Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Алгоритми та методи обчислення

Лабораторна робота №3

**«Обчислювальна складність алгоритмів сортування»**

Виконав:

студент групи ІВ-81

Зікратий Д. О.

Перевірив:

ст.вик. Порєв В. М.

Київ

2020 р.

**Мета:** Закріплення навичок практичної оцінки алгоритмічної складності логічних алгоритмів на прикладі алгоритмів сортування.

**Завдання:** Використовуючи відповідний до варіанту алгоритм сортування

написати програму сортування масиву даних. Застосовуючи дану програму,

дослідити часову складність алгоритму сортування та порівняти її з

теоретичною алгоритмічною складністю.

**Алгоритм за варіантом:** сортування Хоара

**Роздруківка тексту програми:**

Select\_sort.py

**def** select\_sort\_min(arr: list):  
 **if** len(arr) == 0:  
 **return  
 for** j **in** range(len(arr)):  
 \_min = j  
 **for** i **in** range(j+1, len(arr)):  
 **if** arr[i] < arr[\_min]:  
 \_min = i  
 **if** \_min != j:  
 value = arr[\_min]  
 **for** l **in** range(\_min, j-1, -1):  
 arr[l] = arr[l-1]  
 arr[j] = value  
 **return** arr  
  
  
**def** select\_sort\_max(arr: list):  
 **if** len(arr) == 0:  
 **return  
 for** j **in** range(len(arr)-1, -1, -1):  
 \_max = j  
 **for** i **in** range(len(arr)-1, j-1, -1):  
 **if** arr[i] > arr[\_max]:  
 \_max = i  
 **if** \_max != j:  
 value = arr[\_max]  
 **for** l **in** range(j+1, \_max,):  
 print(arr)  
 arr[l] = arr[l-1]  
 arr[j] = value  
 **return** arr  
  
  
**def** count\_sort\_time(length: int, step: int=1):  
 **from** random **import** randint  
 **from** time **import** clock  
  
 n\_time = dict()  
 **for** j **in** range(0, length+1, step):  
 r = []  
 **for** i **in** range(0, j):  
 new = randint(-100000000001, 100000000001)  
 **if** new **not in** r:  
 r.append(new)  
 **else**:  
 i -= 1  
 t1 = clock()  
 select\_sort\_max(r)  
 t2 = clock()  
 n\_time.update([(j, round(t2-t1, 10))])  
 **return** n\_time  
  
  
*# print(count\_sort\_time(100000, 100000))*

GUI.py

*# -\*- coding: utf-8 -\*-***import** matplotlib  
matplotlib.use(**"TkAgg"**)  
**from** matplotlib.backends.backend\_tkagg **import** FigureCanvasTkAgg  
**from** matplotlib.figure **import** Figure  
**import** third\_sem.AMO.lab2.select\_sort **as** my\_alg  
**from** tkinter **import** \*  
**from** random **import** randint  
**from** time **import** clock  
  
  
**def** show\_plot\_1000(max\_len=10000, step=1000):  
 *"""Shows real and teor computational complexity."""* slave = Toplevel(root)  
 slave.title(**'Graphs of real and teor computational complexity'**)  
 my\_time = my\_alg.count\_sort\_time(max\_len, step)  
 x1 = list(my\_time.keys())  
 y1 = list(map(**lambda** key: my\_time[key], x1))  
  
 teor\_time = list(map(**lambda** x: x \*\* 2, list(range(0, max\_len+1, step))))  
  
 f = Figure(figsize=(6, 6), dpi=100)  
 a = f.add\_subplot(211)  
 a.set\_title(**'Real computational complexity'**, fontsize=14)  
 a.plot(x1, y1, **'b'**)  
 a.autoscale(enable=**True**, axis=**u'both'**, tight=**False**)  
 a.set\_xlabel(**'Number of elements'**, fontsize=11)  
 a.set\_ylabel(**'Sorting time, s'**, fontsize=11)  
  
 b = f.add\_subplot(212)  
 b.set\_title(**'Theoretical graph'**, fontsize=14)  
 b.plot(x1, teor\_time, **'r'**)  
 b.autoscale(enable=**True**, axis=**u'both'**, tight=**False**)  
  
 f.tight\_layout()  
  
 canvas = FigureCanvasTkAgg(f, slave)  
 canvas.show()  
 canvas.get\_tk\_widget().pack(side=BOTTOM, fill=BOTH, expand=**True**)  
  
  
**def** random\_arr(length):  
 *"""Generate arr with random inetegers and with a long 'length'"""* \_arr = []  
 **for** i **in** range(length):  
 \_arr.append(randint(-1000, 1000))  
 **return** \_arr  
  
  
**def** but\_random():  
 *"""Insert random\_arr into ent\_arr"""* length = ent\_arr.get()  
 **if** check\_input(length) == **'error'**: **return** \_arr = random\_arr(int(length))  
 ent\_arr.delete(0, END)  
 ent\_arr.insert(0, \_arr)  
  
  
**def** check\_input(arr):  
 *"""Check string 'arr' for input errors"""* **global** l1  
 **if** arr == **''**:  
 **try**:  
 l1.destroy()  
 **except** NameError:  
 **pass  
 finally**:  
 l1 = Label(root, text=**'First input something, then press "Result" or "Random"!'**, font=**'Calibri 13 bold'**, fg=**'red'**, bg=main\_bg)  
 l1.grid(column=0, row=5, columnspan=5, pady=10)  
 **return 'error'  
 elif** arr.isalpha():  
 **try**:  
 l1.destroy()  
 **except** NameError:  
 **pass  
 finally**:  
 l1 = Label(root, text=**'Check your input. There are non-numeric symbols.'**, fg=**'red'**, bg=main\_bg)  
 l1.grid(column=0, row=5, columnspan=5, pady=10)  
 **return 'error'  
 else**:  
 **try**:  
 l1.destroy()  
 **except** NameError:  
 **pass  
 return 'OK'  
  
  
def** but\_result():  
 **global** l1  
 arr = ent\_arr.get()  
  
 **if** check\_input(arr) == **'error'**: **return  
  
 for** i **in** [**','**, **'\n'**, **'\t'**, **'\r'**]:  
 arr.replace(i, **' '**)  
 arr = arr.split(**' '**)  
  
 **for** i **in** range(arr.count(**''**)):  
 **try**:  
 arr.remove(**''**)  
 **except**: **pass  
  
 for** i **in** range(len(arr)):  
 arr[i] = int(arr[i])  
  
 t1 = clock()  
 sorted\_arr = my\_alg.select\_sort\_min(arr)  
 t2 = clock()  
  
 **for** i **in** range(len(sorted\_arr)):  
 sorted\_arr[i] = str(sorted\_arr[i])  
  
 **if** len(sorted\_arr) > 15:  
 **with** open(**r'D:\Python\Brovchenko\_projects\third\_sem\AMO\lab2\res.txt'**, **'w'**) **as** fs:  
 fs.writelines(**' '**.join(sorted\_arr))  
 fs.close()  
  
 Button(root, text=**'Show sorted'**, command=show\_from\_file, bg=but\_bg, font=but\_font).grid(column=0, row=9, columnspan=5)  
 **else**:  
 ent\_arr.delete(0, END)  
 ent\_arr.insert(0, **' '**.join(sorted\_arr))  
  
 Label(root, text=**'Sorting time: {}'**.format(t2-t1), bg=main\_bg, font=**'Calibri 14'**).grid(column=0, row=7, columnspan=5)  
  
  
**def** clear\_entry():  
 ent\_arr.delete(0, END)  
  
  
**def** open\_file():  
 *"""Arr from file"""* **with** open(**r'D:\Python\Brovchenko\_projects\third\_sem\AMO\lab2\f.txt'**, **'r'**) **as** f:  
 \_text = f.read().split(**' '**)  
 f.close()  
  
 **for** i **in** range(\_text.count(**''**)):  
 **try**:  
 \_text.remove(**''**)  
 **except**: **pass  
  
 for** i **in** range(len(\_text)):  
 \_text[i] = int(\_text[i])  
  
 ent\_arr.delete(0, END)  
 ent\_arr.insert(0, \_text)  
  
  
**def** show\_from\_file(file=**r'D:\Python\Brovchenko\_projects\third\_sem\AMO\lab2\res.txt'**):  
 *"""Shows the contents of a file"""* **with** open(**r'{}'**.format(file), **'r'**) **as** fs:  
 arr = fs.read()  
 fs.close()  
 slave = Toplevel()  
 slave.title(**'Sorted array'**)  
 slave.resizable(width=FALSE, height=FALSE)  
 T = Text(slave, height=10, width=60)  
 T.pack()  
 T.insert(END, arr)  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 root = Tk()  
 root.title(**'Select sort'**)  
 root.resizable(width=**False**, height=**False**)  
 main\_bg = **'yellow'** but\_bg = **'chartreuse'** but\_font = **'Calibri 13 bold'** root[**'bg'**] = main\_bg  
  
 Label(root, text=**'Input here your array'**, font=**'Calibri 16 bold'**, bg=main\_bg)\  
 .grid(column=0, row=0, columnspan=5, pady=10)  
 Label(root, text=**'\*if you want to generate an array randomly, input length'**, font=**'Calibri 14'**, bg=main\_bg)\  
 .grid(column=0, row=1, columnspan=5, pady=5)  
 ent\_arr = Entry(root, font=**'Calibri 14'**, width=50, bd=0)  
 ent\_arr.grid(column=0, row=2, columnspan=5, padx=5)  
  
 Button(root, text=**'Random'**, command=but\_random, font=but\_font, bg=but\_bg).grid(column=0, row=3, pady=10)  
 Button(root, text=**'From file'**, command=open\_file, font=but\_font, bg=but\_bg).grid(column=3, row=3, columnspan=2)  
  
 Button(root, text=**'Result'**, command=but\_result, font=but\_font, bg=but\_bg).grid(column=0, row=4, pady=10)  
 Button(root, text=**'Show graphics'**, command=show\_plot\_1000, font=but\_font, bg=but\_bg).grid(column=3, row=4, columnspan=2)  
  
 Button(root, text=**'Clear'**, command=clear\_entry, font=**'Calibri 13'**, bg=**'greenyellow'**, relief=GROOVE).grid(column=0, row=3, columnspan=5)  
 root.mainloop()

**Висновки:** У ході виконання лабораторної роботи я закріпила знання з базових понять алгоритмів, навички практичної оцінки алгоритмічної складності логічних алгоритмів на прикладі алгоритмів сортування.