ПРОГРАМУВАННЯ КОМБІНАТОРНИХ ЗАДАЧ НА ПРИКЛАДІ ЛАТИНСЬКИХ КВАДРАТІВ

Наталя БРОНИЦЬКА, Валентина ДАРМОСЮК, Ростислав ХОМІЧУК, Вікторія БЕРЕЖЕЦЬКА

В статті досліджено можливість поглиблення знань учнів з комбінаторики та отримання ними практичних навичок в програмуванні. Розглянуто приклади задач про латинські квадрати та їх програмування мовою С++ із застосуванням комбінаторних задач та бібліотеки STL. Обгрунтовано необхідність введення в шкільний курс розгляд сучасних мов програмування.

В статье исследована возможность углубления знаний учащихся по комбинаторике и получения ими практических навыков в программировании. Рассмотрены примеры задач о латинских квадратах и их программирование на языке С++ с применением комбинаторных задач и библиотеки STL. Обоснована необходимость введения в школьный курс рассмотрения современных языков программирования.

Вступ

Зміст курсу інформатики включає сукупність двох взаємопов'язаних компонентів теоретичного і практичного. Одне із центральних місць в практичному аспекті займає написанням програм однією з мов програмування. Створення програми зводиться до розробки певного алгоритму. Починати вивчати мову програмування доцільно на відомих алгоритмах, таких як, наприклад, комбінаторні. Сучасна програма з математики передбачає вивчення комбінаторики в 11 класі в невеликому об’ємі, а програма з інформатики взагалі не передбачає вивчення даної теми у розділі програмування. Незважаючи на це комбінаторні задачі посідають чільне місце серед олімпіадних задач та задач підвищеної складності, як математики так і інформатики. Комбінаторні обчислення зазвичай передбачають громіздкі розрахунки. Програмна реалізація таких обчислень дає змогу їх спростити і наряду з тим сприяє кращому розумінню учнями мови програмування і принципів алгоритмізації.

Метою статті є розгляд різних задач про латинські квадрати та їх програмування мовою С++ із застосуванням комбінаторних задач та бібліотеки STL.

1. Історична довідка.

Метою комбінаторного аналізу є вивчення комбінаторних конфігурацій, зокрема, питання їх існування, алгоритми побудови, розв’язання задач на перерахування. Латинські квадрати – це один з прикладів комбінаторних конфігурацій.

Перші спогади про латинські квадрати відносяться до 1723 р. Поняття латинського квадрата довгий час відносилося до тих понять, які не вивчалися на належному рівні, а носили розважальний характер. На протязі двох століть латинськими квадратами займалися з точки зору математичних розваг, для тренування розуму і в міру естетичної

привабливості відповідних задач. І тільки наш час надав розважальній математиці минулого нового важливого значення.

Систематичне вивчення латинських квадратів розпочалося з робіт видатного математика XVIII століття Леонарда Ейлера (1707 – 1783 рр). Хоча латинські квадрати почали використовувати задовго до нього при розв'язані багатьох задач, таких як, наприклад, задачі про переміщення та розміщення фігур на шахматній дошці, розміщення колоди з шістнадцяти гральних карт і т.п., але Л. Ейлер перший дав строге визначення латинському квадрату. Дві його роботи [1], [2] присвячені магічним та латинським квадратам, заклали основи нового напряму комбінаторики – теорії латинських квадратів.

Згідно Ейлера, латинський квадрат – це квадратна таблиця з n^2-клітками, яка складена з n різних елементів, жоден із яких не з'являється двічі в жодному рядку та в жодному стовпчику. Число n називається порядком латинського квадрата.

В 1782 році Ейлер запропонував наступну проблему: «Серед 36 офіцерів порівну уланів, драгунів, гусарів, кірасирів, кавалергардів і гренадерів і, крім того, порівну генералів, полковників, майорів, капітанів, поручиків і підпоручиків, причому кожен рід військ представлений офіцерами всіх шести рангів. Чи можна вишикувати всіх офіцерів в каре 6х6 так, щоб в будь-якій колоні і будь-якій шерензі зустрічались офіцери всіх рангів?» [2, с. 291] Ейлер не зміг знайти розв’язання цієї задачі. Дане питання цікавило багатьох вчених, і лише в 1901 р. було доведено, що такого розв’язку не існує.

Питаннями вивчення та побудови латинських квадратів і в наш час цікавляться геометри, алгебраїсти, вчені в області комбінаторного аналізу, теорії кодування, спеціалісти з планування експеременту і т.п.

2. Приклади задач

Задача 1. Побудувати латинський квадрат n-ого порядку.

Існує досить багато схем побудови латинських квадратів. Деякі за них описані, наприклад, в книзі [4, с. 100]. Один з них дозволяє побудувати латинський квадрат будь-якого порядку n=p-1, де p – просте число, p>=3. Наведемо алгоритми побудови латинських квадратів декількома методами.

а) Побудуємо латинський квадрат з використанням циклічного зсуву.

Задача циклічного зсуву зустрічається досить часто як сама по собі, так і в якості підзадачі при розв’язувані інших задач. І таку задачу інколи набагато зручніше розв’язати без використання масивів.

Для побудови латинського квадрату в якості першого рядка візьмемо послідовність чисел 1, 2, …, n. А кожен наступний рядок отримаємо циклічним зсувом.

```
int main() {
int n;
cin>>n;
int i,j,k;
for(k=0;k<n;++k,cout<<endl) {
for(i=k;i<n;++i)cout<<i+1<<' ';
for(i=0;i<k;++i) cout<<i+1<<' ';
}
}
```

б) Ще один спосіб – побудова латинського квадрату за формулою x+y(mod n). Задана формула реалізує ще один варіант циклічного зсуву.

```
int main() {
int n;
cin>>n;
for(int i=0;i<n;i++,cout<<endl)
for(int j=0;j<n;j++)
cout<<(i+j)%n+1<<" ";
}
```

в) Наступний спосіб описано в книзі [4, с. 101].

Для програмної реалізації такого способу спочатку описано функцію пошуку найбільшого спільного дільника за алгоритмом Евкліда, а в головній функції знаходиться число k (k>=2) взаємно просте з n. Фактично цей спосіб реалізує циклічний зсув з постійним кроком k.

```
int gcd(int a,int b) {
while(true) {
a%=b;
if(!a)return b;
b%=a;
if(!b)return a;
 }
}
int main() {
int n,k;
cin>>n;
for(int i=2;i<n;i++) if(gcd(i,n)==1){k=i;break;}
for(int i=1;i<=n;i++,cout<<endl)
for(int j=1;j<=n;j++) if((i*k+j)%n==0)cout<<n<<" ";
else cout<<(i*k+j)%n<<" ";
}
```

Всі запропоновані вище способи будують латинський квадрат будь-якого порядку без додаткових умов. Програмна реалізація кожного з цих способів має багато варіантів. Способи, наведені в статті,

демонструють багатогранність мови програмування С++. Саме тому в наступній задачі задана додаткова умова n<=9. Звичайно, за заданим першим рядком легко побудувати латинський квадрат будь-якого порядку методом циклічного зсуву, але в наступній програмі ми показали використання класу string стандартної бібліотеки шаблонів STL для циклічного зсуву рядка.

Задача 2. Заповнити по заданому першому рядку латинський квадрат n-ого порядку (n<=9). Рядок вводиться без пробілу.

```
#include <string>
int main() {
string s;
cin>>s;
int n=s.length();
for(int i=1;i<n;i++) {
s.push_back(s[0]);
s.erase(s.begin());
cout<<s<<endl;
}
}
```

Задача 3. Побудувати всі можливі варіанти заповнення квадрату 3×3.

```
int main() {
int a[3][3],b[3],i,j;
for(i=0;i<3;++i,cout<<endl) {
b[i]=i+1;
for(j=0;j<3;++j,cout<<' ') a[i][j]=(i+j)%3+1;
}
cout<<endl;
do{
for(i=0;i<3;++i) cout<<b[i]<<' '; cout<<endl;
for(i=0;i<3;++i) cout<<a[b[i]-1][1]<<' '; cout<<endl;
for(i=0;i<3;++i) cout<<a[b[i]-1][2]<<' '; cout<<endl<<endl;
for(i=0;i<3;++i) cout<<b[i]<<' '; cout<<endl;
for(i=0;i<3;++i) cout<<a[2][b[i]-1]<<' '; cout<<endl;
for(i=0;i<3;++i) cout<<a[1][b[i]-1]<<' '; cout<<endl<<endl;
} while(next_permutation(b,b+3));
}
```

Задача 4. (задача з олімпіади Китеня 2011, м. Ковров) Задано прямокутну матрицю з чисел W×H (W≠H, 2≤W, H≤8). По рядкам і стовбцям вона заповнена числами від 1 до max(W,H). Потрібно довести цю матрицю до латинського квадрату зі стороною max(W,H) або

повідомити, що це неможливо. Забороняється змінювати вміст заданого прямокутника.

Технічні умови.

Вхідні дані. Перший рядок у вхідних даних містить числа W і H. У наступних H рядках записано по W чисел через пропуск.

Вихідні дані. Якщо довести до латинського квадрату цей прямокутник неможливо, то виведіть "Impossible" (без лапок), інакше у першому рядку виведіть довжину сторони латинського квадрату, а у наступних рядках виведіть отриманий латинський квадрат. Числа у стовбцях виводьте через один пропуск. Якщо розв'язків декілька – виведіть довільний. [8]

Наведемо розв'язок цієї задачі мовою програмування C++ з використанням стандартної бібліотеки шаблонів та алгоритмів STL.

Попередня підготовка.

Для коректної компіляції програми нам потрібно підключити наступні бібліотеки:

```
#include <iostream> //бібліотека введення-виведення
#include <algorithm> //стандартна бібліотека алгоритмів
#include <vector> //контейнерний клас вектор стандартної бібліотеки STL
```

Для зручності краще використовувати синоніми для введених складних типів:

```
typedef vector <int> vi;
typedef vector <vector <int> > vvi;
```

Типи та змінні.

Шуканий латинський квадрат зберігається у двовимірному векторі vvi a. Кожен елемент цього вектора містить відповідне число заданої матриці, або 0, якщо це число необхідно знайти.

Глобальні змінні: n – розмір квадрату (більша сторона матриці), m – менша сторона заданої матриці, s – кількість стовпчиків матриці і r – кількість її рядків.

Опис необхідних функцій.

Функції введення та виведення заданої матриці.

Введення відбувається зі стандартного потоку. У функції створюється локальний двовимірний квадратний вектор, який заповнюється спочатку нулями, а потім на відповідні місця цього вектора записуються елементи матриці. Якщо рядків більше, ніж стовпчиків, то для універсальності запропонованого алгоритму у функції задана матриця транспонується.

```
vvi input() {
cin>>s>>r;
n=max(s,r);
```

```
m=min(s,r);
vi temp(n,0);
vvi a(n,temp);
for(int i=0;i<r;i++)
for(int j=0;j<s;j++)
if(s>r) cin>>a[i][j];
else cin>>a[j][i];
return a;
}
```

Виведення побудованого латинського квадрату відбувається у стандартний вихідний потік. Якщо матрицю було транспоновано у функції Input, то виводимо її початковий варіант.

```
void print(vvi a) {
cout<<n<<endl;
for(int i=0;i<n;i++) {
if(s>r) cout<<a[i][0];
else cout<<a[0][i];
for(int j=1;j<n;j++)
if(s>r) cout<<" "<<a[i][j];
else cout<<" "<<a[j][i];
cout<<endl;
}
}
```

Функція перевірки матриці.

Функція isValid перевіряє, чи не суперечить прямокутна матриця ненульових елементів двовимірного вектора означенню латинського квадрата.

Функції inRow (inCol) перевіряють, чи повторюється значення аргументу v у тому ж рядку (стовпчику), в якому він розташований.

Функція inAny перевіряє, чи повторюється значення аргументу v у тому ж рядку або стовпчику, в якому він розташований, використовуючи результати функцій inRow та inCol.

```
bool inRow(vvi a,ii p,int v) {
for(int i=0;i<n;++i)
if(p.first!=i&&a[i][p.second]==v)return true;
return false;
}
bool inCol(vvi a,ii p,int v) {
for(int i=0;i<n;++i)
if(p.second!=i&&a[p.first][i]==v)return true;
return false;
}
```

```
bool inAny(vvi a,ii p,int v) {
return inRow(a,p,v)||inCol(a,p,v);
}
bool isValid(vvi a) {
for(int i=0;i<n;i++)
for(int j=0;j<n;j++)
if(a[i][j]&&inAny(a,ii(i,j),a[i][j])) return false;
return true;
}
```

Функція добудови матриці до латинського квадрату.

Доки не будуть знайдені всі рядки, кожний наступний рядок шукаємо таким чином:

Запишемо в додатковий одновимірний масив останній знайдений рядок.

Знаходимо наступну перестановку (в лексикографічному порядку) додаткового масиву і дописуємо цей масив на місце шуканого рядка.

Перевіряємо, чи відповідає отримана матриця означенню латинського квадрату. Якщо так, переходимо до пункту 1. Інакше переходимо до пункту b.

```
void build(vvi& a) {
for(int i=m;i<n;i++) {
vi temp=a[i-1];
next_permutation(temp.begin(),temp.end());
a[i]=temp;
while(!isValid(a)) next_permutation(temp.begin(),temp.end());
}
}
```

Головна функція.

```
int main() {
vvi a=input();
if(isValid(a)) {
build(a);
print(a);
}
else cout<<"Impossible"<<endl;
}
```

Стандартна бібліотека шаблонів (STL – Standard Template Library) – це бібліотека контейнерних класів, яка включає вектори, списки, черги, стеки а також ряд алгоритмів загального призначення. Мета включення бібліотеки STL в стандарт мови полягає в тому, щоб спростити написання загальноприйнятих алгоритмів. Бібліотека стандартних шаблонів містить більше сотні різних шаблонів та алгоритмів. Ядро

бібліотеки складають три групи шаблонних класів: контейнери, алгоритми та ітератори. [3, с. 482] У вересні 2011 року Міжнародною Організацією зі Стандартизації (ISO) було прийнято і опубліковано новий стандарт мови C++, в якому бібліотеку STL було оновлено та розширено [6].

У наведеній програмі були використані наступні алгоритми [7]:

min(int,int) (max(int,int)) – знаходження мінімального (максимального) значення серед двох заданих чисел;

next_permutation(iterator first,iterator last) – знаходження наступної (в лексикографічному порядку) перестановки елементів від first до last, де first і last – ітератори першого та останнього елемента.

Висновки

Теорія латинських квадратів − один з напрямків комбінаторики, результати якого застосовуються в різних областях математики (алгебрі та теорії чисел, теорії графів і т.п.) та кібернетики (теорії кодування, планування експерименту, захисті інформації, тощо). В наш час теми пов’язані з вивченням латинських квадратів є актуальними, оскільки забезпечують зв’язок між різними галузями знань та допомагають формувати в учнів уміння використовувати знання про властивості чисел в різних нестандартних ситуаціях. На прикладі латинських квадратів можна розглядати і деякі аспекти програмування комбінаторних задач.

В статті наводяться алгоритми та коди програм на мові програмування на C++ для наступних задач:

Побудувати латинський квадрат n-ого порядку.

Заповнити по заданому першому рядку латинський квадрат n-ого порядку (n<=9).

Побудувати всі можливі варіанти заповнення квадрату 3×3.

Доповнити задану прямокутну матрицю (n×m, 2<=n,m<=8) до латинського квадрата, або зробити висновок, що це неможливо.

Сьогодні бути професійним програмістом високого класу означає бути компетентним в мові програмування C++, проте в шкільній програмі не передбачено її розгляд. Високий динамізм становлення методичної системи навчання інформатики зумовлює доцільність впровадження в шкільний курс більш сучасних мов програмування. Мова C++ призначена для розробки високопродуктивного програмного забезпечення, її синтаксис став стандартом для інших професійних мов програмування, а її принципи розробки відображають ідеї розвитку обчислювальної техніки в цілому.

Наведені в статті алгоритми дають змогу поглибити знання з комбінаторики та отримати практичні навички в програмуванні. В них використовується широкий спектр засобів мови програмування C++ та демонструються деякі можливості стандартної бібліотеки шаблонів STL.

БІБЛІОГРАФІЯ

1. Euler L. «De quadratis magicis» // Представлено Санкт-Петербургской Академии Наук 17 октября 1776 г., впервые опубликовано в Mem. Soc. Flessingue, Comm. Arith. Collect. (elogeSt.Petersburg 1783) 2 (1849), P.593-602.
2. Euler L. Recherches sur une nouvelle espece de carrйs magiques // Opera Omnia, 1923, Vol. 1, P.291–392.
3. Глушаков С.В., Смирнов С.В., Коваль А.В. Практикум по С++ // Харьков, Фолио, 2006, 525 с.
4. Олехник С.Н., Нестеренко Ю.В., Потапов М.К. Старинные занимательные задачи. // Москва «Наука» Гл. редакция физико-математической литературы, 1988, 155 с.
5. Холл М. Комбинаторика // Издательство «Мир», Москва, 1970, 424 с.
6. http://www.cplusplus.com/info/history/
7. http://www.cplusplus.com/reference/stl
8. http://www.e-olimp.com.ua/problems/2105

ВІДОМОСТІ ПРО АВТОРА

Наталя Анатоліївна Броницька – доцент кафедри інформатики, кандидат фізико-математичних наук Миколаївського національного університету ім. В.О. Сухомлинського.

Валентина Миколаївна Дармосюк – доцент кафедри математики і механіки, кандидат фізико-математичних наук Миколаївського національного університету ім. В.О. Сухомлинського.

Ростислав Хомічук – студент 3 курсу Миколаївського національного університету ім. В.О. Сухомлинського.

Вікторія Бережецька – студентка 2 курсу Миколаївського національного університету ім. В.О. Сухомлинського.

ПСИХОЛОГО-ПЕДАГОГІЧНІ ОСНОВИ ІНТЕРАКТИВНО ВЗАЄМОДІ ВЧИТЕЛЯ І УЧНІВ НА УРОКАХ МАТЕМАТИКИ

Леся ВАСИЛЕНКО, Ігор ВАСИЛЕНКО

У статті обгрунтовано доцільність утвердження такого поняття, як інтерактивна педагогічна взаємодія; розглядається структура інтерактивної взаємодії вчителя з учнями на уроках математики.

В статье обосновано целесообразность утверждения такого понятия как интерактивное педагогическое взаимодействие ; рассматривается структура интерактивного взаимодействия учителя и учеников на уроках математики

Ефективність сучасної освіти напряму залежить від утілення в ній демократичних засад, цінностей партнерства та співпраці між педагогом і учнями. В основу більшості інноваційних педагогічних технологій покладено підвищення якості взаємодії вчителя з учнями у напрямку її більшої паритетності й духовності, дієвості й активності. Інтерактивна