Контрольная работа по спискам

Теория:

1) Линейный поиск (программа полностью)
2) Двоичный поиск (таблицы поиска значения)
3) Сортировки (алгоритмы + таблицы сортировки)

Практика:

Написать программу полностью (использовать циклы, условные операторы)

Решения оформляются через п/п (надо уметь писать и печатать документацию функций)

1) Создать список (ввод с клавиатуры, инициализация, список со случайными числами)
2) Печатать список
3) Замена элемента (например, поменять все чётные числа на 1000000)
4) Перестановка элементов
5) Удалить элементы (методы, срезы)
6) Вставка элементов (методы, срезы)

Двоичный поиск

Дан отсортированный список a (по возрастанию). Найти в списке элемент со значением T.

Обозначим номер начального элемента списка как p, а номер последнего элемента списка как q.

Пока p не совпадает с q

1) считаем s = (p + q) // 2
2) проверяем a[s]

$a_s < T$	p	q
да	s+1	—
нет	—	s

Анализ поиска:

a[p] = T?

1) да => искомое число T в списке стоит под номером p
2) нет => искомого значения T в списке нет

ПРИМЕР

$a = 3_0, 7_1, 8_2, 10_3, 13_4, 15_5, 16_6, 18_7, 21_8, 23_9, 37_{10}, 39_{11}, 40_{12}, 44_{13}, 53_{14}$

$T = 10$

s	a_s	$a_s < T$	p	q	$p \neq q$
			0	14	+
7	18	—	0	7	+
3	10	—	0	3	+
1	7	+	2	3	+
2	8	+	3	3	—

$a_3 = 10$? да => искомый элемент в списке стоит под номером 3

$T = 45$

s	a_s	$a_s < T$	p	q	$p \neq q$
			0	14	+
7	18	+	8	14	+
11	39	+	12	14	+
13	44	+	14	14	—

$a_{14} = 45$? нет $\Rightarrow$

эл-та со значением 45 в списке нет

В представленных таблицах в верхнем обозначение:

"+" – истина (True)

"–" – ложь (False)

Сортировки

1) сортировка выбором (по возрастанию)

9.1 Сортировка выбором

Алгоритм сортировки выбором (I вариант):

1) найти элемент с наибольшим значением;

2) поменять значениями найденный элемент и последний в рассматриваемом подмассиве;

3) уменьшить на единицу количество просматриваемых элементов;

4) если ⟨количество элементов для следующего просмотра больше единицы⟩, то ⟨повторить пункты, начиная с 1-го⟩.

Сортировка выбором по возрастанию, I вариант

```
for i in range(len(a)-1, 0, -1):
    # пусть макс. значение у I эл-нта в подмассиве
    imax = 0
    for j in range(1, i+1):
        if a[j] > a[imax]:
            imax = j
    # обмен значений
    a[imax], a[i] = a[i], a[imax]
```

Алгоритм сортировки выбором (II вариант).

1) найти элемент с наименьшим значением;

2) поменять значениями найденный элемент и первый в рассматриваемом подмассиве;

3) сдвинуть левую границу просматриваемых элементов вправо (будет на один элемент меньше);

4) если ⟨количество элементов для следующего просмотра больше единицы⟩, то ⟨повторить пункты, начиная с 1-го⟩.

Сортировка выбором по возрастанию, II вариант

```
for i in range(0, len(a)-1):
    # пусть мин. значение у I эл-нта в подмассиве
    imin = i
    for j in range(i+1, len(a)):
        if a[j] < a[imin]:
            imin = j
    # перестановка элементов
    a[imin], a[i] = a[i], a[imin]
```

2) сортировка обменом (по возрастанию)

9.2 Сортировка обменом

Алгоритм сортировки обменом (метод пузырька). Последовательно сравниваются соседние элементы, и если выполняется неравенство

$$x_k > x_{k+1}$$

(сортировка по возрастанию), то поменять элементы x_k и x_{k+1} местами; в результате элемент с наибольшим значением окажется в конце массива.

Следующие просмотры списка применяются ко всем элементам, кроме последнего, и т.д.

Сортировка обменом, I вариант

```
for i in range(0, len(a)-1):
    for j in range(0, len(a)-i-1):
        # если соседи не упорядочены
        if a[j] > a[j+1]:
            # перестановка соседей
            a[j], a[j+1] = a[j+1], a[j]
```

II вариант реализации метода сортировки обменом — экономичный. Заведем переменную `sort` типа `Boolean`. Если `sort = True`, значит перестановка на данном шаге была произведена, в противном случае `sort` остается равным `False`. Если на некотором шаге не было произведено ни одной перестановки, то сортировка прекращается.

Экономичный вариант метода сортировки обменом выгоден для частично упорядоченных списков. Иначе, он будет выполняться даже дольше обычного варианта сортировки обменом.

Сортировка обменом, II вариант

```
sort = True   #  список надо сортировать
i = 0
while sort:
    sort = False
    for j in range(0, len(a)-i-1):
        # если соседи не упорядочены
        if a[j] > a[j+1]:
            sort = True  # есть перестановка
            # перестановка соседних элементов
            a[j], a[j+1] = a[j+1], a[j]
    i += 1
```

3) сортировка методом простой вставки (по возрастанию)

9.3 Сортировка включением

Алгоритм сортировки включением (простыми вставками): просматривать последовательно $a_1, \ldots, a_{n-1}$ и каждый новый элемент a_i вставлять на подходящее место в уже упорядоченную последовательность $a_0, \ldots, a_{i-1}$. Это место определяется последовательным сравнением a_i с упорядоченными элементами $a_0, \ldots, a_{i-1}$.

Сортировка включением, [redacted]

```
for i in range(1, len(a)):
    y = a[i]
    j = i - 1
    while (j >= 0) and (y < a[j]):
        a[j+1] = a[j]
        j = j - 1
    a[j+1] = y
```

ПРИМЕРЫ СОРТИРОВОК

2. Для вектора $(5, 4, 0, 2, 7, 1)$ представьте процесс сортировки по возрастанию *методом выбора* в виде таблицы.

1 шаг

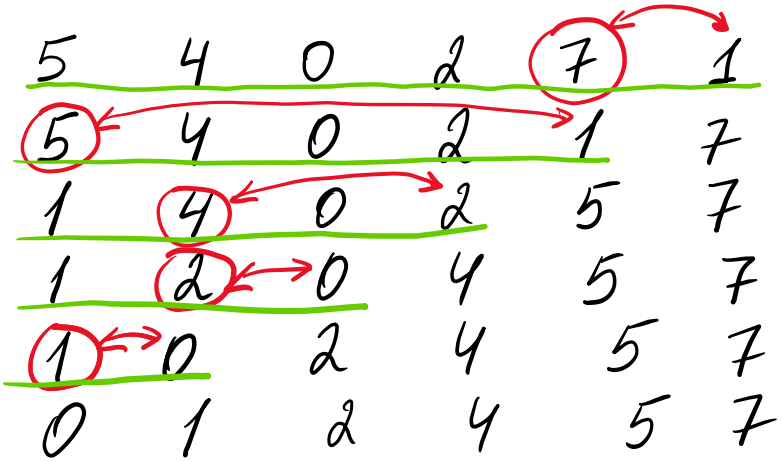
2 шаг

3 шаг

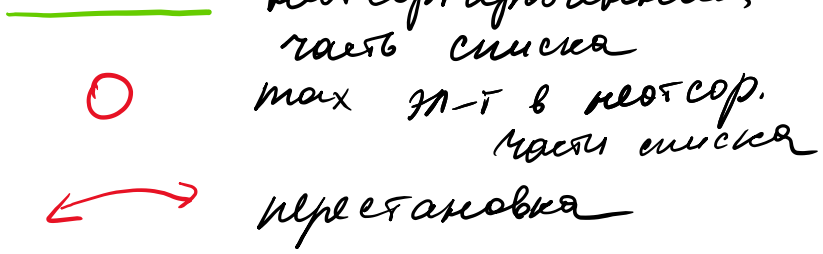
4 шаг

5 шаг

ОТВЕТ



Обозначения:



3. Для вектора $(6, 4, 1, 3, 8, 2)$ представьте процесс сортировки по убыванию *методом простых вставок* в виде таблицы.

1 шаг

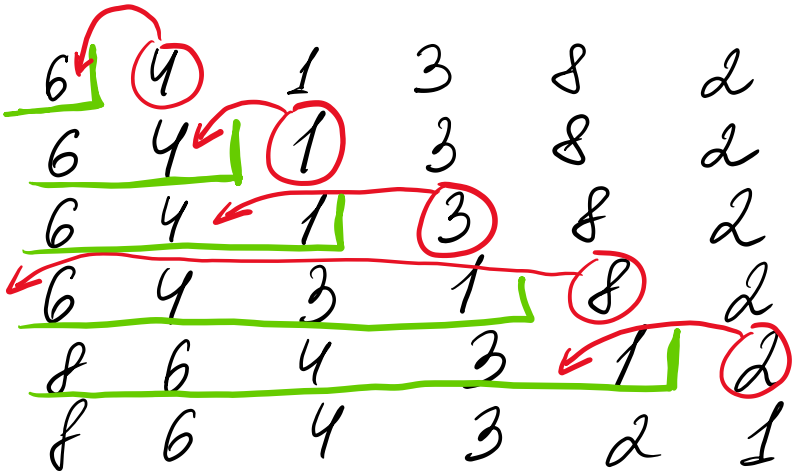
2 шаг

3 шаг

4 шаг

5 шаг

ОТВЕТ



Обозначения: отсортированная часть списка

эл-т и место для вставки в отсортир. часть списка

5. Представьте процессы сортировки по возрастанию списков $A = (10, 4, 6, 2, 5, 3)$ и $B = (2, 1, 0, 4, 3, 6)$ *методом обмена* в виде таблиц.

Список A

	10 4 6 2 5 3		
1	4 6 2 5 3 10	5	5
2	4 2 5 3 6 10	4	3
3	2 4 3 5 6 10	3	2
4	2 3 4 5 6 10	2	1
5	2 3 4 5 6 10	1	0

Список B

	2 1 0 4 3 6		
1	1 0 2 3 4 6	5	3
2	0 1 2 3 4 6	4	1
3	0 1 2 3 4 6	3	0
4	0 1 2 3 4 6	2	0
5	0 1 2 3 4 6	1	0

Будет ли выгода для экономичного варианта сортировки обменом?

Сортируем список A

1 шаг	10 4 6 2 5 3	5	5
2 шаг	4 6 2 5 3 10	4	3
3 шаг	2 4 3 5 6 10	3	2
4 шаг	2 3 4 5 6 10	2	1
5 шаг	2 3 4 5 6 10	1	0
ОТВЕТ	2 3 4 5 6 10		

Сортируем список B

1 шаг	2 1 0 4 3 6	5	3
2 шаг	1 0 2 3 4 6	4	1
3 шаг	0 1 2 3 4 6	3	0
4 шаг	0 1 2 3 4 6	2	0
5 шаг	0 1 2 3 4 6	1	0

ОТВЕТ	0 1 2 3 4 6	

Экономичный вариант сортировки обменом только для списка B даст выгоду в 2 шага (т.е. вместо 5 шагов сортировки необходимо будет выполнить только три шага).