Для більш точного урахування теплофізичних характеристик огороджень попередньо в спеціалізованому програмному продукті THERM було проведено моделювання окремих вузлів складної форми. Також проводилася оцінка дотримання санітарно-гігієнічних вимог стосовно температури поверхні зовнішніх огороджень.

Висновки: проведено енергетичне обстеження серійної житлової будівлі і виконане моделювання за допомогою програмних продуктів ENSI та програми, розробленої на базі EXEL; що дозволяє визначати базовий рівень споживання та потенціал енергозбереження.

Список використаних джерел:

1. ДБН В.2.6-31:2016. Теплова ізоляція будівель – К.: Мінрегіонбуд, 2017. – 33 с.
2. ЗУ № 2118 від 22.06.2017 «Про енергетичну ефективність будівель».
3. ДСТУ ISO 50002:2016 Енергетичні аудити. Вимоги та настанова щодо їх проведення.
4. Аналіз програмних продуктів для оцінювання показників енергоефективності будівель / Шовкалюк М.М., Зіменко С.В. // Збірник наук. праць міжнар. наук.-техн. конф. «Муніципальна енергетика: проблеми, рішення» – [Миколаїв, 20-21 грудня 2017 р.] – С. 72-77.
5. ДСТУ Б А.2.2-12:2015 Енергетична ефективність будівель. Метод розрахунку енергоспоживання при опаленні, охолодженні, вентиляції та ГВП.
6. ДСТУ Б EN ISO 13790:2011 Енергоефективність будівель. Розрахунок енергоспоживання при опаленні та охолодженні. – К. : НДІБК, 2011. – 229 с.
7. ДСТУ Б В.2.6-189:2013. Методи вибору теплоізоляційного матеріалу для утеплення будівель. – К. : Мінрегіонбуд, 2013. – 55 с.

Озеров А.В., Давыдова О.Н.
студенты;
Кралина А.С.
преподаватель,
Колледж информационных технологий и землеустройства
Национального авиационного университета

АЛГОРИТМИЧЕСКАЯ СЛОЖНОСТЬ

Алгоритмическая сложность – незаменимая на практике вещь, о которой часто можно услышать. Что же это такое и для чего это программистам?

Итак, зачем программистам понимать сложность алгоритмов и уметь оценивать их? Ответ: на практике программисты частенько встречаются с необходимостью выбора определенных алгоритмов при реализации задач в программировании [5, с. 84]. Например, нужно будет выбрать какой-нибудь контейнер для хранения и обработки данных. Это может быть массив или связный список, множество или карта и т.д. Они всегда смогут предположить, какие операции будут приоритетными: добавление и удаление элементов, поиск элементов либо всяческие изменения и махинации со значениями. Зная сложность таких операций, как добавление, удаление, поиск и обращение к

элементу, и имея представление о том, какие операции для вашей задачи будут приоритетными, программисты без труда выберут наиболее оптимальный контейнер. Либо же перед ними может стоять другая задача: выбрать наиболее оптимальный алгоритм для обработки числовой последовательности. Умея оценить характер последовательности и сложность алгоритмов для обработки, программист решит задачу с минимальными потерями производительности.

Алгоритмическая сложность – это зависимость объема работы, выполняемой алгоритмом от размера и характера входящих данных [2, с. 22].

Измерить скорость алгоритма реальным временем – секундами и минутами – непросто. Одна программа может работать медленнее другой не из-за собственной неэффективности, а по причине медлительности операционной системы, несовместимости с оборудованием, занятости памяти компьютера другими процессами [2].

Для измерения эффективности алгоритмов придумали универсальные концепции, и они выдают результат независимо от среды, в которой запущена программа [1, с. 124]. Измерение асимптотической сложности задачи производится с помощью нотации О-большого.

Заглядывая вперед, в большинстве случаев нас интересует худший случай работы алгоритма, и обозначается это как О-большое, также есть и наилучший случай – Омега-большое или Ω.

Рассмотрим на примере. Представьте, что в переменной length сохранили значение, равное количеству символов в строке. Например, length = 1000. Чтобы получить длину строки, нужен только доступ к этой переменной, на саму строку можно даже не смотреть. И как не менять длину, всегда можно обратиться к этой переменной. В таком случае асимптотическая скорость = O(1). От изменения входных данных скорость работы в такой задаче не меняется, поиск завершается за постоянное время. В таком случае программа является асимптотически постоянной.

Но допустим надо найти какое-то значение в этой строке с помощью линейного поиска, несложно догадаться, что в худшем случае искомое значение будет на последнем месте или не будет вообще, в такой ситуации будет линейная зависимость «времени» от размера входных данных O(n).

Если вам кажется, что линейное время – это плохо, смеем заверить: бывает гораздо хуже. Например, O(n^2). Такая запись означает, что с ростом n рост времени, нужного для перебора элементов (или другого действия) будет расти всё резче.

Оценивая асимптотическую сложность, необходимо использовать только ту часть, которая возрастает быстрее всего. Предположим, что рабочий цикл описывается выражением $n^3 + n$. В таком случае его сложность будет равна O(n^3). Рассмотрение быстро растущей части функции позволяет оценить поведение алгоритма при увеличении n. Например, при n=100, разница между $n^3 + n$=1000100 и n^3=1000000 равна всего лишь 100, что составляет 0,01%.

При вычислении О можно не учитывать постоянные множители в выражениях. Алгоритм с рабочим шагом $3n^3$ рассматривается, как O(n^3). Это делает зависимость отношения O(n) от изменения размера задачи более очевидной.

Итак, подытожим: работа измеряется в абстрактных единицах. За единицу работы мы принимаем выполнение каких-либо элементарных действий. Это могут быть вызовы функций, какие-нибудь вычисления, логические операции и так далее [4].

В заключение приведём таблицу, которая показывает, как долго компьютер, осуществляющий миллион операций в секунду, будет выполнять некоторые медленные алгоритмы.

Таблица 1

Выполнение алгоритмов

	N=10	N=20	N=30	N=40	N=50
N^3	0.001 с	0.008 с	0.027 с	0.064 с	0.125 с
2^N	0.001 с	1.05 с	17.9 мин	1. 29 дней	35.7 лет
3^N	0.059 с	58.1 мин	6.53 лет	$3.86 * 10^5$ лет	$2.28 * 10^{10}$ лет
$N!$	3.63 с	$7.71 * 10^4$ лет	$8.41 * 10^{18}$ лет	$2.59 * 10^{34}$ лет	$9.64 * 10^{50}$ лет

Список использованных источников:

1. Вирт Н. Алгоритмы и структуры данных. – М.: Мир, 1989. – 256 с.
2. Ахо, Хопкрофт, Ульман. Структуры данных и алгоритмы. – М.: Вильямс, 2000. – 384 с.
3. Habr.com [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/104219/
4. Online-learning.harvard.edu [Електронний ресурс] – Режим доступу до ресурсу: https://online-learning.harvard.edu/course/cs50-introduction-computer-science
5. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих – СПБ.: Питер, 2018. – 288 с.

Сторчак В.С.
аспірант,
Льотна академія
Національного авіаційного університету

ОСНОВНІ ПРИНЦИПИ ПОБУДОВИ ТРЕНАЖЕРНИХ СИСТЕМ ДЛЯ ПІДГОТОВКИ ОПЕРАТОРІВ АВТОМАТИЗОВАНИХ СИСТЕМ УПРАВЛІННЯ ПОВІТРЯНИМ РУХОМ

Для підтримання відповідного рівня професійної підготовки операторів автоматизованих систем управління повітряним рухом (АС УПР) необхідно зосереджувати головну увагу на тренажерній підготовці, її забезпеченості методичним супроводженням, сучасними тренажерними системами, що дозволить підтримати й закріпити отримані знання, придбати навички й уміння в управлінні екіпажами в складній навігаційній обстановці та аварійних ситуаціях. Автоматизація управління істотно розширює можливості оператора