Список

Тисак Таїсія

24.04.2020

- *Список* - базова динамічна структура даних, що складається з вузлів, кожен з яких містить дані та посилання на попередній та/або наступний вузол списку.

  Однозв’язний список містить вказівку на наступний елемент, двозв’язний вказує на наступний та попередній елементи.

  Перший елемент називається “головою”, останній - “хвостом”.

  Застосування: списки інтенсивно застосовуються в програмуванні як самостійні структури. Також на їх основі можуть будуватись складніші структури даних, такі як дерева. На базі списків також можуть бути реалізовані стеки та черги.

  Двобічний vs однобічний список: двобічне зв'язування потребує більше пам'яті на елемент та більших обчислювальних затрат на виконання елементарних операцій. Але такими списками легше маніпулювати, тому що вони дозволяють проходження по списку в обох напрямах.

Переваги над масивами

+ У списках набагато ефективніше (за час O(1), тобто незалежно від кількості елементів) виконуються процедури додавання та вилучення елементів(тоді як масиви для цього потребують часу O(n), тобто час зростає з ростом кількості елементів масиву.)

+ В списках також не існує проблеми «розширення», яка рано чи пізно виникає в масивах фіксованого розміру, коли виникає необхідність включити в нього додаткові елементи. Функціонування списків можливо в ситуації, коли пам'ять комп'ютера фрагментована, тоді як масиви переважно потребують неперервної області для зберігання.

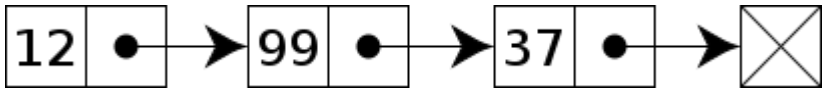
Недоліки

- Масиви набагато кращі в операціях, які потребують безпосереднього доступу до кожного елементу, що у випадку зі зв'язаними списками неможливо та потребує послідовного перебору усіх елементів, які передують даному.

- Іншим недоліком списків є необхідність разом з корисною інформацією додаткового збереження інформації про вказівники, що позначається на ефективності використання пам'яті цими структурами.

Різновиди: однозв'язний список

В однобічно зв'язаному списку кожний елемент складається з двох полів: data та вказівника next на наступний елемент. Якщо вказівник не вказує на інший елемент (інакше: next = NULL), то вважається, що даний елемент — останній в списку.

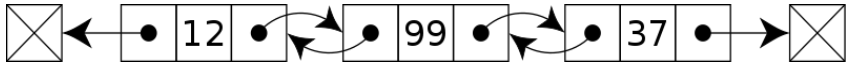


Операції зі списком:

- Додавання вузла в кінець списку: для того, щоб додати вузол **A** у кінець списку, треба знайти останній вузол **B** у цьому списку, заповнити інформаційну частину вузла **A** і вказівнику вузла **A** присвоїти NULL, і «приєднати» його до останнього вузла у списку, тобто до вузла **B**.
- Додавання вузла у початок списку: для того, щоб додати вузол **A** у початок списку, потрібно заповнити інформаційну частину вузла **A**, вказівник **A** направити на голову **head** списку і зробити цей вузол головою.
- Видалення вузла зі списку: для того, щоб видалити необхідний вузол, потрібно послідовно перебирати вузли, запам'ятовуючи попередній вузол **B**. Коли необхідний вузол **A** буде знайдено, потрібно вказівник «попередника» (тобто вузла **B**) зв'язати з наступним вузлом (тим, що йде після вузла **A**) і видалити вузол **A**.

Різновиди: двозв'язний список

В двобічно зв'язаному списку елемент складається з трьох полів — вказівника на попередній елемент *prev*, поля даних *data* та вказівника *next* на наступний елемент. Якщо *prev=NULL*, то в елемента немає попередника (тобто він є «головою» списку), якщо *next=NULL*, то в нього немає наступника («хвіст» списка).

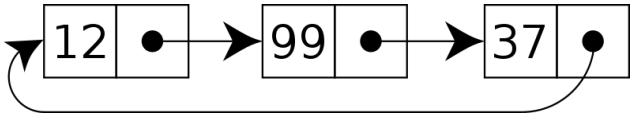


По двобічно зв'язаному списку можна пересуватися в будь-якому напрямку — як від початку до кінця, так і навпаки. Для нього простіше проводити видалення і перестановку елементів, оскільки завжди відомі адреси тих елементів списку, вказівник яких спрямований на змінюваний елемент.

Різновиди: кільцевий список

В кільцевому списку перший та останній елемент зв'язані. Тобто, поле *prev* голови списка вказує на хвіст списка, а поле *next* хвоста списка вказує на голову списка.

Також може бути одно- або двозв’язним(тоді також є вказівник з першого елемента на останній).



Різновиди: XOR-зв’язний список

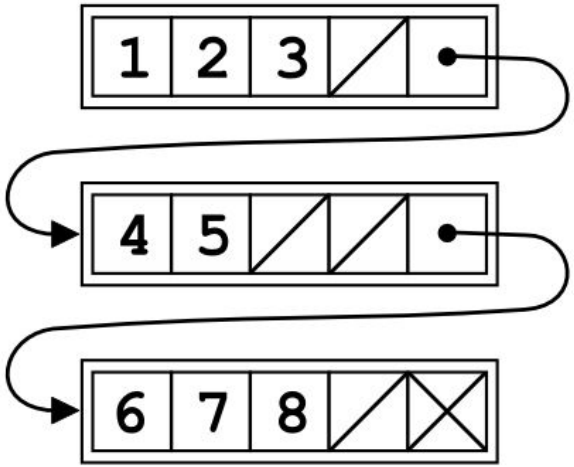
Структура даних, схожа на звичайний двозв’язний список, але в кожному елементі зберігається лише одна адреса - результат операції XOR над адресами попереднього і наступного елементів списка. Для переміщення по списку необхідно знати адреси попереднього та наступного елементів.

Альтернатива - розгорнутий зв’язний список.

Різновиди: розгорнутий зв'язний список

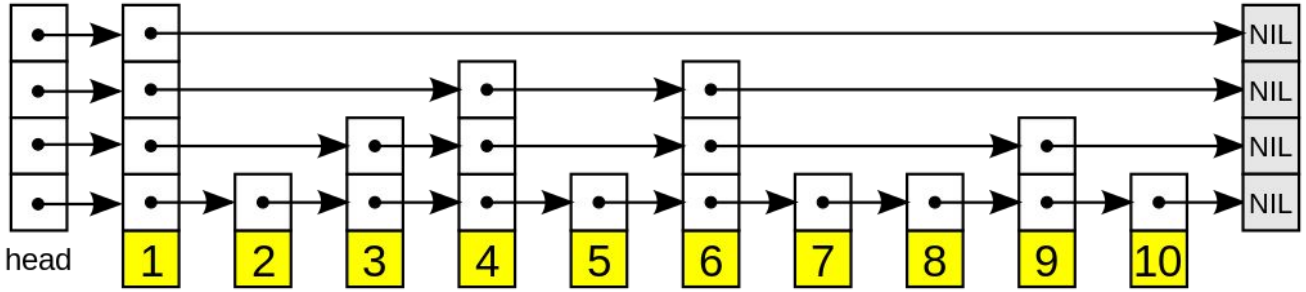
Це список, кожен елемент якого складається з декількох логічних елементів(зазвичай у вигляді масиву, щоб прискорити доступ до окремих елементів).

Такий вид дозволяє значно зменшити використання пам'яті(досягається за малих розмірів логічних елементів та їх великій кількості) і збільшити продуктивність(велика частина операцій проводиться над відносно невеликими масивами) порівняно зі звичайним списком. В такий список легко додавати нові елементи, не виникає необхідності переписувати весь масив.



Різновиди: список з пропусками

Це структура даних, яка дозволяє швидкий пошук в упорядкованій послідовності елементів. Швидкий пошук стає можливим через утримання зв'язаної ієрархії підпослідовностей, кожна з яких пропускає кілька елементів з попереднього списку. Пошук стартує в найрозрідженішому списку, і триває допоки не знайдені два послідовних елементи один менший і один більший від шуканого елементу. Через ієрархічні зв'язки ці два елементи зв'язані з елементами наступного по щільності списку, в якому пошук продовжується. Таким чином ми доходимо до повного списку, без пропусків. Елементи пропущені у розріджених списках можуть обиратись імовірнісно.



Різновиди: hash linking

Вказівники можуть не бути частиною вузла. Якщо дані зберігаються в масиві та є проіндексованим, то вказівник може знаходитися в окремому масиві з тим самим індексом, що й дані.

Дякую за увагу!