2. Лекция 1. Графы

**4. Поиск в глубину. Топологическая сортировка. Вычисление топологического упорядочивания.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 3-4. Графы**

**5. Алгоритм кратчайшего пути Дейкстры.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 3-4. Графы**

**Грокаем алгоритмы стр. 152**

**6. Алгоритм кратчайшего пути Флойда – Уоршелла.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 3-4. Графы**

**7. Волновой алгоритм (Алгоритм Ли).**

**Алгоритмы и структуры данных. Семестр 2. Лекция 3-4. Графы**

**8. Алгоритм кратчайшего пути Форда – Фалкерсона.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 3-4. Графы**

**9. Минимальное остовное дерево. Алгоритм Прима.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 5. Графы**

**10. Минимальное остовное дерево. Алгоритм Краскала.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 5. Графы**

**11. Парадигма проектирования жадных алгоритмов. Жадный алгоритм Хаффмана.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 7.**

**12. Парадигма проектирования жадных алгоритмов. Задача о составлении расписания. Задача о рюкзаке.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 7.**

**Грокаем алгоритмы стр. 183**

**13. Динамическое программирование. Числа Фибоначчи.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 8**

**Алгоритмы и структуры данных. Семестр 2. Лекция 9**

**14. Динамическое программирование. Возрастающая подпоследовательность.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 8**

**Алгоритмы и структуры данных. Семестр 2. Лекция 9**

**15. Динамическое программирование. Путь в лабиринте.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 8**

**Алгоритмы и структуры данных. Семестр 2. Лекция 9**

**16.Динамическое программирование.**

**Общая подпоследовательность.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 8**

**Алгоритмы и структуры данных. Семестр 2. Лекция 9**

**Грокаем алгоритмы стр. 232**

**17. Динамическое программирование. Подзадачи и рекуррентные отношения.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 8**

**Алгоритмы и структуры данных. Семестр 2. Лекция 9**

**18. Динамическое программирование. Задача о лягушке.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 8**

**Алгоритмы и структуры данных. Семестр 2. Лекция 9**

[**https://www.youtube.com/watch?v=UVAfng6OCW0**](https://www.youtube.com/watch?v=UVAfng6OCW0)

**19. Динамическое программирование. Расстояние редактирования. Взвешенное расстояние редактирования.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 8**

**Алгоритмы и структуры данных. Семестр 2. Лекция 9**

**20. Динамическое программирование. Задача о рюкзаке. Задача о рюкзаке с повторением и без повторения.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 10**

**Грокаем алгоритмы стр. 206**

**21. Динамическое программирование. Перемножение матриц.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 10 (15 слайд)**

**22. Недетерминированные полиномиальные задачи (NP-задачи). Задача коммивояжера. Генетический алгоритм.**

**Алгоритмы и структуры данных. Семестр 2. Лекция 13**

[**https://www.youtube.com/watch?v=aB\_6ZsLjzzc**](https://www.youtube.com/watch?v=aB_6ZsLjzzc)

**23. Недетерминированные полиномиальные задачи (NP-задачи). Задача коммивояжера. Метод ветвей и границ (алгоритм Литтла).**

**24. Недетерминированные полиномиальные задачи (NP-задачи). Задача коммивояжера. Муравьиный алгоритм.**

**24. Алгоритмы с возвратом. Задача о ходе коня. Задача о ферзях.**

**25. Алгоритм k-ближайших соседей.**

**26. Асимптотические обозначения. Математическое определение. Обозначение Омега-большое, Тета-большое и о-малое.**

**27. Сортировки данных. Классы алгоритмов сортировки. Оценка алгоритмов сортировки. Обменная сортировка. Пузырьковая сортировка. Быстрая сортировка. Сортировка вставками.**

**28. Динамические структуры данных. Вектор. Связанный список. Двухсвязный список. Стек. Очередь. Словарь. Множество.**

**29. Поиск. Последовательный поиск. Бинарный поиск. Интерполяционный поиск. Поиск по деревьям.**

**30. Рекурсия. Виды рекурсивных функций. Перебор с помощью рекурсии. Решение задач с помощью.**