Дәріс 11

Массивті сұрыптау әдістері

1. Кіріспе

Сұрыптау (Sorting) — мәліметтерді белгілі бір тәртіпке (көбінесе өсу немесе кему ретімен) келтіру процесі.
Мақсаты — ақпаратты іздеу мен өңдеу жылдамдығын арттыру.

Сұрыптау компьютерлік ғылымның ең маңызды алгоритмдік мәселелерінің бірі болып табылады, себебі көптеген есептерде мәліметтерді белгілі бір ретке келтіру қажет.

2. Сұрыптаудың түрлері

Сұрыптау әдістері бірнеше критерийлер бойынша бөлінеді:

Белгісі	Түрлері
Салыстыру принципі бойынша	Салыстыру арқылы (comparison-based), салыстырусыз (counting, radix)
Тұрақтылық бойынша	Тұрақты (stable), тұрақсыз (unstable)
Жылдамдығы бойынша	$O(n^2)$ — жай әдістер, $O(n \log n)$ — тиімді әдістер
Жадты пайдалануына қарай	Ішкі (in-place), сыртқы (external)

3. Негізгі сұрыптау әдістері

3.1. Таңдау арқылы сұрыптау (Selection Sort)

Мәні: массивтен ең кіші элемент таңдалып, бірінші орынға қойылады; келесі итерацияда екінші кіші элемент екінші орынға және т.с.с.

Алгоритм қадамдары:

1. `i`-ші позиция үшін массивтің соңына дейін ең кіші элементті табу.
2. Оны `i`-ші элементпен алмастыру.
3. `i`-ді 1-ге арттыру, соңына дейін қайталау.

Уақыты: $O(n^2)$

Мысал (Python):

```
def selection_sort(a):
    n = len(a)
    for i in range(n-1):
        min_i = i
        for j in range(i+1, n):
```

```
            if a[j] < a[min_i]:
                min_i = j
        a[i], a[min_i] = a[min_i], a[i]
    return a
```

3.2. Кірістіру арқылы сұрыптау (Insertion Sort)

Мәні: әр элемент өз орнын табу үшін алдыңғы элементтермен салыстырылады.

Артықшылығы: кіші массивтер мен ішінара сұрыпталған массивтерге тиімді.

Уақыты: $O(n^2)$

Мысал:

```
def insertion_sort(a):
    for i in range(1, len(a)):
        key = a[i]
        j = i - 1
        while j >= 0 and a[j] > key:
            a[j + 1] = a[j]
            j -= 1
        a[j + 1] = key
    return a
```

3.3. Көпіршіктеу арқылы сұрыптау (Bubble Sort)

Мәні: көршілес элементтер салыстырылады, егер реті бұзылған болса — орындары ауысады.

Уақыты: $O(n^2)$, бірақ оңай түсінікті.

Мысал:

```
def bubble_sort(a):
    n = len(a)
    for i in range(n):
        for j in range(0, n-i-1):
            if a[j] > a[j+1]:
                a[j], a[j+1] = a[j+1], a[j]
    return a
```

3.4. Жылдам сұрыптау (Quick Sort)

Мәні: массив бір элемент (pivot) бойынша екі бөлікке бөлінеді:
сол жағында — кіші элементтер, оң жағында — үлкен элементтер.
Әр бөлік рекурсивті түрде сұрыпталады.

Уақыты: орташа — $O(n \log n)$, ең жаманы — $O(n^2)$.

Мысал:

```
def quick_sort(a):
    if len(a) <= 1:
        return a
    pivot = a[len(a)//2]
    left = [x for x in a if x < pivot]
    middle = [x for x in a if x == pivot]
    right = [x for x in a if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)
```

3.5. Біріктіру арқылы сұрыптау (Merge Sort)

Мәні: массив екіге бөлініп, әр бөлік жеке сұрыпталады, кейін біріктіріледі.

Уақыты: O(n log n)

Мысал:

```
def merge_sort(a):
    if len(a) > 1:
        mid = len(a)//2
        L = a[:mid]
        R = a[mid:]
        merge_sort(L)
        merge_sort(R)
        i = j = k = 0
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                a[k] = L[i]; i += 1
            else:
                a[k] = R[j]; j += 1
            k += 1
        while i < len(L):
            a[k] = L[i]; i += 1; k += 1
        while j < len(R):
            a[k] = R[j]; j += 1; k += 1
    return a
```

4. Салыстырмалы талдау

Әдіс	Орташа күрделілігі	Тұрақтылығы	Артықшылығы	Кемшілігі
Bubble Sort	$O(n^2)$	Иә	Қарапайым	Баяу
Selection Sort	$O(n^2)$	Жоқ	Аз алмасу	Көп салыстыру
Insertion Sort	$O(n^2)$	Иә	Кішкентай массивке тиімді	Үлкен массивте баяу
Merge Sort	O(n log n)	Иә	Тұрақты, сенімді	Көбірек жад қажет

Әдіс	Орташа күрделілігі	Тұрақтылығы	Артықшылығы	Кемшілігі
Quick Sort	O(n log n)	Жоқ	Өте жылдам	Ең жаман жағдайда баяу

5. Қорытынды

Массивті сұрыптау — мәліметтерді өңдеудің негізгі бөлігі.
Таңдалған әдіс массивтің көлеміне, реттелу дәрежесіне және жад шектеулеріне байланысты.