Сортировка выбором

Устойчивость сортировки — сохранение относительного порядка элементов с равными ключами.

Идея сортировки выбором состоит в том, чтобы создавать отсортированную последовательность путем присоединения к ней одного элемента за другим в правильном порядке. На каждой итерации находится наименьший из ещё не упорядоченных элементов затем он перемещается в конец отсортированной части последовательности.

Реализация:

```
template<class T>
void selectSort(T a[], long size) {
	long i, j, k;
	T x;

	for (i = 0; i < size; i ++) {
		k = i;
		x = a[i];

		for (j = i + 1; j < size; j++) {
			if (a[j] < x) {
				k = j;
				x = a[j];
			}
		}

		a[k] = a[i];
		a[i] = x;
	}
}
```

Анализ алгоритма сортировки выбором

$$C = \sum_i = 1¿n-1(n-i) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = \frac{(n^2-n)}{2}$$

Число перестановок в худшем случае:

$M_i=(n-i)+3$

Суммарное число перестановок:

$M_{max}=\frac{(n^2-n)}{2}+3(n-1)$

Вероятность того, что произойдёт обновление текущего минимума на $j-ом$ шаге внутреннего цикла равна $\frac{1}{(j-i+1)}$

В среднем за весь внешний цикл суммарное число перестановок есть $M \approx n \ln n + n(\gamma+1)$

Сортировка обменами («пузырьковая»)

Идея метода: шаг сортировки состоит в проходе снизу вверх по массиву. По пути просматриваются пары соседних элементов. Если элементы некоторой пары находятся в неправильном порядке, то меняем их местами.

Реализация:

```
template<class T>
void bubbleSort(T a[], long size) {
	long i, j;
	T x;

	for (i = 0; i < size; i++) {
		for (j = size - 1; j > i; j--) {
			if (a[j - 1] > a[j]) {
				x = a[j-1];
				a[j-1] = a[j];
				a[j] = x;
			}
		}
	}
}
```

Сортировка вставками

- Будем разбирать алгоритм, рассматривая его действия на $i-м$ шаге.
  Последовательность к этому моменту разделена на две части: готовую $a[0].,a[i]$ и неупорядоченную $a[i+1]..a[n]$
- На следующем, $(i+1)-м$ каждом шаге алгоритма берем $a[i+1]$ и вставляем на нужное место в готовую часть массива.
  Поиск подходящего места для очередного элемента входной последовательности осуществляется путём последовательных сравнений с элементом, стоящим перед ним.

Таким образом, в процессе вставки мы «просеиваем» элемент x к началу массива, останавливаясь в случае, когда

1. Найден элемент, меньший x или
2. Достигнуто начало последовательности.

Реализация:

```
template<class T>
void insertSort(T a[], long size) {
	long i, j;
	T x;

	// цикл проходов, i - номер прохода
	for (i = 0; i < size; i++) {
		x = a[i];

		for (j = i - ; j >= 0 && a[j] > k; j--) {
			// сдвигаем элемент направо, пока не дошли
			a[j+1] = a[j];
		}

		// место найдено, вставить элемент
		a[j+1] = x;
	}
}
```

Анализ алгоритма сортировки вставками

$$C = \frac{1}{2}\sum_{i=2}^{n} i = \frac{1}{2}\left(\frac{n(n+1)}{2}-1\right) = \frac{1}{4}(n(n+1)-2) \approx \frac{1}{4}n^2$$