In this project, you are to use binomial heaps to implement a priority queue in a preemptive scheduling system. A preemptive scheduling mechanism, shown in Fig. 1, operates as follows:

Processes bound to run are put in a priority queue (in this project a binomial heap (BH)) by a priority mechanism (in this project a function $f(e_i, t_{arr}(i))$ of the execution time, e_i, and arrival time, $t_{arr}(i)$, of the process). Whenever the processor (μP) is available, it is allocated for the process with the highest priority waiting in BH to run. Any running process can use μP only for a limited span of time called a *time slice* or *quantum*, q. If the current process runs to completion within q, then the next process with the highest priority waiting in BH attains the right to use μP. Otherwise (i.e., if the process is not finished), it is *preempted* (i.e., it releases μP, its status saved for the next use of μP and μP is allocated to the next highest priority process in BH). This switch of the right of use of μP from one process to another is known as "context switching." The process that is preempted, is re-enqueued, after its new priority is calculated based on its new arrival time so as to reallocate μP later and run to completion. This flow of events iterates until there are no more processes left in BH waiting for μP.

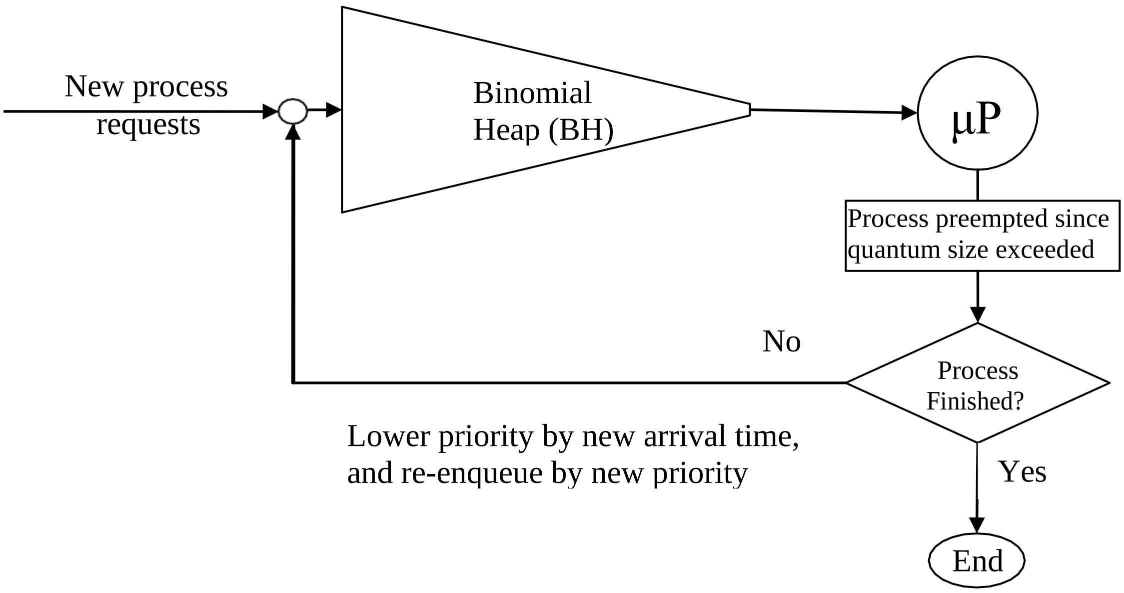

Figure. 1: Preemptive scheduling with a binomial queue

<u>What you are expected to do</u> is to

1. implement the preemptive scheduling system with BH as explained above (and shown in Fig. 1) and
2. optimize the quantum q; i.e., find the quantum value q, such that the average waiting time of BH (*AWT*) is minimized.

<u>Process criteria</u>

The waiting time per process i, WT_i, is defined by the sum of all spans of time passed from each arrival time t_{arr} of process i (assumed to be the time point process i is enqueued) until the time point t_{deq} it is dequeued (*DeleteMin*ed) and starts/continues running.

This is mathematically expressed as follows:

$$WT_i = \sum_{k=1}^{K_i} (t_{deq}(i,k) - t_{arr}(i,k))$$

where

- i is the process index,
- k is the index for the BH visits of process i and
- K_i denotes the number of times process i is enqueued (i.e., visited BH).

AWT, in turn, is the averaged sum of all individual waiting times, WT_i, or

$$AWT = \frac{1}{I}\sum_{i=1}^{I} WT_i = \frac{1}{I}\sum_{i=1}^{I}\sum_{k=1}^{K_i} t_{deq}((i,k) - t_{arr}(i,k))$$

with I the total number of processes under consideration over the analysis period.

<u>Input:</u>

1. You will be given a list of triplets (i, e_i, $t_{arr}(i)$) with i the process id, e_i, the execution time and $t_{arr}(i)$, the arrival time of process i. The sequence of triplets in the list specifies the order in which the processes request for processor allocation. You will use the input to compute the priority values $f(e_i, t_{arr}(i))$. These priority values will, in turn, determine the location of each process in BH.
2. Source code for a sample binomial heap is available for you. You may modify, compile and use it for your project.

<u>The computation of</u> $f(e_i, t_{arr}(i))$:

The piecewise continuous function below is used to compute the priority value of process i.

$$f(e_i, t_{arr}(i)) = \begin{cases} c(e_i) * e_i & for \quad e_i \neq e_j \\ t_{arr}(i) & otherwise \end{cases}$$

$$with \quad c(e_i) = \begin{cases} 1 & \text{, for first insertion to BH} \\ \dfrac{1}{\exp\left(-\left(\dfrac{2e_i}{3e_{\max}}\right)^3\right)} & \text{, for furtherinsertions to BH} \end{cases}$$

What this formula says is that, as long as the execution time of process i, e_i, is different than the execution time of any other process j, e_j, the priority of process i is its execution time. In case the execution times of two processes i and j are the same, then the arrival time $t_{arr}(i)$ of process i becomes its priority among those jobs with the same execution time e_i. If process i is inserted to BH for the first time, the factor $c(e_i)$=1. Otherwise (i.e., for further insertions), $c(e_i)$ is calculated using the above formula and multiplied to the <u>new</u> execution time, e_i^{new}, of process i (i.e., $(e_i^{new} = e_i^{pre} - q)$). Note as a programming detail that the original execution time should be stored in the node record. To determine the priority of processes i and j, their arrival times $t_{arr}(i)$ and $t_{arr}(j)$ are compared. The sooner the arrival of a process in BH, the higher its priority.

The factor of e_i, $c(e_i)$, when $e_i \neq e_j$, slightly diminishes the priority value of process *i* with respect to the remaining execution time of process *i* in case it is preempted before it can finish its execution and re-inserted into BH.

<u>Algorithm:</u>

- Initialize parameters such as q, e_{max};
- While there exist processes in the input list
  - Put the next process *i* arrived in μP
  - While μP allocated
    - Enqueue incoming processes by their priority
    - If $e_i > q$
      - Preempt current process
      - Reassign new priority
      - Re-insert process into BH
    - Else release μP
    - For each process *i* in BH
      - Update WT_i
    - *DeleteMin* (i.e., remove most prior process from BH)
    - Assign μP to this process
  - End of While
- End of While

One potential algorithm is given above for a specific quantum value. You may, but do not have to, use it.

To accomplish the minimization of *AWT* over a variety of quantum values, q, you have to iterate the whole run for many increasing/decreasing q values and store the corresponding *AWT* values. The q value providing the minimum *AWT* is the value we are looking for. This iteration is not considered in the algorithm given above.

A Sample Scenario:

The input file (PID, e, t_{arr}):

P1	3	0
P2	1	2
P3	2	3
P4	2	5
P5	2	6
P6	4	7

The allocation sequence of μP for different q values(**e_{max}=4**):

q=1

	0–1	1–2	2–3	3–4	4–5	5–6	6–7	7–8	8–9	9–10	10–11	11–12	12–13	13–14
μP	P1	P1	P2	P1	P3	P3	P4	P4	P5	P5	P6	P6	P6	P6

Time	Processes in BH	Priority value of processes in BH
0	P1	P1: 3
1	P1	P1: $(1/\exp-(2*2/3*4)^3)*2 = 2.075$
2	P1, P2	P1: $(1/\exp-(2*1/3*4)^3)*1 = 1.005$, P2: 1
3	P1, P3	P1: $(1/\exp-(2*1/3*4)^3)*1 = 1.005$, P3: 2
4	P3	P3: 2
5	P3, P4	P3: $(1/\exp-(2*1/3*4)^3)*1 = 1.005$, P4: 2
6	P4, P5	P4: 5, P5:6 (both have the same *e* value, so priority is t_{arr})
7	P4, P5,P6	P4: $(1/\exp-(2*1/3*4)^3)*1 = 1.005$, P5: 2,P6:4
8	P5, P6	P5: 2, P6: 4
9	P5, P6	P5: $(1/\exp-(2*1/3*4)^3)*1 = 1.005$, P6: 4
10	P6	P6: 4
11	P6	P6: $(1/\exp-(2*3/3*4)^3)*3 = 3.399$
12	P6	P6: $(1/\exp-(2*2/3*4)^3)*2 = 2.075$
13	P6	P6: $(1/\exp-(2*1/3*4)^3)*1 = 1.005$
14	EMPTY	

PID	Waiting time
P1	1
P2	0
P3	1
P4	1
P5	2
P6	3

AWT = 8/6 = 1.33

q=2

	0	2	3	4	6	8	10	12	14
μP	P1	P2	P1	P3	P4	P5	P6	P6	

Time	Processes in BH	Priority value of processes in BH
0	P1	P1: 3
2	P1, P2	P1: $(1/exp-(2*1/3*4)^{3})*1 = 1.005$, P2: 1
3	P1, P3	P1: $(1/exp-(2*1/3*4)^{3})*1 = 1.005$, P3: 2
4	P3	P3: 2
6	P4, P5	P4: 5, P5:6 (both have the same *e* value, so priority is t_{arr})
8	P5, P6	P5: 2, P6: 4
10	P6	P6: 4
12	P6	P6: $(1/exp-(2*2/3*4)^{3})*2 = 2.075$
14	EMPTY	

PID	Waiting time
P1	1
P2	0
P3	1
P4	1
P5	2
P6	3

AWT = 8/6 = 1.33

q=3

	0	3	4	6	8	10	13	14
μP	P1	P2	P3	P4	P5	P6	P6	

Time	Processes in BH	Priority value of processes in BH
0	P1	P1: 3
3	P2, P3	P2: 1, P3: 2
4	P3	P3: 2
6	P4, P5	P4: 5, P5:6 (both have the same *e* value, so priority is t_{arr})
8	P5, P6	P5: 2, P6: 4
10	P6	P6: 4
13	P6	P6: $(1/exp-(2*1/3*4)^{3})*1 = 1.005$
14	EMPTY	

PID	Waiting time
P1	0
P2	1
P3	1
P4	1
P5	2
P6	3

AWT = 8/6 = 1.33