Инвариант цикла

Инвариант цикла — это логическое утверждение, которое остается истинным в каждой итерации цикла. Инвариант часто используется для доказательства корректности алгоритмов, особенно в алгоритмах сортировки.

- Инвариант должен быть истинным перед началом выполнения цикла.
- Инвариант должен сохраняться на каждой итерации.
- После завершения цикла инвариант должен привести к утверждению о корректности результата.

Сортировка вставкой

Алгоритм сортировки вставкой состоит в том, чтобы на каждой итерации текущий элемент вставлялся в уже отсортированную часть массива.

Пример кода

```c
#include <stdio.h>

// Function for insertion sort
void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        // Move elements that are greater than key
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}
```

Инвариант цикла

> После i-й итерации внешний цикл гарантирует, что первые i элементов массива (`array[0..,i-1]`) упорядочены.

Задача о найме. Анализ в худшем и среднем случае.

Описание задачи:

Компания нанимает кандидатов на вакансию, рассматривая их одного за другим в случайном порядке. На каждом шаге доступна информация только о текущем кандидате и уже просмотренных. Решение о найме

принимается немедленно: новый кандидат принимается на работу, если его квалификация лучше всех предыдущих.

Алгоритм решения задачи

Псевдокод:

```
HireCandidate(candidates):
    best = None
    for candidate in candidates:
        if candidate > best:
            best = candidate
            Hire(candidate)
```

Здесь `Hire(candidate)` — процедура найма кандидата.

Анализ в худшем случае

В худшем случае каждый новый кандидат оказывается лучше всех предыдущих, и алгоритм принимает всех кандидатов. Если в списке n кандидатов, потребуется n наймов.

Временная сложность: $O(n)$.

Анализ в среднем случае

Пусть кандидаты поступают в случайном порядке. Вероятность того, что i-й кандидат окажется лучшим среди первых i кандидатов, равна $1/i$. Таким образом, матожидание числа наймов равно:

$$E[\text{число наймов}] = \sum_{i=1}^{n} \frac{1}{i}.$$

Эта сумма равна H_n (гармоническое число), которое аппроксимируется как:

$$H_n \approx \ln n,$$

Средняя сложность: $O(\ln n)$.

Итоговый анализ

- В худшем случае потребуется $O(n)$ наймов.
- В среднем случае потребуется $O(\ln n)$ наймов.

Понятие рандомизации алгоритма

Рандомизация в алгоритмах — это использование случайных чисел или случайных выборов для управления поведением алгоритма. Рандомизированные алгоритмы могут давать разные результаты или проходить разные пути выполнения на одном и том же входе.

Преимущества рандомизации

- Простота реализации.
- Избежание худших случаев, зависящих от структуры входных данных.
- Более высокая производительность в среднем случае.

Недостатки рандомизации

- Возможность некорректных решений (в случае Монте-Карло алгоритмов).
- Потребность в качественном источнике случайных чисел.