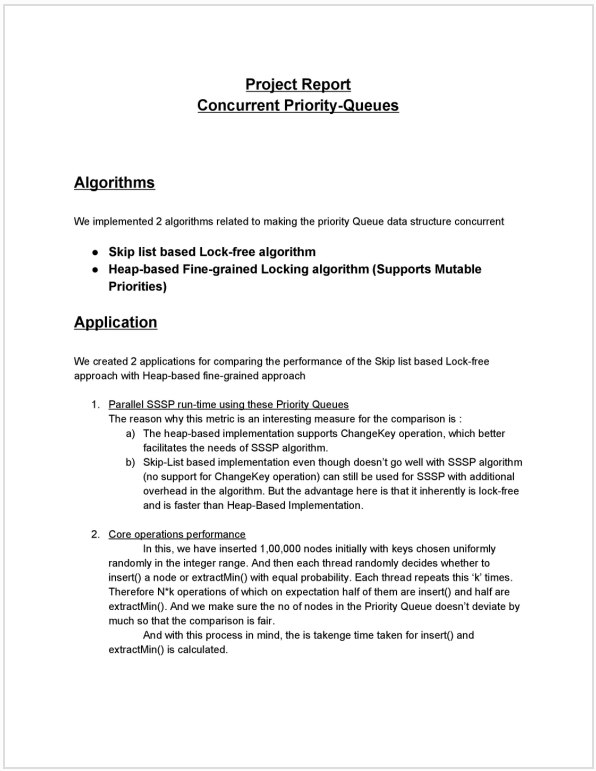
Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	Bhanu Prakash
Assignment title:	Final Course Project
Submission title:	Concurrent Priority Queues
File name:	File size:335.13K
Page count:	7
Word count:	871
Character count:	4,791
Submission date:	29-Apr-2019 10:03AM (UTC+0530)
Submission ID:	1121046498



Project Report
Concurrent Priority-Queues

Algorithms

We implemented 2 algorithms related to making the priority Queue data structure concurrent

- **Skip list based Lock-free algorithm**
- **Heap-based Fine-grained Locking algorithm (Supports Mutable Priorities)**

Application

We created 2 applications for comparing the performance of the Skip list based Lock-free approach with Heap-based fine-grained approach

1. Parallel SSSP run-time using these Priority Queues
   The reason why this metric is an interesting measure for the comparison is :
   a) The heap-based implementation supports ChangeKey operation, which better facilitates the needs of SSSP algorithm.
   b) Skip-List based implementation even though doesn't go well with SSSP algorithm (no support for ChangeKey operation) can still be used for SSSP with additional overhead in the algorithm. But the advantage here is that it inherently is lock-free and is faster than Heap-Based Implementation.

2. Core operations performance
   In this, we have inserted 1,00,000 nodes initially with keys chosen uniformly randomly in the integer range. And then each thread randomly decides whether to insert() a node or extractMin() with equal probability. Each thread repeats this 'k' times. Therefore N*k operations of which on expectation half of them are insert() and half are extractMin(). And we make sure the no of nodes in the Priority Queue doesn't deviate by much so that the comparison is fair.
   And with this process in mind, the is takenge time taken for insert() and extractMin() is calculated.