2019A7PS0020U Dhruv Maheshwari

Lab 4

Function	Big O

int*linear_queue;
int linear_queue_size;
int*circular_queue;
int circular_queue_size;
int opCode;
int frontL=0, rearL=0;
int frontC=0, rearC=0;
	O(1) [declaration]
	O(1)

int isLinearQueueEmpty()
{
if(frontL==rearL)
{
return 1;
}
else return 0;
}
	O(1)

int isLinearQueueFull()
{
if(rearL==linear_queue_size)
{
return 1;
}
else return 0;
}
	O(1)

int linearEnqueue(int x)
{
if(rearL==linear_queue_size)
{
return -1;
}
else {
linear_queue[rearL]=x;
rearL=rearL+1;
return 1;}
}
	O(1)[comparison, return]
	O(1)[index, assignment]
	O(1)[arithmetic, assignment]
	O(1)

int linearDequeue()
{
int x;
if(frontL==rearL)
{
return -1;
}
else {
x=linear_queue[frontL];
frontL=frontL+1;
return 1;}
}
	O(1)[Declaration]
	O(1)[comparison, return]
	O(1)[index, assignment]
	O(1)[arithmetic, assignment]
	O(1)

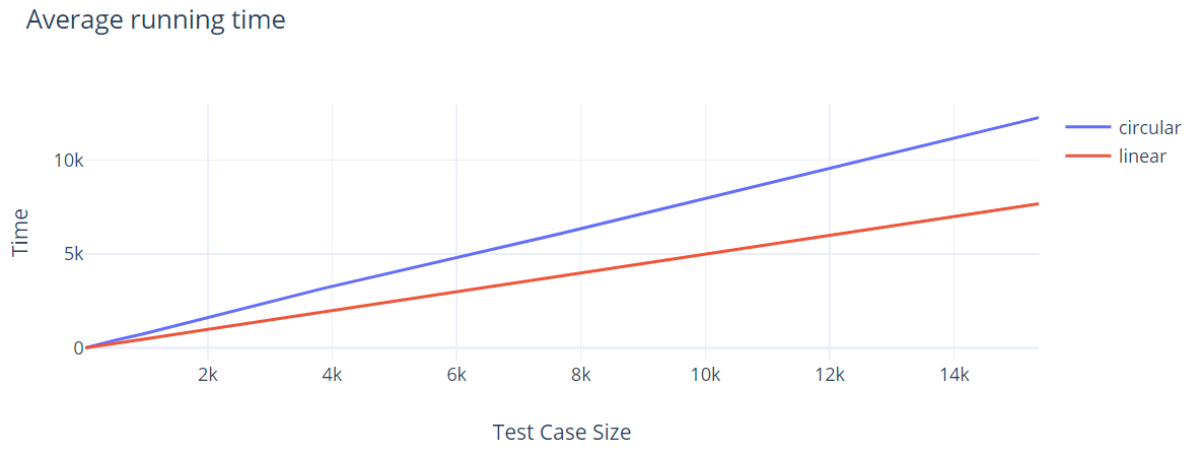
int isCircularQueueEmpty()
{
if(frontC==rearC)
{
return 1;
}
else return 0;
}
	O(1)[comparison, assignment]
	
	O(1)

int isCircularQueueFull()
{
if((frontC==0 && rearC==circular_queue_size-1) || (rearC == frontC-1))
{ return 1; }
else return 0;
}
	O(1)[comparison, logical operator, return]
	
	O(1)

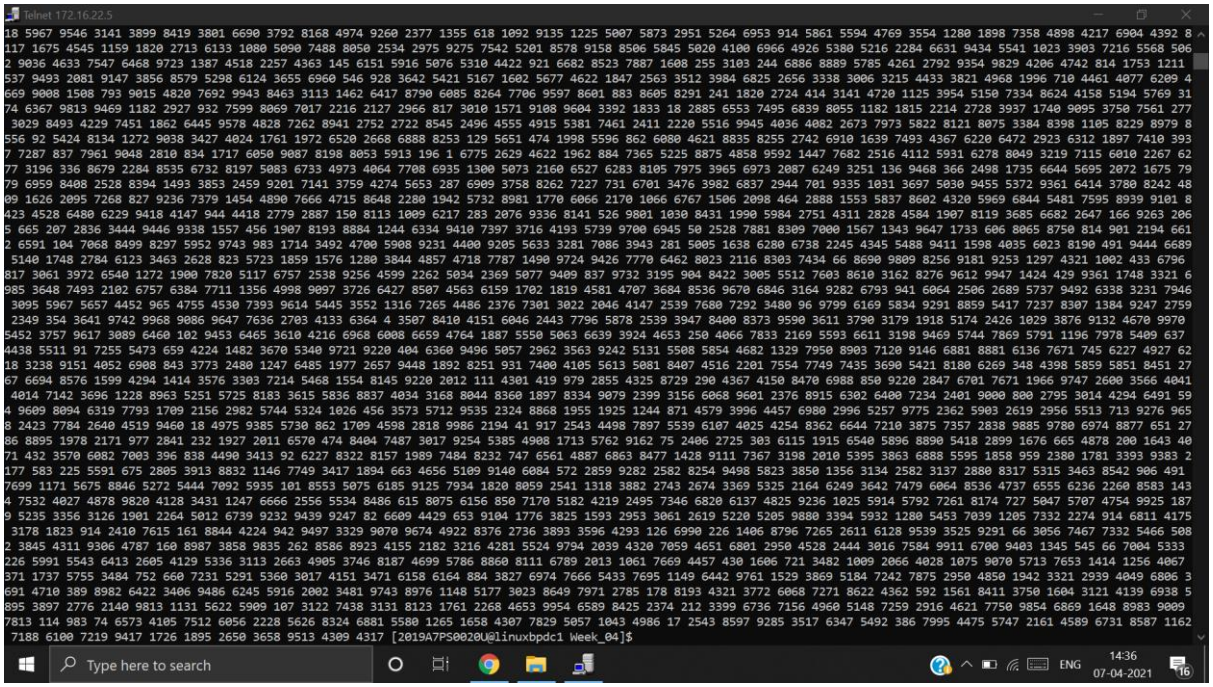
int circularEnqueue(int x)
{
if(isCircularQueueFull())
{
return -1;
}
else
{ circular_queue[rearC]=x;
rearC=(rearC+1)%(circular_queue_size);
return 1;
}
}
	O(1)[function, return]
	
	O(1)[index, assignment]
	O(1)[arithmetic, assignment]
	
	O(1)

int circularDequeue()
{
int x;
if(isCircularQueueEmpty())
{ return -1;
}
else
{
x=circular_queue[frontC];
if(frontC==rearC)
{
frontC=0;
rearC=0;
}
else{ frontC=(frontC+1)%circular_queue_size;
}
return x;
}
}
	O(1)[Declaration]
	O(1)[function, return]
	
	O(1)[index, assignment]
	O(1)[comparison]
	
	O(1)[assignment]
	
	O(1)[arithmetic, assignment]
	
	O(1)

2019A7PS0020U Dhruv Maheshwari



Ex



Function	Big O

void load(int array[],int size)
{
int index=1;
int length=size;
enQueue(array[1]);
while(size>1)
{
if(2*index > length || 2*index+1> length)
{
break;
}
if(array[2*index]!=-1)
{
enQueue(array[2*index]);
	
	O(1)[declaration, assignment]
	O(1)[declaration, assignment]
	O(1)
	O(n-1)
	
	O[1*(n-1)](arithmetic, comparison, or]
	
	O[1*(n-1)](arithmetic, index, comparison]
	
	O(1)

}
if(array[(2*index) + 1]!=-1)
{
enQueue(array[(2*index)+1]);
}
index++;
size--;
}
}
	
	O[1*(n-1)](arithmetic, index, comparison]
	
	O(1)
	
	O(1)[increment]
	O(1)[decrement]
	
	O(n)