Алгоритмы и структуры данных

Инвариант цикла.
Временная сложность

Практическое занятие **01**

2026–2027 учебный год

О формате практических занятий

1. Практические занятия продолжают и закрепляют лекционный материал.

2. Оценка **ПР** складывается из
   - **60%** — решения задач «у доски» во время занятий,
   - **40%** — решения самостоятельных работ (во время занятий, ~30 мин.).

Самостоятельные работы проводятся раз в две недели – письменные задачи или контесты для закрепления материала.

О формате практических занятий

1. По **сложности** задачи подразделяются на:
   - **простые** помечаются значком  и весят **1 балл**,
   - **средние** помечаются значком  и весят **2 балла**,
   - **трудные** помечаются значком  и весят **3 балла**.
2. Задачи, помеченные значком  , предназначены для **совместного** решения и обсуждения.

План

1. Временная сложность **циклических** алгоритмов
2. Инвариант циклического алгоритма
   - Задача **поиска** и **сортировки**
   - Пара слов о **бесконечных** циклах
3. Сортировка **Шелла** • улучшение `INSERTION SORT`

Warm-Up

FAQ Корректность и сложность

1. В чем измеряется **временная сложность** алгоритма? Почему?
2. Чем полезен **асимптотический** анализ сложности?
3. Как соотносятся функции $T(N)$ и $f(N)$, если они связаны соотношением $T(N) = \Theta\big(f(N)\big)$?
4. Инвариант цикла — это ...

Задача 1

Временная сложность вложенных циклов

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 точную функцию временной сложности $T(N)$ и оценить порядок роста этой функции.

Задача 2

Бесполезный и полезный инвариант цикла

MULTIPLY_LOOP.cpp

```cpp
int multiply(int a, int b) {
    int result = 0;

    for (int i = 0; i < a; ++i) {
        result += b;
    }

    return result;
}
```

1. Подходит ли условие $P = \{2 + 2 = 4\}$ в качестве инварианта приведенного циклического алгоритма?
2. Какое условие является более подходящим в качестве инварианта?

Задача 3

Есть ли инвариант у бесконечного цикла?

1. Подходит ли условие $P = \{i \geq 0\}$ в качестве инварианта приведенного циклического алгоритма?
2. Как проверить выполнимость P при выходе из цикла [TRM]?

INFINITE_LOOP.cpp

```cpp
int i = 2;

while (i > 0) {
    i = i + 1;
    std::cout << i;
}

i = i - 1;
```

Задача 3

Есть ли инвариант у бесконечного цикла?

1. Подходит ли условие $P = \{i \geq 0\}$ в качестве инварианта приведенного циклического алгоритма?
2. Как проверить выполнимость P при выходе из цикла [TRM]?

Инвариант применяется при условии **завершаемости** цикла, что требует дополнительных проверок...

INFINITE_LOOP.cpp

```cpp
int i = 2;

while (i > 0) {
    i = i + 1;
    std::cout << i;
}

i = i - 1;
```

Задача поиска

Задача 4

Анализ алгоритма линейного поиска

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. Разработайте циклический алгоритм линейного поиска.
2. Докажите корректность разработанного алгоритма с помощью инварианта.
3. Оцените сложность $\boldsymbol{T(n)}$ разработанного алгоритма.

Задача 5

Анализ алгоритма бинарного поиска

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. Разработайте циклический алгоритм бинарного поиска.
2. Докажите корректность разработанного алгоритма с помощью инварианта.
3. Оцените сложность $\boldsymbol{T(n)}$ разработанного алгоритма.

Задача 5

Анализ алгоритма бинарного поиска

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

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

1. Разработайте циклический алгоритм бинарного поиска.
2. Докажите корректность разработанного алгоритма с помощью инварианта.
3. Оцените сложность $T(n)$ разработанного алгоритма.

Почему основание логарифма **не влияет** на оценку порядка роста $T(n)$?

Задача сортировки

Задача 6

Анализ алгоритма BUBBLE SORT

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. Докажите корректность алгоритма сортировки с помощью инварианта.
2. Оцените сложность $\boldsymbol{T(N)}$ алгоритма.
3. Выделите худший и лучший случай по временной сложности.

Задача 6

Анализ алгоритма BUBBLE SORT

BUBBLE_SORT.cpp

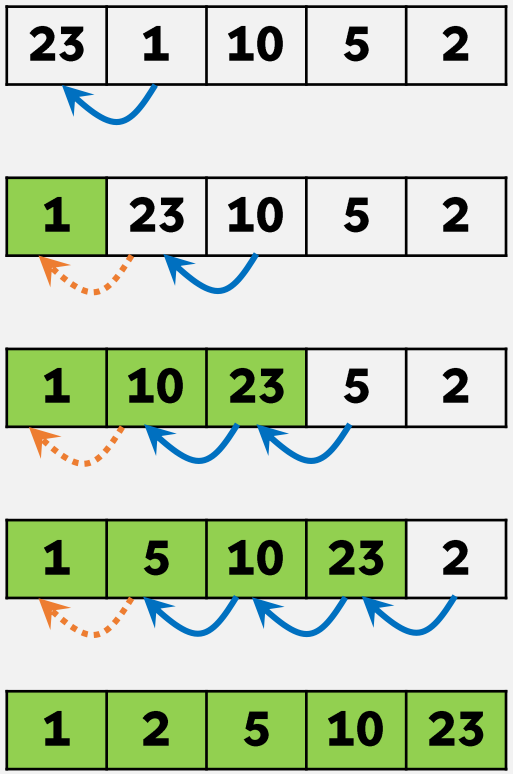
```cpp
void bubbleSort(std::vector<int> arr) {
    size_t N = arr.size();
    for (size_t i = 0; i < N; ++i) {
        for (size_t j = 0; j < N - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
                std::swap(arr[j], arr[j + 1]);
            }
        }
    }
}
```

1. Докажите корректность алгоритма сортировки с помощью инварианта.
2. Оцените сложность $T(N)$ алгоритма.
3. Выделите худший и лучший случай по временной сложности.

Как мы можем **оптимизировать** работу алгоритма BUBBLE SORT?

Вспомним

Анализ алгоритма INSERTION SORT



1. Оценка лучшего случая работы алгоритма — $\Theta(N)$.
2. Оценка худшего случая работы алгоритма — $\Theta(N^2)$.
3. Продвижение элемента в отсортированную часть требует большого количества смежных перестановок.

Для ускорения алгоритма можно **увеличить** интервалы перестановок…

Алгоритм сортировки SHELL SORT

- INSERTION SORT с изменяемыми интервалами
- Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$

Новое

Алгоритм сортировки SHELL SORT

- INSERTION SORT с изменяемыми интервалами
- Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$

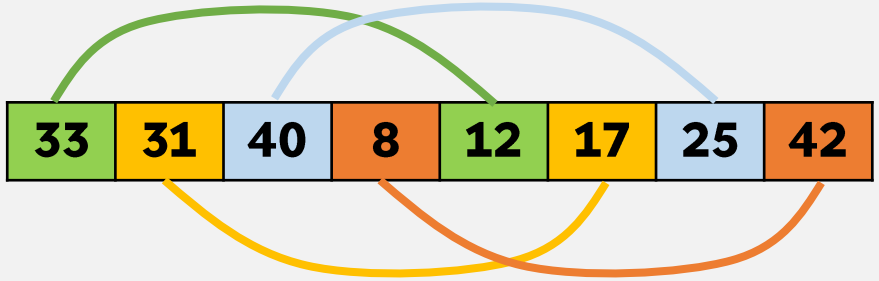
33	31	40	8	12	17	25	42

Установить корректное расположение элементов массива на расстоянии **4**

Новое

Алгоритм сортировки SHELL SORT

- INSERTION SORT с изменяемыми интервалами
- Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$

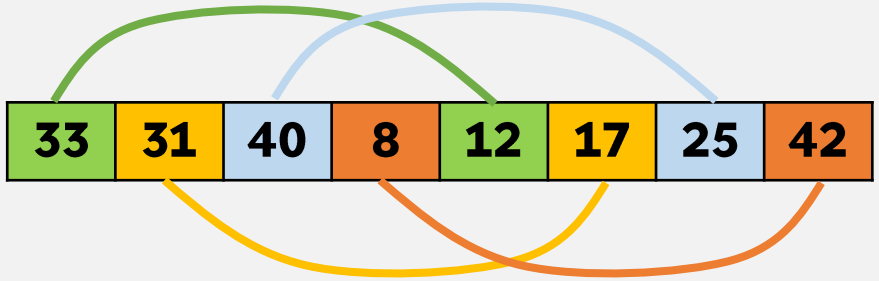


Установить корректное расположение элементов массива на расстоянии $\mathbf{4}$

Новое

Алгоритм сортировки SHELL SORT

- INSERTION SORT с изменяемыми интервалами
- Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$



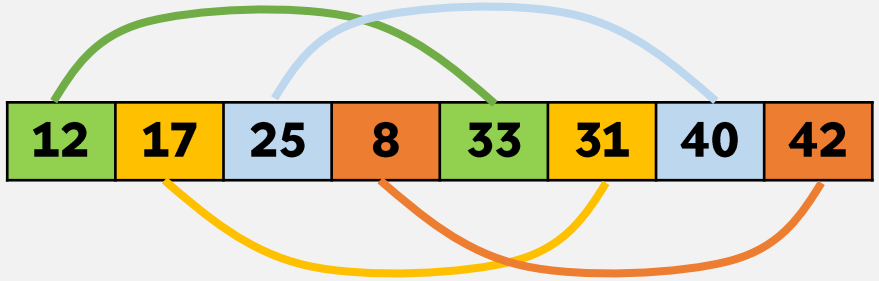
Установить корректное расположение элементов массива на расстоянии **4**

- В подмассивах выполняется обычный INSERTION SORT

Новое

Алгоритм сортировки SHELL SORT

- INSERTION SORT с изменяемыми интервалами
- Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$



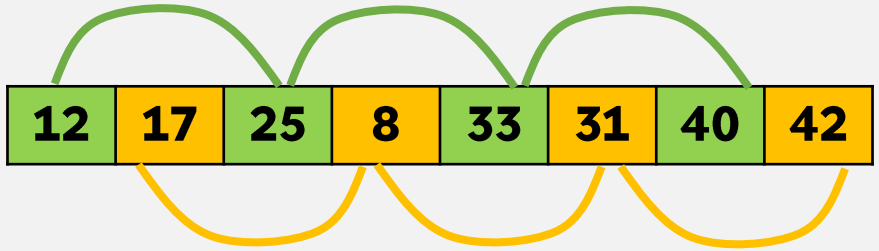
Установить корректное расположение элементов массива на расстоянии $\mathbf{4}$

- В подмассивах выполняется обычный INSERTION SORT

Новое

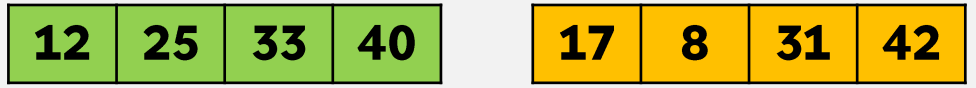
Алгоритм сортировки SHELL SORT

- INSERTION SORT с изменяемыми интервалами
- Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$



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

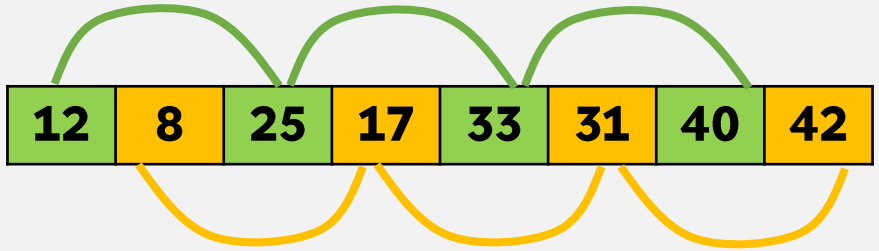
- В подмассивах выполняется обычный INSERTION SORT



Новое

Алгоритм сортировки SHELL SORT

- INSERTION SORT с изменяемыми интервалами
- Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$



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

- В подмассивах выполняется обычный INSERTION SORT

Новое

Алгоритм сортировки SHELL SORT

- INSERTION SORT с изменяемыми интервалами
- Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$

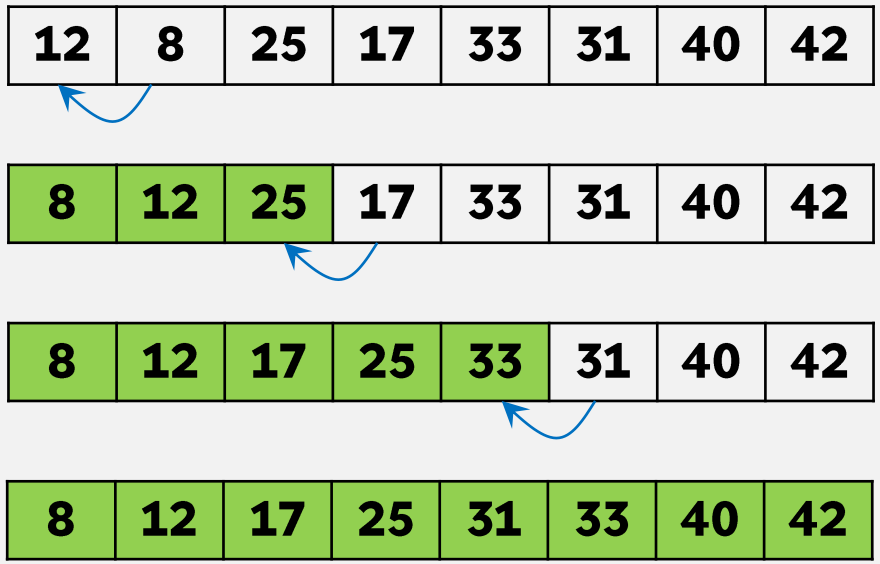
12	8	25	17	33	31	40	42

Установить корректное расположение элементов массива на расстоянии **1**

Новое

Алгоритм сортировки SHELL SORT

- INSERTION SORT с изменяемыми интервалами
- Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \ldots, 1$



Установить корректное расположение элементов массива на расстоянии **1**

На последнем шаге выполняется обычный INSERTION SORT

Новое

Алгоритм сортировки SHELL SORT

- INSERTION SORT 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 — $\Theta(N^2)$... Поясним

Анализ алгоритма SHELL SORT

- INSERTION SORT с изменяемыми интервалами
- Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \dots, 1$

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

и другие ...

Новое

Анализ алгоритма SHELL SORT

- INSERTION SORT с изменяемыми интервалами
- Стандартная последовательность - $^{n}/_{2}, ^{n}/_{4}, \dots, 1$

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

и другие ...

Резюме

1. Доказательство **корректности** циклических алгоритмов с помощью инварианта:
   - инвариант должен быть осмысленным
   - особый случай бесконечных циклов
2. Оценка **временной сложности** и содержательный анализ различных случаев работы алгоритма
3. Алгоритм сортировки SHELL S0RT