

**Experiment: 5**

**Title:** Implementation of Event Scheduling Algorithm for Dump Truck Problem

**Problem Statement:**

Dump truck is used to haul load from the entrance of a small mine to the railroad. Each truck is loaded by one of two loaders. After a loading, truck immediately moves to the scale, to be weighed as soon as possible. Both the loaders and the scale have a first-come-first-served waiting line (queue) for trucks. Travel time from a loader to the scale is considered negligible. After being weighed, a truck begins a travel time (during which the truck unloads) and then afterwards returns to the loader queue.

The distribution of loading time, weighing time and travel time are given along with random digits assigned. Simulate the dump truck problem using C/C++/Java.

**Expected Outcome of Experiment:**

| **Index** | **Outcome** |
| --- | --- |
| CO1 | Understand the concepts of discrete event simulation and its importance in business, science, engineering, industry and other services. |
| CO2 | Ability to analyse and apply general principles of event scheduling algorithm & various statistical methods on different applications. |

**Books/ Journals/ Websites referred:**

1. Jerry Banks, John Carson, Barry Nelson, and David M. Nicol, “Discrete Event System Simulation”; Fifth Edition, Prentice-Hall.

2. Averill M Law, “System Modeling & Analysis”; 4th Edition TMH.

3. Banks C M, Sokolowski J A, “Principles of Modeling and Simulation”, Wiley

**Pre Lab/ Prior Concepts:**

**Theory:**

To estimate the loader and scale utilization (% of time busy)

**Concept in Discrete event simulation:**

1. System: A collection of entities (e.g. peoples & machines) that interact together over time to accomplish one or more goals.

2. Model: An abstract representation of a system, usually containing structures, logical or mathematical relationships that describe a system in terms of state, entities and their attributes, sets, process, events.

3. System state: A collection of variables that contain all the information necessary to describe the system at any time.

4. Entity: Any objector component in the system that requires explicit representation in the model.

5. Attributes: The properties of a given entity.

6. List: A collection of associated entities (permanent / temporary) associated entities, ordered in some logical fashion.

7. Event: An instantaneous occurrence that changes the state of the system.

8. Event notice: A record of an event to occur at the current / future time, along with any associated data necessary to execute the event.

9. Activity: Duration of time of specified length, which is known when it begins.

10. Delay: A duration of time specified length which is not known until it ends.

11. Clock: A variable representing simulated time.

**World View:**

**Even scheduling approach**: When using the event scheduling approach, a simulation analyst concentrates on events & their effect on system state. Loading & weighing are two events which affect the system at loader & weighing queue. Based on the queue is busy or idle, imminent event is loaded.

**Process Interaction approach**: It describes the lifespan of activities. The analyst defines the simulation model in terms of entities or objects and their life cycle as they flow through the system, demanding resource and queuing to wait for resources. This life cycle consists of various events & activities. It is based on fixed time advance. Disadvantage of it is that we need to scan activity again and again.

**Activity scanning approach:** It is also known as three phase approach. It considers the activities of duration zero time units. Based on this, activities are categorized as B activities: It includes activities which are bound to occur, all primary events and unconditional activities. C activities: It includes activities or events that are conditional upon certain conditions being true.

**In three phase approach**, the simulation has

Phase A: Remove imminent event from the FEL and advance the clock to its event time. Remove form FEL any other events that have the same event time.

Phase B: Execute all B-type events that were removed from FEL.

Phase C: Scan the condition that triggers each C type activity and rescan until no additional C- type activities can begin & no events occur.

**Conceptual Model:**

**1.** **System states**: [ LQ(t), L(t), WQ(t), W(t)]

Where, LQ(t) = No. of trucks in loader queue

L(t) = No. of trucks (0,1 or 2) being loaded

WQ(t) = No. of trucks in weigh queue

W(t) = No. of trucks (0 or 1) being weighed

All at simulation time t.

**2. Entities**: Six dump trucks [DT1,DT2, … ,DT6]

**3. Lists**: Loading queue - All trucks waiting to begin loading

Weighing queue – All trucks waiting to be weighed

**4. Events**: An instantaneous occurrence that changes the state of the system.

**5. Event notices**: (ALQ, t, DTi) Dump truck i arrives at loader queue ALQ at time t

(EL, t, DTi) Dump truck i end loading EL at time t

(EW, t, DTi) Dump truck i end weighing EW at time t

**6. Activities**: Loading time, weighing time & travel time

**7. Delay**: Time required at loader queue & scale.

**Random Numbers:**

Random numbers used in this simulation are for loading time & weighing time of dump truck.

**Algorithm / Activity Diagram: (Simulation Approach):**

**The Event scheduling algorithm:**

The sequence of actions which a simulation language must perform to advance the clock and build a new system snapshot is called Event scheduling algorithm / Time advance algorithm.

STEP 1. Remove the event notice for the imminent event from FEL

STEP 2. Advance clock to imminent event time.

STEP 3. Execute imminent event, update system state, change entity attribute and set membership as needed

STEP 4. Generate future events and place their event notices on FEL, ranked by event time.

STEP 5. Update cumulative statistics and counters.

**Code:**

load = [10,5,5,10,15,10,10]

weigh = [12,12,12,16,12,16]

travel = [60,100,40,40,80]

clock = 0

lq = 3;l = 2;wq = 0;w = 1

loader\_queue = ['T4','T5','T6']

loaders = ['T2','T3']

weigh\_queue = []

scale = 'T1'

travelling = []

fel = []

bl = 0

bs = 0

i = 0

j = 0

k = 0

for loader in loaders:

fel.append(['EL',clock+load[i],loader])

i += 1

fel.append(['EW',clock+weigh[j],scale])

j += 1

for traveller in travelling:

fel.append(['ALQ',clock+travel[k],traveller])

k += 1

bl = 0;bs = 0

print("Clock: ",clock)

print("LQ(t): ",lq)

print("L(t): ",l)

print("WQ(t): ",wq)

print("W(t): ",w)

print("Loader Queue: ",loader\_queue)

print("Weigh Queue: ",weigh\_queue)

print("Future Event List: ",fel)

print("BL: ",bl)

print("BS: ",bs)

print("----------------------------------------------------------")

while (i<len(load) or j<len(weigh) or k<len(travel)):

min\_ind = 0

for event\_ind in range(len(fel)):

if fel[event\_ind][1] < fel[min\_ind][1]:

min\_ind = event\_ind

bl += (fel[min\_ind][1] - clock) \* l

bs += (fel[min\_ind][1] - clock) \* w

event = fel[min\_ind][0]

clock = fel[min\_ind][1]

truck = fel[min\_ind][2]

fel.pop(min\_ind)

if event == 'EL':

loaders.remove(truck)

l -= 1

if len(loader\_queue) != 0:

next = loader\_queue.pop(0)

lq -= 1

loaders.append(next)

l += 1

if scale == -1:

scale = truck

else:

weigh\_queue.append(truck)

wq += 1

elif event == 'EW':

travelling.append(truck)

scale = -1

w -= 1

if len(weigh\_queue) != 0:

next = weigh\_queue.pop(0)

wq -= 1

scale = next

w +=1

elif event == 'ALQ':

travelling.remove(truck)

if len(loaders) < 2:

loaders.append(truck)

l += 1

else:

loader\_queue.append(truck)

lq += 1

for loader in loaders:

if loader not in (item for sublist in fel for item in sublist):

fel.append(['EL',clock+load[i],loader])

i += 1

if scale not in (item for sublist in fel for item in sublist):

fel.append(['EW',clock+weigh[j],scale])

j += 1

for traveller in travelling:

if traveller not in (item for sublist in fel for item in sublist):

fel.append(['ALQ',clock+travel[k],traveller])

k += 1

print("Clock: ",clock)

print("LQ(t): ",lq)

print("L(t): ",l)

print("WQ(t): ",wq)

print("W(t): ",w)

print("Loader Queue: ",loader\_queue)

print("Weigh Queue: ",weigh\_queue)

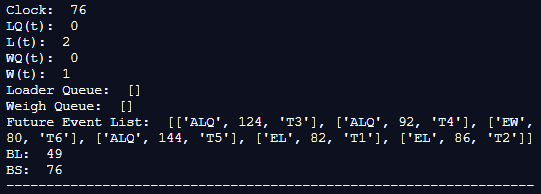
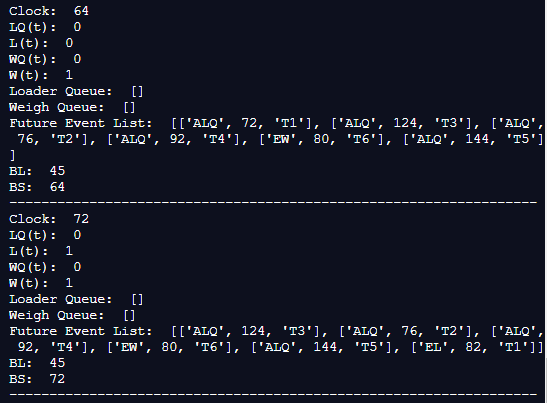
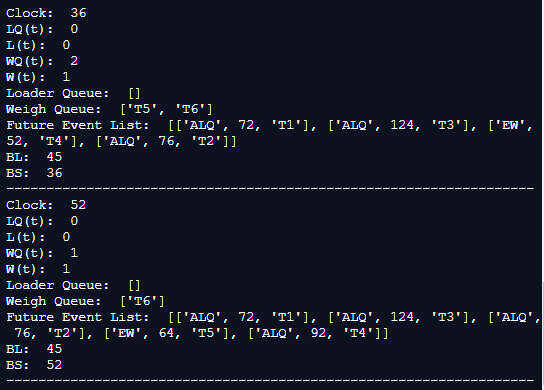
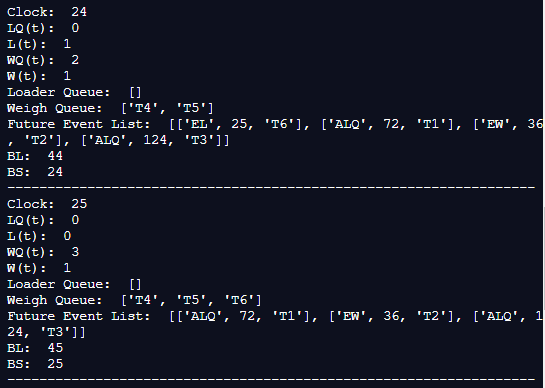
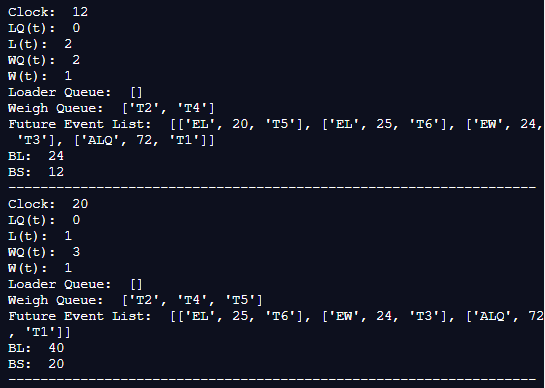
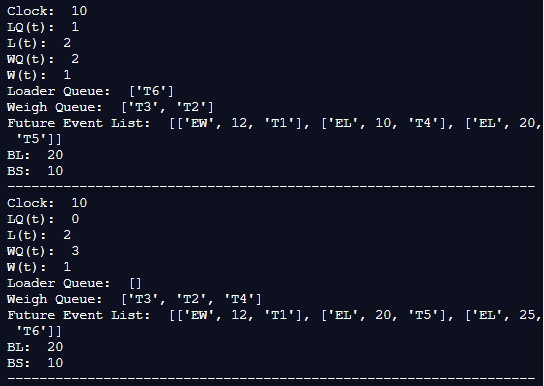
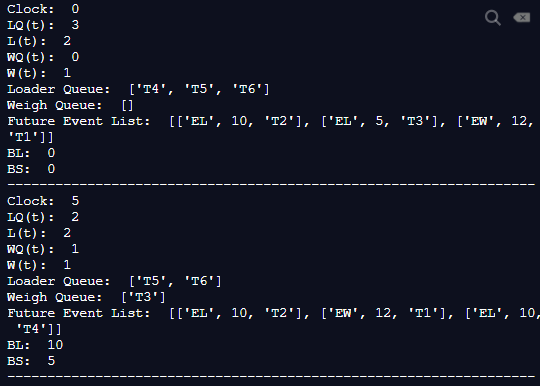
print("Future Event List: ",fel)

print("BL: ",bl)

print("BS: ",bs)

print("---------------------------------------------------------")

**Output:**



**Conclusion:**

We successfully understood & implemented the concept of Event Scheduling Algorithm for Dump Truck Problem in python.