ASANSÖRLERDEKİ TALEP YOĞUNLUĞUNUN MULTITHREAD İLE KONTROLÜ

Hüseyin Yılmaz
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
ylmzhsyn98@gmail.com

Özet

Bu projede, multithread kullanarak bir AVM sistemindeki asansörün eşzamanlı kontrolüyle kuyruk yoğunluğunu azaltacak, tüm müşterilerin hedeflediği kata en kısa sürede ulaşmasını ve müşteri giriş-çıkışlarının kontrol edileceği bir program yazmamız istenmiştir.

Giriş

Çalışmada amaç, multithread kullanımını öğretmektir. Programlama dili belirtilmemiştir, proje yazılırken Python dili kullanılmıştır.

Bizden farklı işlemleri yapacak birden fazla thread yazmamış istenmiştir. Giriş threadi her 0.5 saniyede bile 1 ile 10 arasında rastgele müşterinin AVM'ye giriş yapmasından ve üst katlardan rastgele birini hedef belirlemesinden sorumludur.

Çıkış threadi, saniye başı 1 ile 5 arasında rastgele sayıda müşterinin AVM'den çıkış yapmak için 0. katı hedeflemesinden, bu sayede üst katlarda da kuyruk oluşmasından sorumludur.

Asansör threadi, her 0.2 saniyede bir katlar arasında dolaşmaktadır. Öncelikle hedefi o anda bulunduğu kat olan yolcuları indirmekte, içinde bulunan müşteri sayısı 0 ise ve kontrol threadi durmasını istediyse durmakta, aksi takdirde gittiği yönü hedefleyen yolculardan kapasitesi kadar almaktadır. En üst kata gelindiğinde yönü aşağı, en alt kata gelince ise yukarı olarak değiştirilmektedir.

Kontrol threadi ise sürekli olarak katlardaki kuyrukları kontrol etmektedir. Kuyruktaki toplam kişi sayısı, o anda çalışan asansörlerin toplam kapasitesinin iki katına çıktığı anda yeni bir asansör etkinleştirmekte, o anda çalışan asansörlerin toplam kapasitesinin altına düştüğünde ise son açılan asansörü kapatmaktadır. Ayrıca arayüzün sürekli olarak güncel değerleri göstermesinden de yine kontrol threadi sorumludur. Her döngüde hem asansörleri hem de katları kontrol ederek, değişen değerlerin değiştirilmesini sağlamaktadır.

Yöntem

Projede nesneye yönelik programlama yerine dict ve listeler kullanılması tercih edildi. Programın kontrolünün kolaylaştırılması için bir Mall sınıfı oluşturuldu, hem asansörler listesi hem de müşteriler listesi bu sınıfta tutuldu. Ayrıca threadler de bu sınıfın birer fonksiyonu olarak yazıldı.

Tüm asansörler birer dict objesinde saklanmaktadır. Bu dictler, asansör kapasitesi, asansör aktif olma durumu, bulunduğu kat, yönü gibi bilgileri saklamaktadır. Tüm asansör dictleri bir liste içinde saklanmakta, bu sayede tek elden kontrolü kolaylaştırılmaktadır.

Aynı şekilde müşteriler de birer dict objesinde saklanmaktadır. Bu dictlerde müşterinin bulunduğu kat ve hedef katı bulunmaktadır. Asansörde bulunmayan tüm müşteriler tek bir listede saklanmakta, asansörde bulunanlar ise bulundukları asansörün dictindeki listede tutulmaktadır.

Giriş threadinin mantığı, her 0.5 saniyede bir 1 ile 10 arasında bir rastgele int değeri oluşturulup, bu sayıda yeni müşteri dicti yaratılıp, sınıfta bulunan müşteriler listesine eklenmesi üzerine kuruludur. Müşteri dicti yaratılırken bulunduğu kat olarak 0. kat belirlenmekte, hedef katı da 1 ile 4 arasında rastgele bir int değeri alınıp belirlenmektedir.

Çıkış threadindeki döngü ise o anda bir hedef katı olmayan rastgele 1 ile 5 arasında müşterinin hedef katını 0'a ayarlamaktadır. Böylece müşteri

kuyruğa girmiş olmaktadır. Her döngü sonunda thread 1 saniye uyutulmaktadır.

Asansör threadindeki döngü öncelikle hedefi o anda bulunulan kat olan yolcuları kendi müşteriler listesinden silmektedir. Eğer ki o anda 0. katta bulunuluyorsa müşteri AVM'den çıkacağı için asansörden inen müşteri geriye bir yere yerleştirilmemektedir ama başka bir katta bulunuluyorsa müşterinin hedefi sıfırlanmakta, bulunduğu kat olarak o anki kat belirlenmekte, daha sonra da sınıftaki genel müşteriler listesine geri konmaktadır.

İnecek olan müşteriler indirildikten sonra o anda asansörde başka müşteri kalıp kalmadığına bakılmaktadır. Eğer ki tüm müşteriler indirildiyse, kontrol threadinden gelen değer kontrol edilerek eğer asansöre artık gerek yoksa döngü bozulup, thread sonlandırılmaktadır. Eğer ki asansöre hala gerek varsa bu sefer de yön kontrolü yapılmaktadır. Asansör yukarıya çıkıyorsa, içinde o anda hiç müşteri yoksa ve üst katlarda da hiç sırada bekleyen müşteri yoksa asansör yönünü doğruca aşağıya döndürmektedir.

Daha sonraysa genel yön kontrolü yapılmaktadır, eğer ki en üst kata gelindiyse yön aşağı, en alt kata gelindiyse yön yukarı olarak ayarlanmaktadır.

Yön kontrolü de yapıldıktan sonra yeni müşteri alımına başlanıyor. O anda 0. katta bulunuluyorsa yukarı katları hedefleyen yolcular kapasite dolana kadar asansöre alınıp, genel müşteri listesinden dicti silinmektedir. Üst katlarda bulunuluyorsa da yalnızca yön aşağı olduğu zaman müşteri alınmaktadır. İçeri alınan müşterilerin bulunduğu kat, inerken düzeltilmek üzere sıfırlanmaktadır.

En sonunda ise thread 0.2 saniye uyutulup, eğer yön yukarı ise asansörün bulunduğu kat bir arttırılmakta, yön aşağı ise kat bir azaltılmaktadır. Böylece döngü tamamlanmaktadır.

Kontrol threadi her döngü başında arayüz güncelleme fonksiyonunu çağırarak arayüzdeki değerlerin doğruluğunu sağlamaktadır. Daha sonra ise o anda kuyrukta bulunan yolcuları kontrol ederek mevcut asansör kapasitesinin 2 katını geçtiyse yeni bir asansör için thread açmaktadır, eğer ki o anda çalışan ama durması emredilmiş bir thread varsa yeni thread açmak yerine o threade devam etmesi bildirilmektedir.

Kuyruktaki müşteri sayısı mevcut kapasitenin altına düştüyse de son açılan asansöre durması emredilmektedir. Aşırı kaynak tüketiminin önüne geçilmesi için bu döngüye 0.02 saniyelik bir bekleme süresi konmuştur.

Arayüz için Python'ın standart GUI kütüphanesi olan tkinter tercih edilmiştir. Tkinter'daki grid sistemi sayesinde tablodaki hücrelerin koordinatlarının ayrı ayrı hesaplanmasına gerek kalmamakta, yalnızca satır ve hücre numarası verilerek otomatik olarak yerleştirilmesi sağlanmaktadır.

Bilgileri tutan hücreler birer label objesinin Mall sınıfındaki labels listesinde tutulmasıyla güncellenmektedir. Arayüz güncelleme fonksiyonu her çağırıldığında o anki değerlerle daha önce yazdırılan değerler kıyaslanıp, değişenlerin hücresi güncellenmektedir.

Arayüz ana threadde çalışırken diğer işlemler yan threadlerde yapıldığı için arayüz kapatıldığında tüm threadlerin sonlandırıldığından emin olmak için ayrı bir fonksiyon yazılmıştır. Ancak bu işlemler aynı anda çağırılınca çakıştığı için bu kapatma fonksiyonu da bir thread yaratıp, threadler sonlandırıldıktan 0.25 saniye sonra arayüzü yok etmektedir.

Sonuçlar

İstenilen tüm işlemler eksiksiz olarak gerçekleştirilmiştir. Program çalışınca sırasıyla giriş threadi, ilk asansörün threadi, çıkış threadi ve kontrol threadi çalıştırılmakta ve arayüz oluşturulmaktadır.

Arayüzden anlık olarak tüm katlardaki müşterilerin toplam sayısı, kuyruktaki kişi sayısı, ve hedefleri; ayrıca tüm asansörlerin bulunduğu kat, gittiği yön, o anda içeride bulunan müşterilerin hedeflediği katlar, çalışma durumu gibi bilgiler takip edilmektedir.

Arayüz kapatılınca tüm threadlerle birlikte program otomatik olarak sonlandırılmaktadır.

Çıktılar

asansor

Kat	Toplam	Kuyrukta	Sabit	0'a Gidecek	1'e Gidecek	2'ye Gidecek	3'e Gidecek	4'e Gidecek
0	20	20	0	0	6	4	4	6
1	4	2	2	2	0	0	0	0
2	3	1	2	1	0	0	0	0
3	0	0	0	0	0	0	0	0
4	4	0	4	0	0	0	0	0

Asansor No	Durum	Kapasite	Iceride	Konum	Yon	0'a Giden	1'e Giden	2'ye Giden	3'e Giden	4'e Giden
1	Calisiyor	10	1	4	Asagi	1	0	0	0	0
2	Duruyor	10	0	0	Yukari	0	0	0	0	0
3	Duruyor	10	0	0	Yukari	0	0	0	0	0
4	Duruyor	10	0	0	Yukari	0	0	0	0	0
5	Duruyor	10	0	0	Yukari	0	0	0	0	0

asansor

Kat	Toplam	Kuyrukta	Sabit	0'a Gidecek	1'e Gidecek	2'ye Gidecek	3'e Gidecek	4'e Gidecek
0	8	8	0	0	5	0	0	3
1	82	1	81	1	0	0	0	0
2	63	1	62	1	0	0	0	0
3	71	0	71	0	0	0	0	0
4	91	1	90	1	0	0	0	0

Asansor No	Durum	Kapasite	Iceride	Konum	Yon	0'a Giden	1'e Giden	2'ye Giden	3'e Giden	4'e Giden
1	Calisiyor	10	0	1	Asagi	0	0	0	0	0
2	Calisiyor	10	10	0	Yukari	0	4	2	3	1
3	Duruyor	10	0	0	Yukari	0	0	0	0	0
4	Duruyor	10	0	0	Yukari	0	0	0	0	0
5	Duruyor	10	0	0	Yukari	0	0	0	0	0

asansor

Kat	Toplam	Kuyrukta	Sabit	0'a Gidecek	1'e Gidecek	2'ye Gidecek	3'e Gidecek	4'e Gidecek
0	12	12	0	0	3	3	3	3
1	124	0	124	0	0	0	0	0
2	101	0	101	0	0	0	0	0
3	120	1	119	1	0	0	0	0
4	124	0	124	0	0	0	0	0

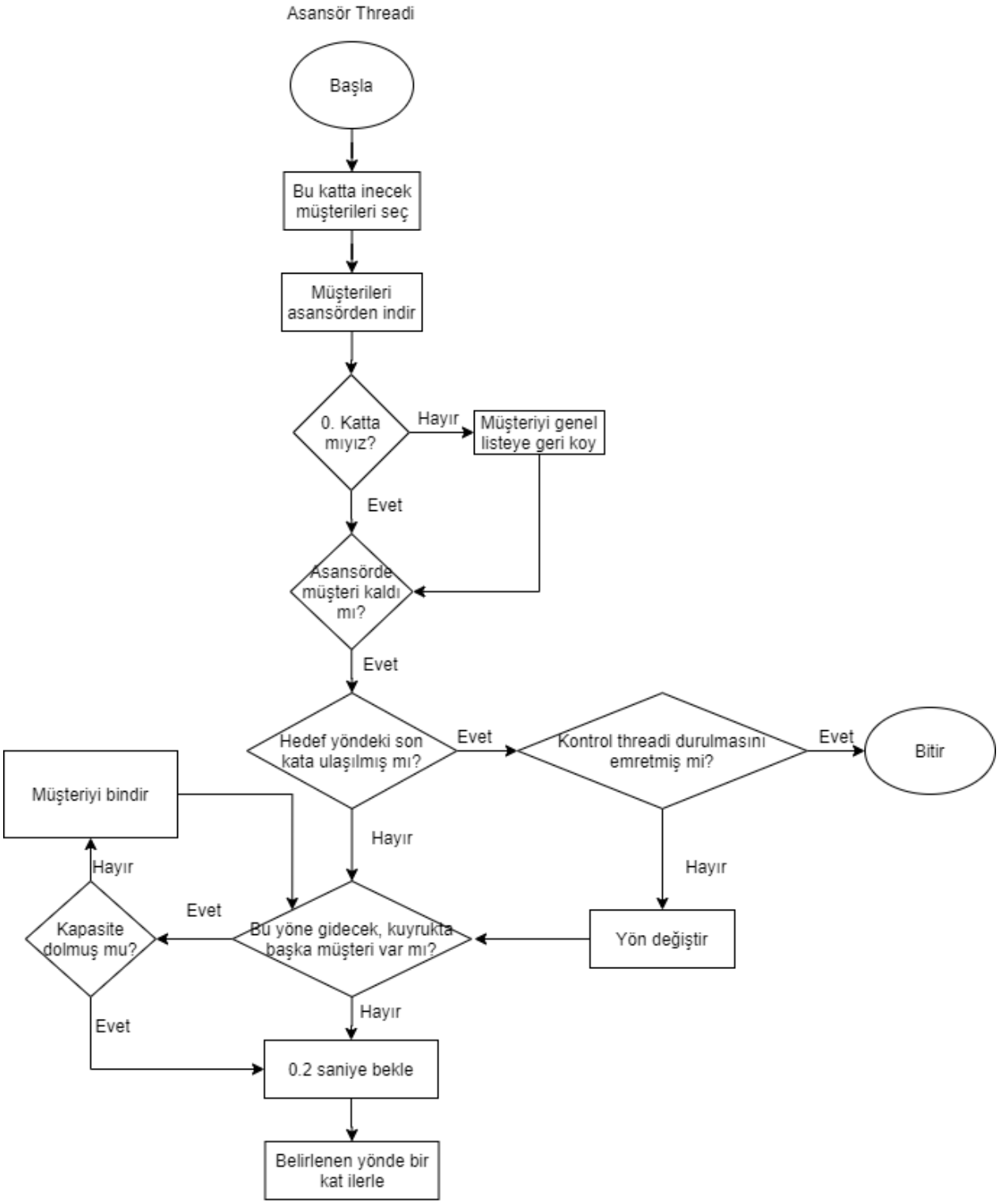
Asansor No	Durum	Kapasite	Iceride	Konum	Yon	0'a Giden	1'e Giden	2'ye Giden	3'e Giden	4'e Giden
1	Calisiyor	10	2	4	Asagi	2	0	0	0	0
2	Duracak	10	4	3	Yukari	0	0	0	0	4
3	Duruyor	10	0	0	Yukari	0	0	0	0	0
4	Duruyor	10	0	0	Yukari	0	0	0	0	0
5	Duruyor	10	0	0	Yukari	0	0	0	0	0

asansor

Kat	Toplam	Kuyrukta	Sabit	0'a Gidecek	1'e Gidecek	2'ye Gidecek	3'e Gidecek	4'e Gidecek
0	17	17	0	0	2	2	7	6
1	167	0	167	0	0	0	0	0
2	144	0	144	0	0	0	0	0
3	152	0	152	0	0	0	0	0
4	157	0	157	0	0	0	0	0

Asansor No	Durum	Kapasite	Iceride	Konum	Yon	0'a Giden	1'e Giden	2'ye Giden	3'e Giden	4'e Giden
1	Calisiyor	10	2	2	Asagi	2	0	0	0	0
2	Duruyor	10	0	4	Yukari	0	0	0	0	0
3	Duruyor	10	0	0	Yukari	0	0	0	0	0
4	Duruyor	10	0	0	Yukari	0	0	0	0	0
5	Duruyor	10	0	0	Yukari	0	0	0	0	0

Diyagramlar



Kaynakça

https://docs.python.org/3/library/threading.html

https://docs.python.org/3/library/tkinter.html

https://www.tutorialspoint.com/python/tk_label.htm

https://www.tutorialspoint.com/python/tk_grid.htm

https://www.programiz.com/python-programming/list-comprehension

https://www.geeksforgeeks.org/self-in-python-class/

https://stackoverflow.com/questions/111155/how-do-i-handle-the-window-close-event-in-tkinter

https://stackoverflow.com/questions/27123676/how-to-update-python-tkinter-window

https://realpython.com/python-main-function/

Kazanımlar

Daha önce indirme yöneticisi gibi multithread kullanan programlar yazmış olsam da gerçek hayatta kullanılan bir sistemin böyle bir yapıyla yazılması güzel bir alıştırma oldu. List comprehension da kullandığım bir yapı olduğu halde bir programda ilk defa bu kadar çok kullanmam gerekti, Python'daki kod yazımını en fazla kolaylaştıran bu yapı için de pratik yapmış oldum.