|  |
| --- |
| CIS 350 – Data Structures |
| Program 2 – Turn in 1 |
| Fall 2014 |

|  |
| --- |
| Daniel Frey  10-29-2014 |

Contents

[1. Problem Summary 2](#_Toc402375220)

[2. Requirements Document 2](#_Toc402375221)

[2.1. Purpose 2](#_Toc402375222)

[2.2. Product Scope 2](#_Toc402375223)

[2.3. Project Scope 2](#_Toc402375224)

[2.4. Assumptions 2](#_Toc402375225)

[3. Decomposition Diagram 3](#_Toc402375226)

[4. Order 4](#_Toc402375227)

[5. Testing Strategy 4](#_Toc402375228)

[6. Test Plan – Version 1 5](#_Toc402375229)

[7. Initial Algorithm 6](#_Toc402375230)

[8. Test Plan – Version 2 7](#_Toc402375231)

# Problem Summary

The program will simulate a processor using a binary heap priority queue, which takes a series of random jobs with their own arrival and processing time.

# Requirements Document

## Purpose

The purpose of this document is to provide the functionality required for a simulation of a processor with a binary heap priority queue.

## Product Scope

* Jobs to be processed will arrive into the system on a random basis
* A processor will process the jobs
* Uncompleted jobs will be placed in priority queue
* Create and report set of metrics for performance
  + Determine optimal number of processors based on metrics

## Project Scope

* Priority queue implemented as binary heap
  + Based on processing time
* Jobs classified as regular or high priority
  + Regular jobs
    - Arrival time 5 ± 1, processing time 3 ± 2
    - Arrival time 10 ± 1, processing time 8 ± 2
    - Arrival time 25 ± 1, processing time 13 ± 2
  + High priority jobs
    - Arrival time 30 ± 5, processing time 10 ± 2
* Jobs run to completion unless interrupted by high priority job
  + Interrupted jobs get placed back into queue
* Simulation runs 500 time unites to pre-fill system with jobs
  + Report queue size
* Run for an additional 9500 time units
  + Report final metrics

## Assumptions

* One job per category may come in at a time in the arrival interval
* Arrival times for jobs will happen in multiples of the base arrival time, e.g. 5 ± 1 will occur in multiples of 5 (5 ± 1, 10 ± 1, 15 ± 1…)

# Decomposition Diagram

# Order

1. Queue/heap
   1. Insert
   2. Delete
2. Generate jobs
   1. Generate random arrival time
   2. Generate random processing time based on arrival time
3. Processor
   1. Run jobs
4. Interrupt jobs
   1. Insert back into queue/heap

# Testing Strategy

* Pre-fill
* Interrupted job
* Generated job
* Simulation

# Test Plan – Version 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Strategy Category** | **Test Number** | **Description** | **Input** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| Pre-fill | 1.1 | Don’t count metrics for pre-fill |  |  |  |  |
| Pre-fill | 1.1 | Queue size reported |  |  |  |  |
| Interrupted job | 2.1 | Interrupted jobs are not counted as completed |  |  |  |  |
| Interrupted job | 2.2 | Interrupted jobs are put back in queue |  |  |  |  |
| Generated job | 3.1 | A job generated has correct random arrival time |  |  |  |  |
| Generated job | 3.2 | A generated job has correct corresponding processing time |  |  |  |  |
| Simulation | 4.1 | Job correctly transfers over to simulation queue |  |  |  |  |
| Simulation | 4.2 | Metrics are counted |  |  |  |  |
| Simulation | 4.3 | Regular job does not interrupt high priority job |  |  |  |  |

# Initial Algorithm

* Create a class structure for minimum binary heap priority queue
  + Use array to store elements in binary heap structure
    - Min at root (index 1)
    - Left child at 2i, right child at 2i+1, parent at i/2
  + Insert
    - Create hole at end
    - Insert element
    - If parent is greater than inserted, swap
      * Continue until no longer greater than
  + Delete
    - Copy last element to root
    - Delete last element
    - If child is less than, then swap with minimum child
  + Size
    - Return size of queue
  + Minimum
    - Return smallest element
* Generate random jobs
  + Using random function generate arrival and processing times for
    - * Regular jobs - 5±1/3±2, 10±1/8±2, 25±1/13±2
      * High priority jobs - 30±5/10±2
    - From time 0 to end time
      * Add ticking time to generated random base times
      * Add created job to a queue sorted by minimum time
* Use a loop to simulate a processor
  + - Each iteration is one time unit
    - Processor
      * Add job to processor
      * Remove job from processor
      * Run job for processing time +1
      * Count finished job when time = 0
  + Compare time root of arrival time job queue
    - If same time, add to processor queue by processing time
    - Delete job off arrival queue
    - Count incoming job and type
  + If processor is empty
    - Add job to be processed
  + If high priority job
    - Stop current processing job, put in queue based on remaining processing time
    - Add high priority job to processor to start processing
    - Count interrupted job
  + If processor has job
    - Add incoming job to queue
    - Count wait time on queue

# Test Plan – Version 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Strategy Category** | **Test Number** | **Description** | **Input** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| Pre-fill | 1.1 | Don’t count metrics for pre-fill | Jobs | None |  |  |
| Pre-fill | 1.1 | Queue size reported | Jobs | Queue size |  |  |
| Interrupted job | 2.1 | Interrupted jobs are not counted as completed | Higher priority job | Lower priority job stopped at current time |  |  |
| Interrupted job | 2.2 | Interrupted jobs are put back in queue | Regular job | Regular job placed back in queue |  |  |
| Interrupted Job | 2.3 | High priority interrupts regular priority | High priority job | High priority job begins processing, regular put in queue |  |  |
| Generated job | 3.1 | A job generated has correct random arrival time | Times for random arrival time | Arrival time based on specifications |  |  |
| Generated job | 3.2 | A generated job has correct corresponding processing time | Times for random processing time | Processing time based on specifications |  |  |
| Simulation | 4.1 | Job transfers over to simulation queue | Job | Simulation processing time queue |  |  |
| Simulation | 4.2 | Metrics are counted | Simulation of processor | Counts on data |  |  |
| Simulation | 4.3 | Regular job does not interrupt high priority job | Regular job | Regular job added to queue |  |  |