Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних інформаційних систем

Алгоритми та складність

Завдання №2

“Болти та гайки”

Виконав студене 2-го курсу

Групи К-22

Гуща Дмитро Сергійович

2020

**Завдання**:

Нехай є n болтів різного розміру та n відповідних гайок. Можна порівнювати, чи підходять гайка і болт одне до одного, або гайка більша чи менша. Порівнювати між собою дві гайки чи два болти не можна. Розробіть і реалізуйте алгоритм розбивки всіх гайок і болтів на відповідні пари за час Θ(n log n).

**Теорія:**

Засновуємо наш алгоритм на швидкому сортуванні. Потрібно взяти перший гвинт і розділити всі гнізда на ті що більше і ті, що менше. Наступний гвинт порівняти з гніздом відповідним до першого і далі порівнювати тільки з тією групою, куди він може підійти.

Алгоритм швидкого сортування працює за nlogn, що задовольняє умові завдання

**Алгоритм:**

Алгоритм складається з трьох кроків:

• Вибираємо елемент з масиву. Назвемо його опорним.

• Розбиваємо: перерозподіляємо елементи в масиві таким чином, що елементи менше опорного помістяться перед ним, а більші або рівні -- після.

• Рекурсивно застосувати перші два кроки до двох підмасивів зліва і праворуч від опорного елемента. Рекурсія не застосовується до масиву, в якому тільки один елемент або елементи відсутні.

**Складність:**

На кожному кроці алгоритму знаходиться одна пара (болт,гайка), і задача розбивається на дві підзадачі меншого розміру.

Найгірший випадок – коли масив гайок розташований у порядку спадання, тому на кожному кроці виникає не дві підзадачі, а всього одна, тільки на 1 менша за попередню. Тоді складність алгоритму *n*(*n*-1)/2 – квадратична.

Кращий та середній випадки – коли на кожному кроці масив болтів ділиться приблизно пополам. Тут складність – Θ(n log n).

**Мова програмування:**

С++

**Модулі програми:**

void NutsAndBoltsMatch::printArray(char array[], int n)

//Просто вивід массивів

int NutsAndBoltsMatch::partition(char arr[], int low, int high, char pivot)

//Повертає індекс розділу масиву на основі елемента зведення іншого масиву.

void NutsAndBoltsMatch::matchPairs(char nuts[], char bolts[], int low, int high)

//Вибираємо останній символ масиву болтів для розділення гайок. Тепер, використовуючи partition з гайками, вибираємо,   
// що для підходить для partition болтів. Повторяємо для [low...pivot-1] & [pivot+1...high] для гайок та болтів.

void NutsAndBoltsMatch::PartitionMatch(char nuts[], char bolts[], int n)

// Цей алгоритм спочатку знаходить partition, вибираючи останній елемент масиву болтів як опорний, переставляє масив  
// nut і повертає індекс розділу „i“ таким чином, що всі гайки, менші за гайки[i], знаходяться з лівого боку та все  
// гайки більші за гайки[i] знаходяться з правого боку. Далі за допомогою гайок[i] ми можемо розділити масив болтів.  
// Операції розділення можуть бути легко реалізовані в O(n). Тепер ми застосовуємо це розділення рекурсивно до лівого та правого підмасиву

**Інтерфейс користувача:**

Вхідні дані завчасно записані до головного методу int main() та результат роботи виводиться на екран.

**Тестові приклади**

|  |  |
| --- | --- |
| **input** | **output** |
| Bolts array: { '$', '%', '&', '^', '@', '#' }  Nuts array:{ '@', '#', '$', '%', '^', '&' } | Sorted bolts array: # $ % & @ ^  Sorted nuts array: # $ % & @ ^ |

**Висновок:**

Вирішив завдання про болти й гайки.

**Література**

* Лекція №4
* «АЛГОРИТМИ ТА СТРУКТУРИ ДАНИХ» Воробйова О.Д., Глазунова Л.В. [Лекція 11. ШВИДКЕ СОРТУВАННЯ, ст. 54]