



دانشگاه صنعتی شریف

دانشکده‌ی مهندسی صنایع

گزارش پروژه‌ی شبیه‌سازی گسسته پیشامد رستوران

فاز دوم

استاد درس

دکتر نفیسه صدقی

دستیاران درس

روزبه آذرگشاسبی

حمیدرضا محمدیها

نگارش

حامد علی اکبری

هستی قادرآزاد

بهار ۱۳۹۹

فهرست

۱	شرح سیستم شبیه سازی
۲	مدل سازی
۲	توصیفات استاتیک مدل
۲	متغیرهای حالت
۴	نهادها
۴	پیشامدها
۴	اعلام پیشامد
۲	فعالیت ها
۲	تأخیر
۲	آماره های تجمعی
۲	ساعت
۲	ساختار بندی لیست پیشامدهای آتی
۲	فرضیات و ساده سازی ها
۲	معیارهای ارزیابی عملکرد سیستم
۳	معیارهای ارزیابی عملکرد مرتبه اول
۴	شاخص های مرتبه ی دوم
۷	شاخص های منتخب جهت ارزیابی سیستم شبیه سازی شده
۷	توصیف پویا
۲۱	شبیه سازی گسسته پیشامد رستوران
۲۲	تحلیل حساسیت
۲۲	تغییر پارامتر تعداد کارمندان بخش پذیرش

۲۴	تغییر پارامتر تعداد کارمندان بخش آشپزخانه
۲۷	تغییر پارامتر تعداد صندلی‌های سالن غذاخوری
۲۹	تعیین برآورد فاصله‌ای و نقطه‌ای
۳۰	بررسی وجود و شناسایی دوره‌ی سرد و گرم سیستم
۳۱	پیوست اول: فایل اکسل
۳۲	پیوست دوم: کد پایتون
۴۸	منابع و مآخذ

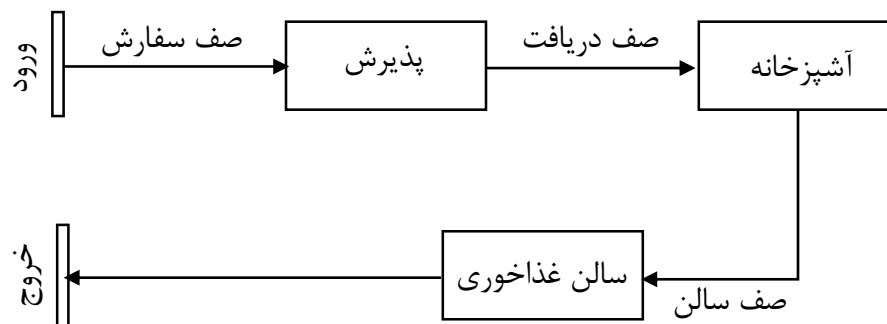
شرح سیستم شبیه سازی

- هدف از این مسئله، شبیه سازی یک رستوران فست فود است. اطلاعات مربوط به صورت زیر ارائه گردیده است:
- طول مدت شبیه سازی از ساعت ۱۰ صبح تا ۳ بعد از ظهر است. افراد به سه صورت به رستوران می رسند:
 - پیاده، به وسیله ی خودرو و اتوبوس.
- پیاده ها با فاصله های زمانی که از توزیع نمایی منفی با میانگین ۳ دقیقه پیروی می کند، وارد رستوران می شوند. اولین شخص با توزیع مذکور بعد از ساعت ۱۰ وارد رستوران می شود.
- تعداد سرنشینان هر خودرو ۱، ۲، ۳ یا ۴ نفر است که احتمال هر یک به ترتیب ۰٫۲، ۰٫۳، ۰٫۳، ۰٫۲ است.
- فاصله های زمانی بین دو ورود خودرو از توزیع نمایی منفی با میانگین ۵ دقیقه پیروی می کند. اولین خودرو در زمان با توزیع مذکور بعد از ساعت ۱۰ وارد رستوران می شود.
- یک دستگاه اتوبوس بین ساعت ۱۰ تا ۳ ظهر به رستوران می رسد که توزیع زمان رسیدن آن به طور یکنواخت است. تعداد سرنشینان این اتوبوس از توزیع پواسون با میانگین ۳۰ نفر تبعیت می کند.
- زمانی که شخصی وارد رستوران می شود، مستقل از نوع ورودش، به سمت قسمت سفارش غذا حرکت می کند. در این بخش مشتری ابتدا غذا را سفارش می دهد که مدت زمان این فعالیت به صورت توزیع مثلی با پارامترهای ۱-۲-۴ است. بلافاصله بعد از سفارش غذا مشتری پول را می پردازد که این فرایند نیز از تابع مثلی با پارامترهای ۱-۲-۳ پیروی می کند.
- پس از سفارش غذا، مشتری به قسمت دریافت غذا می رود که در این قسمت غذای وی طی زمانی که از توزیع یکنواخت بین ۳۰ ثانیه تا ۲ دقیقه برخوردار است، حاضر می شود.
- پس از دریافت غذا، مشتری به سمت سالن غذاخوری حرکت می کند. در سالن غذاخوری ۳۰ عدد صندلی وجود دارد. مدت زمان صرف غذا از توزیع مثلی با پارامترهای ۱۰-۲۰-۳۰ برخوردار است. پس از صرف غذا مشتری به طرف درب خروجی رستوران حرکت می کند.

- مدت زمان حرکت بین هر یک از بخش‌ها از توزیع نمایی منفی با میانگین ۳۰ ثانیه برخوردار است. سیستم صف در هر یک از بخش‌ها از قانون فایفو تبعیت می‌کند. بعد از صرف غذا، مشتری با سرعت کمتری که تابع توزیع آن نمایی با میانگین ۱ دقیقه است، رستوران را ترک می‌کند.
- خدمت‌دهندگان دو میز پذیرش و دریافت غذا، دارای چند زمان استراحت هستند که به طور تصادفی بینشان تقسیم می‌شود. به طور دقیق‌تر، در ساعت‌های ۱۰:۵۰، ۱۱:۵۰، ۱۳:۵۰ و ۱۴:۵۰ از هر میز یک نفر به استراحت می‌رود و پس از ۱۰ دقیقه برمی‌گردد. اگر خدمت‌دهنده در زمان استراحت در حال کار باشد، پس از اتمام سرویس به استراحت خواهد رفت. تعداد خدمت‌دهنده‌ها در میز پذیرش ۵ و در میز دریافت غذا ۲ است.

مدل‌سازی

مدل‌سازی مساله در قالب شبیه‌سازی به همراه معرفی پیشامدها، فعالیت‌ها و متغیرهای حالت (توصیف ایستا).



توصیفات استاتیک مدل

در این بخش به معرفی اجزای مدل شبیه‌سازی می‌پردازیم.

متغیرهای حالت^۱

- طول صف سفارش (LQ_0)
- این متغیر نشان می‌دهد در زمان t چند نفر در صفی که برای سفارش غذا و پرداخت پول تشکیل شده‌است منتظر هستند. این متغیر با $LQ_0(t)$ نمایش داده می‌شود.

^۱ State variables

- طول صف دریافت غذا (LQr)
این متغیر نشان می‌دهد در زمان t چند نفر در صفی که برای دریافت غذا در قسمت آشپزخانه تشکیل شده است منتظر هستند. این متغیر با $LQr(t)$ نمایش داده می‌شود.
- تعداد صندلی‌های خالی (ES)
این متغیر تعداد صندلی‌های خالی در سالن در زمان t را نشان می‌دهد و به صورت $ES(t)$ نمایش داده می‌شود.
- طول صف انتظار سالن (LQc)
این متغیر نشان می‌دهد در زمان t چند نفر در صفی که برای نشستن بر روی صندلی در سالن تشکیل شده است منتظر هستند. این متغیر با $LQc(t)$ نمایش داده می‌شود.
- تعداد کارمند در میز پذیرش در حال استراحت (nRr)
این متغیر یک متغیر 0 و 1 است که اگر فردی از پرسنل پذیرش در حال استراحت باشد مقدار 1 به آن تعلق می‌گیرد و با $nRr(t)$ نمایش داده می‌شود.
- تعداد کارمند در آشپزخانه در حال استراحت (nRk)
این متغیر یک متغیر 0 و 1 است که اگر فردی از پرسنل آشپزخانه و تحویل غذا در حال استراحت باشد مقدار 1 به آن تعلق می‌گیرد و با $nRk(t)$ نمایش داده می‌شود.
- تعداد کارمند پذیرش آزاد (Lr)
این متغیر نشان می‌دهد در زمان t چه تعداد از کارمندان بخش پذیرش آزاد هستند و با $Lr(t)$ نمایش داده می‌شود.
- تعداد کارمند آشپزخانه آزاد (Lk)
این متغیر نشان می‌دهد در زمان t چه تعداد از کارمندان بخش آشپزخانه آزاد هستند و با $Lk(t)$ نمایش داده می‌شود.
- تعداد کارمند پذیرش در انتظار استراحت (WRr)
هنگامی که وقت استراحت یک کارمند پذیرش فرا می‌رسد اما مشغول به کار است، بعد از اتمام کارش باید به استراحت برود. این متغیر تعداد کارمندان پذیرش که در زمان t دچار چنین وضعیتی هستند را نشان می‌دهد و مقدار 0 یا 1 می‌گیرد و با $WRr(t)$ نشان داده می‌شود.
- تعداد کارمند آشپزخانه در انتظار استراحت (WRk)
هنگامی که وقت استراحت یک کارمند آشپزخانه فرا می‌رسد اما مشغول به کار است، بعد از اتمام کارش باید به استراحت برود. این متغیر تعداد کارمندان آشپزخانه که در زمان t دچار چنین وضعیتی هستند را نشان می‌دهد و مقدار 0 یا 1 می‌گیرد و با $WRk(t)$ نشان داده می‌شود.
- تعداد مشتریان حاضر در رستوران (LS)
این متغیر تعداد کل مشتری‌های حاضر در بخش‌های مختلف رستوران در زمان t را نشان می‌دهد و با $LS(t)$

نمایش داده می‌شود.

نهادها^۲

- مشتریان
- کارمندان بخش پذیرش
- کارمندان بخش آشپزخانه

پیشامدها^۳

- ورود به رستوران به صورت پیاده و شروع سفارش‌دهی (Aw)
- ورود به رستوران به وسیله ماشین و شروع سفارش‌دهی (Ac)
- ورود به رستوران به وسیله اتوبوس و شروع سفارش‌دهی (Ab)
- اتمام سفارش‌دهی و پرداخت (rE)
- شروع فرایند دریافت غذا (Er)
- دریافت غذا (R)
- شروع صرف غذا (E)
- پایان صرف غذا (En)
- خروج از رستوران (Ex)
- شروع استراحت کارمندان پذیرش (Srr)
- اتمام استراحت کارمند پذیرش (Err)
- شروع استراحت کارمندان آشپزخانه (Srk)
- اتمام استراحت کارمند آشپزخانه (Erk)
- پیشامد اتمام شبیه‌سازی (End)

اعلام پیشامد^۴

- | | | | |
|--------|---|--------|---|
| (rE,t) | • | (Ac,t) | • |
| (Er,t) | • | (Ab,t) | • |
| (R,t) | • | (Aw,t) | • |

Entities^r

Events^r

Event notice^f

- (Srr,t) •
- (Err,t) •
- (Srk,t) •
- (End,t) •
- (Erk,t) •
- (E,t) •
- (Ex,t) •
- (En,t) •

فعالیت‌ها

- زمان بین ورود مشتری پیاده (WInter)
- زمان بین ورود دو مشتری در حالت پیاده را نشان می‌دهد و از توزیع نمایی منفی با میانگین ۳ دقیقه پیروی می‌کند.
- زمان بین ورود مشتری با ماشین (CInter)
- زمان بین ورود دو ماشین را نشان می‌دهد و از توزیع نمایی منفی با میانگین ۵ دقیقه پیروی می‌کند.
- زمان بین ورود مشتری با اتوبوس (BInter)
- زمان بین ورود دو اتوبوس را نشان می‌دهد و از توزیع یکنواخت پیروی می‌کند.
- سفارش غذا (Ost)
- مدت زمانی که سفارش غذا به صورت مستقل طول می‌کشد و از توزیع مثلی با پارامترهای ۴-۲-۱ پیروی می‌کند.
- پرداخت پول (Pay)
- مدت زمانی که فرآیند پرداخت پول به صورت مستقل طول می‌کشد و از توزیع مثلی با پارامترهای ۳-۲-۱ پیروی می‌کند.
- سرویس‌دهی بخش آشپزخانه_دریافت غذا (Kst)
- مدت زمانی که طول می‌کشد غذای افراد حاضر شود که از توزیع یکنواخت پیروی می‌کند.
- مدت زمان صرف غذا (eat)
- مدت زمانی که طول می‌کشد مشتریان غذای خود را صرف کنند که از توزیع مثلی با پارامترهای ۲۰-۱۰-۳۰ پیروی می‌کند.
- حرکت از بخش پذیرش به سمت آشپزخانه (t1)
- مدت زمانی که پیمودن مسافت بین بخش پذیرش و آشپزخانه طول می‌کشد که از توزیع نمایی منفی با میانگین ۳۰ ثانیه برخوردار است.
- حرکت از بخش آشپزخانه به سمت سالن غذاخوری (t2)
- مدت زمانی که پیمودن مسافت بین آشپزخانه و سالن طول می‌کشد که از توزیع نمایی منفی با میانگین ۳۰ ثانیه برخوردار است.

- حرکت به سمت خروج از رستوران (t_3)
- مدت زمانی که پیمودن مسافت بین سالن و خروجی طول می کشد که از توزیع نمایی با میانگین ۱ دقیقه برخوردار است.
- استراحت کارمند پذیرش (R_r)
- ۱۰ دقیقه
- استراحت کارمند آشپزخانه (R_k)
- ۱۰ دقیقه

تأخیر^۵

- انتظار در صف پذیرش
 - مدت زمان انتظار مشتری تا خالی شدن سرور پذیرش
 - انتظار در صف دریافت غذا
 - مدت زمان انتظار مشتری تا خالی شدن سرور آشپزخانه
 - انتظار در صف سالن غذاخوری
 - مدت زمان انتظار مشتری تا خالی شدن صندلی
 - انتظار برای استراحت کارمند بخش پذیرش
 - مدت زمان بین ساعتی که کارمند بخش پذیرش به استراحت می رود و ساعت استانداری که برای استراحت در نظر گرفته شده بود. منظور از ساعت استاندارد ۱۰:۵۰، ۱۱:۵۰، ۱۳:۵۰ و ۱۴:۵۰ است.
 - انتظار برای استراحت کارمند بخش آشپزخانه
 - مدت زمان بین ساعتی که کارمند بخش آشپزخانه به استراحت می رود و ساعت استانداری که برای استراحت در نظر گرفته شده بود.
- آماره های تجمعی

آماره های تجمعی

- در راستای انجام شبیه سازی نیازمند تعریف برخی از آماره های تجمعی هستیم که باید از آنها در اجرای فرایند استفاده کنیم. ذکر این نکته ضروری است که تمامی آماره های زیر تجمعی هستند و به این همین علت، جهت پیشگیری از تکرار، از آوردن واژه ی تجمعی در کنار آنها پرهیز شده است. این آماره ها عبارت اند از:
- طول صف انتظار پذیرش
 - طول صف سالن صرف غذا
 - طول صف دریافت غذا
 - مدت زمان انتظار در صف پذیرش

- مدت زمان انتظار در صف دریافت غذا
- مدت زمان فعالیت کارکنان بخش پذیرش
- مدت زمان انتظار در صف سالن صرف غذا
- مدت زمان فعالیت کارکنان بخش آشپزخانه
- مدت زمان انتظار جهت دریافت غذا
- مدت زمان حضور مشتریان در رستوران
- تعداد مشتریان رستوران

ساعت^۶

زمان شبیه‌سازی را به ما نشان می‌دهد. این زمان از ساعت ۱۰ صبح الی ۳ بعد از ظهر خواهد بود.

ساختار بندی لیست پیشامدهای آتی^۷

لیست پیشامدهای آتی مجموعه‌ای از اعلام پیشامدها می‌باشد که با استفاده از آنها باید زمان شبیه‌سازی خود را جلو ببریم. در مسئله‌ی مورد بررسی ما به صورت کلی سیزده پیشامد منحصر به فرد وجود دارد که پس از ورود مشتریان و مشغول شدن تمام بخش‌های سیستم به طور معمول دست کم سیزده پیشامد تعریف شده را خواهیم داشت. به عبارت دیگر بعد از عبور از زمان سرد سیستم، لیست پیشامدهای آتی ما به طور معمول شامل حداقل سیزده اعلام پیشامد خواهد بود. با توجه به اینکه تعداد کارمندان بخش پذیرش پنج و کارمندان آشپزخانه دو نفر می‌باشد پیشامدهای مربوط به اتمام فعالیت این افراد می‌تواند بیشتر از یک مورد و تکراری باشد و این امر کاملاً عادی می‌باشد. به همین دلیل از واژه‌ی دست کم سیزده پیشامد استفاده شده‌است. البته شایان ذکر است که در حالت‌هایی استثنایی مانند شروع شبیه‌سازی، اعضای لیست مدنظر ما کمتر خواهد بود.

کمینه مقدار زمان از این لیست انتخاب و زمان شبیه‌سازی به آن منتقل می‌شود. پس از این اقدام اعلام پیشامد مذکور از لیست حذف می‌شود. همچنین با توجه به شرایط خاصی که برای مسئله تعریف شده‌است پس از ورود مشتریان به وسیله‌ی اتوبوس دیگر پیشامد تولید نخواهد شد و بنابراین به لیست باز نمی‌گردد.

نکته‌ی دیگر در مورد وجود پیشامدهای مربوط به اتمام استراحت‌ها می‌باشد که در ابتدا باید پیشامد شروع استراحت اتفاق بیافتد تا بتوان برای اتمام آن برنامه‌ریزی کرد.

برای مثال می‌توانیم لیست زیر را به عنوان نمونه‌ای از لیست پیشامدهای آتی معرفی کنیم که در حالت عادی رخ خواهد داد:

$$FEL = \{(Ac,t_1), (Ab,t_2), (Aw,t_3), (rE,t_4), (Er,t_5), (R,t_6), (Erk,t_7), (E,t_8), (Ex,t_9), (En,t_{10}), (Srr,t_{11}), (Err,t_{12}), (Srk,t_{13})\}$$

^۶Clock

^۷FEL

در مثال بالا هنوز مشتریان مربوط به اتوبوس به رستوران نرسیده‌اند؛ بنابراین پیشامد ورود آنها همچنان در لیست وجود دارد. همچنین تنها یک پیشامد برای اتمام فرایند سفارش‌دهی و دریافت غذا در نظر گرفته شده‌است که به طور معمول اعلام‌های مربوط به این پیشامدها در هر لحظه بیش از یک مورد خواهد بود اما در حال حاضر برای سهولت در خوانایی متن گزارش به معرفی یک مورد بسنده شده‌است.

در نهایت زمان شبیه‌سازی با توجه به زمان‌های لیست مثال فوق پیش خواهد رفت. حال لیست پیشامدهای آتی را برای لحظه شروع سیستم و صفر زمان به عنوان مثال ذکر می‌کنیم:

$$FEL = \{(Ac, 10:05), (Ab, 12:00), (Aw, 10:03), (Srr, 10:50), (Srk, 10:50)\}$$

همانطور که مشاهده می‌شود در حالت اولیه سیستم تنها پنج پیشامد وجود دارد که زمان شبیه‌سازی و حالت سیستم بر اساس کمینه مقدار زمان موجود در این لیست تغییر خواهد کرد.

فرضیات و ساده‌سازی‌ها

- (۱) بین ورودی رستوران و بخش پذیرش فاصله ناچیز است. به همین دلیل فرض می‌شود شروع سفارش و ورود به رستوران یکی است.
- (۲) عملیات سفارش‌دهی و پرداخت پول هر دو در بخش پذیرش انجام می‌شوند. در واقع سروری که سفارش را ثبت می‌کند به پرداخت نیز رسیدگی می‌کند و بین سفارش و پرداخت تغییری در سیستم رخ نمی‌دهد.
- (۳) سرورهای پذیرش یکسان در نظر گرفته شده‌اند و تفاوتی با هم ندارند. به همین دلیل اولییتی در استراحت وجود ندارد.
- (۴) کارمندان آشپزخانه یکسان در نظر گرفته شده‌اند و تفاوتی با هم ندارند. به همین دلیل اولییتی در استراحت وجود ندارد.
- (۵) منظور از آشپزخانه همان بخش دریافت غذا می‌باشد.

معیارهای ارزیابی عملکرد سیستم

معیارهای ارزیابی عملکرد سیستم از مهمترین مواردی است که مدیران باید همواره به آن توجه داشته باشند. محاسبه برخی از این معیارها ساده و برخی دیگر نیازمند تلاش بیشتر است، به این دلیل که به دست آوردن اطلاعات لازم برای آنها دشوار است. در این قسمت از پروژه ما ابتدا سه معیار را به عنوان شاخص‌های مرتبه اول معرفی می‌کنیم که در محاسبه شش معیار مهم مرتبه دوم ارزیابی عملکرد سیستم از آنها استفاده خواهیم کرد. در پایان نیز شاخص‌های منتخب جهت ارزیابی سیستم شبیه‌سازی شده را گزارش خواهیم داد. لازم به ذکر است که کلیه میانه‌ها به ازای تمامی مشتریان در زمان شبیه‌سازی محاسبه خواهد شد.

معیارهای ارزیابی عملکرد مرتبه اول

• زمان خدمت‌دهی^۸

اولین مورد زمان خدمت‌دهی می‌باشد که به خودی خود می‌تواند بینش مناسبی از سیستم را برای ما فراهم آورد. این شاخص به صورت زیر محاسبه می‌شود:

$$\text{زمان خدمت‌دهی} = (\text{میانگین مدت زمان انتظار در صف سفارش} + \text{میانگین زمان سفارش غذا}) + (\text{میانگین مدت زمان انتظار در صف دریافت غذا} + \text{میانگین زمان دریافت غذا})$$

برای محاسبه‌ی قسمت اول این شاخص نیازمند استفاده از فلوچارت‌های «پیشامد ورود به صورت پیاده»، «پیشامد ورود به وسیله‌ی ماشین» و «پیشامد ورود به وسیله‌ی اتوبوس» هستیم. در این فلوچارت‌ها زمان فعالیت سفارش غذا تولید می‌شود و باید از داده‌های آن استفاده کنیم و همچنین باتوجه به تأخیری که مشتریان در صف دریافت غذا متحمل می‌شوند میانگین مدت زمان انتظار آنها را نیز می‌توانیم محاسبه کنیم. البته شایان ذکر است که برای محاسبه‌ی زمان انتظار به فلوچارت «پیشامد اتمام سفارش‌دهی»، بخش کاهش طول صف هم نیاز خواهیم داشت که جزئیات اجرای آن متناسب با شرایط کد برنامه‌نویسی خواهد بود.

برای محاسبه‌ی قسمت دوم نیازمند استفاده از فلوچارت «پیشامد شروع فرایند دریافت غذا» به این دلیل که زمان فعالیت فرایند دریافت غذا در آن تولید می‌شود. همچنین برای محاسبه‌ی میانگین مدت زمان انتظار در صف دریافت غذا نیازمند به استفاده از فلوچارت «پیشامد اتمام دریافت غذا» نیز هستیم به این دلیل که صف در آنجا تغییر حالت می‌دهد.

• تعداد کل مشتریان سیستم در بازه‌ی مورد نظر

این شاخص می‌تواند دید مناسبی از لحاظ کارایی سیستم و شرایط بازاریابی را به ما بدهد. برای محاسبه این شاخص به صورت زیر عمل می‌کنیم.

$$\text{تعداد کل مشتریان سیستم در بازه‌ی مورد نظر} = \text{تعداد کل مشتریانی که از رستوران در بازه‌ی مورد نظر خدمت‌گرفته و خارج شده‌اند}$$

با توجه به شرایط سیستم که مشتری ناراضی در آن وجود ندارد، برای محاسبه‌ی این شاخص تنها از فلوچارت «پیشامد خروج از رستوران» استفاده می‌کنیم. البته لازم به ذکر است که برای محاسبه‌ی این شاخص باید در

^۸Service Time

کدنویسی از یک متغیر کمکی استفاده کنیم چون تعریف متغیر حالت کمکی به حل بهتر مسئله‌ی شبیه‌سازی نمی‌کند.

- میانگین مدت زمان حضور مشتری در رستوران

این شاخص به‌طور پیش‌فرض دید مناسبی در مورد مشتریان ما و عملکرد سیستم را به ما می‌دهد. برای محاسبه این معیار به صورت زیر عمل می‌کنیم:

میانگین مدت زمان حضور مشتری در رستوران = زمان خدمت‌دهی + میانگین زمان حرکت بین بخش‌های رستوران + میانگین زمان صرف غذا + میانگین زمان انتظار در صف سالن صرف غذا

نحوه‌ی محاسبه زمان خدمت‌دهی که پیش‌تر به طور کامل ذکر شد. برای محاسبه‌ی زمان حرکت بین بخش‌های رستوران از فلوچارت‌های «پیشامد اتمام صرف غذا»، «پیشامد اتمام سفارش‌دهی» و «پیشامد اتمام دریافت غذا» استفاده می‌کنیم به این دلیل که زمان فعالیت‌های مدنظر ما در دید رو به جلوی آنها محاسبه شده‌است و تنها باید در یک متغیر کمکی ذخیره شود. برای محاسبه‌ی میانگین زمان صرف غذا و میانگین زمان انتظار در صف سالن در فلوچارت‌های «پیشامد شروع صرف غذا» و «پیشامد اتمام صرف غذا» استفاده می‌کنیم. زمان فعالیت مورد نظر و شرایط حالت صف در این دو فلوچارت تولید و تغییر می‌کند.

حال با استفاده از شاخص‌های مرتبه اول ذکر شده به معرفی شش معیار اصلی ارزیابی عملکرد سیستم تحت عنوان شاخص‌های مرتبه دوم می‌پردازیم.

شاخص‌های مرتبه‌ی دوم

- نسبت‌های پذیرش

$$\text{نسبت اول} = \frac{\text{میانگین زمان سفارش غذا}}{\text{زمان خدمت‌دهی}}$$

$$\text{نسبت دوم} = \frac{\text{میانگین مدت زمان انتظار در صف سفارش}}{\text{زمان خدمت‌دهی}}$$

محاسبه‌ی اجزای این شاخص همگی در شاخص‌های مرتبه اول انجام شده‌است. بازه‌ی این نسبت‌ها بین صفر و یک می‌باشد. هرچه این مقدار بزرگتر باشد سیگنالی ایجاد می‌کند که زمان فرایند سفارش‌دهی زیاد است و باید اصلاحاتی صورت گیرد. به همین دلیل مطلوب ما این است که تا حد ممکن این شاخص‌ها را کمینه کنیم. طبیعی

است که مقدار آن تا حدی می‌تواند کمتر شود و در حالت بهینه‌ی سیستم بازهم مقداری قابل توجه را خواهد داشت.

- نسبت‌های دریافت

$$\text{نسبت اول} = \frac{\text{میانگین زمان دریافت غذا}}{\text{زمان خدمت دهی}}$$

$$\text{نسبت دوم} = \frac{\text{میانگین مدت زمان انتظار در صف دریافت غذا}}{\text{زمان خدمت دهی}}$$

این دسته از معیارها نیز بسیار مشابه دسته‌ی قبلی می‌باشد، با این تفاوت که در معیارهای به ارزیابی عملکرد بخش آشپزخانه می‌پردازیم. هرچه این نسبت‌ها بزرگتر باشد می‌تواند این مفهوم را برساند که فعالیت بخش آشپزخانه در حالت بهینه قرار ندارد و بهتر است که شرایط کاری آنها مورد ارزیابی بیشتر قرار گیرد. عناصر این شاخص نیز در معیارهای مرتبه اول همگی محاسبه شده‌است.

- نسبت زمان خدمت‌دهی به زمان کل

این شاخص به صورت کلی بیانگر این امر است که چه بخشی از زمان حضور مشتری در رستوران صرف فعالیت‌های مربوط به سفارش‌دهی و دریافت غذا شده‌است. هرچه این مقدار کمتر باشد به این مفهوم است که بخش زیادی از زمان حضور مشتری در رستوران به صرف غذا و استراحت پرداخته شده‌است و نه فعالیت‌های آزاردهنده‌ای مانند انتظار، بنابراین برای ما بهتر است. این شاخص به صورت زیر به دست می‌آید که محاسبه عناصر آن پیش‌تر ذکر شده‌است.

$$\frac{\text{زمان خدمت‌دهی}}{\text{میانگین مدت زمان حضور مشتری در رستوران}}$$

- نسبت گردش^۹

Turn over^۹

نسبت‌های گردش همواره یکی از مهمترین شاخص‌های هوشمند ارزیابی عملکرد سیستم می‌باشند. کوچکتر بودن این معیار بیانگر سرعت و عملکرد مناسب سیستم است و بزرگتر بودن آن از مقدارهای استاندارد بیانگر سکون و لختی سیستم است. این شاخص را به صورت زیر تعریف می‌کنیم:

زمان کل شبیه‌سازی

تعداد کل مشتریان سیستم در بازه‌ی مورد نظر

برای رستوران خود ابتدا باید استاندارد را تعریف کنیم و سپس این شاخص را با آن بسنجیم. اگر نتیجه از مقدار تعریف بزرگتر بود یعنی مشتریان زمان زیادی را در سیستم سپری می‌کنند و می‌تواند به شرایط بازاریابی ما آسیب بزند. اگر که این شاخص از استاندارد کوچکتر باشد به معنای عدم رضایت مشتری و خروج سریع آنها می‌باشد. بنابراین باید تلاش کنیم تا حول استاندارد عمل کنیم. زمان کل شبیه‌سازی ورودی مسئله می‌باشد و نحوه‌ی محاسبه‌ی مخرج کسر در شاخص‌های مرتبه‌ی اول ذکر شده‌است.

• نسبت تعداد مشتریان به کارمندان

این معیار یکی از بهترین شاخص‌های ارزیابی عملکرد کارمندان می‌باشد. به طور کلی هر صنعت و حرفه‌ای استاندارد برای تعداد کارمندان به ازای تعداد مشتریان دارد و ما هم برای بررسی رستوران خود باید این مورد را در نظر بگیریم. محاسبه‌ی این شاخص به صورت زیر می‌باشد:

تعداد کل مشتریان سیستم در بازه‌ی مورد نظر

تعداد کارمندان رستوران

برای بررسی این شاخص نیز نیازمند استاندارد تعریف شده هستیم. اگر نتیجه از استاندارد بزرگتر باشد یعنی تعداد کارمندان ما کم است و در خدمت‌دهی عملکرد مناسبی نداریم. کوچکتر بودن هم به این معنی است که تعداد کارمندان ما زیاد و بهره‌وری سیستم پایین می‌باشد. بنابراین باید در ابتدا استاندارد برای این شاخص تعریف کنیم و در ادامه تلاش کنیم که از این مقدار استاندارد انحرافی را نداشته باشیم. مخرج کسر جزء اطلاعات ورودی مسئله‌ی شبیه‌سازی و صورت کسر جزء شاخص‌های مرتبه‌ی اول می‌باشد.

• میانگین طول صف سالن غذاخوری در حالت پر بودن سالن

این معیار شاخص بسیار مناسبی برای برآورد کمبود صندلی و ظرفیت سالن در زمان اوج حضور مشتریان است. برای محاسبه‌ی این شاخص باید از فلوچارت‌های «شروع فرایند صرف غذا» و «اتمام فرایند صرف غذا» استفاده کنیم. البته لازم به ذکر است که چالش محاسبه‌ی این شاخص در کدنویسی آن به علت شرطی بودن حالت می‌باشد.

بزرگ بودن این معیار می‌تواند زنگ خطری برای رضایت و تجربه مشتریان باشد و باید سریعاً در راستای بهبود آن اقدامی را انجام دهیم.

شاخص‌های منتخب جهت ارزیابی سیستم شبیه‌سازی شده

با توجه به توضیحاتی که بخش‌های قبلی ذکر شد و اخذ نظر متخصصین پنج شاخص زیر جهت ارزیابی سیستم شبیه‌سازی شده استفاده خواهد شد:

- میانگین مدت زمان حضور مشتری در سیستم
- میانگین مدت انتظار مشتری جهت دریافت غذا
- حداکثر و میانگین طول صف انتظار صرف غذا
- میانگین بهره‌وری کارکنان قسمت پذیرش و دریافت غذا
- نسبت گردش

ذکر این نکته ضروری است که مقدار مبنی برای نسبت گردش با بررسی‌های انجام شده در این سیستم عدد ۱ تعریف می‌شود که توضیحات نحوه‌ی استفاده از این شاخص پیش‌تر گفته شده‌است.

توصیف پویا

در این قسمت به بررسی فلوچارت‌های پیشامدهای موجود می‌پردازیم. در واقع قصد داریم به پاسخ سوالات زیر برسیم:

- (۱) چگونه هر پیشامد بر حالت سیستم، ویژگی‌های نهاد و محتوای مجموعه تاثیر می‌گذارد؟
- (۲) فعالیت‌ها چگونه تعریف می‌شود؟ قطعی هستند یا احتمالی؟ کدام پیشامد معرف شروع یا پایان هر فعالیت است؟ آیا فعالیت می‌تواند صرف نظر از حالت سیستم شروع شود یا شروع آن مشروع به بودن سیستم در حالت خاصی است؟

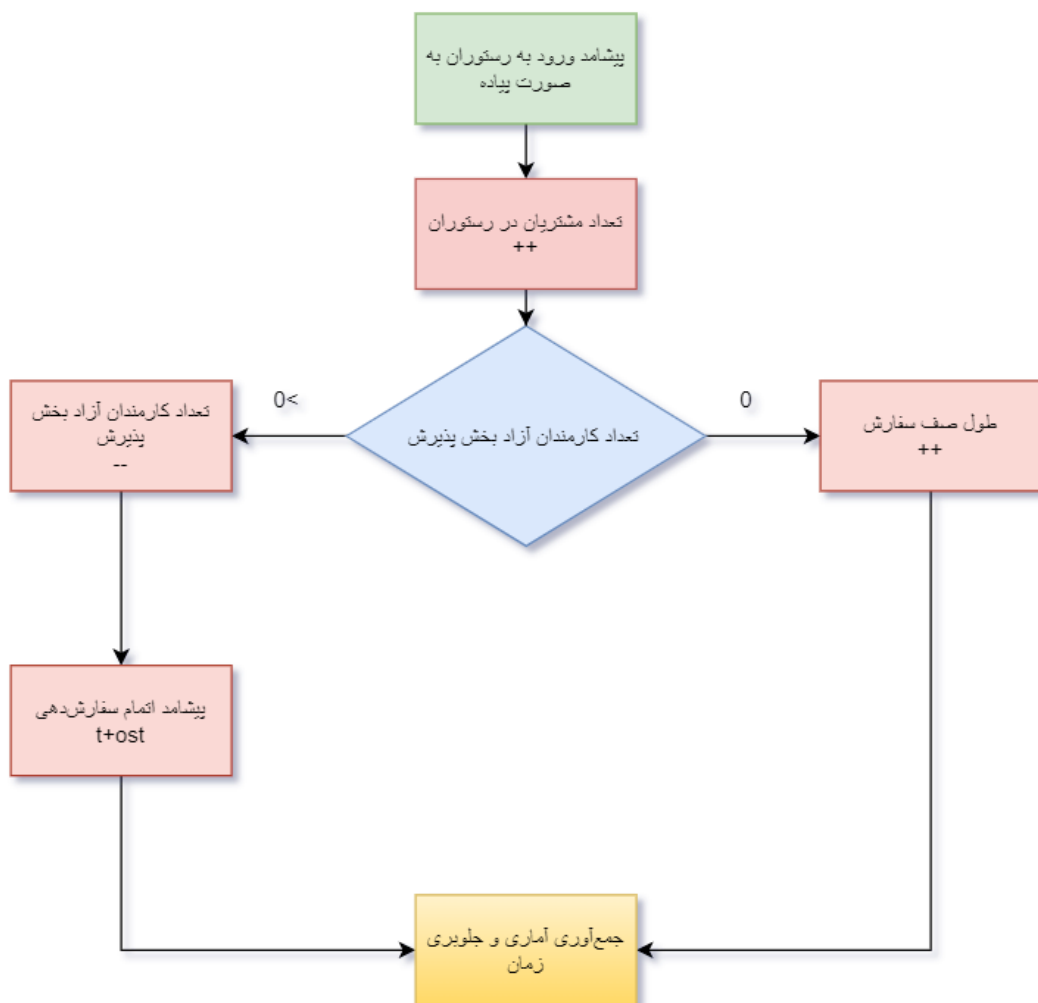
۳) کدام پیشامدها آغاز و پایان هر نوع تاخیر را سبب می‌شود؟ هر تاخیر در کدام شرایط شروع یا تمام می‌شود؟

۴) حالت سیستم در حالت صفر چیست؟ در زمان صفر چه پیشامدهایی برای راه‌اندازی شبیه‌سازی باید تولید شود؟

در ابتدا باید اشاره کنیم که فرض ما در حالت صفر بر این است که هیچ مشتری در رستوران وجود ندارد. همچنین تمامی کارمندان رستوران نیز در حالت آزاد قرار دارند و مشغول نیستند. اولین پیشامد سیستم هم به طور طبیعی ورود یک مشتری خواهد بود که باعث راه‌اندازی سیستم (رستوران) خواهد شد.

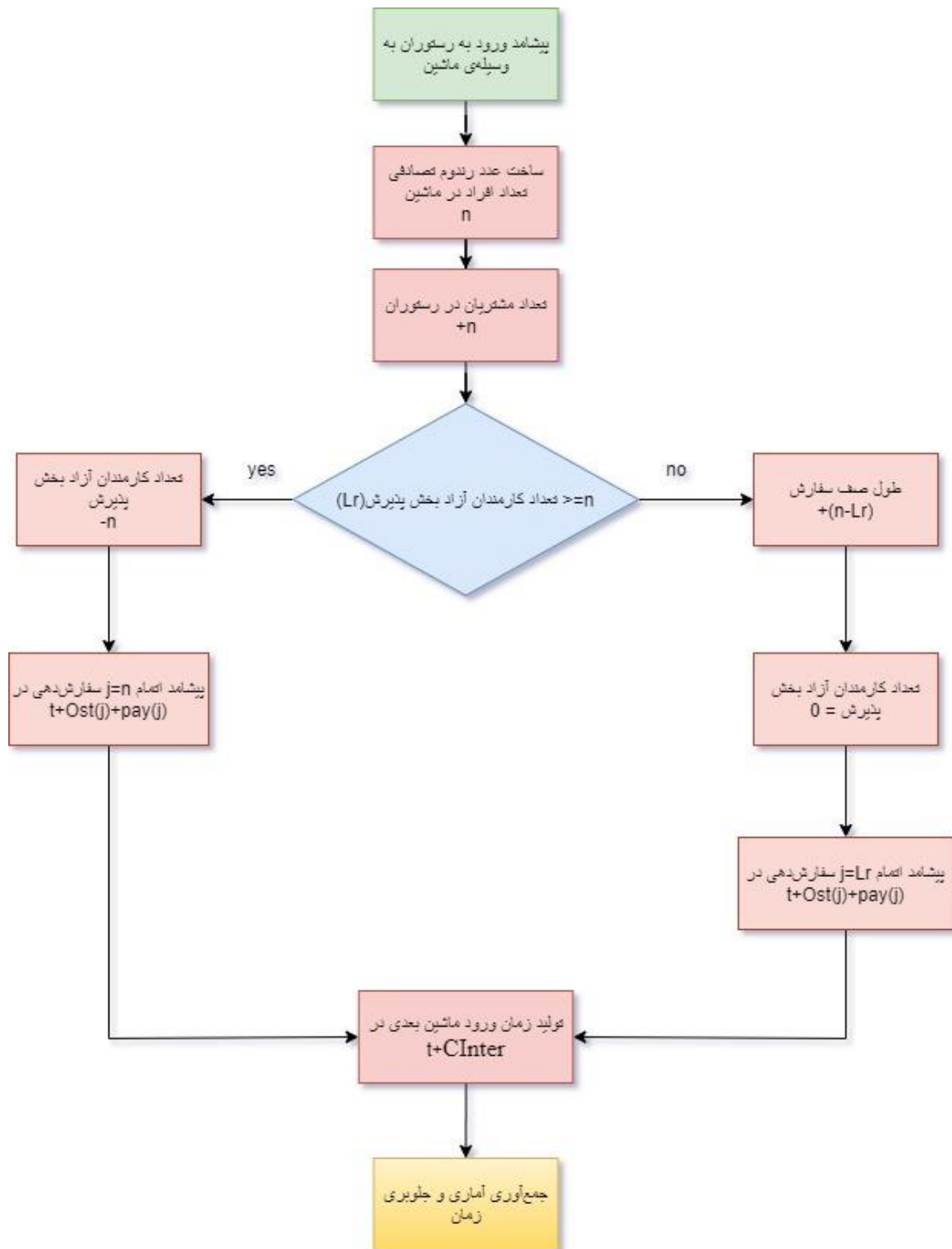
حال به معرفی نمودار جریان مربوط به هر پیشامد می‌پردازیم.

- پیشامد ورود به رستوران به صورت پیاده



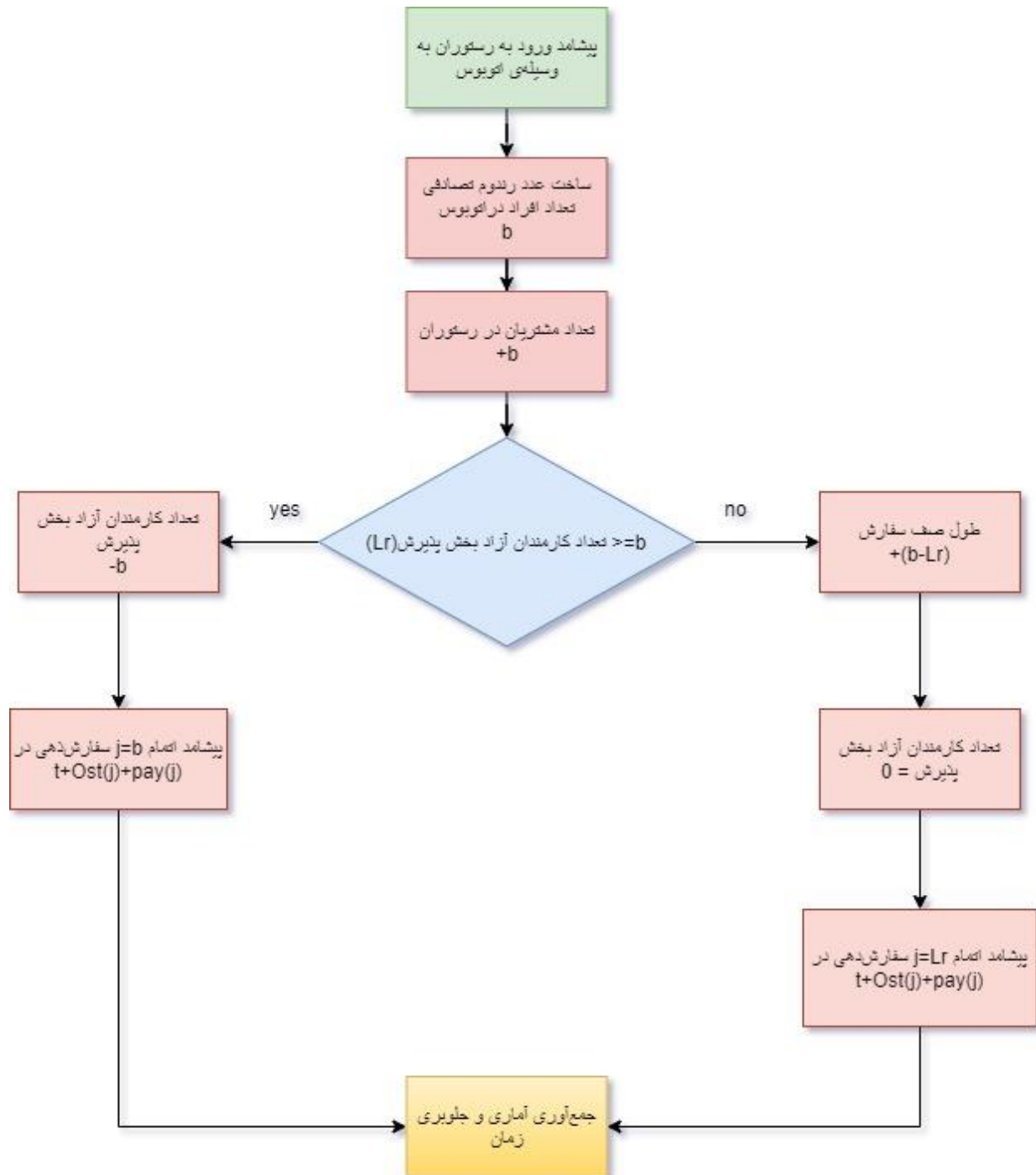
نمودار ۱ - پیشامد ورود به رستوران به صورت پیاده

- پیشامد ورود به رستوران به وسیله ماشین



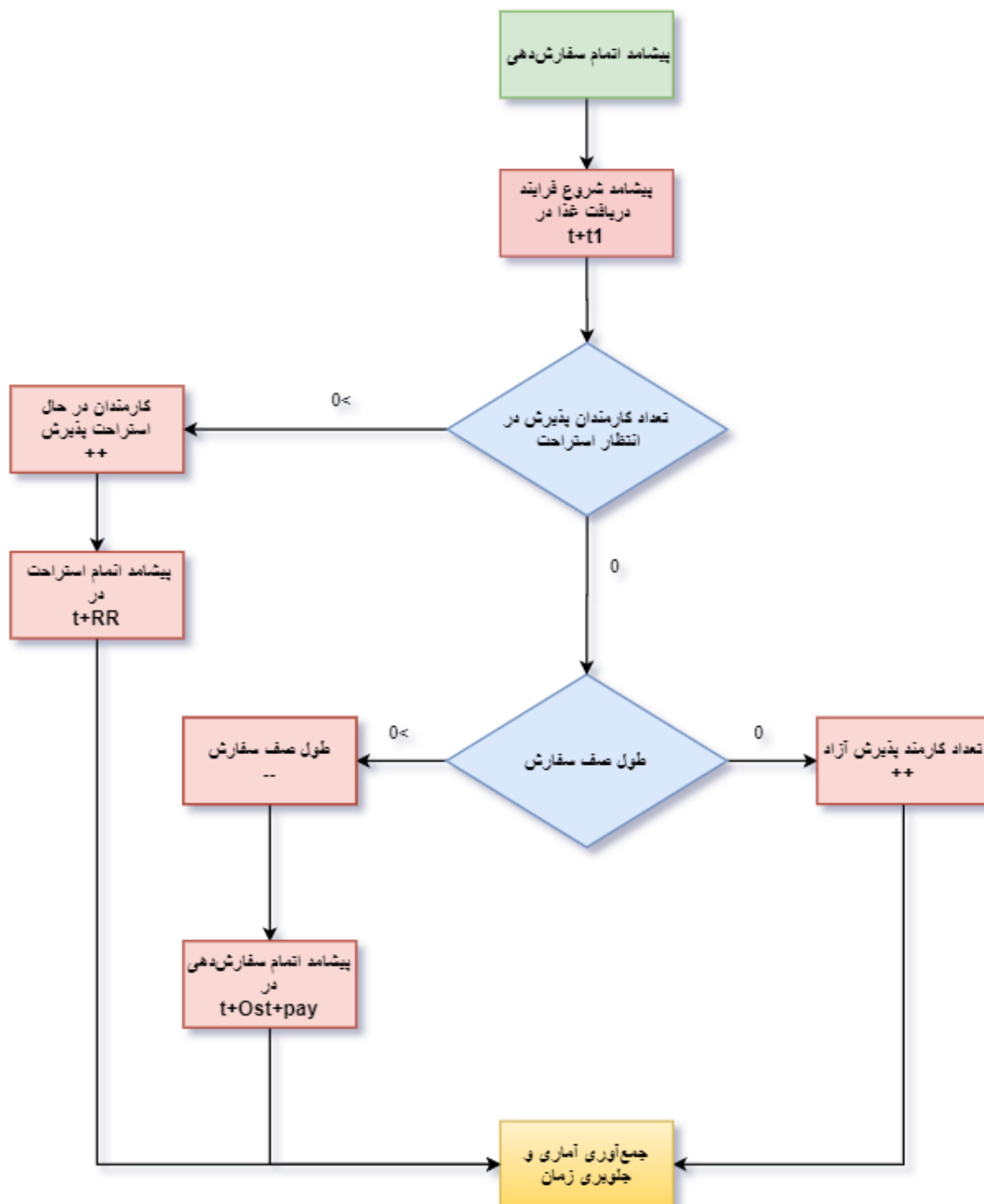
نمودار ۲ - پیشامد ورود به رستوران با ماشین

- پیشامد ورود به رستوران به وسیله اتوبوس



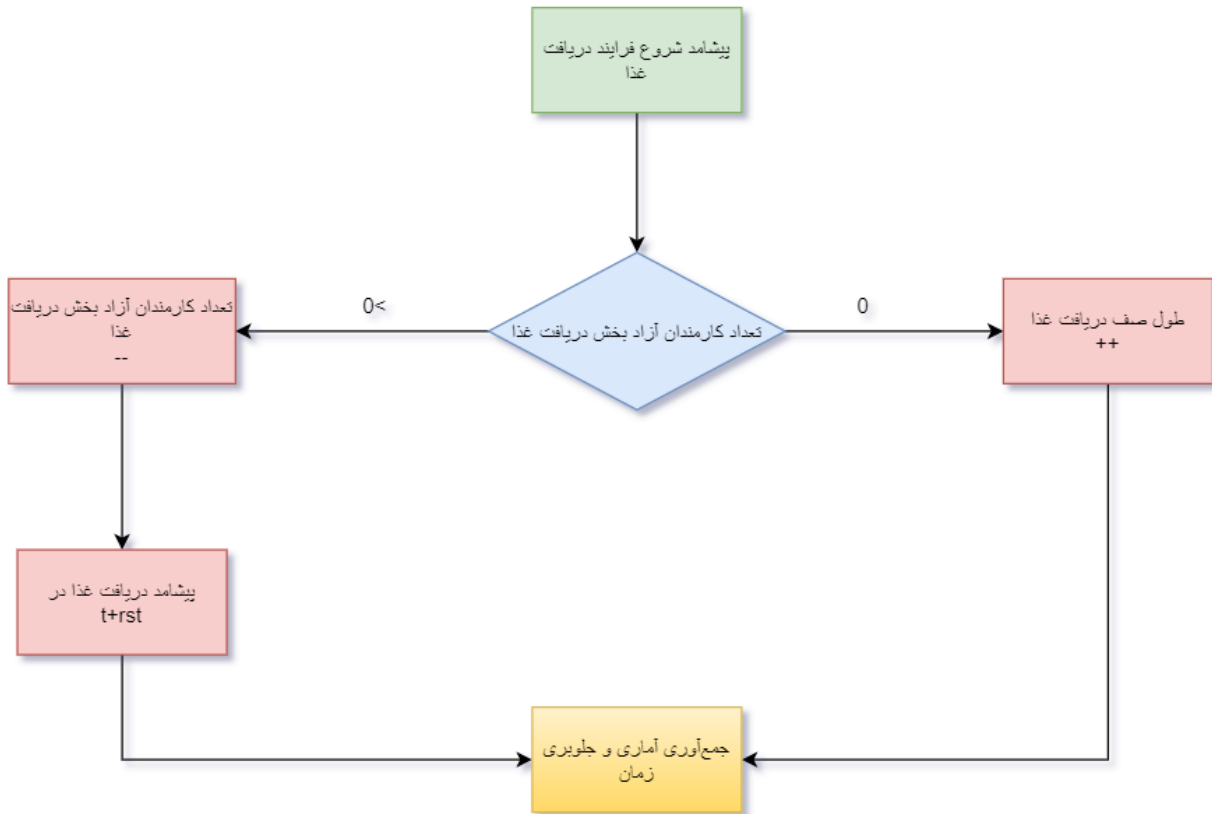
نمودار ۳ - پیشامد ورود به رستوران با اتوبوس

- پیشامد اتمام سفارش دهی



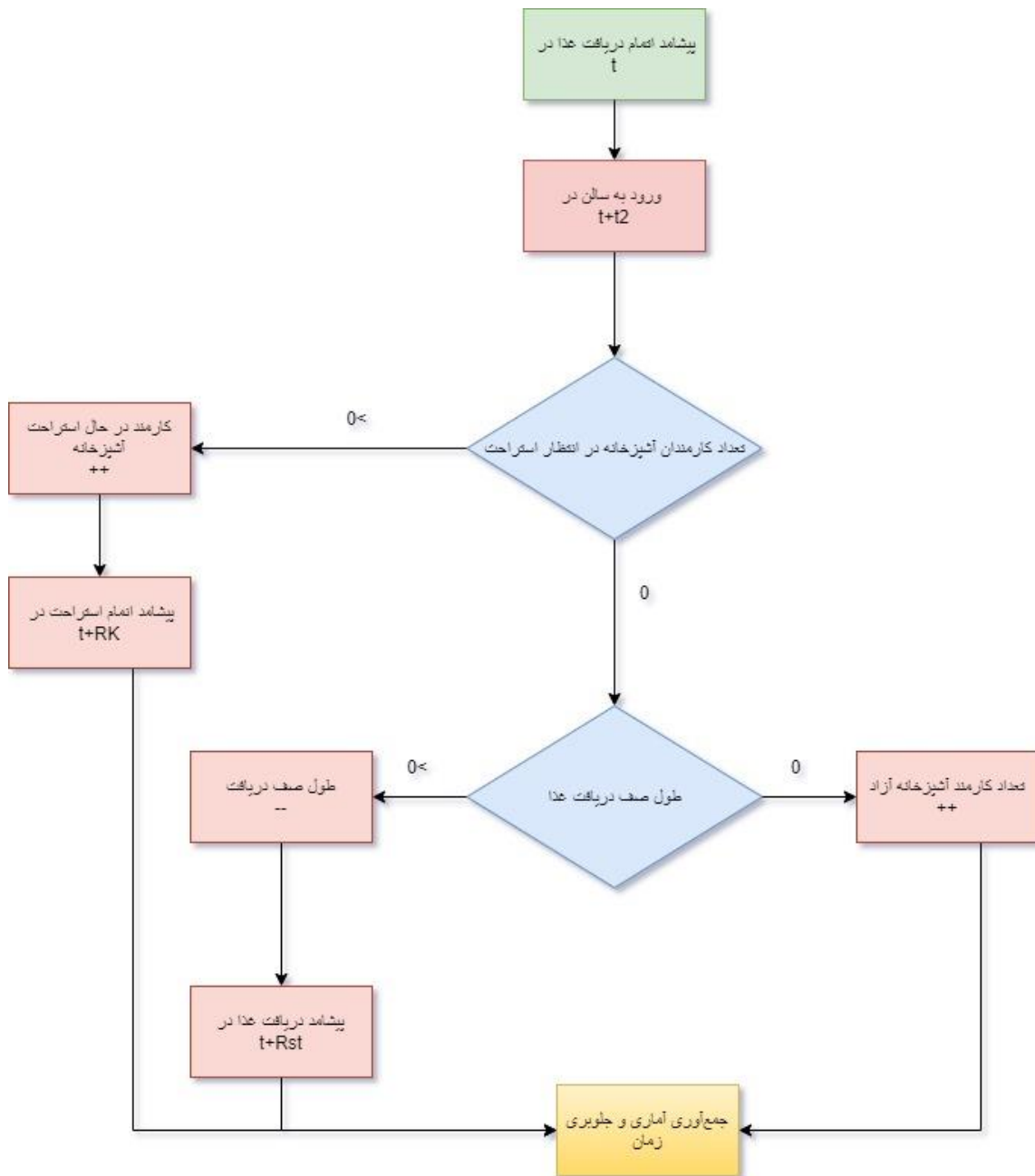
نمودار ۴ - پیشامد اتمام سفارش دهی

- پیشامد شروع فرایند دریافت غذا



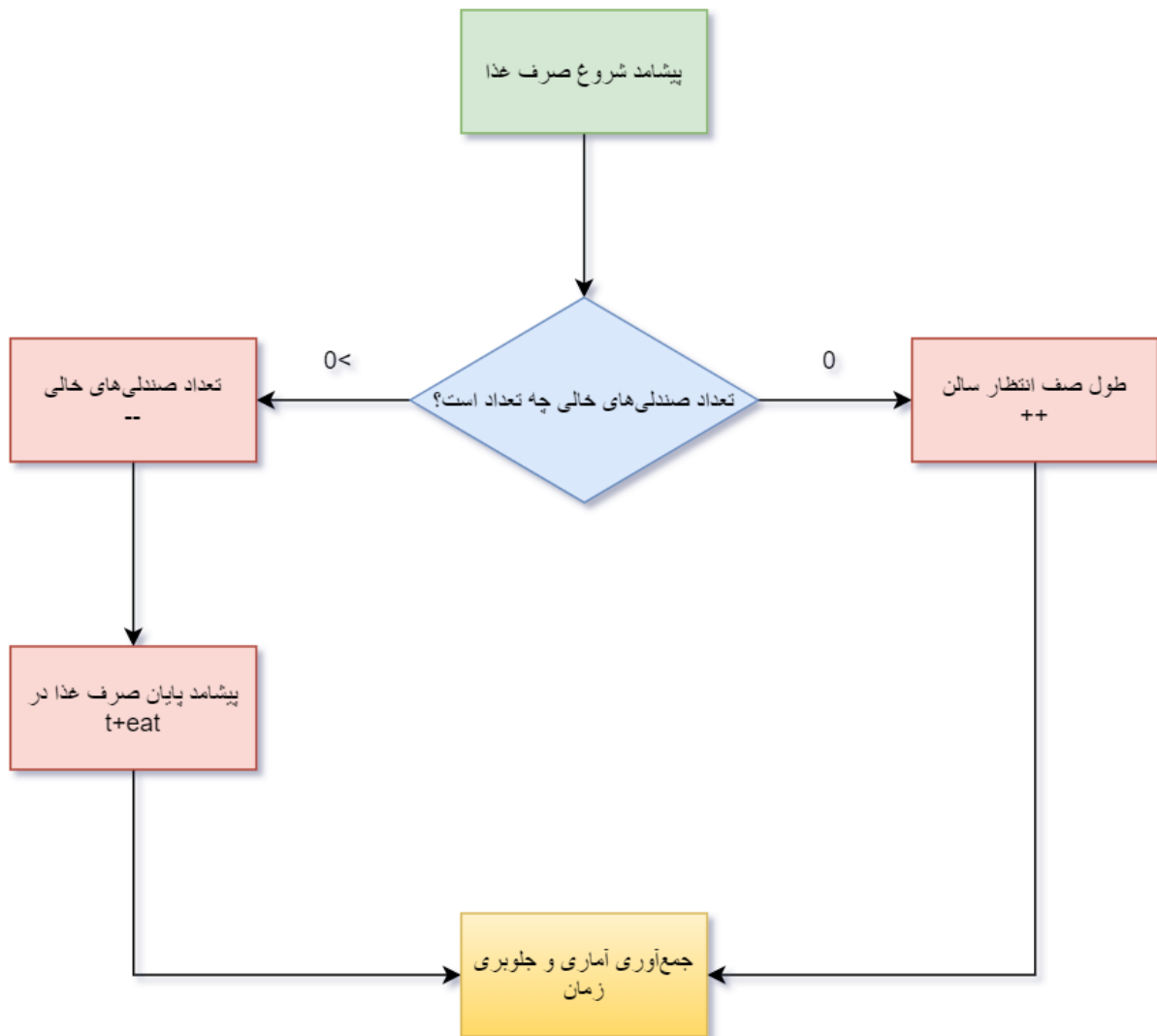
نمودار ۵ - پیشامد شروع فرایند دریافت غذا

- پیشامد اتمام دریافت غذا



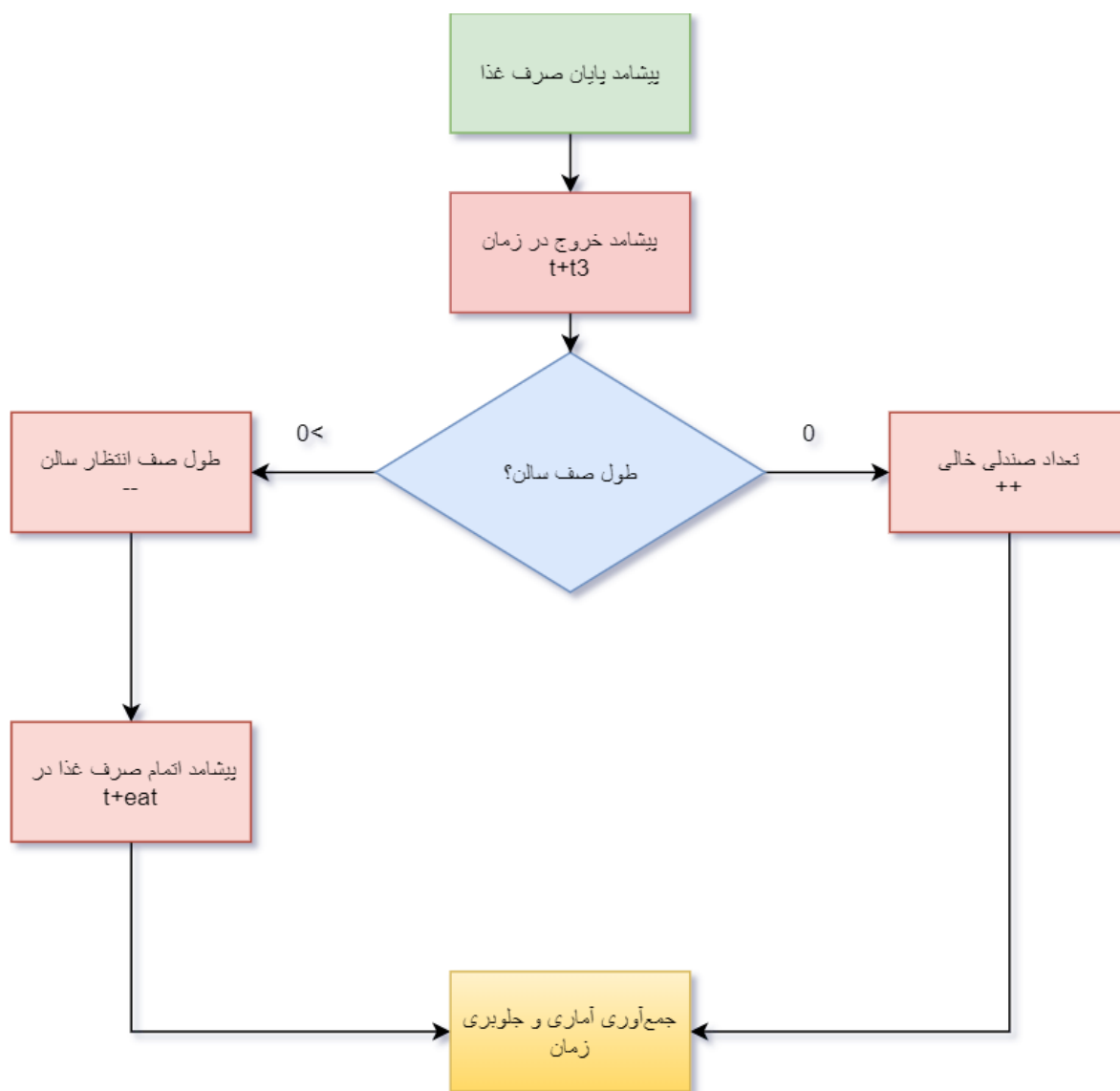
نمودار ۶ - پیشامد اتمام دریافت غذا

- پیشامد شروع صرف غذا



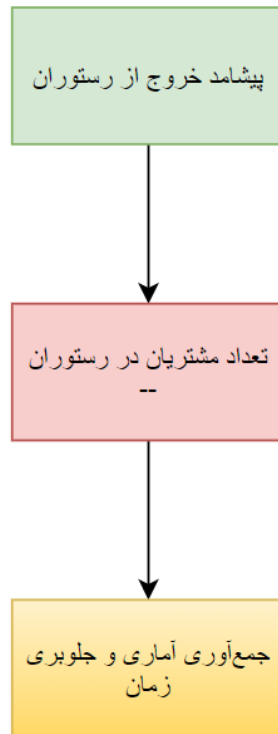
نمودار ۷ - پیشامد شروع صرف غذا

- پیشامد پایان صرف غذا



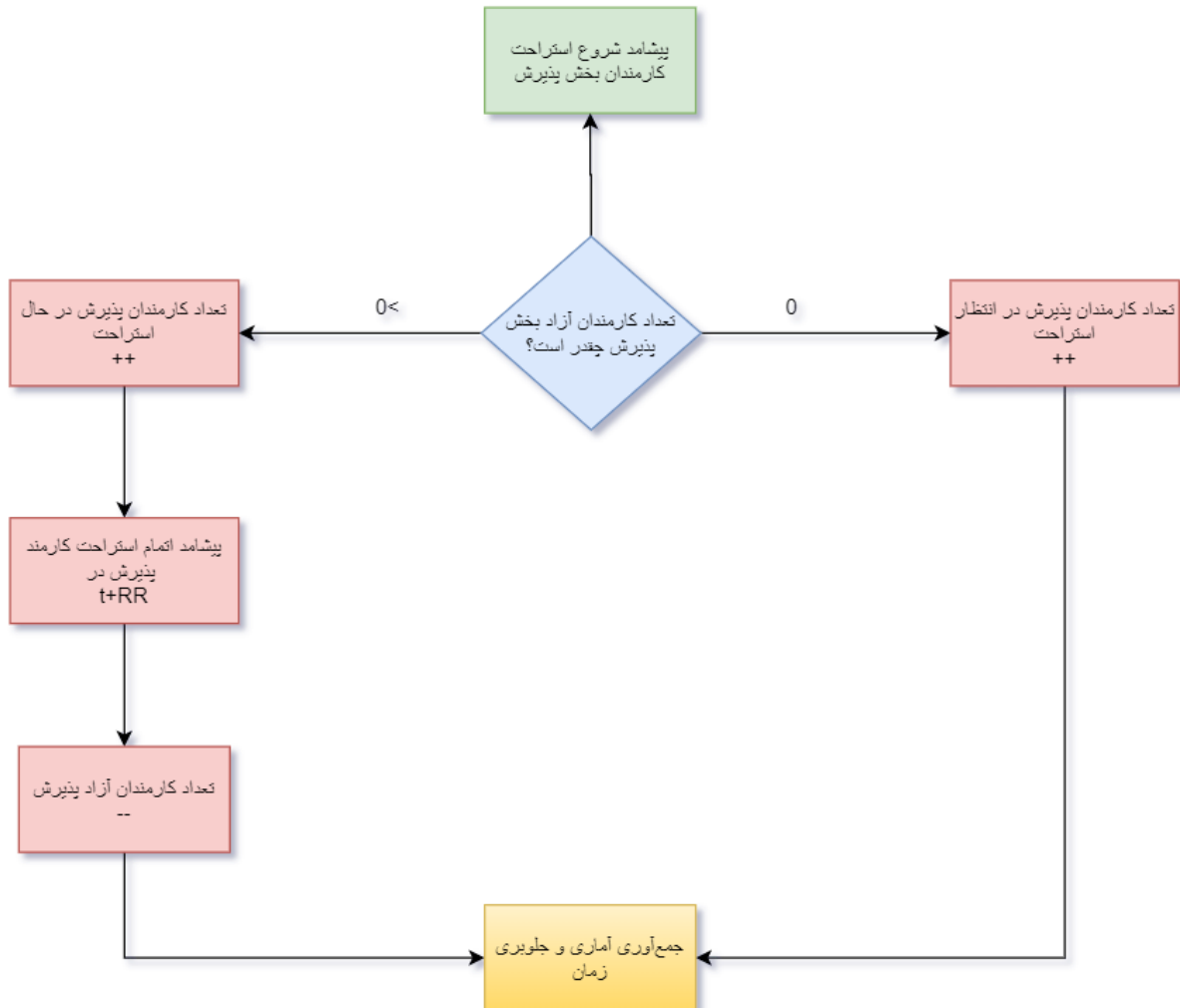
نمودار ۸ - پیشامد پایان صرف غذا

- پیشامد خروج از رستوران



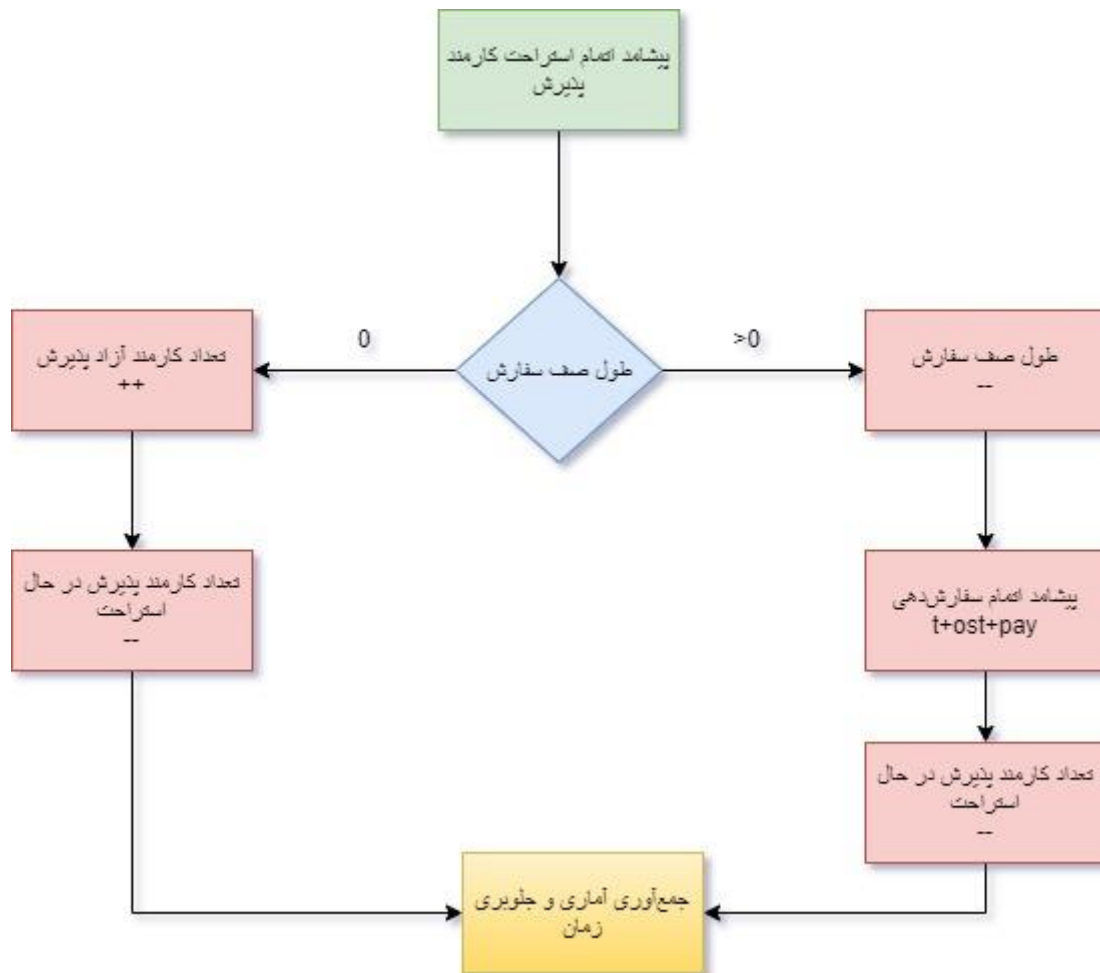
نمودار ۹ - پیشامد خروج از رستوران

- پیشامد شروع استراحت کارمندان بخش پذیرش



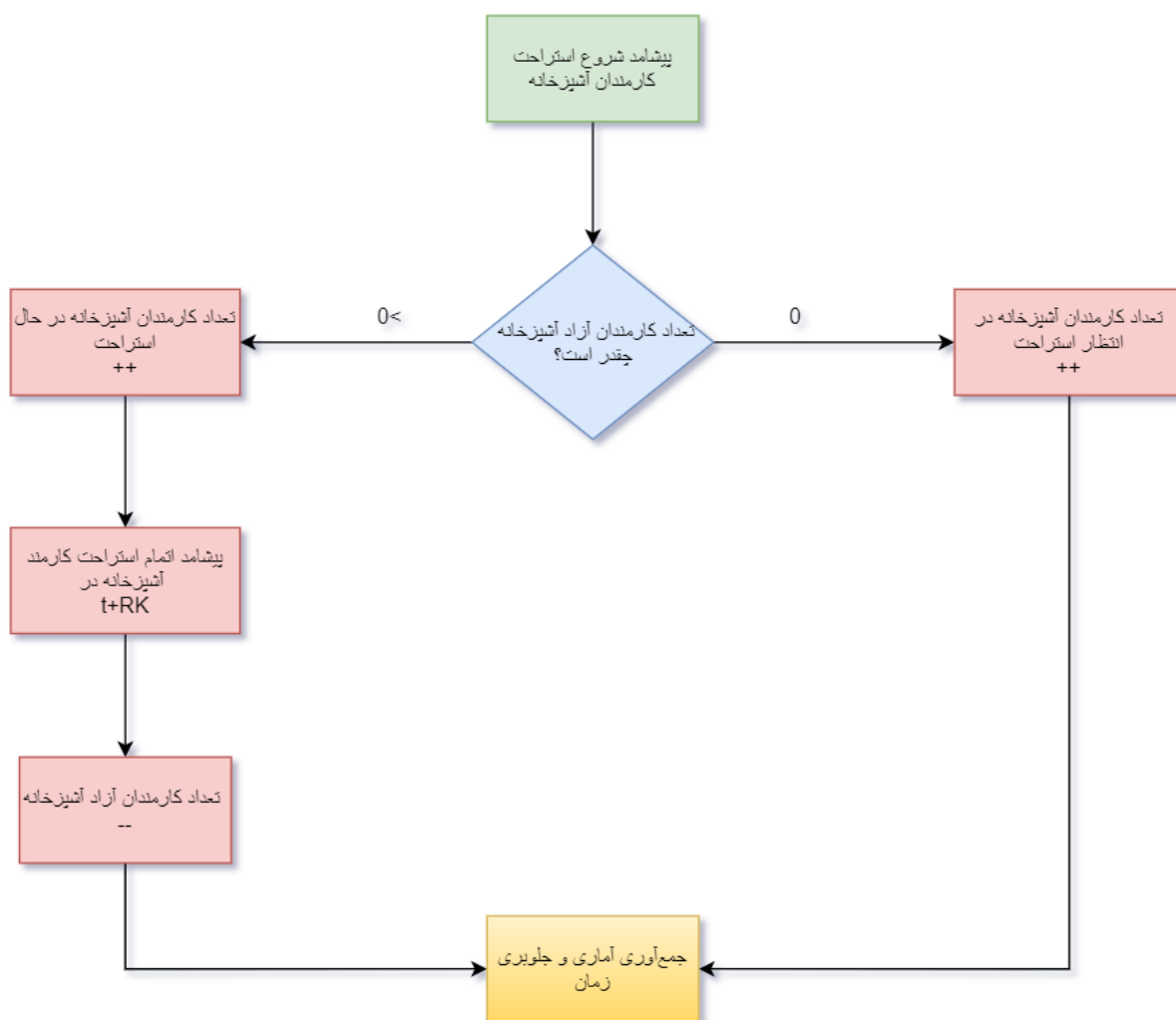
نمودار ۱۰ - پیشامد شروع استراحت کارمندان بخش پذیرش

- پیشامد اتمام استراحت کارمند پذیرش



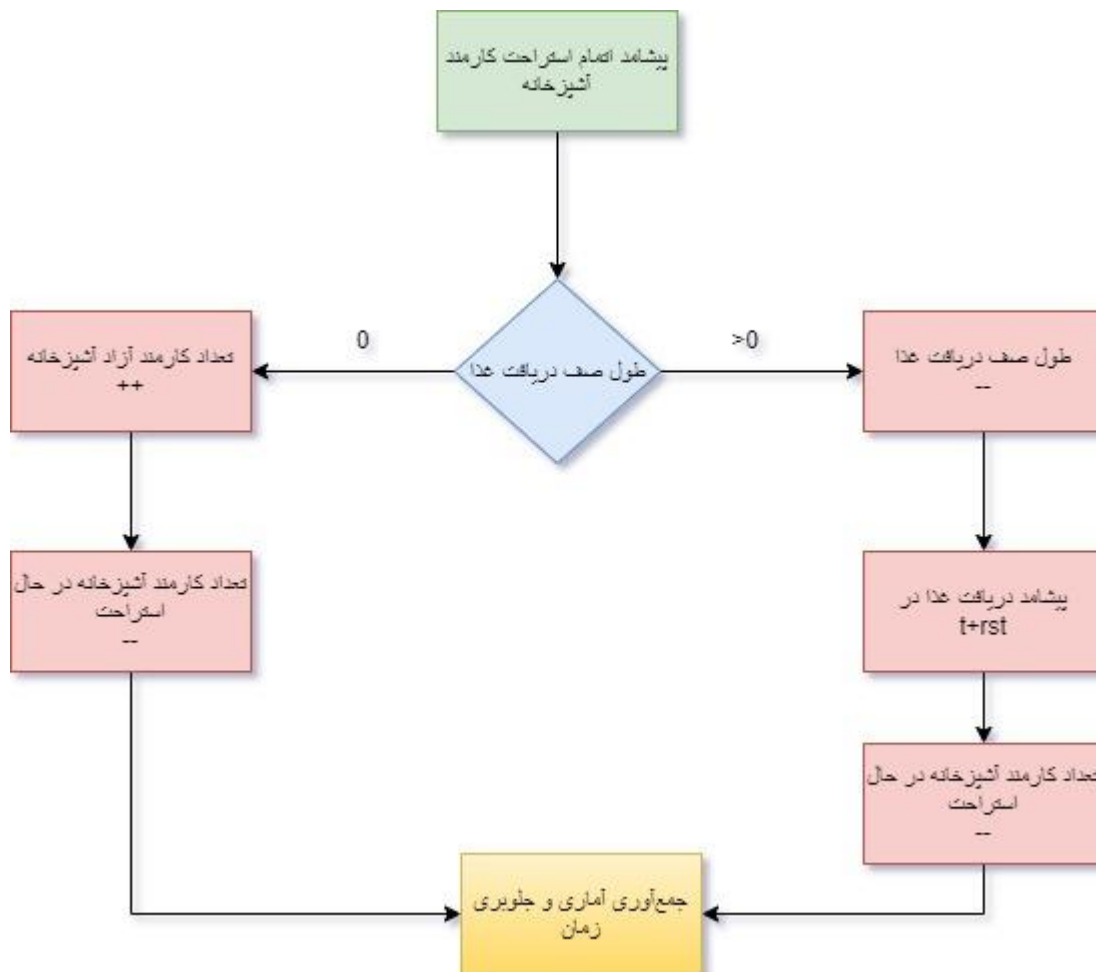
نمودار ۱۱ - پیشامد اتمام استراحت کارمند پذیرش

- پیشامد شروع استراحت کارمندان آشپزخانه



نمودار ۱۲ - پیشامد شروع استراحت کارمندان آشپزخانه

- پیشامد اتمام استراحت کارمند آشپزخانه



نمودار ۱۳ - پیشامد اتمام استراحت کارمند آشپزخانه

شبیه‌سازی گسسته پیشامد رستوران

در این بخش از اجرای شبیه‌سازی به کمک بستر کدزنی پایتون سیستم رستوران خود را مطابق با نمودارهای بخش قبل شبیه‌سازی کردیم و خروجی‌های آن را شناسایی کردیم. در این بخش ذکر یک نکته خالی از لطف نیست. در انتهای زمان شبیه‌سازی دو رویکرد وجود دارد که می‌توان آن را اتخاذ کرد. رویکرد اول این است که در زمان پایان خروجی‌های شبیه‌سازی سیستم را در لحظه محاسبه کنیم و فرایندهای ناقص را در نظر نگیریم. رویکرد دوم نیز این است که پس از پایان زمان مدنظر ورودی‌های سیستم قطع شود و زمان تا اتمام تمامی فرایندها ادامه داشته باشد. بررسی‌های ما نشان داد که این دو خروجی تفاوت معنی‌داری با یک دیگر ندارند و با وجود اینکه کد طراحی شده هر دو خروجی را محاسبه می‌کند، جهت پیشگیری از اطاله‌ی کلام تنها خروجی‌های مربوط به رویکرد اول را در این سند مورد بررسی قرار می‌دهیم. لازم به ذکر است که کد شبیه‌سازی سیستم و بخشی از خروجی

فایل اکسل در پیوست‌های اول و دوم آورده شده‌است. در ادامه خروجی‌های مدل و تحلیل حساسیت این نتایج را مورد بررسی قرار می‌دهیم.

تحلیل حساسیت

در ابتدای امر حالت اولیه‌ای سیستم را بدون تغییر پارامتری، گزارش می‌دهیم. این حالت را پایه در نظر می‌گیریم. سپس با تغییر پارامتر مدنظر حول مقدار اصلی آن تحلیل حساسیت را انجام می‌دهیم. در ادامه به گزارش فرایند ذکر شده می‌پردازیم. جهت سهولت در استفاده از نمودارها تلاش بر این بوده‌است که به طور موازی از دو محور عمودی استفاده شود.

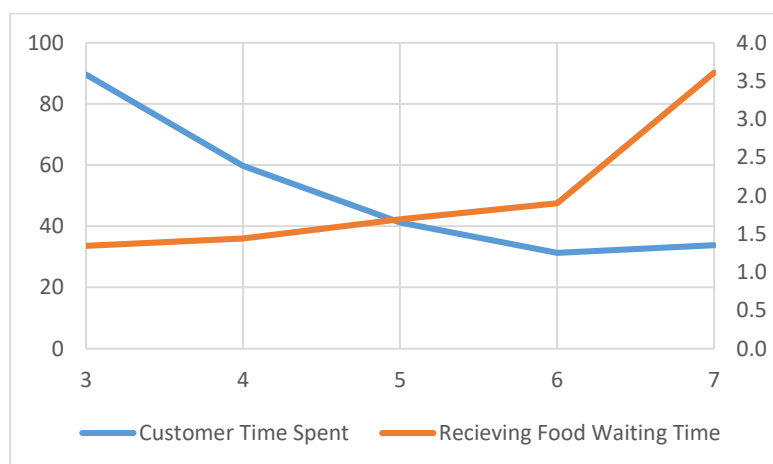
تغییر پارامتر تعداد کارمندان بخش پذیرش

در حالت پایه مقدار این پارامتر ۵ می‌باشد و جهت تحلیل مقدار آن را حول ۵ تغییر می‌دهیم. خروجی تحلیل حساسیت به شرح زیر است:

Receptionists	Customer Time Spent	Receiving Food Waiting Time	Seat Queue Length	Max Seat Queue	Receptionists Utilization	Kitchen Staff Utilization	Turnover Ratio
3	89.5740	1.3443	0.0000	0	0.9550	0.4302	1.0453
4	59.7246	1.4404	0.0000	0	0.9450	0.5566	0.9174
5	41.2585	1.6922	0.0000	0	0.8569	0.6048	0.9677
6	31.2675	1.8999	0.0462	4	0.6541	0.5717	1.0791
7	33.8137	3.6084	0.4238	8	0.5983	0.6230	1.0169

جدول ۱ - تحلیل حساسیت تغییر تعداد خدمت‌دهندگان بخش پذیرش

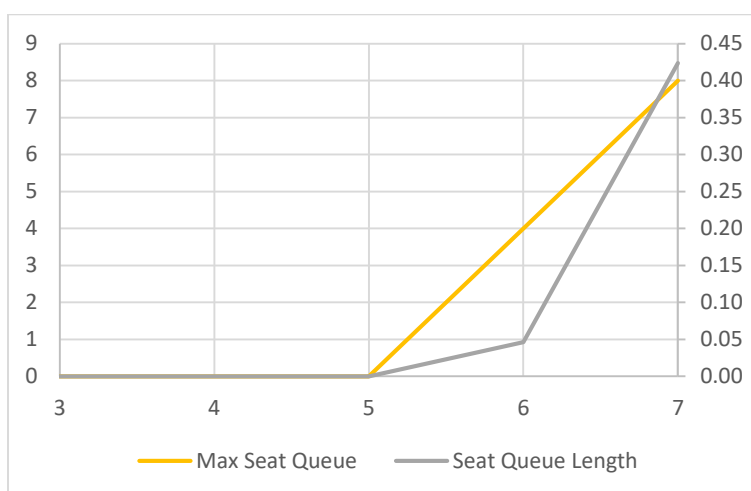
در ادامه به بررسی دقیق این تغییر می‌پردازیم.



نمودار ۱۴

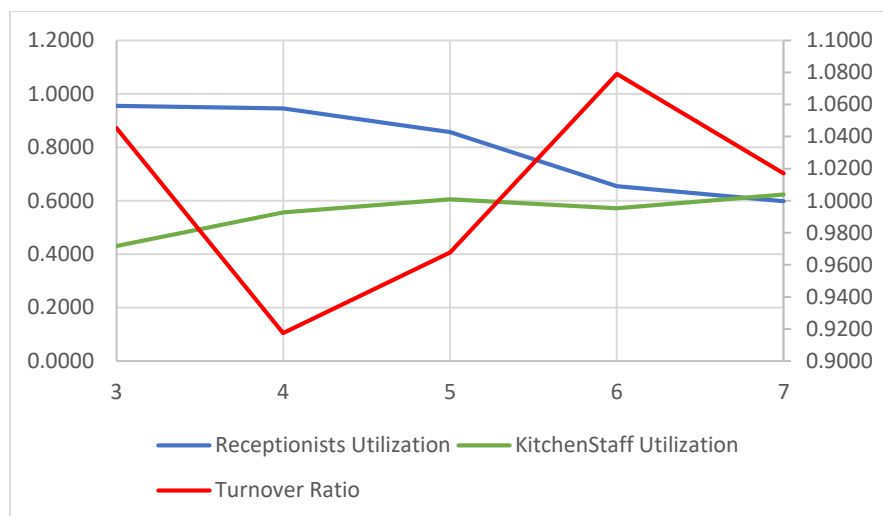
همانطور مشاهده می‌شود مطابق نمودار ۱۴ افزایش تعداد کارمندان بخش پذیرش به طور قابل ملاحظه‌ای زمان حضور مشتریان در رستوران را کاهش می‌دهد در به همین ترتیب کاهش آن به طور فزاینده‌ای زمان حضور مشتریان را افزایش می‌دهد. بنابراین سیاست صحیح این است که تا حد امکان از کاهش تعداد این افراد پیشگیری کنیم.

از سویی دیگر این افزایش در حد ملموسی زمان انتظار مشتریان برای دریافت غذا را نیز بیشتر می‌کند و به عبارت بهتر به نظر می‌آید که باعث ایجاد گلوگاه در بخش دریافت غذا خواهد شد. بنابراین این تغییر اثر جانبی در بخش دیگری از سیستم خواهد داشت. کاهش پارامتر مذکور اما اثر قابل توجهی بر زمان انتظار مشتریان برای دریافت غذا ندارد.



نمودار ۱۵

در نمودار ۱۵ می‌توان دید که کاهش تعداد کارمندان بخش پذیرش اثری بر شرایط صف ندارد. اما افزایش این افراد باعث می‌شود تا میانگین طول صف و حداکثر مقدار صف صرف غذا افزایش چشمگیری داشته باشد. همانطور که در تحلیل پیشین نیز ذکر شد، افزایش این افراد باعث می‌شود تا سرعت بخش پذیرش شود و باعث ایجاد گلوگاه‌هایی در بخش‌های بعدی رستوران بشود. بنابراین به نظر می‌آید که افزایش زیاد این افراد اصلاً برای سیستم مناسب نخواهد بود و در صورت نیاز تغییرات باید با دقت انجام شود.



نمودار ۱۶

نمودار ۱۶ به ما نشان می‌دهد که افزایش تعداد کارمندان بخش پذیرش باعث کاهش بهره‌وری و بازده آنها می‌شود و این موضوع هشدار برای ما می‌دهد که نباید تعداد این افراد را به سادگی تغییر دهیم. افزایش بازده کارمندان بخش آشپزخانه با این تغییر نیز رخ می‌دهد اما به نظر می‌آید که این تغییر آنچنان چشمگیر نیست و به نظر می‌آید که باید از طرق دیگری این موضوع را بهبود بخشید.

نسبت گردش نیز تغییرات نوسانی با تغییر کارمندان بخش پذیرش دارد و به نظر می‌آید علت تغییر آن، موارد دیگری در سیستم هستند و نمی‌توان با قاطعیت در مورد این شاخص سخنی گفت. اما نکته مهم این است که در تمامی موارد شاخص حول مقدار مبنی قرار دارد و این نکته مثبتی می‌باشد.

در نتیجه به صورت خلاصه می‌توان گفت تغییر کوچک در حد افزایش یک نفر به کارمندان بخش پذیرش می‌تواند اثر مثبتی بر سیستم بگذارد اما افزایش بیش از اندازه‌ی این افراد اثرات جانبی دیگری برای سیستم خواهد داشت که باید از این بازخورد جلوگیری شود.

تغییر پارامتر تعداد کارمندان بخش آشپزخانه

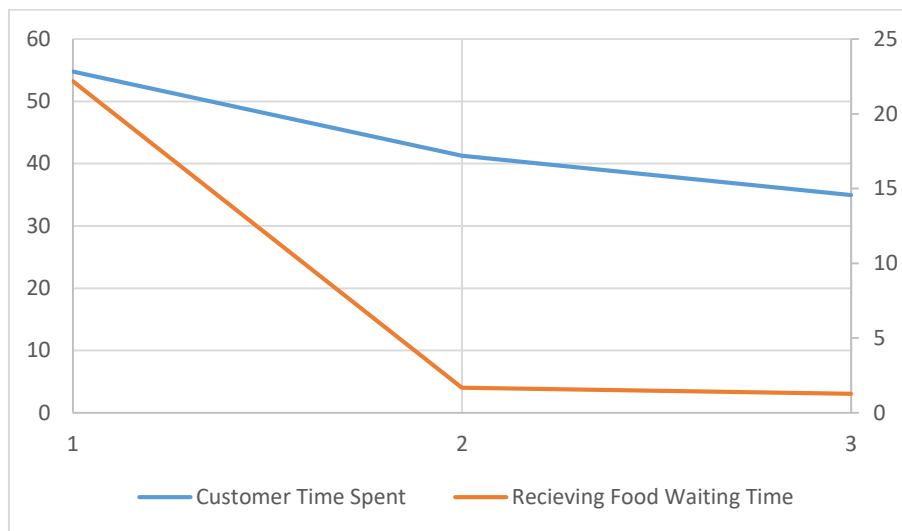
در حالت پایه مقدار این پارامتر ۲ می‌باشد و جهت تحلیل مقدار آن را حول ۵ تغییر می‌دهیم. خروجی تحلیل حساسیت به شرح زیر است:

Kitchen Staff	Customer Time Spent	Receiving Food Waiting Time	Seat Queue Length	Max Seat Queue	Receptionists Utilization	Kitchen Staff Utilization	Turnover Ratio
1	54.7694	22.1807	0.0000	0	0.7444	0.8175	1.1111
2	41.2585	1.6922	0.0000	0	0.8569	0.6048	0.9677

3	34.9464	1.2780	0.0000	0	0.7157	0.3444	1.1858
---	---------	--------	--------	---	--------	--------	--------

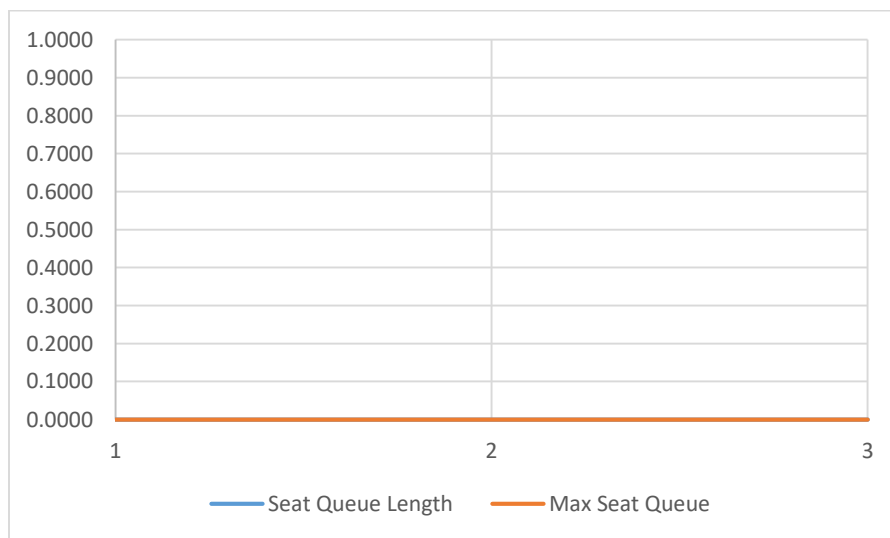
جدول ۲ - تحلیل حساسیت تغییر تعداد کارمندان بخش آشپزخانه

حال به بررسی دقیق این تغییرات می پردازیم.



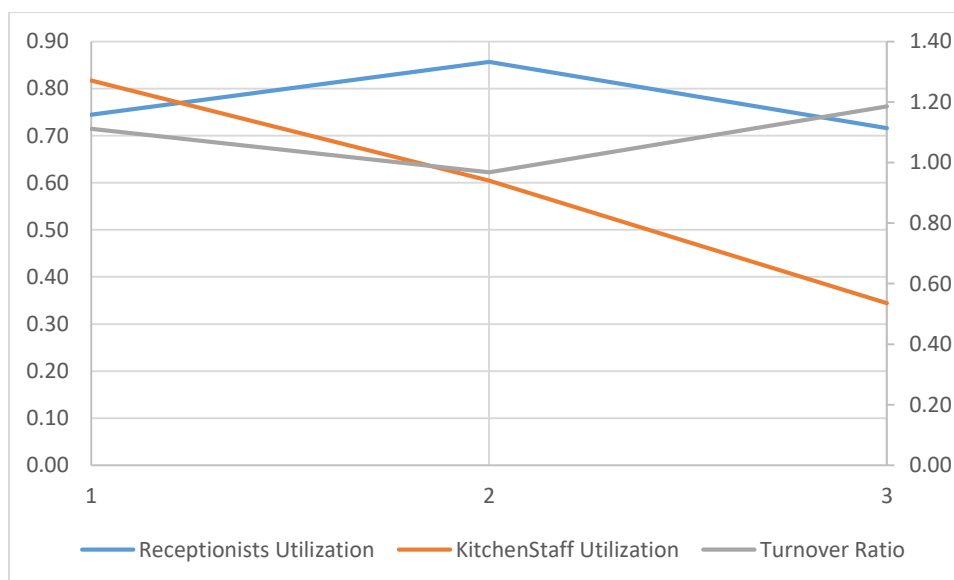
نمودار ۱۷

در نمودار ۱۷ می توان دید که افزایش تعداد کارمندان بخش پذیرش به طور قابل توجهی می تواند زمان حضور مشتریان در سیستم را کاهش دهد و در کنار این موضوع شاید بتوان علت اصلی را کاهش زمان دریافت غذا دانست. با مشاهده نمودار به دست آمده می توان گفت که افزایش افراد مورد بررسی سرعت سیستم را افزایش می دهد و در حال حاضر می توان این تغییر را مثبت ارزیابی کرد. البته قطعاً این موضوع نیازمند بررسی دقیق تر می باشد.



نمودار ۱۸

بررسی‌های انجام شده نشان می‌دهد که افزایش و یا کاهش کارمندان آشپزخانه اثری بر شرایط صف صرف غذا ندارد و گلوگاه اصلی سیستم پیش از مورد تحت بررسی می‌باشد. به همین دلیل آن بخشی که باعث تغییر عمده در صف صرف غذا و شرایط ورود به سالن می‌باشد بخش آشپزخانه نیست و باید بخش‌های قبلی را مورد بازبینی قرار داد.



نمودار ۱۹

همانطور قابل پیش‌بینی نیز بود نمودار ۱۹ به ما نشان می‌دهد افزایش تعداد کارمندان بخش آشپزخانه اثری بر بازده و بهره‌وری کارمندان بخش پذیرش ندارد. اما به طور قابل توجهی می‌توان دید که افزایش کارمندان بخش آشپزخانه بازده آنها را کاهش می‌دهد و این به طور چشمگیری نیز رخ می‌دهد. بنابراین افزایش کارمندان این بخش اصلاً توصیه نمی‌شود و در صورت نیاز باید بر کاهش آنان تحلیلی انجام شود. نکته جالب اما تغییرات جزئی نسبت گردش است که شاید بتوان از آن به این عنوان برداشت کرد که در این حالت تعداد افراد خدمت گرفته در رستوران به صورت کلی تغییرات زیادی نداشته است. البته این موضوع با توجه به نوسان موجود در نمودار تنها یک گمان است و نیازمند بررسی بیشتر می‌باشد.

بنابراین می‌توان این گونه گفت سیستم در حال حاضر با توجه به سایر پارامترها نیازی به افزایش کارمندان بخش آشپزخانه ندارد و در صورت کاهش این افراد نیز آسیبی زیادی به سیستم وارد نمی‌شود. اما عنصر اصلی تعیین‌کننده شرایط این پارامتر، بخش‌های پیشین این سیستم می‌باشد.

تغییر پارامتر تعداد صندلی‌های سالن غذاخوری

در حالت پایه مقدار این پارامتر ۲ می‌باشد و جهت تحلیل مقدار آن را حول ۵ تغییر می‌دهیم. خروجی تحلیل حساسیت به شرح زیر است:

Seats	Customer Time Spent	Receiving Food Waiting Time	Seat Queue Length	Max Seat Queue	Receptionists Utilization	Kitchen Staff Utilization	Turnover Ratio
30	41.2585	1.6922	0.0000	0	0.8569	0.6048	0.9677
25	45.8955	1.7515	0.1302	4	0.8105	0.5954	0.9146
20	46.9131	1.5882	7.8814	21	0.8460	0.6124	0.9901
10	100.4526	1.7101	65.0930	127	0.8050	0.5801	1.0417

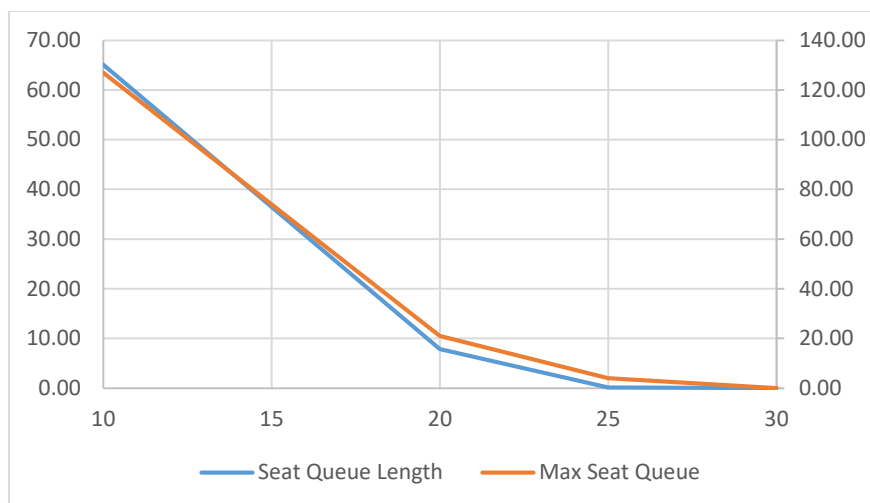
جدول ۳ - تحلیل حساسیت تعداد صندلی

در این تغییر پارامتر چون در حالت پایه میانگین طول صف سالن و حداکثر مقدار آن صفر می‌باشد تنها حالت کاهش آن را به دقت مورد بررسی قرار می‌دهیم. حال به بررسی دقیق این تغییرات می‌پردازیم.



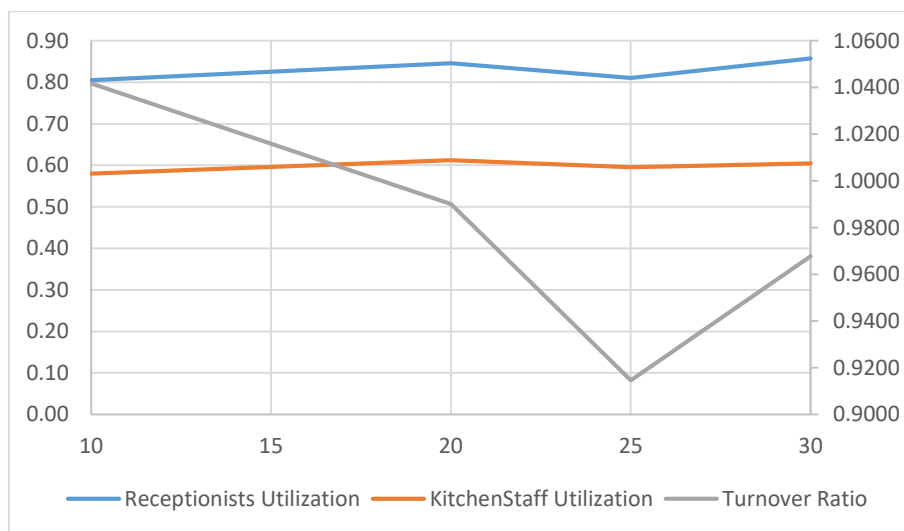
نمودار ۳۰

نتایج حاصل از نمودار ۲۰ نشان می‌دهد که کاهش تعداد صندلی‌های تأثیر بسیار زیادی در مدت زمان حضور مشتریان در رستوران دارد و باعث می‌شود این مدت زمان به طور ملموسی افزایش پیدا کند. بنابراین نباید به طور قابل توجهی تعداد صندلی‌ها را کاهش دهیم. اما همانطور که انتظار می‌رفت میانگین مدت زمان دریافت غذای مشتریان حرکت نوسانی دارد و نمی‌توان رابطه‌ای بین تعداد صندلی‌های سالن غذاخوری و زمان دریافت غذا در نظر گرفت.



نمودار ۲۱

نمودار فوق نشان می‌دهد که شاخص‌های مذکور مهمترین شاخص‌های مربوط به تعداد صندلی‌ها می‌باشد. کاهش تعداد صندلی‌ها به طور چشمگیری شرایط صف سالن را تضعیف می‌کند و باعث می‌شود که میانگین طول صف حتی تا ۶۵ واحد افزایش پیدا می‌کند. شرایط برای شاخص حداکثر طول صف نیز به همین صورت می‌باشد. بنابراین به نظر می‌آید که باید از کاهش صندلی‌ها پیشگیری کنیم و حتی اگر نیاز به چنین اقدامی داریم نباید این کاهش بیش از ۵ صندلی باشد.



نمودار ۲۲

همانطور که انتظار می‌رفت نسبت‌ها مربوط به بازده کارمندان نباید رابطه‌ی معنی‌داری با تعداد صندلی‌های سالن داشته باشد و نتایج نیز به همین صورت است. اما نسبت گردش با کاهش صندلی‌ها افزایش چشمگیری دارد

و حتی از مقدار مبنی یعنی ۱ نیز بیشتر می شود که بیانگر نارضایتی مشتریان است. بنابراین کاهش تعداد صندلی ها اصلاً اقدام مناسبی نمی باشد.

بررسی ها به صورت کلی نشان می دهد کاهش تعداد صندلی ها در صورت نیاز تا ۵ عدد قابل اجرا می باشد و آسیبی به سیستم وارد نمی شود. اما همانطور که در ابتدا نیز ذکر شد افزایش تعداد صندلی ها سودی برای رستوران نخواهد داشت.

در پایان این بخش ذکر این نکته ضروری است که بررسی ها انجام شده به صورت ایزوله و در شرایط شبیه سازی انجام شده اند و همواره باید به این نکته توجه کرد که این تغییرات در سیستم اثرات جانبی به همراه دارد و به همین دلیل نمی تواند در مورد تغییرات مذکور به صورت قاطعانه نظر داد اما خروجی های مذکور می تواند دید مناسبی از شرایط سیستم را برای ما ایجاد کند.

تعیین برآورد فاصله های و نقطه های

خروجی های انتخاب شده در ایبن بخش، میانگین مدت ماندن مشتری در سیستم، میانگین مدت انتظار مشتری جهت دریافت غذا و میانگین بهره وری کارکنان قسمت پذیرش که به ترتیب با TimeSpent، ReceivingTime، RUtil و KUtil نشان داده شده اند هستند. محاسبات این بخش در اکسل انجام شده است.

برای تعیین برآورد نقطه ای، خروجی های مذکور برای ۵ تکرار مجزا محاسبه شدند که نتایج آن در جدول زیر قابل مشاهده است. سپس میانگین نتایج هر کدام به صورت مجزا تحت عنوان \bar{Y} به دست آمده است و این مقدار برآورد نقطه ای خروجی مد نظر را به ما نشان می دهد.

R	Across-Rep Data			
	TimeSpent	ReceivingTime	RUtil	Kutil
1	40.173208	1.616067	0.750841	0.525252
2	38.615792	1.576181	0.780569	0.52979
3	35.981477	1.584905	0.799434	0.53979
4	32.067689	1.716872	0.717635	0.478415
5	39.160109	1.641003	0.880636	0.554858

جدول ۴ - برآورد نقطه ای و فاصله ای

برآورد نقطه ای در نهایت به صورت زیر خواهد بود:

	TimeSpent	ReceivingTime	RUtil	Kutil
$\bar{Y}_{..}$	37.1997	1.6270	0.7858	0.5256

جدول ۵ - برآورد نقطه‌ای

برای محاسبه‌ی برآورد فاصله‌ای از روابط زیر استفاده شده است:

$$\bar{Y}_{..} = \frac{1}{5} \sum_{i=1}^5 \bar{Y}_i$$

$$S^2 = \frac{1}{4} \sum_{i=1}^4 (\bar{Y}_i - \bar{Y}_{..})^2$$

$$H = \frac{S}{\sqrt{5}} \times t_{\alpha/2,4}$$

نتایج به دست آمده به شرح زیر هستند و Left ابتدای بازه و Right انتهای بازه را نشان می‌دهد:

	TimeSpent	ReceivingTime	RUtil	Kutil
S^2	10.6280	0.0032	0.0038	0.0008
S	3.2601	0.0565	0.0614	0.0287
H	4.0479	0.0701	0.0762	0.0357
Left	33.1518	1.5569	0.7096	0.4899
Right	41.2475	1.6971	0.8621	0.5613

جدول ۶ - برآورد فاصله‌ای

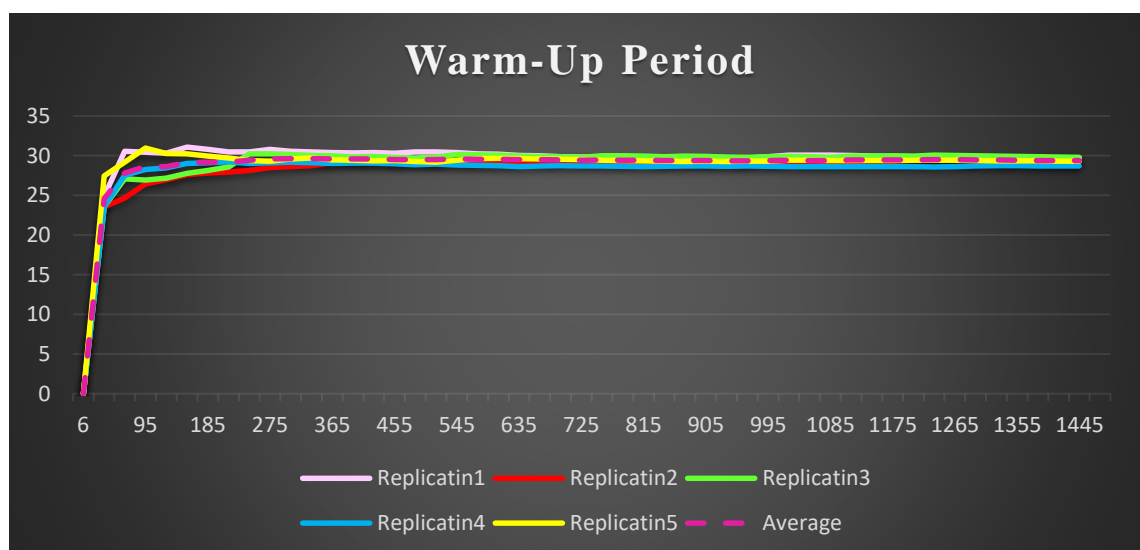
بررسی وجود و شناسایی دوره‌ی سرد و گرم سیستم

در این حالت فرض کنید اتوبوس وجود ندارد و همچنین در واقعیت فست فود شبانه‌روزی است. برای این حالت نیز Replications=5 در نظر بگیرید.

برای شناسایی این مورد نیز از اکسل استفاده شده است و جزئیات محاسبات در فایل‌های پیوست آمده است.

در ابتدا با توجه به این که توزیع‌های ورود به سیستم، زمان‌های خدمت‌رسانی، زمان صرف غذا و سایر پارامترها مانند تعداد کل سرورها، فاصله‌ها و رفتار کلی مشتری‌ها در طول زمان بررسی ثابت است و از لحاظ قواعد، قانون و فرایند، تغییری در سیستم رخ نمی‌دهد به نظر می‌رسد که سیستم بعد از گذراندن دوره‌ی سرد، به دوره‌ی گرم خود برسد. در ادامه این فرضیه را به کمک نمودارهای رسم شده مورد بررسی قرار می‌دهیم.

نتایج به دست آمده نشان می‌دهند که در ابتدا در نمودارها یک روند صعودی داریم که بعد از مدتی به ثبات می‌رسد. بنابراین این سیستم دارای دوره‌ی سرد و گرم می‌باشد که در ادامه بیشتر به آن می‌پردازیم.



نمودار ۲۳

مبنای سنجش دوره‌ی سرد و گرم در این بخش، میانگین مدت ماندن مشتری در سیستم در نظر گرفته شده است و برای زمان‌های ۳۰ دقیقه‌ای این مقدار را ثبت می‌کنیم. این کار را ۵ بار تکرار می‌کنیم که نمودار هر تکرار در شکل بالا دیده می‌شود. در نهایت از داده‌های به دست آمده میانگین می‌گیریم که نمودار آن با خط‌چین نشان داده شده است. با توجه به این که داده‌ها نویز زیادی ندارند استفاده از میانگین پاسخ مناسبی در اختیار ما خواهد گذاشت. در نهایت می‌توان گفت که به صورت حدودی تا زمان ۱۰۰ دقیقه دوره‌ی سرد سیستم است و پس از آن دوره‌ی گرم آغاز می‌شود. در واقع نقطه‌ی برش برای این سیستم را ۱۰۰ در نظر می‌گیریم.

پیوست اول: فایل اکسل

در این پیوست بخش کوچکی از خروجی فایل اسکل جهت نمایش آورده شده است:

STEP	CURRENT EVENT	CLOCK	ORDER QUEUE LENGTH	RECEIVE QUEUE LENGTH	SEAT QUEUE LENGTH	EMPTY SEATS
1	Start of Simulation	0	0	0	0	30
2	Entrance On foot	0.285707246	0	0	0	30
3	Entrance On foot	0.403944281	0	0	0	30
4	Entrance On foot	0.680715076	0	0	0	30
5	Entrance On foot	0.713896938	0	0	0	30
6	End of Reception Service	4.070524305	0	0	0	30
7	End of Reception Service	4.830813207	0	0	0	30
8	End of Reception Service	4.878766831	0	0	0	30
9	Start of Recieving Food	4.887381588	0	0	0	30
10	End of Reception Service	5.038435164	0	0	0	30

11	Start of Recieving Food	5.16040473	0	0	0	30
12	Start of Recieving Food	5.238792558	0	1	0	30
13	Entrance On foot	5.375203042	0	1	0	30
14	Start of Recieving Food	5.550398516	0	2	0	30
15	Recieving Food	5.886232579	0	1	0	30
16	Start of Eating Food	6.025882042	0	1	0	29
17	Entrance By Car	6.469724013	0	1	0	29
18	Entrance On foot	6.571903859	0	1	0	29
19	Recieving Food	6.664628105	0	0	0	29
20	Start of Eating Food	6.778499856	0	0	0	28

پیوست دوم: کد پایتون

```
# Simulation of fast food restaurant:

# There exists 3 different forms of entrance:
# 1.Entering on foot dist.~ Negative exponential with 3 min as mean
#First entrance after 10AM

# 2.Entering by car dist.~ Negative exponential with mean = 5 min #First
entrance after 10AM
# Number of passengers in a car      Probability
#           1                        0.2
#           2                        0.3
#           3                        0.3
#           4                        0.2
# 3.Entering by bus dist.~Uniform dist.[11AM,13PM] #There is only one bus
# Number of passengers in the bus dist.~ Poisson with 30 person per hour
as mean
# Ordering food dist.~triangular[1,2,4]
# Payment dist.~[1,2,3]
# Receiving food dist.~uniform dist. [0.5,2]
# Eating food dist.~triangular[10,20,30]
# Travel time dist between parts except "Exit and salon".~Negative
exponential with 0.5min as mean
# Travel time dist "Exit and salon" parts.~Negative exponential with 1min
as mean
# People get service in a FIFO system
# No limit on any Queue's length
# Number of receptionists = 5 , Number of kitchen staff = 2
# Determined rest times:10:50, 11:50, 13:50, 14:50 #Due to circumstances
these times might change
# Rest duration = 10 min

# Outputs : 1- Customer Time Spent in System's Mean
#           2- Time waiting to Receive food Mean
#           3- Max of Seat queue's length
#           4- Seat queue's length Mean
#           5- Receptionist's efficiency mean
#           6- Kitchen staff's efficiency mean
#           7-our output
# Starting State = System is empty and servers are idle
```

```

# importing the libraries
import numpy as np
import xlswriter as xs
import pandas as pd
import matplotlib.pyplot as plt

# Generating exponential random number
ExpRandom = lambda y: -(1 / y) * (np.log(np.random.random()))

# Generating uniform random number
UniRandom = lambda a, b: a + (b - a) * (np.random.random())

# Generating triangular random number
TriRandom = lambda a, c, b: a + np.sqrt((b - a) * (c - a) *
np.random.random()) \
    if np.random.random() < (c - a) / (b - a) else b - np.sqrt((b - a) *
(b - c) * (np.random.random()))

# Generating poisson random number
def PosRandom(alpha):
    P = 1
    i = -1
    while np.exp(-alpha) <= P:
        P = P * np.random.random()
        i += 1
    return i

# Calculating the number of people in a car
def CarRandom():
    Rnd = np.random.random()
    if Rnd < 0.2:
        i = 1
    elif Rnd < 0.5 and Rnd >= 0.2:
        i = 2
    elif Rnd < 0.8 and Rnd >= 0.5:
        i = 3
    else:
        i = 4
    return i

# This function shows our initial state
def starting_state():
    state = dict()
    # These are the state variables of this simulation
    state['Order Queue Length'] = dict()
    state['Receive Queue Length'] = dict()
    state['Empty Seats'] = 30
    state['Seat Queue Length'] = dict()
    state['Idle Receptionists'] = 5

```

```

state['Idle Kitchen Staff'] = 2
state['Resting Receptionists'] = 0
state['Resting Kitchen Staff'] = 0
state['WRr'] = 0
# number of receptionists waiting to rest which is a binary variable
state['WRk'] = 0
# number of kitchen staff waiting to rest which is a binary variable
state['Number of Current Customers'] = 0

# Data Collecting Dict: saves the times below for each customer
# t0- Entrance, t1- Start of reception process, t2- End of reception
process
# t3- Entering kitchen queue, t4-Start of Receiving food, t5- End of
receiving food
# t6- Entering seat queue, t7- Start of Eating food, t8- End of
eating food, t9- Exit
data = dict()
data['Event Clock'] = 0 # The event clock
data['Customers'] = dict()
# The customer {'Ci':[t0,t1,t2,t3,t4,t5,t6,t7,t8,t9]}

# Cumulative statistics
cum_stat = dict()
# cum_stat records history cumulatively from the beginning till a
given time
cum_stat['Order Queue Length'] = 0
cum_stat['Receive Queue Length'] = 0
cum_stat['Seat Queue Length'] = 0
cum_stat['Order Queue Waiting Time'] = 0
cum_stat['Receive Queue Waiting Time'] = 0
cum_stat['Seat Queue Waiting Time'] = 0
cum_stat['Receiving Food Waiting Time'] = 0
# It consists of the time spent in the Receive queue
# and the time it takes for the food to be prepared by the kitchen
staff

cum_stat['Total Number of Customers'] = 0
# Total number of customers from the beginning till a specific time
cum_stat['Receptionists Busy Time'] = 0
# For all 5 servers
cum_stat['Kitchen staff Busy Time'] = 0
# For both servers

cum_stat['Customer Time Spent in System'] = 0
# For all the customers till a specific time

# At the beginning, the first entrance on foot,by car and by bus
should be generated as C1,Car1 and B1
future_event_list = list()
FEL_maker(future_event_list, 'Entrance On foot', 0, 'C1')
# Difference: make an entrance of specific customer (C1)
FEL_maker(future_event_list, 'Entrance By Car', 0, 'Car1')
# Difference: make an entrance of specific Car (Car1)
FEL_maker(future_event_list, 'Entrance By Bus', 0, 'B1')

```

```

    # All predefined rest times are converted into minutes and appended in
    the FEL at the beginning.
    # The 'RR' and the 'RK' show that we are talking about server's rest.
    future_event_list.append({'Event Type': 'Start of Receptionist
Resting', 'Event Time': 50, 'Customer': 'RR'})
    future_event_list.append({'Event Type': 'Start of Receptionist
Resting', 'Event Time': 110, 'Customer': 'RR'})
    future_event_list.append({'Event Type': 'Start of Receptionist
Resting', 'Event Time': 230, 'Customer': 'RR'})
    future_event_list.append({'Event Type': 'Start of Receptionist
Resting', 'Event Time': 290, 'Customer': 'RR'})
    future_event_list.append({'Event Type': 'Start of Kitchen staff
Resting', 'Event Time': 50, 'Customer': 'RK'})
    future_event_list.append({'Event Type': 'Start of Kitchen staff
Resting', 'Event Time': 110, 'Customer': 'RK'})
    future_event_list.append({'Event Type': 'Start of Kitchen staff
Resting', 'Event Time': 230, 'Customer': 'RK'})
    future_event_list.append({'Event Type': 'Start of Kitchen staff
Resting', 'Event Time': 290, 'Customer': 'RK'})

    return state, data, future_event_list, cum_stat

# This function makes the FEL.
def FEL_maker(future_event_list, event_type, clock, customer):
    global simulation_time
    event_time = 0
    if event_type == "Entrance On foot":
        event_time = clock + ExpRandom(1 / 3)
        if event_time > simulation_time:
            # It prevents another entrance on foot after 15:00 which is
300 min
            return
    elif event_type == "Entrance By Car":
        event_time = clock + ExpRandom(1 / 5)
        if event_time > simulation_time:
            # It prevents another entrance by car after 15:00 which is 300
min
            return
    elif event_type == "Entrance By Bus":
        event_time = clock + UniRandom(60, 180)
        if event_time > simulation_time:
            # It prevents another entrance by bus after 15:00 which is 300
min
            return
    elif event_type == "End of Reception Service":
        event_time = clock + TriRandom(1, 2, 4) + TriRandom(1, 2, 3)
    elif event_type == "Start of Receiving Food":
        event_time = clock + ExpRandom(2)
    elif event_type == "Receiving Food":
        event_time = clock + UniRandom(0.5, 2)
    elif event_type == "Start of Eating Food":
        event_time = clock + ExpRandom(2)

```

```

elif event_type == "End of Eating Food":
    event_time = clock + TriRandom(10, 20, 30)
elif event_type == "Exit":
    event_time = clock + ExpRandom(1)
elif event_type == 'End of Receptionist Resting':
    event_time = clock + 10
elif event_type == "End of Kitchen staff Resting":
    event_time = clock + 10
new_event = {'Event Type': event_type,
             'Event Time': event_time, 'Customer': customer}
# As mentioned above, for resting events, customer component is equal
to 'RR'.
# additional element in event notices (Customer No.)
future_event_list.append(new_event)

def Entrance_On_Foot(future_event_list, state, data, clock, customer,
cum_stat):
    state['Number of Current Customers'] += 1
    # Number of customers who are still in the system
    cum_stat['Total Number of Customers'] += 1
    data['Customers'][customer] = []
    # Add a place for the new customer

    if state['Idle Receptionists'] > 0:
        data['Customers'][customer].extend([clock, clock])
        # In this case, the time in which the customer enters the queue is
equal to
        # the time he/she leaves the queue. t0 = t1 = clock
        state['Idle Receptionists'] -= 1
        # Make server busy
        FEL_maker(future_event_list, "End of Reception Service", clock,
customer)
        # determine when this customer's service ends.
    else:
        state['Order Queue Length'][customer] = clock
        data['Customers'][customer].append(clock)
        # t0
        data['Event Clock'] = clock
        # put the current clock on the last event clock for the next event

        # Extracting the customer number
        customer_num = cum_stat['Total Number of Customers'] + 1
        # Every customer needs a label to be individually recognized
        FEL_maker(future_event_list, 'Entrance On foot', clock, 'C' +
str(customer_num))
        # predict the next customer's Arrival

def Entrance_By_Car(future_event_list, state, data, clock, car, cum_stat):
    Car_Customers = CarRandom()
    # The number of customers in the car
    customer = []
    for i in np.arange(cum_stat['Total Number of Customers'] + 2,

```

```

        cum_stat['Total Number of Customers'] +
Car_Customers + 2, 1):
    customer.append('C' + str(i))
    # This loop labels the customers of the car. +2 is to prevent
    repeating customer labels

    cum_stat['Total Number of Customers'] += Car_Customers
    state['Number of Current Customers'] += Car_Customers

    for i in np.arange(0, Car_Customers, 1):
        data['Customers'][customer[i]] = []
        # Checking whether the server is busy or not
        if state['Idle Receptionists'] > 0:
            data['Customers'][customer[i]].extend([clock, clock]) # t0 =
t1 = clock
            state['Idle Receptionists'] -= 1 # Make a server busy
            FEL_maker(future_event_list, "End of Reception Service",
clock,
                    customer[i]) # Determine when this customer's
service ends.
        else:
            state['Order Queue Length'][customer[i]] = clock
            data['Customers'][customer[i]].append(clock) # t0

    data['Event Clock'] = clock
    # put the current clock on the last event clock for the next event

    # Extracting the car number
    car_num = int(car[3:])
    car_num += 1
    FEL_maker(future_event_list, 'Entrance By Car', clock, 'Car' +
str(car_num))
    # predict the next car's Arrival

def Entrance_By_Bus(future_event_list, state, data, clock, bus, cum_stat):
    Bus_Customers = PosRandom(30)
    # The number of customers in the bus
    customer = []
    for i in np.arange(cum_stat['Total Number of Customers'] + 2,
                        cum_stat['Total Number of Customers'] +
Bus_Customers + 2, 1):
        customer.append('C' + str(i))
        # This loop labels the customers of the bus. +2 is to prevent
        repeating customer labels

        cum_stat['Total Number of Customers'] += Bus_Customers
        state['Number of Current Customers'] += Bus_Customers

        for i in np.arange(0, Bus_Customers, 1):
            data['Customers'][customer[i]] = []
            # Checking whether the server is busy or not
            if state['Idle Receptionists'] > 0:
                data['Customers'][customer[i]].extend([clock, clock])

```

```

        # t0 = t1 = clock
        state['Idle Receptionists'] -= 1
        # Make server busy
        FEL_maker(future_event_list, "End of Reception Service",
clock,
                    customer[i]) # Determine when this customer's
service ends.
    else:
        state['Order Queue Length'][customer[i]] = clock
        data['Customers'][customer[i]].append(clock) # t0

data['Event Clock'] = clock
# put the current clock on the last event clock for the next event

def End_of_Reception_Service(future_event_list, state, data, clock,
customer, cum_stat):
    FEL_maker(future_event_list, "Start of Receiving Food", clock,
customer)
    # Forward look to generate the time to start receiving food
    data['Customers'][customer].append(clock)
    # t2 for the customer
    if state['WRr'] > 0:
        state['WRr'] -= 1
        state['Resting Receptionists'] += 1
        FEL_maker(future_event_list, "End of Receptionist Resting", clock,
"RR")
    else:
        if len(state['Order Queue Length']) > 0:
            # Accessing the first person in the queue
            first_in_queue = min(state['Order Queue Length'], key=lambda
k: state['Order Queue Length'][k])
            data['Customers'][first_in_queue].append(clock) # t1

            cum_stat["Order Queue Waiting Time"] += (
                data['Customers'][first_in_queue][1] -
data['Customers'][first_in_queue][0])
            FEL_maker(future_event_list, "End of Reception Service",
clock, first_in_queue)
            del state['Order Queue Length'][first_in_queue]
        else:
            state['Idle Receptionists'] += 1
            data['Event Clock'] = clock

def Start_of_Receiving_Food(future_event_list, state, data, clock,
customer, cum_stat):
    if state['Idle Kitchen Staff'] > 0:
        state['Idle Kitchen Staff'] -= 1
        data['Customers'][customer].extend([clock, clock]) # t3 = t4
        FEL_maker(future_event_list, "Receiving Food", clock, customer)
    else:
        state['Receive Queue Length'][customer] = clock
        data['Customers'][customer].append(clock) # Queue entering time

```

```

or t3
    data['Event Clock'] = clock

def Receiving_Food(future_event_list, state, data, clock, customer,
cum_stat):
    cum_stat['Receiving Food Waiting Time'] += clock -
data['Customers'][customer][3] # clock - t3
    FEL_maker(future_event_list, "Start of Eating Food", clock, customer)
    data['Customers'][customer].append(clock)
    # Appending t5 to customer's times
    if state['WRk'] > 0:
        state['WRk'] -= 1
        state['Resting Kitchen Staff'] += 1
        FEL_maker(future_event_list, "End of Kitchen staff Resting",
clock, "RK")
    else:
        if len(state['Receive Queue Length']) > 0:
            first_in_queue = min(state['Receive Queue Length'], key=lambda
k: state['Receive Queue Length'][k])
            data['Customers'][first_in_queue].append(clock)
            cum_stat["Receive Queue Waiting Time"] += (
                data['Customers'][first_in_queue][4] -
data['Customers'][first_in_queue][3])
            FEL_maker(future_event_list, "Receiving Food", clock,
first_in_queue)
            del state['Receive Queue Length'][first_in_queue]
            # Bringing out the first person in the queue
        else:
            state['Idle Kitchen Staff'] += 1
            data['Event Clock'] = clock

def Eating_Start(future_event_list, state, data, clock, customer,
cum_stat):
    if state['Empty Seats'] > 0:
        state['Empty Seats'] -= 1
        data['Customers'][customer].extend([clock, clock]) # t6 = t7
        FEL_maker(future_event_list, "End of Eating Food", clock,
customer)
    else:
        state['Seat Queue Length'][customer] = clock
        data['Customers'][customer].append(clock) # t6
        data['Event Clock'] = clock

Seat_Queue_Length = []
def Eating_End(future_event_list, state, data, clock, customer, cum_stat):
    FEL_maker(future_event_list, "Exit", clock, customer)
    data['Customers'][customer].append(clock) # t8
    global Seat_Queue_Length
    Seat_Queue_Length.append(len(state['Seat Queue Length']))
    if len(state['Seat Queue Length']) > 0:
        first_in_queue = min(state['Seat Queue Length'], key=lambda k:

```



```

state['Seat Queue Length'][k])
    data['Customers'][first_in_queue].append(clock)

    cum_stat["Seat Queue Waiting Time"] += (
        data['Customers'][first_in_queue][7] -
data['Customers'][first_in_queue][6])
    FEL_maker(future_event_list, "End of Eating Food", clock,
first_in_queue)
    del state['Seat Queue Length'][first_in_queue]
    # Bringing out the first person in the queue
else:
    state['Empty Seats'] += 1
    data['Event Clock'] = clock

def Exit(future_event_list, state, data, clock, customer, cum_stat):
    data['Customers'][customer].append(clock) # t9
    state['Number of Current Customers'] -= 1
    # A customer gets out of the system
    cum_stat["Customer Time Spent in System"] +=
(data['Customers'][customer][9] - data['Customers'][customer][0])
    # t9-t0
    data['Event Clock'] = clock

def Start_of_Receptionists_Resting(future_event_list, state, data, clock,
customer, cum_stat):
    if state['Idle Receptionists'] > 0:
        state['Resting Receptionists'] += 1
        FEL_maker(future_event_list, "End of Receptionist Resting", clock,
customer)
        state['Idle Receptionists'] -= 1
        # Sending a receptionist to get some rest
    else:
        state['WRr'] += 1
        # If no server is idle to go to rest, the number of servers
waiting for a rest increases.
        data['Event Clock'] = clock

def Start_of_kitchen_staff_Resting(future_event_list, state, data, clock,
customer, cum_stat):
    if state['Idle Kitchen Staff'] > 0:
        state['Resting Kitchen Staff'] += 1
        FEL_maker(future_event_list, "End of Kitchen staff Resting",
clock, customer)
        state['Idle Kitchen Staff'] -= 1
    else:
        state['WRk'] += 1
        # If no server is idle to go to rest, the number of servers
waiting for a rest increases.
        data['Event Clock'] = clock

```

```

def End_of_Receptionists_Resting(future_event_list, state, data, clock,
customer, cum_stat):
    if len(state['Order Queue Length']) > 0:
        first_in_queue = min(state['Order Queue Length'], key=lambda k:
state['Order Queue Length'][k])
        data['Customers'][first_in_queue].append(clock)
        FEL_maker(future_event_list, "End of Reception Service", clock,
first_in_queue)
        del state['Order Queue Length'][first_in_queue]
        state['Resting Receptionists'] -= 1
    else:
        state['Idle Receptionists'] += 1
        state['Resting Receptionists'] -= 1
    data['Event Clock'] = clock

def End_of_kitchen_staff_Resting(future_event_list, state, data, clock,
customer, cum_stat):
    if len(state['Receive Queue Length']) > 0:
        first_in_queue = min(state['Receive Queue Length'], key=lambda k:
state['Receive Queue Length'][k])
        data['Customers'][first_in_queue].append(clock)
        FEL_maker(future_event_list, "Receiving Food", clock,
first_in_queue)
        del state['Receive Queue Length'][first_in_queue]
        state['Resting Kitchen Staff'] -= 1
    else:
        state['Idle Kitchen Staff'] += 1
        state['Resting Kitchen Staff'] -= 1
    data['Event Clock'] = clock

def End_of_Simulation(future_event_list, state, data, clock, customer,
cum_stat, max_Seat_Queue_Length):
    Order_Queue_Waiting_Time = 0
    Receive_Queue_Waiting_Time = 0
    Number_Of_Total_Customers = 0
    Order_Queue_Waiting_Time += cum_stat["Order Queue Waiting Time"]
    Receive_Queue_Waiting_Time += cum_stat['Receive Queue Waiting Time']
    Number_Of_Total_Customers += cum_stat['Total Number of Customers']
    #In the lines above, the values of the cum stats are stored at the end
of the simulation time

    Time_Spent = 0
    for i in data['Customers'].keys():
        # data['Customers'].keys() provides the times allocated to each
customer
        if len(data['Customers'][i]) == 10: # checking whether the
customer is still in the system or not by checking whether "Exit" time has
been generated or not
            Time_Spent += data['Customers'][i][9] -
data['Customers'][i][0]

        #Calculating the number of customers who are not in the system anymore
        Number_Of_Completely_Served_Customers = 0

```

```

    Number_Of_Completely_Served_Customers += cum_stat['Total Number of
Customers'] - state['Number of Current Customers']

    #Calculating the number of customers which the process of receiving
food has been completed for them with their waiting time.
    Number_Of_Customers_Received_Food = 0
    Receiving_Food_Waiting_Time = 0
    for i in data['Customers'].keys():
        # data['Customers'].keys() provides the times allocated to each
customer
        if len(data['Customers'][i]) >= 6: # checking whether the customer
has received his/her food or not
            Receiving_Food_Waiting_Time += data['Customers'][i][5] -
data['Customers'][i][3]
            Number_Of_Customers_Received_Food += 1

    #Calculating the length and the max length of the seat queue.
    max_Seat_Queue = 0
    max_Seat_Queue += max_Seat_Queue_Length
    Seat_Queue_Length = cum_stat['Seat Queue Length']
    if len(state['Seat Queue Length']) > 0:
        max_Seat_Queue = max(max_Seat_Queue, len(state['Seat Queue
Length']))

    Receptionists_Busy_Time = 0
    Receptionists_Busy_Time += cum_stat['Receptionists Busy Time']
    Kitchen_staff_Busy_Time = 0
    Kitchen_staff_Busy_Time += cum_stat['Kitchen staff Busy Time']

    return Number_Of_Total_Customers,
Number_Of_Completely_Served_Customers, Number_Of_Customers_Received_Food,
Order_Queue_Waiting_Time, Receive_Queue_Waiting_Time, Time_Spent,
Receiving_Food_Waiting_Time, Seat_Queue_Length, max_Seat_Queue,
Receptionists_Busy_Time, Kitchen_staff_Busy_Time

def output_excel(worksheet, future_event_list, state, row_num):
    global max_fel
    global header_list
    # we update the header list in this function
    future_event_list = sorted(future_event_list, key=lambda x: x['Event
Time'])
    # print(future_event_list)
    new_row = [row_num, future_event_list[0]['Event Type'],
future_event_list[0]['Event Time'],
len(state['Order Queue Length']), len(state['Receive Queue
Length']),
len(state['Seat Queue Length']), state['Empty Seats'],
state['Idle Receptionists'], state['Idle Kitchen Staff'],
state['Resting Receptionists'],
state['Resting Kitchen Staff'], state['WRr'], state['WRk'],
state['Number of Current Customers']]
    # Creating new row

```

```

# Update the header list and max_fel
if len(future_event_list) - 1 > max_fel:
    for fel_counter in range(max_fel, len(future_event_list) - 1):
        header_list.extend(
            ("Future Event Type " + str(fel_counter + 1), "Future
Event Time " + str(fel_counter + 1)))
        max_fel = len(future_event_list) - 1

    else:
        for add_number in range(max_fel - len(future_event_list) + 1):
            future_event_list.append({"Event Type": "", "Event Time": ""})

    for fel in future_event_list[1:]:
        new_row.extend((fel['Event Type'], fel['Event Time']))
    for col in range(len(header_list)):
        worksheet.write(0, col, header_list[col])
        worksheet.write(row_num, col, new_row[col])

    return worksheet

#Setting the excel format
def excel_formatting(workbook, worksheet, row_num):
    cell_format_header = workbook.add_format()
    cell_format_header.set_align('center')
    cell_format_header.set_align('vcenter')
    cell_format_header.set_font('Times New Roman')
    cell_format_header.set_bold(True)
    worksheet.set_row(0, None, cell_format_header)

    worksheet.set_column(0, 0, 5)
    worksheet.set_column(1, 1, 13)
    worksheet.set_column(2, 2, 9)
    worksheet.set_column(3, 4, 8)
    worksheet.set_column(5, 4 + 2 * max_fel, 19)
    cell_format = workbook.add_format()
    cell_format.set_align('center')
    cell_format.set_font('Times New Roman')
    for row in range(row_num):
        worksheet.set_row(row + 1, None, cell_format)

    return workbook

def simulation(simulation_time):
    state, data, future_event_list, cum_stat = starting_state()
    # Initial state
    max_Seat_Queue_Length = 0
    clock = 0
    check = 6 #This variable is for the warm-up period detection part to
prevent the denominator to be 0
    mem = [] # A list for the warm-up period detection part which holds
the avg of customer spent time.
    memc = [] # A list which stores the times in the warm-up period
detection part

```

```

future_event_list.append({'Event Type': 'End of Simulation', 'Event
Time': simulation_time, 'Customer': ''})
future_event_list.append({'Event Type': 'Start of Simulation', 'Event
Time': 0, 'Customer': ''})

workbook = xs.Workbook('Simulation_Project.xlsx')
worksheet = workbook.add_worksheet('Restaurant')
row_num = 1

# The simulation continues til the last customer in the restaurant
exits.
# The point to make here is that the restaurant will not accept any
new customers after 15:00
while clock < simulation_time or state['Number of Current Customers']
> 0:
    sorted_fel = sorted(future_event_list, key=lambda x: x['Event
Time'])
    #print("sorted fel=")
    #print(sorted_fel)
    current_event = sorted_fel[0]
    # The first element is the thing that's happening now
    clock = current_event['Event Time']
    # Move the time forward
    # print(current_event['Event Type'])
    cum_stat['Order Queue Length'] += len(state['Order Queue Length'])
* (clock - data['Event Clock'])
    cum_stat['Receive Queue Length'] += len(state['Receive Queue
Length']) * (clock - data['Event Clock'])
    cum_stat['Seat Queue Length'] += len(state['Seat Queue Length']) *
(clock - data['Event Clock'])

    temp = len(state['Seat Queue Length'])
    max_Seat_Queue_Length = max(temp, max_Seat_Queue_Length)

    cum_stat["Receptionists Busy Time"] += (5 - state['Idle
Receptionists'] - state['Resting Receptionists']) * (
        clock - data['Event Clock'])

    cum_stat["Kitchen staff Busy Time"] += (2 - state['Idle Kitchen
Staff'] - state['Resting Kitchen Staff']) * (
        clock - data['Event Clock'])

    if current_event['Event Type'] == 'End of Simulation':
        Number_Of_Total_Customers,
Number_Of_Completely_Served_Customers, Number_Of_Customers_Received_Food,
Order_Queue_Waiting_Time, Receive_Queue_Waiting_Time, Time_Spent,
Receiving_Food_Waiting_Time, Seat_Queue_Length, max_Seat_Queue,
Receptionists_Busy_Time, Kitchen_staff_Busy_Time =
End_of_Simulation(future_event_list, state, data, clock,
current_customer, cum_stat, max_Seat_Queue_Length)

    # print("state=")
    # print(state)
    # print("cum_stat=")

```

```

# print(cum_stat)
# print("data=")
# print(data)

if clock < simulation_time or state['Number of Current Customers']
> 0:
    current_customer = current_event['Customer']
    if current_event['Event Type'] == 'Entrance On foot':
        Entrance_On_Foot(future_event_list, state, data, clock,
current_customer, cum_stat)
    elif current_event['Event Type'] == 'Entrance By Car':
        Entrance_By_Car(future_event_list, state, data, clock,
current_customer, cum_stat)
    elif current_event['Event Type'] == 'Entrance By Bus':
        Entrance_By_Bus(future_event_list, state, data, clock,
current_customer, cum_stat)
    elif current_event['Event Type'] == 'End of Reception
Service':
        End_of_Reception_Service(future_event_list, state, data,
clock, current_customer, cum_stat)
    elif current_event['Event Type'] == 'Start of Receiving Food':
        Start_of_Receiving_Food(future_event_list, state, data,
clock, current_customer, cum_stat)
    elif current_event['Event Type'] == 'Receiving Food':
        Receiving_Food(future_event_list, state, data, clock,
current_customer, cum_stat)
    elif current_event['Event Type'] == 'Start of Eating Food':
        Eating_Start(future_event_list, state, data, clock,
current_customer, cum_stat)
    elif current_event['Event Type'] == 'End of Eating Food':
        Eating_End(future_event_list, state, data, clock,
current_customer, cum_stat)

    elif current_event['Event Type'] == 'Start of Receptionist
Resting':
        Start_of_Receptionists_Resting(future_event_list, state,
data, clock, current_customer, cum_stat)
    elif current_event['Event Type'] == 'Start of Kitchen staff
Resting':
        Start_of_kitchen_staff_Resting(future_event_list, state,
data, clock, current_customer, cum_stat)

    elif current_event['Event Type'] == 'End of Receptionist
Resting':
        End_of_Receptionists_Resting(future_event_list, state,
data, clock, current_customer, cum_stat)
    elif current_event['Event Type'] == 'End of Kitchen staff
Resting':
        End_of_kitchen_staff_Resting(future_event_list, state,
data, clock, current_customer, cum_stat)

    elif current_event['Event Type'] == 'Exit':
        Exit(future_event_list, state, data, clock,
current_customer, cum_stat)

```

```

        output_excel(worksheet, future_event_list, state, row_num)
        row_num += 1
        future_event_list.remove(current_event)
        if clock >= check:
            if cum_stat['Total Number of Customers'] - state['Number
of Current Customers'] == 0:
                mem.append(cum_stat['Customer Time Spent in System'] /
state['Number of Current Customers'])
                memc.append(clock)
                check += 30 #This part is for detecting warm-up period
and checks the times each 30 min
            else:
                mem.append(cum_stat['Customer Time Spent in System'] /
(cum_stat['Total Number of Customers'] - state['Number of Current
Customers']))
                memc.append(clock)
                check += 30

    # outputs
    #This part is more detailed in the excel file provided for the warm-up
period detection
    plt.plot(memc, mem, c='b')
    plt.xlabel('Clock')
    plt.ylabel('Length')
    plt.show()

    # simulation time outputs
    # Output1
    Customer_Time_Spent_1 = Time_Spent /
Number_Of_Completely_Served_Customers
    print("Customer_Time_Spent_1")
    print(Customer_Time_Spent_1)
    # Output2
    Receiving_Food_Waiting_Time_1 = Receiving_Food_Waiting_Time /
Number_Of_Customers_Received_Food
    print("Receiving_Food_Waiting_Time_1")
    print(Receiving_Food_Waiting_Time_1)
    # Output3
    Seat_Queue_Length_1 = Seat_Queue_Length / simulation_time
    print("Seat_Queue_Length_1")
    print(Seat_Queue_Length_1)
    max_Seat_Queue_1 = max_Seat_Queue
    print("max_Seat_Queue_1")
    print(max_Seat_Queue_1)
    # Output4
    Receptionists_util_1 = Receptionists_Busy_Time / (simulation_time * 5)
    print("Receptionists_util_1")
    print(Receptionists_util_1)
    KitchenStaff_util_1 = Kitchen_staff_Busy_Time / (simulation_time * 2)
    print("KitchenStaff_util_1")
    print(KitchenStaff_util_1)
    # Output5
    Turnover_Ratio_1 = simulation_time / Number_Of_Total_Customers

```

```

print("Turnover_Ratio_1")
print(Turnover_Ratio_1)

# outputs till the restaurant gets empty
#Note that since the results were almost the same as the outputs
above, these parts do not have a "print" section.
# Output1
Customer_Time_Spent_2 = cum_stat['Customer Time Spent in System'] /
len(data['Customers'])
# Output2
Receiving_Food_Waiting_Time_2 = cum_stat['Receiving Food Waiting
Time'] / len(data['Customers'])
# Output3
Seat_Queue_Length_2 = cum_stat['Seat Queue Length'] / clock
max_Seat_Queue_2 = max_Seat_Queue_Length
# Output4
Receptionists_util_2 = cum_stat['Receptionists Busy Time'] / (clock *
5)
KitchenStaff_util_2 = cum_stat['Kitchen staff Busy Time'] / (clock *
2)

#print("final clock")
#print(clock)

print("End!")
#return Customer_Time_Spent_1, Receiving_Food_Waiting_Time_1,
Receptionists_util_1, KitchenStaff_util_1
return mem, memc

max_fel = 0
# Maximum length that FEL gets (the current event does not count in)
header_list = ['Step', 'Current Event', 'Clock', 'Order Queue Length',
'Receive Queue Length', 'Seat Queue Length',
'Empty Seats', 'Idle Receptionists', 'Idle Kitchen Staff',
'Resting Receptionists',
'Resting Kitchen Staff', 'WRr', 'WRk', 'Number of Current
Customers']
simulation_time = int(input("Enter the Simulation Time: "))
simulation(simulation_time)
"""
#Performing 5 replications
Result = []
for i in np.arange(0, 5, 1):
    Result.append(simulation(simulation_time))

Result = pd.DataFrame(Result)
#print(np.mean(Result), axis=0)
#Result.columns = ['TimeSpent', 'ReceivingTime', 'RUtil', 'KUtil']
print(Result)
"""

```


منابع و مآخذ

- Crason. B, (2013), Discrete event system Simulation. Pearson.
- Dunne. S, 2018, *5 KPIs Every Savvy Multi-Location Restaurant Manager Should Track*, <<https://www.bizimply.com/blog/5-kpis-every-savvy-multi-location-restaurant-manager-should-track/>>.
- Sigurdson. S, 2018, *15 Important Restaurant KPIs*, <<https://www.franchiseblast.com/restaurant-kpis/>>
- صدقی. ن، (۱۳۹۸-۱۳۹۹)، اسلایدهای تدریسی درس شبیه‌سازی در دانشگاه صنعتی شریف: آموزش شبیه‌سازی برگرفته از کتاب Discrete event system simulation.