

Література

1. Карашецкий В.П. Расчет коэффициента Картера при наличии магнитного клина в пазу электрической машины методом конечных элементов// Электромеханика: Изв. вузов СССР. – 1988, № 2. – С. 14-19.
2. Никитенко А.Г., Пеккер И.И. Расчет электромагнитных механизмов на вычислительных машинах. – М.: Энергоатомиздат, 1985. – 215 с.
3. Шимони К. Теоретическая электротехника. – М.: Мир, 1964. – 773 с.

УДК 681.3.06

Студ. Ю.Т. Ликтей; доц. Ю.І. Грицюк, канд. техн. наук –
НЛТУ України, м. Львів

ПРОЕКТУВАННЯ РОЗРІДЖЕНИХ МАСИВІВ ДЛЯ РОБОТИ З ЕЛЕМЕНТАМИ ТОПОЛОГІЧНИХ МАТРИЦЬ

Розглянуто питання створення структурованого типу даних – *розріджених масивів* (sparse arrays) для здійснення ефективної роботи з матрицями великих розмірів, у яких значна кількість елементів не використовується. Здатність розріджених масивів оптимізувати доступ до таких даних робить їх надзвичайно корисними в багатьох задачах, які виникають під час розроблення сучасних програмних продуктів. До таких задач належить робота з топологічними матрицями, які використовуються для відображення зв'язків елементів у електронних схемах, що асоціюються з її клітинами. Завдяки використанню розріджених масивів пам'ять для зберігання кожної з таких клітин виділяється з пули вільної пам'яті тільки в міру потреби.

Stud. Yu.T. Lyktej, assist. prof. Yu.I. Grytcyuk – NUFWT of Ukraine, L'viv

Organization of disperse an array is for work with the elements of topology matrices

The question of creation of the structured type of information is considered – disperse (sparse arrays) an array for realization of effective work with the matrices of ageneses which the far of elements is not utilized in. To optimize ability of disperse an array access to such information does them extraordinarily useful in many tasks which arise up during development of modern software products. To such tasks work belongs with topology matrices, which are utilized for displaying of connections of elements in electronic charts which are associated with its cages. Due to the use of disperse an array memory is for storage each of such cages selected from pools of free space only to the extent of necessity..

Вступ

Під час виконання різних інженерних розрахунків часто доводиться мати справу з так званими розрідженими масивами, значна кількість елементів яких фактично не використовується. У середовищі фахівців [1, с. 112], робота яких пов'язана з дослідженням розріджених масивів, у обігу широко вживають два терміни: логічний і фізичний масиви. Логічний масив є уявним, фігурує тільки у математичних записах і графічних представленнях, а фізичний – фактично існує тільки в програмі. Наприклад, якщо певна електронна схема описується топологічною матрицею (суміжності чи інциденцій) розміром 100×5000 елементів, то логічний масив складатиметься з такої ж самої кількості елементів навіть у тому разі, якщо цей масив у програмі фізично не існує. Якщо у цій матриці фактично використовуються тільки 100 елементів, то саме вони і займатимуть фізичну пам'ять комп'ютера.

Поодинокі елементи чи навіть структури (записи), які зберігаються у звичайному масиві (чий індекс є аналогом ключа), можуть бути знайдені дуже швидко і без будь-яких порівнянь. Якщо відомо ключ (тобто індекс масиву), розмір запису і початкову адресу масиву, то адресу потрібного запису обчислюємо за допомогою множення і додавання. Зазвичай, це питання стає цікавим тільки тоді, коли нам потрібно зберігати значну кількість записів. Нехай, замість 100 елементів у нас є 5 000, а розмір топологічної матриці залишається попереднім. Наш гіпотетичний масив, який складається з такої кількості елементів, все ще на 99 % порожній, тобто ми тепер не можемо знехтувати проблемою пошуку потрібного запису серед їх півмільйонної кількості. Структурований тип даних, який має назву *розріджений масив*, якраз і призначений для вирішення таких неоднозначних ситуацій шляхом ефективного пошуку відповідних об'єктів.

Для зберігання розрідженого масиву, як правило, використовують одну з динамічних структур даних – переважно однозв'язні списки. Зазвичай кожен об'єкт такого списку містить як мінімум два поля: логічний індекс елемента і його значення. Розгляд основних особливостей проектування одно- і двовимірних розріджених масивів для роботи з матрицями, більшість елементів яких не використовується, є актуальним, становить науковий і практичний інтерес, є основною метою даної роботи.

1. Проектування розріджених масивів

Більшість питань, які виникають у програмістів під час оброблення матриць великих розмірів, стосуються в реалізації механізму індексування елементів масивів. Деякі з сучасних прикладних програм працюють з дуже великими масивами, значна частина елементів яких дорівнює нулю або відповідає якомусь іншому значенню за замовчуванням. У процесі розроблення таких програм вигідним є усунення витрат пам'яті не тільки для їх зберігання, але й для здійснення ефективного пошуку потрібного елемента. Розріджені масиви [5, с. 118] є динамічними структурами об'єктів, які поводяться як масиви, але не вимагають місця в пам'яті комп'ютера для зберігання значень даних, які містять нулі або певні значення за замовчуванням.

а)

Col	Data	Right
-----	------	-------

б)

-1		•
----	--	---

 →

0	4	•
---	---	---

 →

2	12	X
---	----	---

Заголовний об'єкт Об'єкт 0 Об'єкт 2

Рис. 1. Розподіл пам'яті у одно- і двовимірному розрідженому масиві:
а) структура об'єкта; б) схема динамічної структури об'єктів

На рис. 1 показано розподіл пам'яті у одновимірному розрідженому масиві, об'єкти якого зберігають значення цілих чисел. Кожен об'єкт цієї динамічної структури має три поля: номер стовпця, самі дані і покажчик, який посилається на наступний об'єкт справа. Заголовний об'єкт, крайній зліва, має номер стовпця -1 і не містить даних. Перший значущий об'єкт (з номером 0) містить значення 4. Другий об'єкт (з номером 1) у структурі не показаний взагалі. В цьому випадку вважається, що об'єкт 1 зберігає значення за замовчуванням. Третій об'єкт (з номером 2) зберігає значення 12.

На рис. 2 показано розподіл пам'яті у двовимірному розрідженому масиві цілих чисел. Незважаючи на те, що це пряме узагальнення одновимір-

го розрідженого масиву, кожен об'єкт цієї структури входить у два списки і, окрім номера стовпця, має ще і номер рядка. Хоча двовимірний варіант масиву є дещо складнішим у реалізації порівняно з одновимірним, проте він набагато вигідніший для економного зберігання значень даних. У деяких випадках найбільш важливою частиною розріджених масивів є "дірки" – об'єкти, які явно не зберігаються в пам'яті комп'ютера.

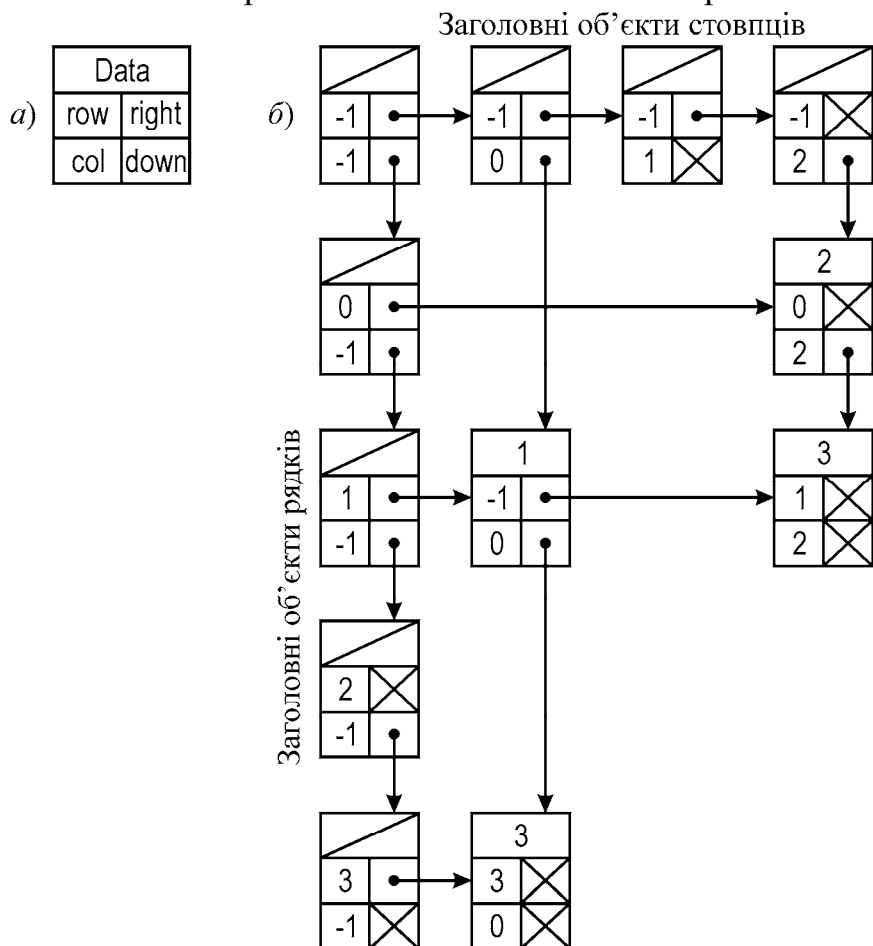


Рис. 2.
Розподіл пам'яті у
двовимірному
розрідженому масиві:
а) структура об'єкта;
б) схема динамічної
структури об'єктів

Проектуючи динамічну структуру даних – одновимірний розріджений масив, зазвичай використовують перевантаження операторної функції `operator[]()`, яка одночасно реалізує запис і зчитування об'єктів. Проте такий підхід має свої недоліки, їх вдало усувають під час реалізації роботи двовимірного розрідженого масиву, у якому ця функція взагалі не використовується.

1.1. Реалізація одновимірного розрідженого масиву

Найчастіше реалізація одновимірного розрідженого масиву здійснюється у вигляді однозв'язного списку. У нього немає фіксованого розміру і він зростає у міру додавання об'єктів. Оскільки для зчитування і запису значень даних використовується одна операторна функція `operator[]()`, то її внутрішній код повинен правильно управляти доступом у обох випадках, не знаючи наперед, яка саме дія виконується в конкретний момент. Мінімальним допустимим індексом для цієї функції є нуль, тобто у ній спочатку відбувається перевірка на заборону застосування негативних індексів.

Як уже було наголошено, головний недолік реалізації операторної функції `operator[]()` полягає в тому, що вона наперед не знає, для чого викликається – зчитування чи запису. Щоби зрозуміти важливість таких дій, спро-

буємо розрізнити окремо доступи для зчитування і запису. Проте навіть для нефахівців з області програмування, очевидно, що під час роботи програми дію доступу для зчитування даних потрібно робити не так вже й часто. Якщо у процесі пошуку необхідний об'єкт знайдено, то операторна функція має повернути посилання на його поле даних. Інакше вона має повернути значення даних за замовчуванням.

Стосовно дії доступу для запису, то операторна функція, знайшовши необхідний об'єкт, має поводитися точно так, як і при зчитуванні, тобто має повернути посилання на його поле даних. Якщо ж потрібний об'єкт не знайдено, то функція повинна його створити, інакше просто нікуди буде записати чергове значення даних. Це питання вирішується шляхом надання кожному виклику операторної функції статусу доступу для запису. Як тільки запрошується неіснуючий об'єкт, то в потрібному стовпці створюється новий об'єкт, який ініціалізується значенням за замовчуванням.

1.2. Реалізація двовимірного розрідженого масиву

Хоча двовимірний розріджений масив не є "концептуальним проривом" у створенні динамічних структур даних порівняно з одновимірним, проте сумніву не викликає і те, що реалізація операторної функції `operator[]()` не особливо вдало узагальнюється на двовимірний випадок. Однією з причин є неможливість розрізнити дію зчитування від дії запису, що шкодить ефективній її роботі. Згадаймо, щоразу, коли у одновимірному розрідженому масиві зчитувався неіснуючий об'єкт, то треба було створювати новий та ініціалізувати його дані значенням за замовчуванням. Якщо це трапляється часто, то зазнає краху вся ідея використання розрідженого масиву. Очевидно, під час реалізації двовимірного розрідженого масиву це питання стає набагато серйознішим.

Друга причина полягає в тому, що операторна функція `operator[]()` може приймати тільки один аргумент. Це примусило б нас реалізувати двовимірну модель розрідженого масиву як масив масивів. Як на перший погляд, такий підхід вартий уваги, але згодом можна потрапити у безвихідь, як тільки спробуємо визначити значення за замовчуванням. Однією з вимог під час реалізації розрідженого масиву є також те, щоб введений користувачем тип об'єкта, як і в одновимірному випадку, мав тип `tType`. Проте, якщо взяти за модель нашу просту одновимірну реалізацію, то масив масивів повинен мати за замовчуванням тип *масив*.

Для ефективного вирішення цих питань, зазвичай у двовимірній моделі зовсім відмовляються від операторної функції `operator[]()` і замінюють її двома функціями доступу – окремо для зчитування і для запису. Це дає змогу акуратному користувачеві уникнути створення непотрібних нових об'єктів та ініціалізації їх даних значенням за замовчуванням. Інші мало зацікавлені користувачі можуть застосовувати функцію доступу винятково тільки для запису значень даних.

З метою спрощення програмної реалізації зарезервований двовимірний розріджений масив може мати фіксовані розміри, які перед виконанням встановлює користувач. Вводячи об'єкт у двовимірний масив, спочатку ство-

рюється рядок заголовних стовпців "над" рядком 0 і стовпець заголовних рядків "зліва" від стовпця 0. За потреби, такий підхід можна модифікувати так, щоб розмір масиву встановлювався на програмному рівні. Інше можливе удосконалення – не створювати заголовні рядки (стовпці) доти, доки в цей рядок (стовпець) не буде дійсно поміщено об'єкт.

Розроблені функції доступу для зчитування і запису мають приймати як аргументи індекси рядка і стовпця. Функція доступу для зчитування повертає `const tType &`. Після перевірки діапазону зміни індексів масиву функція перевіряє список заголовних рядків доти, доки не знайде той об'єкт, який вказано аргументом `row`. Потім вона перевіряє рядок зліва направо у пошуках об'єкта, який відповідає аргументові `col`. Якщо функція зчитування потрапляє на об'єкт, поле `col` якого є більшим від вказаного аргументу або сягає кінця списку, то вона повертає значення даних за замовчуванням.

Функція доступу для запису повертає `tType &` і слідує схожому алгоритму пошуку, як і у функції зчитування. Проте, коли функція запису виявляє, що шуканого об'єкта не існує, то вона його створює, ініціалізує дані значенням за замовчуванням і вставляє його у потрібне місце масиву. Оскільки це місце визначається шляхом перегляду відповідного рядка, то функція має достатньо інформації, щоб за потреби вставити у цей рядок новий об'єкт. Але ця функція повинна ще вставити цей об'єкт і в другий однозв'язний список – в його стовець. Вона робить це так: спочатку переглядаюся заголовні стовпці, відшукуючи серед них потрібний. Знайшовши його, далі виконуються дії, аналогічні перегляду рядка. За дії перегляду стовпця відповідає внутрішній цикл `while`, який є відсутнім у функції доступу для зчитування.

За бажанням користувача, можна мінімізувати кількість заголовних об'єктів і надати масиву здатність програмно міняти свої розміри. Ці зміни здійснюються шляхом модифікування параметризованого конструктора і двох функцій доступу. Модифікований конструктор матиме тільки один параметр (`const tType &`) і, замість того, щоб створювати всі заголовні рядки і стовпці, створить тільки головний заголовний об'єкт у рядку -1 і стовпці -1. Модифіковані функції доступу переглядатимуть заголовні об'єкти не шляхом підрахунку, а шукатимуть рядок або стовець з номером, більшим від нього або який дорівнює потрібному. За потреби буде створено новий заголовний об'єкт. Такий підхід імітує наявну перевірку рядка і створення об'єкта з даними у міру потреби.

2. Приклади створення розріджених масивів

Розглянемо процес створення двовимірних розріджених масивів на конкретному навчальному прикладі, який, для розуміння цього механізму, має невелику кількість рядків і стовпців. Хоча такі приклади не завжди відповідатимуть реальним інженерним розрахункам, проте у нашому випадку розроблена програма здатна працювати з надзвичайно великими масивами.

Отож, основне завдання топологічного опису будь-якої електронної схеми (рис. 3, а) полягає в тому, щоби перевести графічну інформацію про з'єднання її компонент у формальні математичні позначення, які потім використовуються для написання алгебричних рівнянь та нерівностей [1, с. 114].

Основою топологічного опису є граф, який складається з *гілок* – сукупності відрізків довільної довжини і форми, а також *вершин* – точок перетину вершин. Важливим поняттям теорії графів є дерево графу (рис. 3, б), яке складається з сукупності гілок електронної схеми, охоплює усі її вузли, але не містить жодного контура. Графічне подання зв'язків у електронній схемі переводиться у формальні математичні позначення за допомогою топологічних матриць (рис. 3, в).

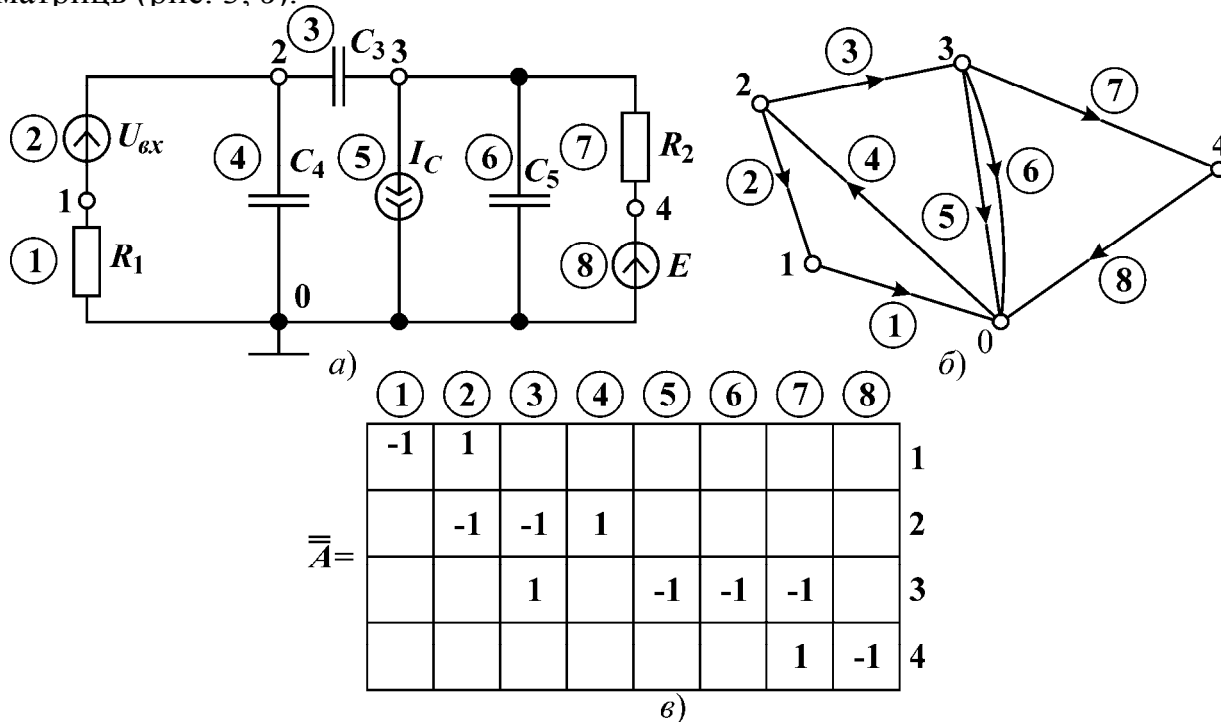


Рис. 3. Топологічний опис електронного ключа: а – еквівалентна електронна схема; б – дерево графу; в – топологічна матриця

До топологічних належать структурні матриці (з'єднань – інциденцій), які вперше запропонував Кірхгоф. Для їх побудови попередньо нумерують вузли і гілки графу, а потім довільно вибирають додатні напрямки його гілок. У матриці інциденцій (рис. 3, в) рядки відповідають різним вузлам графу, а стовпці – гілкам графу. Кожен i -ий рядок матриці показує, які гілки з'єднані з i -им вузлом, а кожен j -ий стовпець показує, з якими вузлами з'єднана j -а гілка графу. Кожен a_{ij} елемент цієї матриці набуває значення +1, якщо j -та гілка підходить до i -го вузла, або -1, якщо j -та гілка виходить з i -го вузла; а також 0, якщо j -та гілка не має спільної точки з i -им вузлом. Матриця інциденцій дає змогу записати інформацію про зв'язки в електронній схемі, яка міститься в графі, у векторно-матричному вигляді.

Наприклад, використовуючи 1-ий закон Кірхгофа у матричному вигляді

$$\bar{A} \times \vec{i} = \vec{0} \Rightarrow \{ a_{ij} \cdot i_j = 0, j = \overline{1, n}; i = \overline{1, m} \}, \quad (1)$$

отримаємо систему лінійних рівнянь для знаходження струмів (i_j) у гілках електронної схеми. Використовуючи такий матричний запис

$$\bar{A}^T \times \vec{\varphi} = \vec{u} \Rightarrow \{ a_{ji}^T \cdot \varphi_i = u_j, i = \overline{1, m}; j = \overline{1, n} \}, \quad (2)$$

отримаємо систему лінійних рівнянь для знаходження напруг (u_j) між вузлами цієї схеми, де φ_i – електричний потенціал у вузлі.

У фаховій літературі [3, 4] з моделювання електронних схем детально розглянуто питання побудови та алгоритми розв'язання алгебро-топологічних рівнянь з використанням комп'ютерної техніки. У них також зазначено, що, використовуючи у топологічних матрицях певні закономірності розміщення нульових елементів, під час розроблення програми можна істотно зменшити об'єм використаної пам'яті комп'ютера і водночас скоротити тривалість процесу розрахунку. Скорочення тривалості розрахунку електронних схем досягається як шляхом зменшення кількості обчислень (не виконуючи арифметичних операцій з нульовими елементами), так і за рахунок заміни довгих операцій короткими. Основна властивість топологічної матриці полягає в тому, що у ній кількість нульових елементів є достатньо великою, водночас як усі ненульові елементи дорівнюють або $+1$, або -1 . Окрім того, топологічна матриця складної електронної схеми має, як правило, блочну структуру. Це дає змогу замінити довгі операції множення і додавання операціями порівняння, додавання і віднімання, тобто більш короткими операціями, і значно зменшити кількість виконуваних дій.

З огляду на сказане вище, останнє завдання цієї роботи полягає в тому, щоби за допомогою розробленого програмного проекту зберегти топологічну матрицю (рис. 3, в) і забезпечити ефективний доступ до її елементів, тобто реалізувати роботу двовимірного розрідженого масиву. Оскільки у виразах (1) і (2) матрицю інцидентів елементів представлено у звичайному $\overline{A}$ та транспонованому A^T виглядах, то перед введенням об'єктів у двовимірний розріджений масив необхідно спочатку для значущих елементів матриці записати значення їхніх індексів.

Отож, фрагмент результатів роботи розробленого програмного проекту, який показує збереження значущих об'єктів у двовимірному розрідженому масиві, має такий вигляд:

Звичайний вигляд матриці Транспонований вигляд матриці

$[(1,1) -1]$	$[(1,1) -1] [(1,2) 1]$
$[(2,1) 1] [(2,2) -1]$	$[(2,2) -1] [(2,3) -1] [(2,4) 1]$
$[(3,2) -1] [(3,3) 1]$	$[(3,3) 1] [(3,5) -1] [(3,6) -1] [(3,7) -1]$
$[(4,2) 1]$	$[(4,7) 1] [(4,8) -1]$
$[(5,3) -1]$	
$[(6,3) -1]$	
$[(7,3) -1] [(7,4) 1]$	
$[(8,4) -1]$	

Як на перший погляд, збереження у пам'яті комп'ютера об'єктів, які відображають елементи транспонованої матриці інцидентів, є більш компактним, ніж у звичайній матриці. Проте – це тільки екранне представлення об'єктів у відповідних двовимірних розріджених масивах, водночас як у пам'яті комп'ютера з 32 булевих елементів кожної з матриць зберігається тільки 11 значущих.

Висновки

1. Розглянуто питання застосування динамічної структури даних – *розріджених масивів* (sparse arrays) для роботи з топологічними матрицями, значна більшість елементів яких не використовуються.

Табл. 1. Значення індексів та елементів матриці інцидентій у звичайному та транспонованому вигляді при $m = 4$ і $n = 8$ (всього 32 булевих елементи)

№ п/п	Звичайний вигляд матриці			Транспонований вигляд матриці		
	i	j	a_{ij}	j	i	a_{ji}
1	①	1	-1	1	①	-1
2	②	1	1	1	②	1
3	②	2	-1	2	②	-1
4	③	2	-1	2	③	-1
5	③	3	1	2	④	1
6	④	2	1	3	③	1
7	⑤	3	-1	3	⑤	-1
8	⑥	3	-1	3	⑥	-1
9	⑦	3	-1	3	⑦	-1
10	⑦	4	1	4	⑦	1
11	⑧	4	-1	4	⑧	-1

2. З'ясовано, що розріджені масиви мають здатність як компактно зберігати відповідні об'єкти, так і здійснювати ефективний доступ до них, що робить їх надзвичайно корисними в багатьох ситуаціях, які виникають під час розроблення сучасних програмних продуктів.
3. Наведено приклади зберігання об'єктів, які представляють елементи матриці інцидентій електронного ключа у звичайному та транспонованому виглядах, що свідчить про придатність роботи розробленого програмного проекту.
4. Встановлено, завдяки використанню розріджених масивів пам'ять для зберігання кожного з об'єктів (елементів булевої матриці) виділяється з пули вільної пам'яті комп'ютера тільки в міру потреби.

Література

1. Молчанов А.А. Моделирование и проектирование сложных систем. – К.: Вища шк., 1988. – 360 с.
2. Павловская Т.А., Щупак Ю.А. С++. Объектно-ориентированное программирование: Практикум. – СПб.: Питер, 2005. – 265 с.
3. Сигорский В.П., Петренко А.И. Алгоритмы анализа электронных схем. – М.: Сов. радио, 1976. – 608 с.
4. Ильин В.Н. Основы автоматизации схемотехнического проектирования. – М.: Энергия, 1979. – 392 с.
5. Фридман А., Кландер Л., Михаэлис М., Шильт Х. С/С++. Алгоритмы и приемы программирования. – М.: ООО "Бином-Пресс", 2007. – 560 с.