|  |
| --- |
| **Algorithm**: Bus Scheduling and Allocation in both directions for a bus route  **Input:** List of trips in both direction *up\_trips and down\_trips;* Bus travel time for the route *travel\_time*in minutes  **Output:** Bus Schedule for the given route  *1:* Initialize *Bus\_tag← 0, curr\_hour ← 4, next\_hour ← 5, curr\_min ← 0*  2: Compute *up\_count*= ∑𝑁𝑢𝑚𝑏𝑒𝑟 𝑜𝑓 𝑇𝑖𝑚𝑒 𝑆𝑙𝑜𝑡𝑠 𝑢𝑝\_𝑡𝑟𝑖𝑝𝑠[𝑖]and *down\_count*= 𝑖=0  𝑁𝑢𝑚𝑏𝑒𝑟 𝑜𝑓 𝑇𝑖𝑚𝑒 𝑆𝑙𝑜𝑡𝑠  ∑ 𝑑𝑜𝑤𝑛\_𝑡𝑟𝑖𝑝𝑠[𝑖]  𝑖=0  3: **while** j ≤ Number of Time Slots do  4: **If** *up\_trips[j]>1* then generate the schedule for *curr\_hour* and append to *up\_schedule* with headway between trips = Number of minutes in 1 hour / *up\_trips[j]*  5: **Else** if *up\_trips[j] == 1* then schedule the trip for *curr\_hour* at 30 minutes past *curr\_hour* and append to *down\_schedule*  6: **If** *down\_trips[j]>1* then generate the schedule for *curr\_hour* and append to *down\_schedule* with headway between trips = Number of minutes in 1 hour / *down\_trips[j]*  7: **Else** if *down\_trips[j] == 1* then schedule the trip for *curr\_hour* at 30 minutes past *curr\_hour* and append to *down\_schedule*  8: j ← j+1, *curr\_hour = curr\_hour + 1*, *next\_hour ←next\_hour + 1*, *curr\_min ← 0* 9:  **end while**  10: Create a sorted combined list of *up\_schedule* and *down\_schedule* named combined with *dir ← ‘U’* appended to *up\_schedule* records and *dir ←‘D’* appended to *down\_schedule* records  11: *up\_schedule\_index ← 0*, *down\_schedule\_index ← 0* |
| 12: **while** j ≤ len (*combined*) do  13: **If** *combined[j].dir== ‘U’* then *down\_count ← down\_count – 1 //as a bus from down will arrive at up*  //down trips are left then  14: **If** *down\_count> 0* then  // and there are buses came to up  15: **If** *len (up\_arrival)>0* then sort *up\_arrival* by time in ascending order  // their time <= current time, up bus arrived at 4 and current is 4 or 5  16: **If** *up\_arrival [0].time ≤combined[j].time* then  *17: up\_schedule [up\_schedule\_index].Bus\_Tag ← up\_arrival [0].Bus\_tag 18: down\_arrival.append (up\_arrival [0].time + travel\_time,up\_arrival [0].Bus\_tag)*  // remove the scheduled bus from up\_arrival  *19: up\_arrival [0].drop()*  *20: up\_schedule\_index ← up\_schedule\_index + 1*  21: **Else** sort *down\_arrival* by time in ascending order  // if a bus goes up->down and comes back at up before the current time  22: **If** *down\_arrival[0].time + travel\_time ≤ combined[j].time* then  *23: up\_schedule [up\_schedule\_index].Bus\_Tag ← down\_arrival*  *[0].Bus\_tag*  *24: down\_arrival.append (down\_arrival [0].time +*  *2\*travel\_time,down\_arrival[0].Bus\_tag)*  *25: down\_arrival [0].drop()*  *26: up\_schedule\_index ← up\_schedule\_index + 1*  //No bus available can do it  27: **Else** create new bus for the trip  *28: Bus\_tag ← Bus\_tag + 1*  *29: up\_schedule [up\_schedule\_index].Bus\_Tag ← Bus\_tag*  *30: down\_arrival.append (combined[j].time +travel\_time,Bus\_tag)*  *31: up\_schedule\_index ← up\_schedule\_index + 1*  32: **Else** goto step 20  33: **Else** goto step 20  34: **Else** *up\_count ← up\_count – 1*  //up trips are left then  35: **If** *up\_count> 0* then  // buses are available at down  36: **If** *len (down\_arrival)>0* then sort *down\_arrival* by time in ascending order  37: **If** *down\_arrival [0].time ≤ combined[j].time* then  *38: down\_schedule [down\_schedule\_index].Bus\_Tag ← down\_arrival[0].Bus\_tag*  *39: up\_arrival.append(down\_arrival[0].time+ travel\_time,down\_arrival[0].Bus\_tag)*  *40: down\_arrival [0].drop()*  *41: down\_schedule\_index ← down\_schedule\_index + 1*  *42:* ***Else*** *sort up\_arrival by time in ascending order*  *43:* ***If*** *up\_arrival[0].time + travel\_time ≤ combined[j].time then*  *44: down\_schedule [down\_schedule\_index].Bus\_Tag ← up\_arrival[0].Bus\_tag*  *45: up\_arrival.append(up\_arrival[0].time+ 2\*travel\_time,up\_arrival[0].Bus\_tag)*  *46: up\_arrival [0].drop()*  *47: down\_schedule\_index ← down\_schedule\_index + 1*  *48:* ***Else*** *create new bus for the trip*  *49: Bus\_tag ← Bus\_tag + 1*  *50: down\_schedule [down\_schedule\_index].Bus\_Tag ← Bus\_tag*  *51: up\_arrival.append (combined[j].time + travel\_time,Bus\_tag)*  *52: down\_schedule\_index ← down\_schedule\_index + 1*  *53:* ***Else*** *goto step 40*  *54:* ***Else*** *goto step 40*  *55: j ← j+1*  *56:* ***end while***  *57: Total\_buses ← Bus\_tag*  *58: Return up\_schedule, down\_schedule, Total\_buses* |