Лабораторна робота № 9_10

Тема: Технології програмування алгоритмів сортування та пошуку у масивах даних

Мета роботи

• Отримання практичних навичок в обробці масивів, у сортуванні елементів масивів різними методами та за різними реквізитами.

• Дослідження і вивчення методів пошуку ключових елементів у масивах.

• Здійснення порівняння та аналізу ефективності використовуваних методів сортування і пошуку.

ТЕОРЕТИЧНІ ВІДОМОСТІ

1. Бульбашкове сортування (сортування обміном)

Метод "бульбашкового сортування" ґрунтується на перестановці сусідніх елементів. Для впорядкування елементів масиву здійснюються повторні проходи по масиву. Переміщення елементів масиву здійснюється таким чином: масив переглядається зліва направо, здійснюється порівняння пари сусідніх елементів; якщо елементи в парі розміщені в порядку зростання, вони лишаються без змін, а якщо ні - міняються місцями. В результаті 1-го проходу найбільше число буде поставлено в кінець масиву. У 2-му проході операції виконуються над елементами з першого до (N -1)-ого, у 3-му - від першого до (N-2)-ого і т.д. Впорядкування масиву буде закінчено, якщо при проході масиву не виконається жодної перестановки елементів масиву. Факт перестановки фіксується за допомогою деякої змінної *(is),* яка на початку має значення 0 і набуває значення 1, коли виконається перестановка в будь-якій парі.

Масив до впорядкування	22	20	-1	-40	88	-75	-22
Перший перегляд масиву	20	-1	-40	22	-75	-22	**88**
Другий перегляд масиву	-1	-40	20	-75	-22	**22**	**88**
Третій перегляд масиву	-40	-1	-75	-22	**20**	**22**	**88**
Четвертий перегляд масиву	-40	-75	-22	**-1**	**20**	**22**	**88**
П'ятий перегляд масиву	-75	-40	**-22**	**-1**	**20**	**22**	**88**

```
const n=10;
int a[n],  i,  c, is;
    /* ... */
do  { is=0;
  for i=1;i<n;i++)
  if (a[i-1]>a[i])
      {   c=a[i];  a[i]=a[i-1]; a[i-1]=c;
      is=1 ;
}
} while   (is) ;
```

2. Сортування методом вибору

Сутність методу така: масив переглядається перший раз, знаходиться мінімальний елемент масиву і міняється місцями з першим елементом. Другий раз масив переглядається, починаючи з другого елементу. Знову знаходиться мінімальний елемент і міняється місцями з другим. Даний процес виконується доти, поки не буде поставлений на місце *N-1* елемент.

Масив до впорядкування	22	20	-1	-40	88	-75	-22
Перший перегляд масиву	**-75**	20	-1	-40	88	**22**	-22
Другий перегляд масиву	-75	**-40**	-1	**20**	88	22	-22
Третій перегляд масиву	-75	-40	**-22**	20	88	22	**-1**
Четвертий перегляд масиву	-75	-40	-22	**-1**	88	22	**20**
П'ятий перегляд масиву	-75	-40	-22	-1	**20**	22	**88**
Шостий перегляд масиву	-75	-40	-22	-1	20	22	88

```
const int n=20;
int b[n];
int imin,  i,  j, a;
 /* ... */
for (i=0;i<n-1;i++)
{   imin=i;
     for (j=i+1;j<n;j++)
        if (b[j]<b[imin]) imin=j;
     a=b[i];
     b[i]=b[imin];
     b[imin]=a;
     }
```

Сортування вставками

При використанні даного методу на *i-му* етапі відбувається "вставка" елемента *a[i]* в потрібну позицію серед елементів *a[1], …,a[i-1],* які вже впорядковані. Після цієї вставки перші *i* елементів будуть впорядковані.

Масив до впорядкування	22	20	-1	**-40**	**88**	-75	-22
Перший перегляд масиву	20	22	-1	**-40**	**88**	-75	-22
Другий перегляд масиву	-1	20	22	**-40**	**88**	-75	-22
Третій перегляд масиву	**-40**	-1	20	22	**88**	-75	-22
Четвертий перегляд масиву	**-40**	-1	20	22	**88**	-75	-22
П'ятий перегляд масиву	-75	**-40**	-1	20	22	**88**	-22
Шостий перегляд масиву	-75	**-40**	-22	-1	20	22	**88**

```
const int n=20;
int b[n];
int i,j,c;
/* ... */
for (i=1;i<n;i++)
   {   c=a[i];
       for  (j=i-1;j>=0&&a[j]>c;j- -)
            a[j+1]=a[j];
       a[j+1]=c; }
```

Швидке сортування

Швидке сортування полягає в тому, що множина елементів B $\{k_1,..,\ k_n\}$ перетворюється на множину B_1, $\{k_1\}$, B_2, де B_1 - підмножина B з елементами, не більшими за k_1, а B_2 - підмножина B з елементами більшими k_1. Причому k_1 після розбиття множини B буде перебувати на потрібному місці. Далі до множин B_1 і B_2 знову застосовують впорядкування швидким сортуванням. На практиці цей алгоритм виявляється одним із найшвидших.

```
double * quick(double *s,int low,int hi)
{  double cnt,aux;
int i,j;
if (hi>low)
{   i=low; j=hi;
     cnt=s[i];
     while(i < j)
     {    if (s[i+1]<=cnt)
          {    s[i]=s[i+1];
               s =cnt;
               i++;
          }
          else
           {       if (s[j]<=cnt)
                   {    aux=s [j];
                        s [j]=s[i+1] ;
                        s =aux;
                    }
                   j--
          }
     }
     quick(s,low,i-1);
     quick(s,i+1,hi);
  }
  return(s);
}
```

Сортування методом Шелла

Головний недолік простих методів - обмін ведеться в основному між сусідними елементами. Тому бажано робити якомога ширші обміни.

Метод Шелла - метод сортування включеннями з відстанями, що зменшуються. Візьмемо для прикладу масив:

60 16 41 06 59 79 34 15

На першому етапі масив уявно ділиться на підмасиви з елементами, що, скажімо, відстоять один від одного на 4 елементи:

60 59
16 79
41 34
06 15

Кожен з них впорядковується окремо:

59 60
16 79
34 41
06 15

Сортування робиться у кожному підмасиві методом простого включення. На другому етапі підмасиви утворюються елементами через один:

59 34 60 41
16 06 79 15

Одержуємо:

34 41 59 60
06 15 16 79

Після цього весь масив сортується разом. За рахунок попередніх етапів він виявляється вже близьким до відсортованого, тому обмінів необхідно вже не так багато. Цей метод дає кращі результати, якщо розподіл на підмасиви роблять кроками, що не є степенями двійки, а, навпаки, не є множниками один одного.

```
void SortShell(int* arr, int n)
{
   int step = n/2;   // не обов'язково!
  while   (step > 0)
  {
    for(int i = 0; i < n-step;i++)
    {
      int j=i;
          while  (j>=0 && a[j]>s[j+step])
          {       int t=a [j];
                  a[j]=a[j+step];
                  a[j + step]=t;
                  j--
          }
      }
    step/=2;
  }
}
```

Лінійний пошук

Лінійний пошук порівнює кожен елемент масиву з ключем пошуку. Оскільки масив не впорядкований, цілком імовірно, що відшукують значення виявиться першим елементом масиву. Але в середньому програма повинна порівняти з ключем пошуку половину елементів масиву.

```
int LinearSearch (int array [], int size, int key) {
   for  (int i = 0;  i <size;  i + +)
   if (array[i]== key)
   return i;
   return -1;
}
```

Метод лінійного пошуку добре працює для невеликих або для несортованих масивів. І хоча для великих масивів лінійний пошук неефективний, він абсолютно надійний.

Двійковий пошук

Двійковий (бінарний) пошук - алгоритм пошуку елемента у відсортованому масиві. Бінарний пошук знайшов собі застосування в математиці і інформатиці.

Двійковий пошук можна використовувати тільки в тому випадку, якщо є масив, всі елементи якого впорядковані (відсортовані).

Нехай *x* - елемент, пошук якого здійснюється; *l (left)* - ліва границя пошуку;

r (right) - права границя пошуку.

```
int binary search(int x, int l ,int r)
{
   while (r - l > 1)
   {
       int mid = (l + r) / 2;
       //ділимо відрізок  [l,r] навпіл
       if(a[mid]<x)  l = mid;
       else r = mid;
   }
   for (int i = l; i<= r;i++)
        if(a[i]==x)
           return i;
   return -1;// не знайшли такого елемента
}
```

Порядок виконання роботи

1. Засвоїти теоретичний матеріал (теоретичні відомості до лаб.роб. та конспекти лекцій). Розібратись у сутності кожного методу сортування та пошуку.

2. Розробити програму згідно індивідуального завдання. Для сортування і пошуку повинні бути розроблені окремі функції.

3. Підготувати звіт:

- варіант і текст завдання;
- схеми роботи функцій сортування і пошуку;
- лістинг програми;
- результати виконання і аналізу сортування (вигляди екрану, таблиця і графік);
- висновки.

Варіанти індивідуальних завдань

1. Сгенерувати масив (масив оголошувати динамічно) і вивести його на екран (використати при цьому власну функцію для виведення масиву на екран).

2. Відсортувати масив, обравши вказані методи та параметри сортування.

3. Запускаючи програму не менше десяти разів для кожного методу (або передбачити це у програмі), задавати різну кількість *n* елементів масиву, отримати час *t* сортування. Побудувати залежність *t=f(n)* на одному графіку для різних методів сортування (у табличному і графічному виглядах).

4. Здійснити пошук вказаного (з клавіатури) елемента у масиві, використовуючи вказаний метод пошуку, згідно варіанту).

№	Методи сортування	Тип елементів масиву	Напрям сортування	Порядок сортування	Метод пошуку
1	Вибору Швидкий	Ціле	З початку	За убуванням	Бінарний
2	Вибору Вставки	Символьне	З початку	За зростанням	Лінійний
3	Шелла. Вибору	Дійсне	З кінця	За убуванням	Бінарний
4	Вставки. Бульбашковий	Довге ціле	З кінця	За зростанням	Бінарний
5	Шейкерний. Вибору	Символьне	З початку	За убуванням	Лінійний
6	Швидкий. Бульбашковий	Дійсне (подв. точність)	З початку	За зростанням	Лінійний
7	Вставки. Шейкерний	Коротке ціле	З кінця	За убуванням	Бінарний
8	Шелла. Шейкерний	Символьне	З кінця	За зростанням	Бінарний
9	Бульбашковий . Вибору	Дійсне	З початку	За убуванням	Лінійний
10	Вставки. Шелла	Ціле	З початку	За зростанням	Бінарний
11	Шейкерний. Швидкий	Символьне	З кінця	За убуванням	Бінарний
12	Вибору. Шейкерний	Дійсне (подв. точність)	З кінця	За зростанням	Лінійний
13	Бульбашковий . Шелла	Довге ціле	З початку	За убуванням	Лінійний
14	Вставки. Швидкий	Символьне	З початку	За зростанням	Бінарний
15	Шелла. Швидкий	Дійсне	З кінця	За убуванням	Бінарний

Контрольні питання

1. Які групи методів сортування ви знаєте? В чому їх сутність?
2. Поясніть алгоритм методу сортування бульбашками.

3. Наведіть приклад фрагмента коду, що здійснює сортування символів бульбашками у порядку зростання (починати сортування з першого елементу).

4. Наведіть схему роботи функції для реалізації бульбашкового сортування (сортування починати з кінця).

5. Поясніть алгоритм методу сортування вставками.

6. Наведіть схему роботи функції для реалізації сортування вставками (сортування починати з початку).

7. Наведіть приклад приклад фрагмента коду, що здійснює сортування цілих чисел вставками у порядку убування (починати сортування з першого елементу).

8. Поясніть алгоритм методу сортування вибором.

9. Наведіть схему роботи функції для реалізації сортування вибором (сортування починати з початку).

10. Наведіть приклад приклад фрагмента коду, що здійснює сортування дійсних чисел вибором у порядку убування (починати сортування з кінця).

11. Наведіть схему роботи функції для реалізації бульбашкового сортування.

12. Як оцінюються методи сортування?

13. Які удосконалення методів сортування ви знаєте?

14. В чому сутність шейкерного сортування? Наведіть приклад.

15. В чому сутність методу швидкого сортування? Його переваги.

16. Наведіть алгоритм методу Шелла для сортування? Його переваги.