**Сортування елементів масиву**

***(методи сортування, сортування перестановкою, вибором, швидке сортування, задача кількість різних чисел в масиві)***

Розглянемо способи сортування. Сама тема сортування є однією з найбільш досліджених задач.

Є три способи сортування масивів:

* сортування вибором;
* сортування обміном;
* сортування вставкою.

Для кожного способу є багато алгоритмів, які відрізняються часом сортування, який злежить від числа операцій порівняння і операцій обміну.

Традиційно розрізняють внутрішнє сортування, яке обробляє дані оперативної пам’яті, і зовнішнє сортування, яке оперує з даними розміщеними на дисках.

Розглянемо сортування числового одномірного масиву.

Відсортувати числовий масив: 7, 3, 8, 4,8, 5, 9, 1.

Звичайне сортування : 1, 3, 4, 5, 7, 8, 8, 9,.

Адресне сортування: 7, 3, 8, 4, 8, 5, 9, 1.

5, 2, 6, 3, 7, 4, 8, 1 (адреса).

Є багато різноманітних алгоритмів сортування (сортування бульбашкою, сортування за допомогою дерева, пірамідальне сортування, швидке сортування (половинного поділу).

Розглянемо деякі з них в дещо видозміненому вигляді.

Метод бульбашки:

*Опис:*

Найпростіший і найпопулярніший із них - це “сортування бульбашкою”.

Назва його походить від образної інтерпретації, при котрій у процесі виконання алгоритму більш “легкі” елементи мало-помалу випливають на “поверхню”.

Нехай а – числовий масив

а[1], а[2], ... ,а[n]

Говорять, що елементи а[і] і а[j] із а утворюють інверсію, якщо і<j

і а[і]<а[j].

Алгоритм “сортування бульбашкою” складається в послідовних проглядах знизу вверх (від початку до кінця) масиву s і обміну місцями сусідніх елементів.

З даної програми по введеному числу n створюється масив, заповненням його з клавіатури.

Цикл від 1 до до n здійснює перестановку місцями елементів масиву. Перестановка здійснюється, поки масив не стане відсортованим, за що відповідає змінна s.

Відсортований масив виводиться на екран. “Сортування бульбашкою” не потребує для реалізації додаткової пам’яті. Однак через погані характеристики він має лише історичну цікавість і навряд чи може бути рекомендована для практичного використання.

##### Сортування вибором

*(складніший спосіб сортування з точки реалізації)*

арг А[1..n]

рез А[1..n]

Приклад n=5

2 7 4 3 5

* Знайти максимальний елемент max та його номер k.
* Стерти елемент з номером k.
* Поставити елемент з значенням max в кінець.

2 4 3 5 7

2 4 3 5 7

2 3 4 5 7

2 3 4 5 7

поч

ввести масив А[1..n]

kol:=n

поки kol>1 пц

знайти max, k

стерти елемент з номером k

вставити max в позицію kol

kol:=kol-1

кц

вивести масив А[1..n]

кін

##### Швидке рекурсивне сортування

В основі швидкого сортування лежить метод розбиття.

Дано цілочисельна таблиця А і деяке число Х. За один прохід переставляються таблиці так, щоб спочатку були елементи менші Х, потім рівні Х та більші Х.

k1- кількість елементів менша Х;

k2- кількість елементів більша Х;

k-номер елемента з яким працюємо.

<x =x >x

1 k1 k k2 n

Якщо k-й елемент >х тоді

змінити місцями елементи з номером k, n-k2

k2 збільшуємо

k, k1 не міняємо

Якщо k-й елемент <х тоді

міняємо місцями k-й елемент і елемент з номером 1+k1

k, k1 збільшуємо

k2 не міняємо

Якщо k-й елемент = х тоді

нічого не переставляємо

k збільшуємо

k1, k2 не міняємо

Приклад

5 7 4 3 8 k=1, k1=0, k2=1

x

8 7 4 3 5 k=1, k1=0, k2=2

3 7 4 8 5 k=2, k1=1, k2=2

3 4 7 8 5 k=2, k1=1, k2=3

3 4 7 8 5 k=3, k1=1, k2=3

І ІІ

Межі першого масиву: L=1 (L), R=1 (L+k1-1).

Межі першого масиву: L=3 (R-k2+1), R=1 (R)

**4. Перебір**

***(перестановки, лексичний перебір, перебір з поверненням)***

###### Переборні задачі

Класичним прикладом переборної задачі служить задача комівояжера.

Дано множини з N міст , відстань між якими відома. В якому порядку повинний проходити їх комівояжер , ходячи в кожне місто один раз , щоб пройдений шлях був найкоротший ? Скільки існує перестановок із N міст?

Почнемо з простіших задач.

1. Утворити всі можливі перестановки трьох цифр 1,2,3

для і від 1 до 3 пц

для j від 1 до 3 пц

для k від 1 до 3 пц

якщо ( І <>j ) і (j<>k) і (k<>І ) тоді

вивести І,k,j

все

кц

кц

кц

2) Отримати всі перестановки чисел від 1,2,3….,n

поч

ввести масив A[1..n]

t:=0

виконати рекурсивну процедуру генерації

кін

Підпрограма генерації

поч

якщо t<n тоді

для j від t+1 до n пц

переставляємо елементи a[t+1], a[j]

збільшуємо t

виконуємо генерацію

зменшуємо t

переставляємо елементи a[t+1], a[j]

кц

якщо t=n тоді виводжу масив все

кін

3)Генеруємо перестановки, використовуючи множинний тип величин.

**Лексичний перебір**

1.Повернемось до перебору:

а). Ми мали: 1,2,3

1,3,2

2,1,3

2,3,1

3,2,1

3,1,2

б). А якщо ми маємо

3,8,7

Як утворити всі можливі перестановки?

1. Утворити перестановки 1,2,3 і використати їх як індексний масив.

в) Маємо 1,1,2

1,2,1

2,1,1

2. Побудуємо лексичний перебір для довільних елементів масиву

X=3 2 4 2 4 3 1

а) Рухаємось справа наліво. Крок вперед можна зробити, якщо наступне число більше за попереднє. Ми зупинилися перед числом 2. Це число потрібно помітити.

X=3 2 4 2 4 3 1

б) Рухаємось справа наліво. Крок вперед можна зробити, якщо число менше за знайдене число(2). Ми зупинилися перед числом 3. Це число потрібно помітити.

X= 3 2 4 2 4 3 1

в) Переставляємо знайдені числа.

X= 3 2 4 3 4 2 1

г) Запишемо числа, розміщені після першого знайденого в зворотному порядку.

X=3 2 4 3 1 2 4

*функція наступна : логічна*

*поч.*

*і:=n*

*пошук:=хибно*

1

2

3

*і:=k+1;*

*j:=n;*

*поки І>j*

*пц*

*t:=x[j];*

*4) x[j]:=x[і];*

*x[і]:=t; іnc(і);*

*і:=і+1;*

*кц.*

*все*

*наступна:=пошук;*

*кін.*

*поч*

*x[1..n];*

*ввести х ;*

*поки наступна пц*

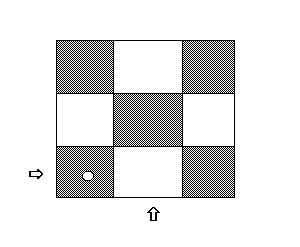
*вивести х*

*кц*

*кін.*

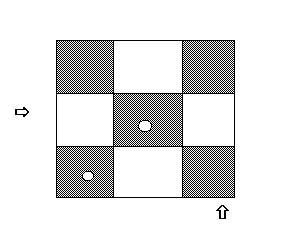
### Перебір з поверненням

Розглянемо метод розв’язку цілого ряду переборних задач на прикладі відомої задачі про тури, які треба розставити на шахівниці так, щоб вони не били один одного. Для наочності візьмемо дошку 3х3 клітинки і розставляти будемо відповідно 3 тури. З відомих формул комбінаторики випливає, що ми будемо мати 3!=6 варіантів розміщень. Очевидно. що при будь-якім розміщенні на кожній горизонталі і на кожній вертикалі повинно бути по одній турі. При відомій вправності і фантазії 3 тури можна ще розставити “вручну” . Ну, а вісім чи більше ( на відповідній дошці, звичайно)? Важкувато...Спробуємо перекласти цю задачу на плечі машини.



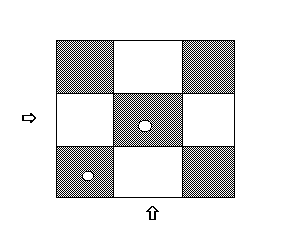
Нехай ми маємо два покажчики - стрілки 

Одна з них указує на горизонталь дошки, інша - на вертикаль. Ставимо першу туру і покажчики, як показано на малюнку, і переміщаємо горизонтальний покажчик на одну позицію вправо. Пробуємо ставити в клітинку, на яку вказують покажчики. Але це зробити не можна. Піднімаємо вертикальний покажчик на одну позицію нагору. Клітка, на яку вказують покажчики, не бита, можна ставити туру. Переміщаємо горизонтальний покажчик на одну клітинку вправо. Знову пробуємо поставили туру туди, куди вказує покажчик, але клітка бита.

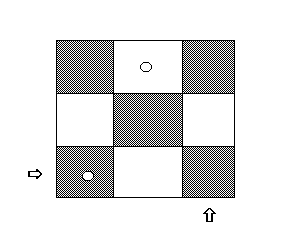


Піднімаємо вертикальний покажчик на одну позицію вгору. Клітка вільна, ставимо туру, горизонтальний покажчик вийде за межі дошки.

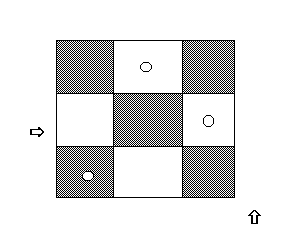
Це ознака того, що дане розміщення довершене. Виводимо результат і повертаємо горизонтальний покажчик на одну позицію вліво, вертикальний покажчик установлюємо на туру в даній вертикалі і намагаємося підняти туру, на яку він указує . Вільних кліток немає. Знімаємо туру, горизонтальний покажчик уліво на одну позицію, а вертикальний поміщаємо на ту горизонталь, де є тура в даній вертикалі.



Намагаємося знову підняти туру, на яку вказує покажчик. Це можливо. Піднімаємо її і горизонтальний покажчик на 1 позицію вправо, а вертикальний - на 1.

`

Пробуємо помістити туру по покажчиках, якщо немає - піднімаємо вертикальний наверх, поки не знайдемо не биту клітку. Ставимо туру, і знову горизонтальний покажчик піде на одну позицію вправо. Готове чергове розміщення.



Знову після виведення результату повернемо горизонтальний покажчик на одну позицію вправо , а вертикальний установимо на туру в цій вертикалі і спробуємо підняти туру, на яку він указує. Вільних кліток немає. Знімаємо туру і, перемістивши горизонтальний покажчик ще раз вправо , вертикальний виставляємо проти тури у відповідній вертикалі і намагаємося підняти її. У нас знову нічого не вийде, ми знімаємо і цю туру і переміщаємо горизонтальний покажчик ще лівіше, а вертикальний - на туру в стовпці, на який вказує горизонтальний покажчик. Пробуємо підняти її. Це зробити вдається. Повторюємо всі ці дії , при цьому, якщо горизонтальний покажчик виходить за межі дошки вправо, виводимо розміщення, а ознакою того, що всі комбінації вже були, стане те, що горизонтальний покажчик вийде за межі дошки вліво.

Виберемо структуру даних. Поле представимо у виді матриці A[n:n], де N кількість кліток у кожній горизонталі і вертикалі ( та й тур у нашому випадку теж N). Якщо в даній клітинці немає тури - A[і,j]=0 , а якщо є то A[і,j]=1. Ще нам знадобляться дві змінні цілого типу для збереження в них значення покажчиків П\_В и П\_Г.

Для конструювання алгоритму на високому рівні деталізації нам необхідно мати такі процедури і функції:

**ПОСТАВ\_І\_ВПРАВО** - ставить туру в клітинку з заданими координатами і переміщає горизонтальний покажчик на одну позицію вправо , а вертикальний установлює на першу позицію (процедура)

**ЗНІМИ\_І\_ВЛІВО** - знімає туру з даної клітки і переміщає горизонтальний покажчик на одну позицію вліво , вертикальний покажчик встановлює в позицію тури, на яку тепер указує горизонтальний покажчик ( процедура )

**ПРАВИЛЬНА\_КЛІТИНКА** - логічна функція. Повертає ***істина***, якщо туру можна поставити в дану клітку, і ***хибно***, якщо поставити в клітинку не можна.

**ВЛІВО** - переміщає горизонтальний покажчик на одну позицію вліво і установлює вертикальний покажчик на туру в цій вертикалі (процедура).

**ВИВЕДЕННЯ**  - виводить на екран результат (процедура)

Тепер наш алгоритм може бути представлений так:

***алг*** Пошук\_з\_поверненням

***поч***

**для** і **від** 1 **до** n

**нц**

**для** j **від 1 до** n

**нц**

A[і,j] :=0

**кц**

**кц**

П\_Г:=1

П\_В:=1

ПОСТАВ\_І\_ВПРАВО П\_В:= П\_В+1

**поки** П\_Г**<>**0

**пц поки** ( не ПРАВИЛЬНА\_КЛІТИНКА ) і (П\_Г < n+1)

**пц**

П\_В:=П\_В+1

кц

**якщо** П\_В < n+1

**то**  ПОСТАВ\_І\_ВПРАВО

**інакше** ЗНІМИ\_І\_ВЛІВО

**все**

**якщо** П\_Г =n+1

**то** ВИВЕДЕННЯ

ВЛІВО

все

**кц**

**до** П\_Г=0

**кін**