Indian Institute of Technology Kharagpur

CS29003: Algorithms Laboratory, Spring 2021

Assignment 4: Greedy Algorithms

2PM – 5PM 9th February, 2021

General Instructions (to be followed strictly)

Submit a single C/C++ source file.
Do not use global variables unless you are explicitly instructed so.
Do not use Standard Template Library (STL) of C++.
Use proper indentation in your code and comment.
Name your file as `<roll_no>_<assignment_no>`.
Write your name, roll number, and assignment number at the beginning of your program.

A café serves fast food and beverages all day. It opens at time $t = 0$ with customers arriving at opening time and placing orders. Suppose that there are n customers and it takes time t_i (an integer) to process and serve the i-th customer's order (for $1 \leq i \leq n$), known to the café manager apriori. The café is huge and has k different counters where orders can be placed. The café's objective is to schedule the orders so that all customers are served as early as possible. The café closes once all orders are processed. Computing the optimal schedule is believed to be a difficult problem. On the other hand, 'fairly good' solutions can be obtained using greedy strategies.

(a) In this part, you will write a function *schedule_orders_a*() that on input $t_1, \ldots, t_n$ and k, schedules the orders using the following greedy approach. Assume that first i orders are already scheduled and for $j = 1, \ldots, k$ the j-th counter finishes the currently assigned orders at time s_j. We (greedily) pick the counter that finishes earliest to assign the next order. More precisely, we assign the $(i+1)$-th order to counter j if $s_j = \min(s_1, s_2, \ldots, s_k)$. If there are more than 1 counters finishing earliest, we break ties by picking the one with the smallest index (i.e., pick the smallest j). The function *schedule_orders_a*() should compute a schedule for each counter and return the closing time for the café given the schedule. You may use a 2-D array to store the schedule so that row j contains the schedule of orders for the j-th counter.

(b) It is often possible to close the café earlier by scheduling orders with longer processing time before the shorter ones. Write a function *schedule_orders_b*() that first sorts $t_1, \ldots, t_n$ in decreasing/non-increasing order and uses this sorted sequence to schedule the orders following the same strategy as explained in Part a. The function should return the closing time corresponding to the schedule computed. Write your own function to sort an array of integers in non-increasing order that runs in time $O(n \log n)$.

(c) Suppose the i-th customer's order is served at counter j starting at time w_i. Call w_i the waiting time for customer i (given by the sum of times taken to process orders assigned to counter j prior to order i). The total waiting time defined by $w_T = \sum_{i=1}^{n} w_i$ is an indicator of customer satisfaction. Larger value of w_T indicates higher dissatisfaction among customers. Write a function *wait_time*() that, given a schedule, computes and returns w_T. Write another function *schedule_orders_c*() running in time $O(n)$ to minimize w_T without delaying the closing time of the café as achieved by the schedule of Part b.

In the *main()* function,

- Read the number of customers n and the order processing times $\{t_i\}_{i=1}^{n}$ from the user.
- Run the function from Part a. Print the order schedule computed using an auxiliary function *print_schedule*(), followed by the closing time of the café. Call *wait_time*() and print the total waiting time.
- Repeat the above for functions from Parts b and c in that order.

Sample Output 1

```
n = 15
k = 5

Enter order processing times:

4 24 29 24 13 20 18 15 30 12 3 11 7 17 24

Part (a)

Order Schedule (processing time for each order within parantheses)
  Counter  1:    1 ( 4)    6 (20)    8 (15)   15 (24)
  Counter  2:    2 (24)    9 (30)
  Counter  3:    3 (29)   11 ( 3)   13 ( 7)
  Counter  4:    4 (24)   10 (12)   14 (17)
  Counter  5:    5 (13)    7 (18)   12 (11)
Cafe closes at time 63
Total waiting time = 256

---

Part (b)

Order Schedule
  Counter  1:    9 (30)    5 (13)   13 ( 7)
  Counter  2:    3 (29)    8 (15)    1 ( 4)
  Counter  3:    2 (24)    6 (20)   11 ( 3)
  Counter  4:    4 (24)    7 (18)   12 (11)
  Counter  5:   15 (24)   14 (17)   10 (12)
Cafe closes at time 53
Total waiting time = 345

---

Part (c)

Order Schedule
  Counter  1:   13 ( 7)    5 (13)    9 (30)
  Counter  2:    1 ( 4)    8 (15)    3 (29)
  Counter  3:   11 ( 3)    6 (20)    2 (24)
  Counter  4:   12 (11)    7 (18)    4 (24)
  Counter  5:   10 (12)   14 (17)   15 (24)
Cafe closes at time 53
Total waiting time = 157
```