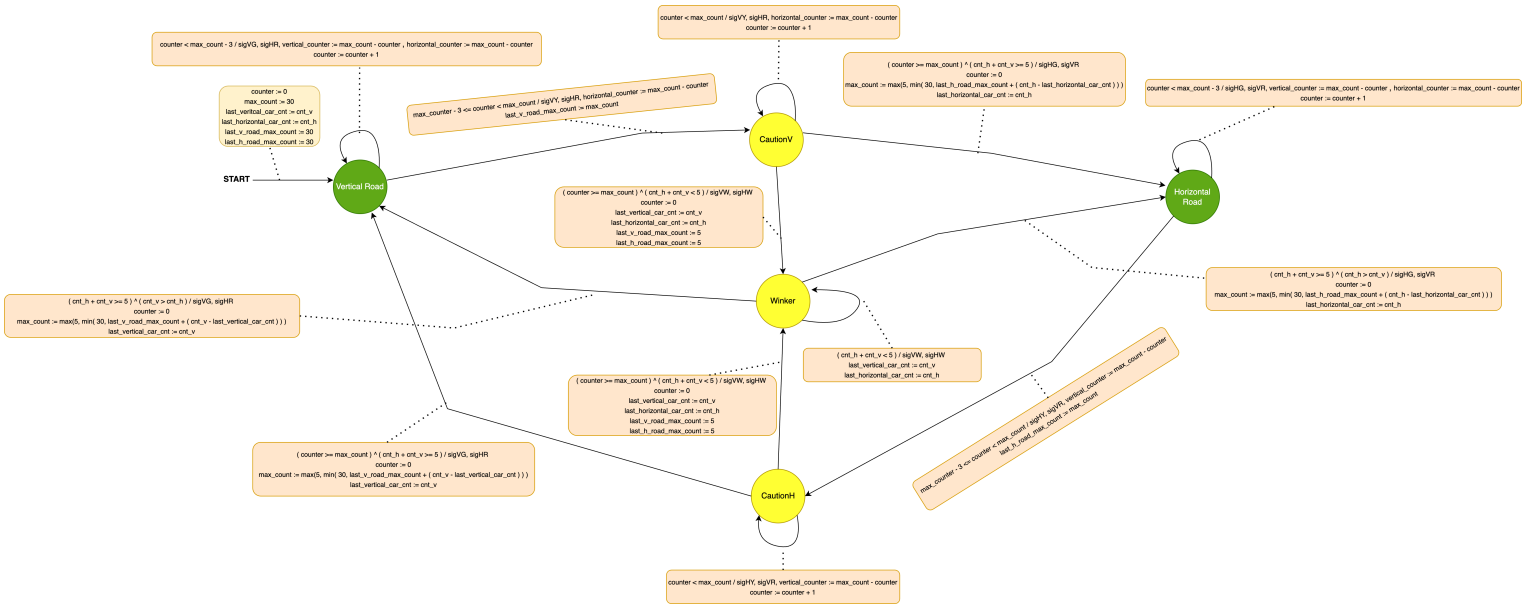
علیرضا آقایی ۴۰۱۲۰۳۳۷۱

استوت چارت طراح شده برا کنترل کننده چراغ راهنما:



در این استیت چارت ۲ ورودی cnt_v و cnt_h داریم که در واقع همان تعداد ماشین‌های موجود در جاده‌ی عمودی و افقی مربوط به چهارراه است. همچنین متغیرهای زیر نیز در داخل استیت چارت تعریف شده‌اند:

last_vertical_car_cnt: تعداد ماشین‌های پشت چراغ جاده‌ی عمودی مربوط به چراغ قرمز قبلی
last_horizontal_car_cnt: تعداد ماشین‌های پشت چراغ جاده‌ی افقی مربوط به چراغ قرمز قبلی
last_v_road_max_count: مدت زمان تعیین شده برای چراغ قرمز قبلی جاده‌ی عمودی
last_h_road_max_count: مدت زمان تعیین شده برای چراغ قرمز قبلی جاده‌ی افقی
counter: این متغیر میزان زمان سپری شده را می‌شمارد
max_count: مدت زمان تعیین شده برای چراغ قرمز کنونی

سیگنال‌های خروجی نیز عبارتند از:

sigVG: Vertical Road Green Light
sigVR: Vertical Road Red Light
sigVY: Vertical Road Yellow Light
sigVW: Vertical Road Winker

sigHG: Horizontal Road Green Light
sigHR: Horizontal Road Red Light
sigHY: Horizontal Road Yellow Light
sigHW: Horizontal Road Winker

vertical_counter: timer value for vertical road
horizontal_counter: timer value for horizontal road

استیت های طراحی شده نیز عبارتند از:

Vertical_Road: این استیت به معنی سبز بودن چراغ برای جاده‌ی عمودی است.
Horizontal_Road: این استیت به معنی سبز بودن چراغ برای جاده‌ی افقی است.
CautionV: این استیت به معنی احتیاط یا همان چراغ زرد برای جاده‌ی عمودی است.
CautionH: این استیت به معنی احتیاط یا همان چراغ زرد برای جاده‌ی افقی است.
Winker: استیتی است که در آن، هر دو چراغ راهنمای افقی و عمودی در حالت چشمک زن هستند.

منطق مشخص کردن زمان مربوط به یک چراغ قرمز با استفاده از مشتق گسسته میزان ترافیک هر جاده مشخص می‌شود. علت استفاده از متغیرهای داخلی که با پیشوند last شروع می‌شوند نیز به دست آوردن این مشتق است. در واقعا مدت زمان چراغ راهنما بر اساس صعودی بودن یا نزولی بودن میزان ترافیک جاده نسبت به چراغ قرمز قبلی می‌توان کاهش یا افزایش داشته باشد.

همچنین فرض شده است که مدت زمان عبور از چهارراه در حالت چشمک زن بر اساس توان ۲ نمایی است و به همین دلیل عدد ۵ برای threshold تبدیل چراغ‌ها به چشمک زن استفاده شده است.

- برخی موارد پیاده سوال که نیاز به توضیح دارند:

۱. توزیع نامتوازن ورودی و خروجی‌های ماشین‌ها با استفاده از تولید عدد یک عدد رندوم در مرحله‌ی اول و سپس انتخاب یک ورودی و خروجی بر اساس عدد رندوم و threshold قرار داده شده که در این سوال مقدار ۰.۷ قرار داده شده است، انجام می‌شود.

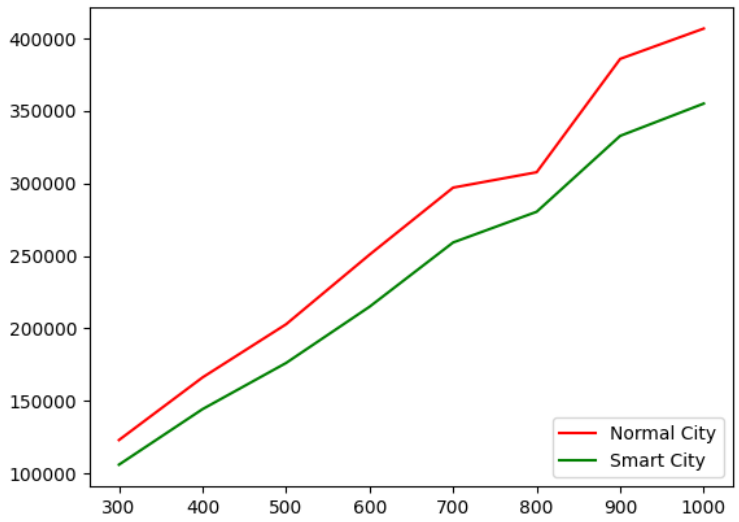
۲. مسیریابی بین ورودی و خروجی ماشین‌ها با استفاده از الگوریتم DFS به دست آمده است. با توجه به این که در سوال نیاز به پیدا کردن کوتاه‌ترین مسیر برای ماشین‌ها نبود با استفاده از الگوریتم DFS برای هر ماشین یک مسیر صحیح پیدا شده است.

۳. در این پیاده‌سازی زمان logical مورد بررسی و آزمایش قرار گرفته است. به این صورت که در هر مرحله از زمان، همه‌ی آبجکت‌ها شامل ماشین‌ها و چراغ‌های راهنما به اندازه‌ی یک واحد زمان جلو می‌روند. تمامی فواصل مسیرها نیز اعداد طبیعی در نظر گرفته شده‌اند.

۴. کلاس‌های مورد نیاز برای ساخت یک شهر عادی یا شهر هوشمند در فایل city_objects.py قرار داده شده‌اند. همچنین کلاس City و SmartCity نیز در فایل‌های city.py و smart_city.py قرار داده شده‌اند.

۵. آزمایش‌ها با استفاده از دستور python main.py انجام می‌شود. به این صورت که به ازای هر کدام از تعداد ماشین‌های ۳۰۰ تا ۱۰۰۰، ۳ بار آزمایش متفاوت انجام می‌دهیم و میانگین زمان سپری شده برای ماشین‌ها را محاسبه کرده و در نمودار قرار می‌دهیم.

نمودار به دست آمده برای ۲ حالت شهر عادي و شهر هوشمند به صورت زیر است:



همان طور که مشخص است، در آزمایش‌های انجام شده، در شهر هوشمند ماشین‌ها میزان زمان کمتری در ترافیک گذرانده‌اند که به این معنی است که کنترل کننده‌ی طراحی شده تاثر گذار بوده است.

۶. پیاده سازی استیت چارت طراحی شده در کلاس SmartTrafficLightController انجام شده است. این کلاس مسئولیت کنترل ۲ عدد آبجکت از کلاس SmartTrafficLight را بر عهده دارد که یکی از آن‌ها چراغ راهنمایی عمودی و دیگر چراغ راهنمای افقی است.

۷. فایل مربوط به استیت چارت در فولدر statechart_diagram قرار داده شده است و می‌توان ورژن با کیفیت‌تر آن را با استفاده از برنامه‌ی draw.io مشاهده کرد.

۸. عدد جاده‌ها به صورت تصادفی از بین ۱۰ تا ۴۰ انتخاب می‌شود و عدد ثابت چراغ راهنما در شهر عادي برابر با ۳۰ است.