Алгоритмы и структуры данных-1

Инвариант цикла.
Временная сложность

Практическое занятие №1 02.09 — 07.09.2024

2024–2025 учебный год

ПЛАН

Временная сложность циклических алгоритмов

Инвариант циклического алгоритма – поиск и сортировка

Сортировка Шелла — улучшенный вариант INSERTION SORT

Разминка...

Упражнение 1 Временная сложность

NESTED_LOOPS.cpp

```cpp
int sum = 0;

for (size_t i = 1; i < N; i *= 2) {
    for (size_t j = N; j > 0; j /= 2) {
        for (size_t k = j; k < N; k += 2) {
            sum += (i + j * k);
        }
    }
}
```

С помощью пошаговой трассировки составить точное выражение для функции временной сложности $T(N)$, а также оценить порядок роста этой функции

Задача поиска

Упражнение 2 Линейный поиск

Вход: Последовательность ключей $\boldsymbol{A = \langle a_1, \ldots, a_n \rangle}$ и значение $\boldsymbol{v}$.

Выход: Индекс $\boldsymbol{1 \leq i \leq n}$, для которого $\boldsymbol{A[i] = v}$ или сообщение **NOT FOUND**, если такого индекса не существует.

1. Разработать циклический алгоритм линейного поиска
2. Доказать корректность разработанного алгоритма с помощью инварианта
3. Оценить сложность $\boldsymbol{T(n)}$ разработанного алгоритма

Упражнение 3 Бинарный поиск

Вход: Последовательность ключей $\boldsymbol{A = \langle a_1, \ldots, a_n \rangle}$, для которой $\boldsymbol{a_1 \leq a_2 \leq \cdots \leq a_n}$, и значение $\boldsymbol{v}$.

Выход: Индекс $\boldsymbol{1 \leq i \leq n}$, для которого $\boldsymbol{A[i] = v}$ или сообщение **NOT FOUND**, если такого индекса не существует.

1. Разработать циклический алгоритм бинарного поиска.
2. Доказать корректность разработанного алгоритма с помощью инварианта.
3. Оценить сложность $\boldsymbol{T(n)}$ разработанного алгоритма.

Упражнение 3 Бинарный поиск

Вход: Последовательность ключей $\boldsymbol{A = \langle a_1, \ldots, a_n \rangle}$, для которой $\boldsymbol{a_1 \leq a_2 \leq \cdots \leq a_n}$, и значение $\boldsymbol{v}$.

Выход: Индекс $\boldsymbol{1 \leq i \leq n}$, для которого $\boldsymbol{A[i] = v}$ или сообщение **NOT FOUND**, если такого индекса не существует.

1. Разработать циклический алгоритм бинарного поиска.
2. Доказать корректность разработанного алгоритма с помощью инварианта.
3. Оценить сложность $\boldsymbol{T(n)}$ разработанного алгоритма.

[Q] Почему функции округления, а также основание логарифма не влияет на оценку порядка роста $\boldsymbol{T(n)}$?

Задача сортировки

Упражнение 4 Сортировка пузырьком

BUBBLE_SORT.cpp

```cpp
void bubbleSort(std::vector<int> arr) {
    size_t N = arr.size();
    for (size_t i = 0; i < N; ++i) {
        for (size_t j = 0; j < N - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
                std::swap(arr[j], arr[j + 1]);
            }
        }
    }
}
```

1. Доказать корректность алгоритма сортировки с помощью инварианта.
2. Оценить сложность $\boldsymbol{T(N)}$ алгоритма. Выделить худший и лучший случай по временной сложности.

Упражнение 4 Сортировка пузырьком

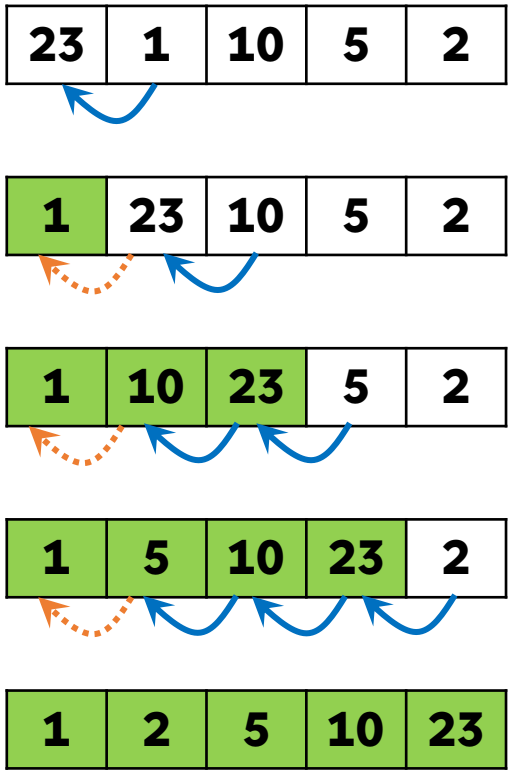
BUBBLE_SORT.cpp

```cpp
void bubbleSort(std::vector<int> arr) {
    size_t N = arr.size();
    for (size_t i = 0; i < N; ++i) {
        for (size_t j = 0; j < N - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
                std::swap(arr[j], arr[j + 1]);
            }
        }
    }
}
```

1. Доказать корректность алгоритма сортировки с помощью инварианта.
2. Оценить сложность $\boldsymbol{T(N)}$ алгоритма. Выделить худший и лучший случай по временной сложности.

[Q] Как мы можем улучшить алгоритм пузырьковой сортировки?

Сортировка вставками – замечания



1. Оценка худшего случая работы алгоритма - $\boldsymbol{\Theta(N^2)}$.
2. Продвижение элемента в отсортированную часть требует большого количества взаимных перестановок.

Сортировка Шелла

Сортировка вставками с изменяемыми интервалами

Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$

Сортировка Шелла

Сортировка вставками с изменяемыми интервалами

Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$

33	31	40	8	12	17	25	42

Сортировка Шелла

Сортировка вставками с изменяемыми интервалами

Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$

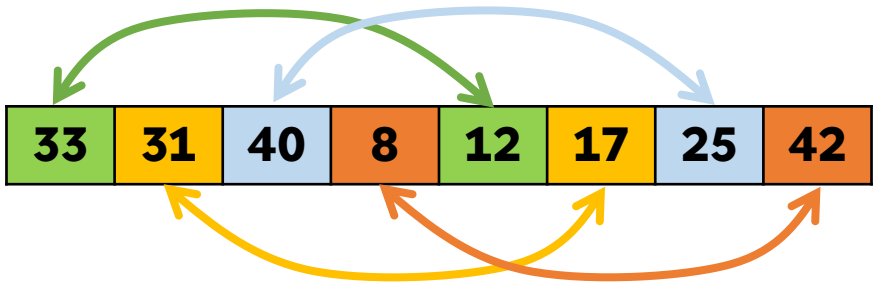
33	31	40	8	12	17	25	42

Установить корректное расположение элементов массива на расстоянии $\mathbf{4}$.

Сортировка Шелла

Сортировка вставками с изменяемыми интервалами

Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \dots, 1$



Установить корректное расположение элементов массива на расстоянии $\mathbf{4}$.

Сортировка Шелла

Сортировка вставками с изменяемыми интервалами

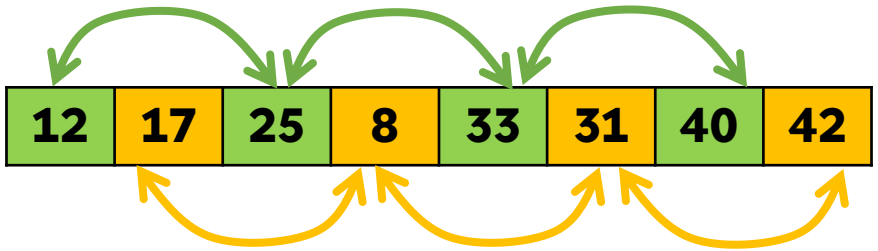
Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \dots, 1$

Установить корректное расположение элементов массива на расстоянии $\mathbf{4}$.

Сортировка Шелла

Сортировка вставками с изменяемыми интервалами

Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \dots, 1$



Установить корректное расположение элементов массива на расстоянии **2**.

Сортировка Шелла

Сортировка вставками с изменяемыми интервалами

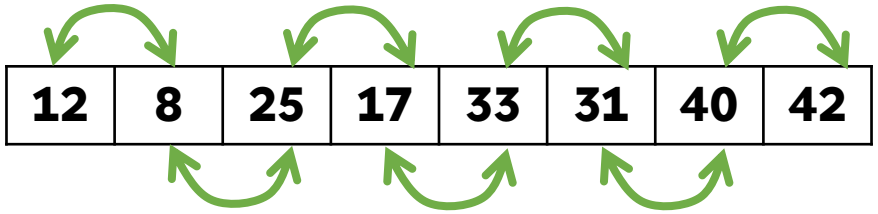
Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$

Установить корректное расположение элементов массива на расстоянии **2**.

Сортировка Шелла

Сортировка вставками с изменяемыми интервалами

Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \dots, 1$

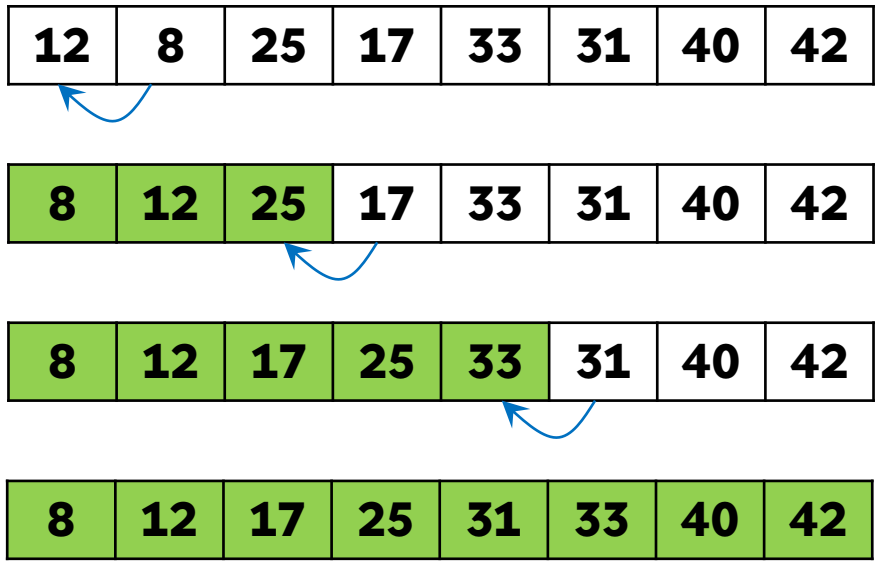


Установить корректное расположение элементов массива на расстоянии $\mathbf{1}$.

Сортировка Шелла

Сортировка вставками с изменяемыми интервалами

Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$



Установить корректное расположение элементов массива на расстоянии $\mathbf{1}$.

Сортировка Шелла

Сортировка вставками с изменяемыми интервалами

Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$

SHELL_SORT.cpp

```cpp
void shellSort(std::vector<int> arr) {
    size_t N = arr.size();
    for (size_t interval = N / 2; interval > 0; interval /= 2) {
        for (size_t i = interval; i < N; ++i) {
            int tmp = arr[i];
            for (size_t j = i; j >= interval && arr[j - interval] > tmp; j -= interval) {
                arr[j] = arr[j - interval];
            }
            arr[j] = tmp;
        }
    }
}
```

Сортировка Шелла

Сортировка вставками с изменяемыми интервалами

Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$

SHELL_SORT.cpp

```cpp
void shellSort(std::vector<int> arr) {
    size_t N = arr.size();
    for (size_t interval = N / 2; interval > 0; interval /= 2) {
        for (size_t i = interval; i < N; ++i) {
            int tmp = arr[i];
            for (size_t j = i; j >= interval && arr[j - interval] > tmp; j -= interval) {
                arr[j] = arr[j - interval];
            }
            arr[j] = tmp;
        }
    }
}
```

Временная сложность в худшем случае - $\Theta(n^2)$

Пояснение на доске...

Сортировка Шелла

Сортировка вставками с изменяемыми интервалами

Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \dots, 1$

- Последовательность Хиббарда - $1, 3, 7, 15, 31, 63, \dots, 2^k - 1$
- Последовательность Кнута - $1, 4, 13, 40, 121, \dots, \frac{3^k - 1}{2}$
- Последовательность Седжвика - $1, 8, 23, 77, 281, \dots, 4^k + 3 \cdot 2^{k-1} + 1$
- Последовательность Циура - $1,4, 10, 23, 57, 132, 301, 701$

и другие ...

Сортировка Шелла

Сортировка вставками с изменяемыми интервалами

Стандартная последовательность - $n/2, n/4, \dots, 1$

- Последовательность Хиббарда - $1, 3, 7, 15, 31, 63, \dots, 2^k - 1$ $\Theta\left(n^{3/2}\right)$
- Последовательность Кнута - $1, 4, 13, 40, 121, \dots, \frac{3^k-1}{2}$ $\Theta\left(n^{3/2}\right)$
- Последовательность Седжвика - $1, 8, 23, 77, 281, \dots, 4^k + 3 \cdot 2^{k-1} + 1$ $\Theta\left(n^{4/3}\right)$
- Последовательность Циура - $1,4, 10, 23, 57, 132, 301, 701,1750$???

и другие ...

РЕЗЮМЕ

Доказательство корректности
циклических алгоритмов с помощью инварианта

Оценка временной сложности и содержательный
анализ различных случаев работы алгоритма