Santa’s Workshop

Description

In this project, you will get more familiar with the concepts of scheduling, synchronisation, multi-threading and deadlock prevention in operating systems by using POSIX threads (pthreads) API.

Santa’s workshop

You have been hired by Santa to verify whether he can complete his job this Christmas with only 2 employee elves left due to economical crisis. Your job is to implement a safe and deadlock-free simulator of his workshop. His workshop operates as shown on the image below. There are 3 types of gifts: just chocolate, toy and a chocolate and for the best kids a one with also a GameStation on top. Every present needs to be packaged by one of the elves and delivered by Santa. Santa’s workshop is unable to produce anymore, so all the gifts are manufactured in China. Toys require additional customization, and due to being ex- pensive, GameStations need to pass quality assurance (QA) by Santa as shown on the image.

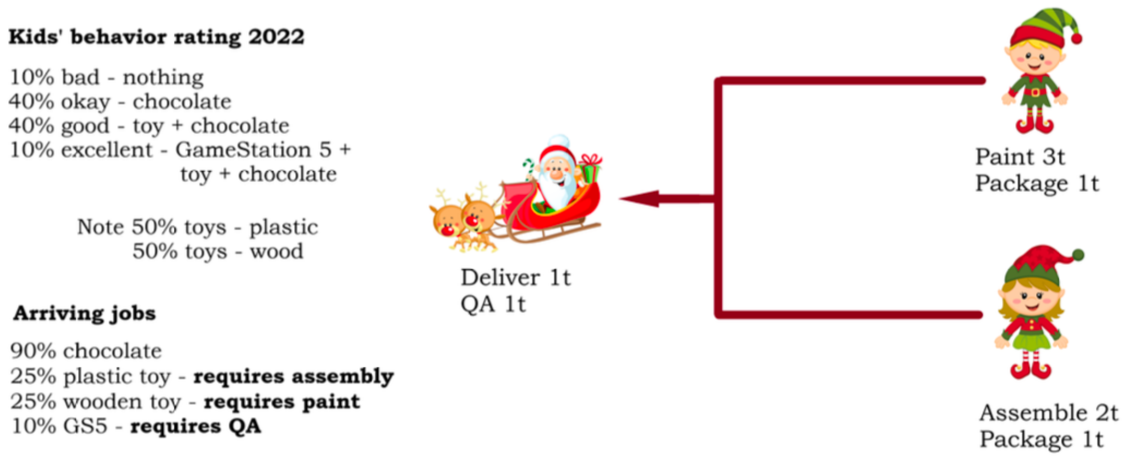
Kids' behavior rating 2022
10% bad - nothing
40% okay - chocolate
40% good - toy + chocolate
10% excellent - GameStation 5 + toy + chocolate
Note 50% toys - plastic
50% toys - wood
Arriving jobs
90% chocolate
25% plastic toy - **requires assembly**
25% wooden toy - **requires paint**
10% GS5 - **requires QA**
Deliver 1t
QA 1t
Paint 3t
Package 1t
Assemble 2t
Package 1t

Figure 1: Santa’s workshop

Here is how the workshops simulation is going to work. Every t secs (it is assumed to be 1 sec) there is a 90% probability that a gift request arrives. From the tables on Figure 1 you can see that every t seconds

1. There is a 40% chance that a gift (containing only a chocolate) will only need packaging, followed by delivery.
2. There is a 20% chance that a gift (containing a wooden toy and a chocolate) will require painting, followed by packaging, followed by delivery.
3. There is a 20% chance that a gift (containing a plastic toy and a chocolate) will require assembly, followed by packaging, followed by delivery.
4. There is a 5% chance that a gift (containing a GS5, wooden toy and a chocolate) will require painting and QA, followed by packaging, followed by delivery.
5. There is a 5% chance that a gift (containing a GS5, plastic toy and a chocolate) will require assembly and QA, followed by packaging, followed by delivery.

The same information can be represented in a form of Directed Acyclic Graphs (DAG). You can see from them, that in gifts of type 1, 2 and 3 all tasks are strictly one after the other. In those of type 4 and 5, however, QA can be done by Santa at the same time as Painting or Assembly is done by one of the elves. Note that Packaging only has to start after both of the previous tasks are completed.

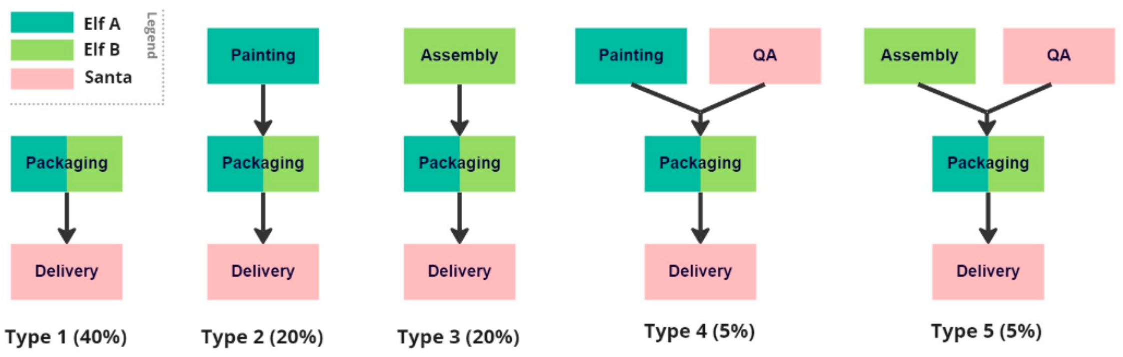

Figure 2: Task DAGs for every gift type

Here are more detailed rules for the simulation environment.

- The simulation should use real-time. Get the current time of the day, run the simulation until current time + simulation time.
- Elf A can paint wooden toys or package gifts. Elf B can assemble plastic toys and package gifts as well. Santa handles delivery of packaged gifts and quality check of GameStations.
- There is only one job being done by each worker at a time.
- To reduce the waiting time of gifts, elves always prioritize packaging tasks, while Santa prioritizes delivery.
- Each painting task takes 3t secs to complete. Assembly takes 2t secs, packaging, delivery and QA take 1t secs.

Assume t is 1 sec for the purpose of the computer simulation.

Implementation

- The starter code given to you outlines a possible structure of your project, but you are free to make changes.
- You may want to keep a queue for each task type, add tasks to queues at the appropriate times. The queue code is provided to you, but you can edit it if you want to, especially the Task objects.
- You can use random number generator to generate gifts at the specified probability. For easy debugging, use the same seed.
- You should represent each worker as a thread. Each worker's thread will sleep for a given amount of time to simulate a task being performed.

Output the snapshot of the waiting tasks with their IDs in every second starting from nth secs to the terminal, where n is also a command line argument. The numbers indicates the task IDs waiting in the queue at time n. Feel free to come up with a better representation or GUI.