Фундаментальные структуры данных

Понятие

Структура данных –

- *это класс однородных математических объектов, ориентированный на эффективное представление данных в некотором классе задач.*
- *это систематизированный способ организации данных и доступа к ним.*

Классификация

1. Простые
2. Статические
3. Полустатические
4. Динамические
5. Файловые

Простые структуры данных

- Числовые
- Символьные
- Логические
- Перечисление
- Интервал
- Указатели

Статические структуры данных

- Вектор
- Массивы
- Множества
- Записи
- Таблицы

Полустатические структуры данных

- Стеки
- Очереди
- Деки
- Строки

Динамические структуры данных

- Связные списки
- Графы
- Деревья

Файловые структуры

- Последовательные
- Прямого доступа
- Комбинированного доступа

Требования, предъявляемые к структурам данных:

- Эффективность представления
- Экономное расходование памяти
- Быстрый доступ к элементам структуры.

Массив

Массив – структура данных для представления множества элементов, однотипных по структуре и способу использования.

Массивы относятся к структурам данных со случайным доступом : для выделения некоторой компоненты к имен массива добавляется индекс (значение специального типа), который можно вычислить, потому ***элементы массива иногда называют переменным с индексами***

Базовые операции

1. поиск элемента, равного x
   - В произвольном массиве $T(n) = \Theta(n)$.
   - В отсортированном массиве $T(n) = \Theta(\log n)$
2. поиск максимального (минимального) элемента.
   - В произвольном массиве $T(n) = \Theta(n)$.
   - В отсортированном массиве $T(n) = \Theta(1)$.

Базовые операции

Обычный прием работы с массивами – это

- поиск
- выборочное изменение отдельных компонент массива.

Основные операции над массивами не предполагают включения новых элементов или исключения элементов.

Однако на практике часто необходимо выполнять процедуры включения и исключения отдельных элементов, что приводит к необходимости разработки структур данных, поддерживающих эти операции.

Линейный список

Список

– упорядоченная последовательность данных, характеризующих однородные объекты, отличающиеся значениями своих признаков.

– конечная последовательность элементов, порядок которых определяется с помощью ссылок.

Линейный список – конечная последовательность элементов (множество), структурные свойства которой ограничиваются лишь линейным (одномерным) относительным порядком элементов.

Базовые операции

1. формирование списка ($T(n) = \Theta(n)$)
2. просмотр списка ($T(n) = \Theta(n)$)
3. поиск некоторого заданного элемента с ключом x ($T(n) = \Theta(n)$)
4. поиск максимального (минимального) элемента ($T(n) = \Theta(n)$)
5. включение элемента с ключом x ($T(n) = \Theta(1)$)
6. исключение элемента ($T(n) = \Theta(1)$)

Виды списков

Однонаправленный список – список, в котором предусмотрен жесткий порядок перебора элементов – от первого к последнему.

Двунаправленный список – список, структура которого предусматривает возможность перебора элементов как в прямом, так и в обратном порядке.

Способы представления одно- и двунаправленного списков аналогичны.

Способы реализации

1. В виде двух массивов A и B:

Пусть i - индекс элемента в списке, тогда

A[i] - сам элемент,

B[i] - индекс следующего элемента в списке A. B[k] = 0, где k - индекс последнего элемента в списке.

Две переменные:

nz -индекс начала занятых компонент,

ns - индекс начала свободных компонент.

Способы реализации

Список: 8, 13, 5

A:

8	5	13	*	*	*
1	2	3	4	5	6

B:

3	0	2	5	6	0

nz = 1, ns = 4

Способы реализации

2. С использованием последовательности связанных компонент (линейный список):

Каждая компонента списка состоит из двух ячеек памяти (первая содержит сам элемент либо указатель на его местоположение, а вторая – указатель на следующий элемент)

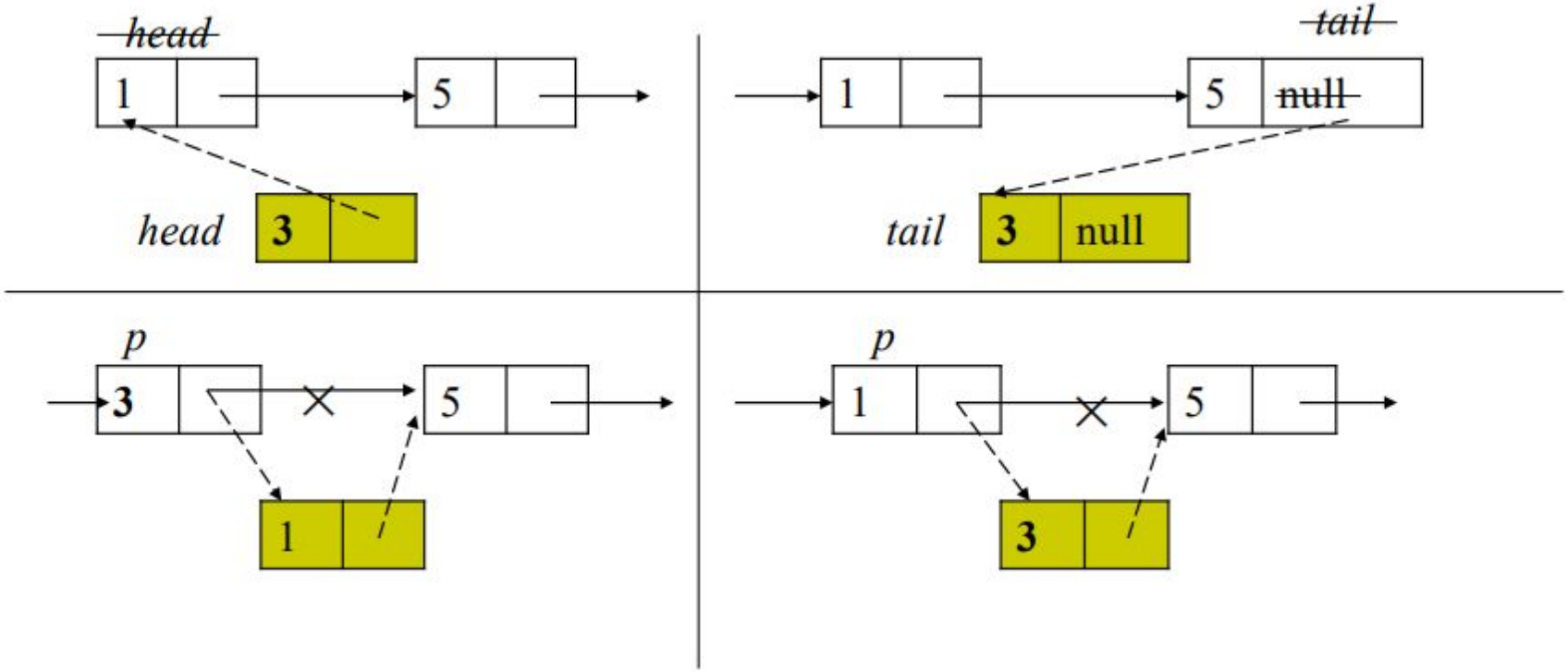
Способы реализации

Список: 8, 13, 5

Включение элемента

- в начало (конец) списка
- до (после) элемента, на который указывает заданная ссылка p

Включение элемента



Формирование списка

1. Начиная с пустого списка последовательно включаем элементы в начало списка.
   - не надо обрабатывать отдельно ситуацию, когда включается элемент в пустой список
   - порядок следования элементов в списке обратен порядку их включения
2. Включаем элементы в конец списка.
   - порядок следования элементов совпадает с порядком их включения
   - необходимо ввести указатель на последний поступивший элемент (tail)
   - первый включаемый элемент обрабатывается иначе, чем остальные элементы.

Исключение элемента из списка

1. стоящего после элемента, на который указывает заданная ссылка p.
2. на который указывает заданная ссылка p
   - p – указатель не на последний элемент
   - p – указатель на последний элемент.

Поиск элемента x в неупорядоченном списке

1. Осуществляется последовательным просмотром элементов
   - заканчивается либо при обнаружении требуемого элемента, либо при достижении конца списка
   - чтобы оптимизировать условие окончания просмотра, будем использовать технику барьера (s.key ← x)
   - переменная head указывает на начало списка

   для того, чтобы процедура отработала правильно в случае когда список пустой, необходимо при формировании списка выполнить оператор: head ← s.

Поиск элемента x в упорядоченном списке

- можно заканчивать при обнаружении первого ключа, со значением большем x.
- упорядоченность списка достигается путем включения нового элемента в подходящее для него место, что позволяет полностью использовать гибкость списковой структуры.
- Однако, даже упорядоченные линейные списки не позволяют организовать ничего подобного на двоичный поиск в массивах.

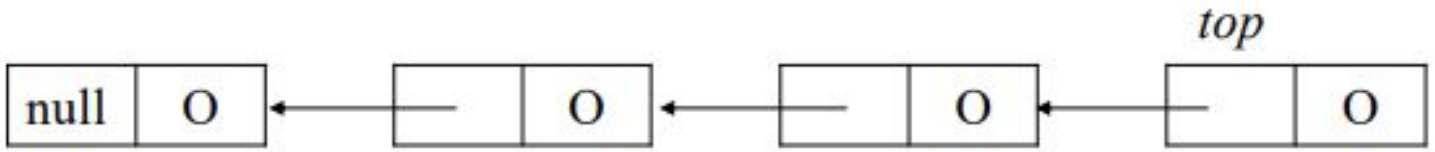
Операции работы со списками

- Конкатенация (сцепление) двух списков. В результате образуется единый список.
  - T = Θ(1), если имеются адреса первого и последнего элементов списков.
  - Если известны только начальные адреса, то T = Θ(n+ m), где n, m – количество элементов в сцепляемых списках. Или T = O(max{n, m})
- Расцепление списка.
  - T = Θ(1), если известен указатель на элемент непосредственно предшествующий месту расцепления.

Стек

Стек – линейный однонаправленный список, в котором все включения и исключения элементов (и обычно всякий доступ) делаются в одном конце списка.

Реализуется принцип “последний вошел – первый вышел” (last-in, first-out – LIFO).



Представление стека

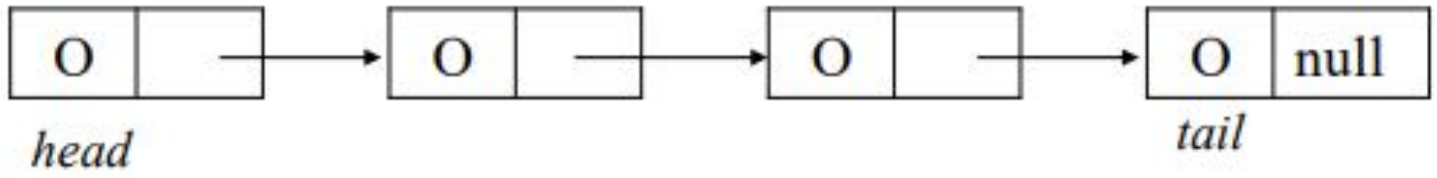
- Если заранее известно максимальное количество элементов, одновременно хранящихся в стеке, то целесообразно моделировать стек на массиве постоянной длины.
  - Переменная top содержит индекс последнего включенного стек элемента (первоначально top = 0).
- В случае, когда количество элементов заранее неизвестно, стек может быть реализован виде списковой структуры.
  - Переменная top содержит адрес последнего включенного в стек элемента (первоначально top = null).

Базовые операции

- Операция добавления элемента в стек часто обозначается Push, а операция удаления – Pop.
- Трудоемкость процедур включения и исключения элемента из стека есть $\Theta(1)$.

Очередь

- ***Очередь*** – линейный список, в котором все включения элементов производятся в одном конце списка, а все исключения (и обычно всякий доступ) делаются в другом его конце.
- Реализуется принцип “первый вошел первый вышел” (first-in, first-out – FIFO).



Очередь

Если максимальное количество элементов, одновременно хранящихся в очереди не превосходит l, то можно смоделировать очередь на массиве постоянной длины l.

- Переменные head и tail, определяют индексы первого и последнего элементов очереди (первоначально head= 1 и tail =0);
- В случае циклической очереди необходима переменная булевского типа empty, которая равна true в случае, когда очередь пуста.

Очередь

Если количество элементов очереди заранее не известно, то для реализации очереди используются списковые структуры.

- Переменная head содержит адрес первого элемента очереди, а tail – адрес последнего элементов очереди (первоначально head, tail = null).

Трудоемкость процедур включения и исключения элемента из очереди есть Θ (1).

Дек

Дек–линейный список, в котором все включения и исключения элементов (и обычно всякий доступ) делаются на обоих концах списка.

Deque(double-ended queue)–очередь с двусторонним доступом