СОРТИРАНЕ НА ДАННИ. АЛГОРИТМИ ЗА СОРТИРАНЕ

ВТУ, 23.10.2008 г.

СОРТИРАНЕ - ОБЩИ ПОНЯТИЯ

Сортиране – процес на пренареждане (пермутиране) на елементите на дадено множество в определен ред

Сортирането е основна дейност с широка сфера на приложение – речници, телефонни указатели, индексни справочници и др.

Примери:

- Цели числа – $A = \{2, 5, 1, 4, 3\} \rightarrow A' = \{1, 2, 3, 4, 5\}$;
- Имена в азбучен ред (кирилица) – {Веселин, Здравко, Йордан, Филип};
- Имена в азбучен ред (кирилица) – {Filip, Jordan, Veselin, Zdravko};
- Какво липсва в една дамска чанта? – ред (сортировка);

Сортирането се извършва по най-различни начини. Практически в основата на почти всеки компютърен алгоритъм има някакъв вид сортировка.

КЛАСИФИКАЦИЯ НА АЛГОРИТМИТЕ

Съществуват различни класификации на алгоритмите за сортиране.

- В зависимост от местонахождението на данните – вътрешно (с директен достъп) и външно (с последователен достъп);
- В зависимост от операцията – чрез сравнение $(<, >, =)$ и чрез трансформация (с помощта на аритметични операции);
- Устойчиви (относителният ред на елементите с равни ключове остава непроменен) и неустойчиви.

Нека $M = \{a_1, a_2, \ldots, a_n\}$ и f е функция върху M. Под ***сортиране*** на елементите на M ще разбираме пермутирането им в подходящ ред $a_{i_1}, a_{i_2}, \ldots, a_{i_n}$, така че да е изпълнено

$$f(a_{i_1}) \leq f(a_{i_2}) \leq \ldots \leq f(a_{i_n})$$

ПРИНЦИПИ НА СОРТИРАНЕТО

- При универсалните методи за сортиране ще представяме елементите на множеството чрез масив;
- Основно изискване към алгоритмите за сортиране е минималният разход на допълнителна памет;
- Друго изискване е минимален брой сравнения и размени на елементи;
- По принцип сортирането се извършва чрез размяна на два елемента.

Представяне на елементите:

```
const N = 100;
TypeElem array[N];
```

Размяна на елементи:

```
void swap(TypeElem array[ ], int i, int j) {
    TypeElem temp = array[i]; array[i] = array[j]; array[j] = temp;
}
```

ТЪРСЕНЕ НА МИН. ЕЛЕМЕНТ

Вход: Масив *array* от N елемента
Изход: Минималният елемент на масива

1. Инициализация – обявяваме първия елемент на масива за минимален;

2. Сравняване – проверяваме един по един останалите елементи. Ако текущия елемент е по-малък от минималния, минималния става равен на текущия;

3. Край – когато проверим всички елементи.

```
TypeElem min = array[0];
for(int i = 1; i < N; i++)
    if(array[i] < min)
        min = array[i];
```

МЕТОД НА ПРЯКАТА СЕЛЕКЦИЯ

Вход: Несортиран масив $array$ с N елемента.
Изход: Сортиран масив $array$.

1. Разделяме масива на сортирана част (с дължина i_1) и несортирана част (с дължина i_2). Първоначално $i_1 = 0, i_2 = n$;

2. Намираме минималния елемент в несортираната част на масива;

3. Разменяме първия и минималния елемент в несортираната част на масива;

4. Увеличаваме размера на сортираната част с 1 ($i_1 + +$) и намаляваме размера на несортираната част с 1 ($i_2 - -$);

5. Ако $i_2 > 1$ – стъпка 2;

6. Край.

МЕТОД НА ПРЯКАТА СЕЛЕКЦИЯ

```
for(i = 0; i < N-1; i++) {
    min = array[i]; index = i;
    for(j = i+1; j < N; j++) {
        if(array[j] < min) {
            min = array[j]; index = j;
        }
    }
    if(index != i)
        swap(array,i,index);
}
```

Сложност: $O(n^2)$

МЕТОД НА МЕХУРЧЕТО

Идеята на метода е 'леките' елементи да 'изплуват', т.е. да се преместват наляво (при сортиране във възходящ ред).

1. Разделяме масива на сортирана част (с дължина i_1) и несортирана част (с дължина i_2). Първоначално $i_1 = 0, i_2 = n$;

2. Сравняваме два съседни елемента x_{j-1} и x_j в несортираната част на масива (първоначално индексът j сочи последния елемент на масива);

3. Ако $x_{j-1} > x_j$, ги разменяме. Намаляваме j с 1, ако $j > i_1$ – стъпка 2;

4. Увеличаваме размера на сортираната част с 1 ($i_1 + +$) и намаляваме размера на несортираната част с 1 ($i_2 - -$);

5. Ако $i_2 > 1$ – стъпка 2;

6. Край.

МЕТОД НА МЕХУРЧЕТО

```
for(i = 0; i < N–1; i++) {
    for(j = N–1; j > i; j – –) {
        if(array[j–1] > array[j])
            swap(array,j–1,j);
    }
}
```

Сложност: $O(n^2)$

СОРТИРАНЕ ЧРЕЗ ВМЪКВАНЕ

Това е добре познатия на картоиграчите метод за нареждане на картите, при който играчът, държейки в едната си ръка картите, ги изважда една по една и ги поставя на правилната позиция. За вмъкване на картата на правилното място се налага последователното ѝ сравнение с вече наредените карти, до откриване на правилната позиция.

Масивът $array$ се разделя на сортирана и несортирана област. В началото сортираната област обхваща само първия елемент на масива. Сортирането протича на $N - 1$ стъпки. На i-та стъпка сортираната част се увеличава с 1, като за целта добавяният елемент x се вмъква на подходящо място в сортираната област.

Как става вмъкването? – последователно сравняваме (и евентуално разменяме) x със стоящия вляво от него елемент. Процесът продължава до възникване на една от следните ситуации:

1. достигане на елемент, който е по-малък или равен на x;

2. достигане на първия елемент на масива.

СОРТИРАНЕ ЧРЕЗ ВМЪКВАНЕ

```
for(i = 1; i < N; i++) {
    x = array[i];
    for(j = i; j > 0 && array[j–1] > x; j – –) {
        array[j] = array[j–1];
    }
    array[j] = x;
}
```

Сложност: $O(n^2)$