Ege Eğitim Teknolojileri Dergisi
Journal of Ege Education Technologies
e-ISSN 2667-4270
Cilt 3, Sayı 2 Aralık 2019, Sayfa 42- 51

Yükseköğretimde Programlama Derslerine Yönelik Bir Otomatik Ödev Notlandırma Sistemi Önerisi[1]

Ahmet Arslan
Eskişehir Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü
aarslan2@eskisehir.edu.tr

Nilgün Özdamar
Anadolu Üniversitesi, Uzaktan Öğretim Bölümü
nozdamar@anadolu.edu.tr

Geliş Tarihi: 29.10.2019 Kabul tarihi: 25.10.2019 Yayınlanma Tarihi: 31.12.2019

Özet

Türkiye'de eğitim gören üniversite öğrencisi sayısı son yıllarda hızla yükselmektedir. Artan öğrenci mevcudu sebebiyle eğitim – öğretim faaliyetlerinin planlanması zorlaşmakta, böylece daha fazla öğretim elemanı ve fiziksel imkâna ihtiyaç duyulmaktadır. Öğretim elemanları, bir dönem süresince öğrencilerden neredeyse haftalık gelen ödevleri tek tek değerlendirmek için zamana ihtiyaç duymakta, bazen yüzlerce notlandırılması gereken dosyaların yükü altında fazlaca çaba göstermek zorunda kalmaktadır. Bu nedenle de öğretim elemanı tarafından öğrencilere verilen geribildirim süreleri can sıkıcı bir şekilde uzamakta, bu süreler öğrenme için gerekli tepki sürelerini aşmaya başlamaktadır. Dolayısıyla, yaşanan zorlayıcı bu süreç, öğrenciler açısından memnuniyetsizlikle karşılanmaktadır.

Oysaki ödevlerin verilmesi ve değerlendirilmesinde yeni bilgi ve iletişim teknolojilerinden faydalanarak fiziksel ve insan gücü kaynaklarına olan gereksinimleri azaltmak mümkündür. Bu doğrultuda, GitHub Classroom ev ödevlerinin dağıtımı ve geri toplaması için son zamanlarda dünya çapında eğitim kurumlarında yaygın olarak kullanılmaya başlanan bir teknolojidir. Bu çalışmada, programlama ödevlerini dağıtma ve geri toplamanın yanı sıra otomatik olarak notlandırılabilmesi için GitHub Classroom sisteminin üzerine ne tür eklentiler yapılması gerektiği anlatılmaktadır. Otomatik notlandırmada benimsenen yaklaşım Yazılım Mühendisliği alanında kullanılan birim testine benzer bir mekanizmadır. Çalışmada bir programlama uğraşının otomatik notlandırılabilmesi için sağlaması gereken özelliklerin üzerinde de durulmaktadır.

Bu tarz bir otomatik notlandırma sisteminin uygulanması, öğretim elemanının iş yükünü azaltacağı gibi oldukça hızlı geribildirim sağlayarak öğrencilerin öğrenmelerini kolaylaştırabilecektir. Bu düşünceden motive olan bu çalışmanın temel amacı yükseköğretim kurumlarında yürütülen programlama derslerinde kullanılabilecek bir otomatik programlama ödevi notlandırma sisteminin tanıtılmasıdır.

Bu bağlamda Anadolu Üniversitesi Bilgisayar Mühendisliği bölümü müfredatında bulunan iki ayrı programlama dersi tasarlanan otomatik notlandırma sisteminin uygulanması için örnek ders olarak seçilmiştir. Derslere kayıtlı öğrencilerin gönderdikleri ödevlerin bağımsız bir sunucu üzerinde derlenip

[1] Bu çalışma 12-13 Eylül 2019 tarihinde düzenlenen 2nd International Instructional Technologies in Engineering Education konferansında sözlü bildiri olarak sunulmuştur.

çalıştırılması için otomatik derleme aracı olan Apache Maven kullanılmıştır. Dahası, programlama ödevlerini otomatik notlandırmak için kullanılan betik dosyaları diğer eğitimcilerinde faydalanabilmesi amacıyla https://education.github.community/t/automatic-grading-script/6940 adresinden umuma açık halde paylaşılmıştır.
Çalışmada tanıtılan bu sistem, yükseköğretimde Bilgisayar Mühendisliği programlama derslerinin yanı sıra, sosyal bilimler gibi diğer disiplinlerde de GitHub Classroom platformuyla destekli eğitim – öğretim faaliyetlerinin yaygınlaşması için yol gösterici olacaktır.

Anahtar Kelimeler: Otomatik notlandırma, Durumlu öğrenme, GitHub Classroom, Test güdümlü programlama

Ege Eğitim Teknolojileri Dergisi
Journal of Ege Education Technologies
e-ISSN 2667-4270
Volume 3, Issue 2, December 2019, Pages 42- 51

A Proposal for an Automatic Homework Grading System for Programming Courses in Higher Education

Abstract

The number of university students in Turkey is increasing rapidly in recent years. Due to the increasing number of students, it is getting difficult to plan education and training activities; thus, more teaching staff and physical resources are needed. The faculty members spend reasonable time to grade each and every assignment and sometimes they have to put in a lot of effort under the burden of hundreds of assignments that need to be graded. Therefore, the feedback given to the students by the instructor is annoyingly prolonged, and these periods begin to exceed the reaction time required for learning. Therefore, the students are dissatisfied with this compelling process, which causes disappointment and frustration.

However, it is possible to reduce the labor and physical resource requirements by using the new information and communication technologies (ICTs) in assigning and evaluating assignments. In this respect, GitHub Classroom is the cutting-edge technology that has been widely used in educational institutions worldwide for the distribution and collection of homework assignments. In this study, it is explained how to build an automatic grading system on top of the GitHub Classroom system, which is traditionally used for distributing and collecting programming assignments. The approach adopted in automatic grading is a mechanism that is similar to that of the unit testing (a type of software testing), which is a subfield of software engineering. The study also introduces the required and desirable criteria that a programming assignment must meet in order to be auto gradable.

The implementation of such automatic grading system will not only reduce the workload of the instructor, but will also provide rapid feedback to students, which facilitate their learning process. Motivated by this rationale, the main purpose of this study is to introduce an automatic grading system for programming assignments, which can be used in programming courses in higher education.

For proof of concept, the proposed automatic grading system was used in two different programming courses at Department of Computer Engineering in Anadolu University. The Apache Maven, a build automation tool, was used to compile and run the programming assignments submitted by the students on a standalone server. Moreover, the script files used for automatic grading of programming assignments are shared publicly at https://education.github.community/t/automatic-grading-script/6940 for the benefit of the education community.

We believe that the system introduced in this study will make the educational activities supported by the GitHub Classroom more prevalent among not only computer science, but also social sciences.

Keywords: Automated grading, Situated learning, GitHub Classroom, Test driven development

Giriş

İnsanların iletişim şekillerini ve bilgiye erişim yollarını zenginleştiren bilgi ve iletişim teknolojilerinde yaşanan muazzam gelişmeler, yükseköğretim kurumlarında dijital dönüşüm çağı olarak adlandırılan süreci başlatmıştır. Özellikle Nisan 2001 yılında Massachusetts Institute of Technology (MIT) tarafından duyurulan ve açıklık felsefesi temelinde ortaya çıkan açık ders kaynaklarının yükseköğretim derslerinde kullanılması, öğretim süreçlerinde geleneksel yöntemlerden ziyade çağa uygun dijital ve yenilikçi öğretim yöntemlerine geçilmesi gerektiğine yönelik fikirlerin ortaya çıkmasını sağlammıştır. Bununla birlikte Harvard, Stanford, MIT, Boston, gibi prestijli üniversitelerinin oluşturduğu konsorsiyumlar tarafından tasarlanan kitlesel açık çevrimiçi derslerin herkes tarafından ücretsiz bir şekilde erişilebilir olması, dijital ortamda bulunması, kullanıma ve devşirilmeye açık olması ve yüksek kalitede tasarlanmış farklı ve çeşitli ders malzemeleri sunmasından dolayı yüksek ilgi görmesi, yükseköğretim kurumlarına karşı bir Tsunami etkisi yaratacağı savının ortaya çıkmasına neden olmuştur. Bu hareketin yükseköğretim kurumlarına kasırga kadar yıkıcı bir etkisi olmasa da üniversitelerde yenilikçi ve yaratıcı yeni öğretim modellerinin entegrasyonuna yönelik vizyonun geçilmesine katkı sağladığı söylenebilir. Ayrıca uzaktan öğretim teknolojilerinin hem öğretim elemanları hem de öğrenciler açısından birçok fırsat sağlaması (yer ve zamandan bağımsız olma, bağlama dayalı öğrenme gibi…) üniversitelerin öğretimde karma model yöntemine (hem yüz yüze hem de uzaktan) geçilmesine yönelik teşviklerini arttırdıkları gözlenmektedir. Dünya da birçok kurum uzaktan eğitim yöntemini derslerde zorunlu tutarken Türkiye'de son yıllarda üniversitelerde öğretim elemanlarının uzaktan öğretim yöntemini kullanımının arttığı ve bu amaç doğrultusunda daha kaliteli eğitim hizmeti vermek amacıyla uzaktan eğitim merkezlerinin kurulduğu gözlenmektedir. Özellikle Google Classroom, Moodle, Canvas gibi birçok açık kaynak kodlu yazılım ile ücretsiz bir şekilde derslerin yer ve zamandan bağımsız bir şekilde uzaktan verilmesini olanak tanıması, bu merkezlerde uzaktan eğitim teknolojilerinin entegrasyon sürecini kolaylaştırmıştır.

Açık ders malzemelerinin en yoğunluklu kullanıldığı programlardan birisi ise bilgisayar mühendisliği eğitim programıdır. Bu programda sıklıkla teknoloji destekli esnek öğrenme (flipped) yöntemi ile MIT, Harvard gibi üniversitelerde alanında ün yapmış saygın hocaların programlama bilgisi, yapay zekâ gibi teknik konularda hazırladıkları ders videolarını derse katılmadan önce öğrencilerin izlemeleri, ders sürecinde öğretim elamanı eşliğinde bu ders videolarına yönelik uygulama ve tartışma yapmaları istenmektedir. Böylece üniversiteler öğretim sürecinde kaliteli ve kapsamlı zengin içeriğe erişme fırsatı bulmuşlardır. Bugün Coursera, EDx, Futurelearn gibi birçok kitlesel çevrimiçi açık kurs platformlarında programla bilgisine yönelik derslere ve sertifikalara rastlamak mümkün. Bu bağlamda yükseköğretimde dijital dönüşüme öncülük edebilecek en uygun programlardan birisi olarak bilgisayar mühendisliği eğitim programı olduğu söylenebilir. Bilgisayar bilimlerinin küresel dünyada çağın öncelikli ve gerekli alanlarından birisi olması, bu nedenle öğrenci talebinin yüksek olması bilgisayar bilimleri eğitiminin daha nitelikli ve yenilikçi modellerle desteklenmesi gerektiğini işaret etmektedir.

Öte yandan bilgisayar bilimleri programlarında öğrenci sayılarının fazla olması öğretim faaliyetlerinin planlanmasını zorlaştırmakta, daha fazla öğretim elemanı ve fiziksel imkânlara ihtiyaç gün geçtikçe daha da fazlalaşmaktadır. Öğretim elemanları, bir dönem sürecinde öğrencilerden neredeyse haftalık gelen ödevleri tek tek değerlendirmek için zamana ihtiyaç duymakta, bazen yüzlerce notlandırılması gereken dosyaların yükü altında fazlaca çaba göstermek zorunda kalmaktadır. Bu nedenle de öğretim

elemanı tarafından öğrencilere verilen geribildirim süreleri can sıkıcı bir şekilde uzamakta, bu süreler öğrenme için gerekli tepki sürelerini aşmaya başlamaktadır. Dolayısıyla, yaşanan zorlayıcı bu süreç, öğrenciler açısından memnuniyetsizlikle karşılanmaktadır. Oysaki ödevlerin verilmesi ve değerlendirilmesinde yeni enformasyon ve iletişim teknolojilerinden faydalanılarak fiziksel imkânlara yönelik gereksinimleri azaltmak mümkündür. Bu doğrultuda, GitHub Classroom ev ödevlerinin dağıtımı ve geri toplanması için son zamanlarda dünya çapında eğitim kurumlarında yaygın olarak kullanılmaya başlanan yeni bir teknolojidir. Bu çalışmada, programlama ödevlerini dağıtma ve geri toplamanın yanı sıra otomatik olarak notlandırılabilmesi için GitHub Classroom sistemi üzerine ne tür eklentiler yapılması gerektiği anlatılmaktadır. Otomatik notlandırmada benimsenen yaklaşım Yazılım Mühendisliği alanında kullanılan birim testine benzer bir mekanizmadır. Çalışmada bir programlama uğraşının otomatik notlandırılabilmesi için sağlaması gereken özelliklerin üzerinde de durulmaktadır.

Bu tarz bir otomatik notlandırma sisteminin uygulanması, öğretim elemanının iş yükünü azaltacağı gibi oldukça hızlı geribildirim sağlayarak öğrencilerin öğrenmelerini kolaylaştırabilecektir. Bu düşünceden yola çıkarak bu çalışmanın temel amacı yükseköğretimde programlama derslerinde kullanılabilecek bir otomatik programlama ödevi notlandırma sisteminin tanıtılmasıdır. Böylece bilgisayar bilimleri eğitiminde özellikle programlama derslerini veren öğretim elemanlarına farklı bir bakış açısı ve yöntemi sunularak dijital dönüşüm sürecini tetikleyici bir sistem model önerisi sunulması hedeflenmektedir.

İlgili Alanyazın

Bilgisayar Mühendisliği eğitiminde GitHub ya da Maven gibi profesyonel yazılım geliştirme araçlarının kullanımının incelenmesi yeni olmayıp bu konuda birçok çalışma ve yayın mevcuttur. Bu çalışmalarda özellikle Yazılım Mühendisliği dersi vaka durumu olarak kullanılmıştır (Feliciano, Storey, & Zagalsky, 2016; Fiksel, Jager, Hardin, & Taub, 2019; Raibulet & Arcelli Fontana, 2018; Soundarajan, Joshi, & Ramnath, 2015).

Aynı şekilde programlama ödevlerinin otomatik notlandırılması için yapılmış birçok çalışma mevcuttur (Cheang, Kurnia, Lim, & Oon, 2003; Heckman & King, 2018; Helmick, 2007; Liu, Wang, Wang, & Wu, 2019; Morris, 2003; Parihar et al., 2017). Bu bildiride tanıtılan sisteme en yakın çalışma Bilgisayar Mühendisliği eğitimi alanında oldukça prestijli bir konferans olan Special Interest Group on Computer Science Education[2] konferansında 2018 yılında yayımlanmış çalışmadır (Heckman & King, 2018). Bu çalışmada GitHub ve birim testi mekanizması kullanılmıştır. Otomatik notlandırmada, öğrencilerin görebildikleri birim testlerinin yanı sıra öğrencilerin göremediği öğretim elemanı tarafından hazırlanmış birim testleri de kullanılmıştır.

Bizim çalışmamızın özgün tarafları; (i) bir programlama ödevinin otomatik notlandırılabilmesi için hangi özelliklere sahip olması gerektiğinin anlatılması, (ii) oyunlaştırma ve liderlik sıralamasının nasıl uygulanabileceği ve (iii) otomatik notlandırmada kullanılan betik dosyalarının ve günümüze dek verilmiş ödevlerin umuma açık halde paylaşılması olarak sıralanabilir.

[2] https://www.sigcse.org

Otomatik Notlandırma Sistemi

GitHub[3] bir kod barındırma platformudur. Bir projeye ait kaynak kodları içeren dosyalar depo adı verilen bir yerde tutulur. Her depo için özel bir Web adresi oluşturulur ve bu adres ile depoya erişilir. Bir depo herkese açık (umumi) ya da gizli/özel olabilir. GitHub üzerinde umumi depo yaratmak ücretsiz iken, sadece tanımlı kullanıcıların erişebildiği özel depo yaratmak ücretli bir servistir. Ancak, GitHub eğitim-öğretim camiasının sempatisini kazanmak istediği için normalde ücretli olan bu özelliği akademik personele ücretsiz olarak sunmaktadır. Bu servisten ücretsiz yararlanmak için akademik personel kimliğinin fotokopisi ve @edu uzantılı bir e-posta adresi ile akademik üyelik başvurusunda bulunulması gerekmektedir. GitHub platformunun eğitim-öğretim camiasına sunduğu diğer özelliklere https://education.github.com adresinden erişilebilmektedir.

GitHub sürüm kontrol sistemi olarak arka planda Git kullanır. Sürüm kontrol sistemi sayesinde ve projede yapılan değişlikler kaydedilerek projenin geçmişi saklanmış olur. GitHub yazılım geliştirici topluluğunu bir araya getirerek "sosyal kodlama" yapmaya olanak sağlar. Böylece, yazılım geliştiricilerin iş birliği yaparak ortak bir şekilde çalışmasına imkân vermesinin yanında yazılım projelerinin keşfedilmesini ve paylaşılmasını da mümkün kılmış olur.

GitHub Classroom

GitHub Classroom[4] GitHub platformu üzerine inşa edilmiş ödev dağıtma ve toplama sistemidir. Bu sistem kullanıcı dostu bir Web ara yüzüne sahiptir ve bütün işlemler İnternet üzerinden yapılır. Verilecek ödevin açıklamalarını içeren bir GitHub deposu oluşturulduktan sonra, Classroom sistemi vasıtası ile yeni bir ödev tanımlanır. Bu işlemin sonunda bir davetiye linki elde edilir. Bu link öğrencilere ilan edilerek duyurulur. Bu link bir Web adresinden başka bir şey değildir. Bu linke sahip ve bir GitHub kullanıcı hesabı olan herkes davetiyeyi kabul edebilir. Öğrenci davetiyeyi kabul eder etmez öğrenciye özel bir depo otomatik olarak yaratılır. Öğrenci ödevini bu depoya yükler. Ödevi tanımlayan eğitmen davetiyeyi kabul eden öğrencilerin listesini ve bu öğrenciler için yaratılmış depoların içeriklerini görebilir. Ayrıca, yapılan değişiklikleri de adım adım takip edebilir.

Otomatik Derleme – Apache Maven

Bir ödeve gönderilen cevapları içeren depoların hepsini indirmek mümkündür. Fakat bu depolarda kaynak kodlar bulunduğu için direk olarak çalıştırılamazlar. Kaynak kodlardan çalıştırabilir bir dosya elde etme işini otomatikleştirme için Maven, Ant, Gradle gibi araçlar mevcuttur. Bu çalışmada tanıtılan otomatik notlandırma sisteminde Apache Maven kullanılmaktadır. Apache Maven[5] ile yapılandırılmış bir proje komut satırı üzerinden derlenip çalıştırabilmektedir.

[3] https://github.com
[4] https://classroom.github.com
[5] https://maven.apache.org

Birim Testi

Bir ödev gönderisi çalıştırabilir hale getirildikten sonra ödevde istenen gereksinimler teker teker sınanır. Bu sınama işlemi programa örnek bir girdi verilerek yapılır. Eğer program bu girdi için beklenen çıktıyı doğru bir şekilde üretiyorsa o gereksinim sağlanmış demektir.

Bu şekilde örnek girdi ve buna karşılık gelen beklendik çıktı ikililerinden oluşan bir dizi sınama verisi hazırlanır. Komut satırı üzerinden girdiler programa beslenerek, programın çıktısı ile beklenen çıktının birebir aynı olup olmadığı yine otomatik olarak grep komutu ile kontrol edilir. Eğer o gereksinim için yanlış bir çıktı üretiyor ya da başka bir deyişle program beklendiği gibi çalışmıyorsa puan verilmez. Bu bahsedilen yapı yazılım mühendisliği alanında kullanılan birim testi (Runeson, 2006) paradigmasının çalışma prensibi ile aynıdır. Tek farkı, birim testinde öğrenciler sınama verisini görebilmektedirler. Anlatılan yapıda öğrenciler notlandırmada kullanılacak sınama verisinden haberdar değildirler.

Oyuncak Örnek

Verilen bir sayının faktöriyelini hesaplama ödevi, bahsedilen mekanizmayı kesin olarak açıklamak için pek çok detayın kasıtlı olarak çıkarıldığı basit bir örnek olarak kullanılabilir. Tablo 1'in ilk sütununda verilmiş örnek girdilerle öğrenci ödevi çalıştırılır ve ikinci sütundaki beklenen değerler elde ediliyor mu diye kontrol edilir. Başarılı olan her satır için 25 puan verilerek ödeve 100 üzerinden bir not atanmış olur.

Tablo 1. Faktöriyel Bulma Ödevi

Örnek Girdi	Beklenen Çıktı	Puan
3	6	25
0	1	25
-5	IllegalArgumentException	25
merhaba	IllegalArgumentException	25

Geri Bildirim – Hata ve Bilgi Mesajları

Bütün bu anlatılan süreçler komut satırından tüm öğrenci kullanıcıları için otomatik gerçekleştirilir. Bu süreç içerisindeki çıkan bilgi mesajları ve hata mesajları kaydedilir. Her öğrenci kendi süreci için kaydedilmiş bu mesajlara bir Web ara yüzünden erişebilir. Böylece bir hata varsa öğrenci en azından hata mesajına bakarak ödevini düzeltme şansına sahip olmuş olur. Bu aşama daha çok derleme hatalarını içerir. Hatalı konfigürasyon ve benzeri nedenlerden dolayı düzgün olarak derlenmeyen ödevler 0 (sıfır) puan ile notlandırılacaktır. Bu sebepten dolayı, derleme hata mesajlarının öğrenciye gösterilmesi önemlidir.

Programlama Ödevinin Sağlaması Gereken Özellikler

Bir programlama uğraşının otomatik olarak değerlendirebilmesi için birtakım standartları sağlamalıdır. Bunlardan en önemlisi ödevin çözümünün deterministik (rasgele olmayan) olmalıdır. Bir başka deyişle, verilen bir girdi değeri için her zaman aynı değeri çıktı olarak üretmelidir. Eğer bir veri kümesinin üzerinde birtakım istatistikler hesaplama yapma uğraşı ödev olarak verilmiş ise, umumi

veri kümeleri (örneğin The Signal Media One-Million News Articles Dataset[6]) kullanılmalıdır. Böylece tüm öğrenciler aynı veri kümesini indirip onun üzerinde çalışabilirler. Ancak bu şekilde ödevi otomatik notlandırmak için gerekli olan beklendik değerler önceden hesaplanabilir ve otomatik notlandırma yapılabilir.

Bulgular ve Tartışma

Öğrencilerden toplanan olumlu ve olumsuz g Tablo 2'de özetlenmiştir. Hızlı geribildirim, objektif olması ve ödevleri süreç boyunca güncelleyerek hataları düzeltme şansı olması otomatik notlandırma sisteminin olumlu yönleri olarak değerlendirilmiştir. Öte yandan öğrenciler büyük oranda Maven ve GitHub gibi profesyonel araçları erkenden kullanmaya başlamanın mesleki kariyerlerine olumlu katkısı olduğunu düşünmektedirler. Maven ve GitHub kullanımının öğrenmesinin zaman alması ve yaşanan zorluklar sistemin en kötü tarafı olarak listelenmiştir. Gerçekten de bu iki araç lisans öğrencileri için oldukça karmaşık ve öğrenmesi zaman alan araçlardır. Öğrencilerin en çok şikâyet ettikleri bir diğer nokta ise derleme hatası gibi sistemsel hatalarda ödev notunun 0 (sıfır) olmasıdır.

Tablo 2. Sistemin Olumlu ve Olumsuz Yönleri

Olumlu	Olumsuz
Hızlı geri dönüş sağlaması	Ödevi GitHub'a yüklemenin zor olması
Adaletli/objektif/herkese eşit olması	Sistemi öğrenmenin zor ve karmaşık olması
Ödevleri sık sık güncelleyebilmek	Maven'ı konfigüre etmenin zor olması
Hatasız/net/anlaşılır olması	Yükleme ya da derleme hatasında sıfır almak
Notları istenilen zamanda görebilmek	Ödevi göndermenin uğraştırıcı olması

Oyunlaştırma

Bazı programlama uğraşları %100 doğru çalışan ödevleri kendi içerisinde bir kritere göre (örneğin çalışma zamanı) sıralamayı mümkün kılar. Büyük veri üzerinde hesaplama yapmayı gerektiren bir ödev buna bir örnek olarak verilebilir. Bir programın hızlı çalışması arzu edilen bir özellik olduğu için ödevler kendi arasında çalışma zamanına göre küçükten büyüğe sıralanarak liderlik sıralaması yapılabilir. Bu konsepte uygun verilen bir ödevde listenin başına girebilmek için öğrencilerin birbirleriyle rekabete girip bir yarış havasında çalıştıkları gözlemlenmiştir. Burada öğrenciler programın gereksinimlerini sağlamanın yanı sıra, programın çalışmasını hızlandırmak içinde ayrı çaba gösterip paralelleştirme ve dinamik programlama (multi-threading and dynamic programming) gibi teknikleri uygulamışlardır.

Bir başka sıralama kriteri ise sadece optimizasyon problemleri için mümkün olan amaç fonksiyondur. Bu tür problemlerde bir amaç fonksiyonu en küçüklenmeye ya da en büyüklenmeye çalışılır. Optimizasyon problemlerinin tipik örneği gezgin satıcı problemidir (Rodríguez-Pereira, Fernández, Laporte, Benavent, & Martínez-Sykora, 2019). Bu problemdeki amaç fonksiyonu kat edilen yolun kilometre cinsinden ifadesidir. Endüstri ve Bilgisayar mühendisliği öğrencilerin birlikte çalışarak yaptıkları bu ödevde en kısa kilometre kat eden çözüm birinci seçilmiştir.

[6] https://research.signal-ai.com/newsir16/signal-dataset.html

Anlatıldığı gibi otomatik notlandırma sistemi birinci ikinci üçüncünün listelendiği liderler sıralaması oluşturmak içinde kullanılabilmektedir. Bu yapı "ACM RecSys Spotify Automatic Playlist Continuation" [7] (Zamani, Schedl, Lamere, & Chen, 2019) yarışmasından esinlenilmiştir. Böyle bir sistem uygulandığında ödev yapma ve gönderme süreci oyunlaştırılarak daha zevkli hale getirilmiş olur.

Sonuç

Bu çalışmada programlama ödevlerinin otomatik notlandırma için geliştirilmiş bir sistem tanıtılmıştır. Bu sistem GitHub Classroom üzerine inşa edilmiştir. Tamamen ücretsiz servislerden oluşturulabilen ve herhangi bir sunucu/yazılım kurulumu veya bakımı gerektirmeyen bir yapıdır. Tanıtılan bu sistem iki ayrı programlama dersinde iki yıl boyunca uygulanmış olup, şimdiye dek verilen ödevler Tablo 3'te sunulmuştur. Bu iki yılın sonunda elde edilen deneyimler çalışmada özetlenmiştir.

Ayrıca, programlama ödevlerini otomatik notlandırmak için kullanılan betik dosyaları diğer eğitimcilerinde faydalanabilmesi için https://education.github.community/t/automatic-grading-script/6940 adresinden herkese açık halde paylaşılmıştır.

Tablo 3. Geçmişte Verilmiş Ödevlerin Listesi

Ödevin Web Adresi
https://github.com/AnadoluUniversityCeng/DuplicateFiles
https://github.com/AnadoluUniversityCeng/Signal-1M-Statistics
https://github.com/AnadoluUniversityCeng/ClueWeb12SpamRankings
https://github.com/AnadoluUniversityCeng/Triangle-Inequality-Violation
https://github.com/AnadoluUniversityCeng/LongestShortestRoute
https://github.com/AnadoluUniversityCeng/TrendingHashtags
https://github.com/AnadoluUniversityCeng/LocaleSensitiveSort
https://github.com/AnadoluUniversityCeng/multiple-tsp

Çalışmada tanıtılan bu sistemin, yükseköğretimde Bilgisayar Mühendisliği programlama derslerinin yanı sıra, sosyal bilimler gibi diğer disiplinlerde de GitHub Classroom platformuyla destekli eğitim – öğretim faaliyetlerinin yaygınlaşması için yol gösterici olacağı düşünülmektedir.

[7] https://recsys-challenge.spotify.com/static/final_main_leaderboard.html

Kaynakça

Cheang, B., Kurnia, A., Lim, A., & Oon, W.-C. (2003). On automated grading of programming assignments in an academic institution. *Computers & Education*, 41(2), 121-131. doi:10.1016/S0360-1315(03)00030-7

Feliciano, J., Storey, M.-A., & Zagalsky, A. (2016). Student experiences using GitHub in software engineering courses: a case study. Paper presented at *the Proceedings of the 38th International Conference on Software Engineering Companion,* Austin, Texas.

Fiksel, J., Jager, L. R., Hardin, J. S., & Taub, M. A. (2019). Using GitHub Classroom To Teach Statistics. *Journal of Statistics Education*, 27(2), 110-119. doi:10.1080/10691898.2019.1617089

Heckman, S., & King, J. (2018). Developing Software Engineering Skills using Real Tools for Automated Grading. Paper presented at the Proceedings of the *49th ACM Technical Symposium on Computer Science Education*, Baltimore, Maryland, USA.

Helmick, M. T. (2007). Interface-based programming assignments and automatic grading of java programs. Paper presented at *the Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, Dundee, Scotland.

Liu, X., Wang, S., Wang, P., & Wu, D. (2019). Automatic grading of programming assignments: an approach based on formal semantics. Paper presented at *the Proceedings of the 41st International Conference on Software Engineering: Software Engineering Education and Training*, Montreal, Quebec, Canada.

Morris, D. S. (2003, 5-8 Nov. 2003). Automatic grading of student's programming assignments: an interactive process and suite of programs. Paper presented at *the 33rd Annual Frontiers in Education, 2003*. FIE 2003.

Parihar, S., Dadachanji, Z., Singh, P. K., Das, R., Karkare, A., & Bhattacharya, A. (2017). Automatic Grading and Feedback using Program Repair for Introductory Programming Courses. Paper presented at *the Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, Bologna, Italy.

Raibulet, C., & Arcelli Fontana, F. (2018). Collaborative and teamwork software development in an undergraduate software engineering course. *Journal of Systems and Software*, 144, 409-422. doi:10.1016/j.jss.2018.07.010

Rodríguez-Pereira, J., Fernández, E., Laporte, G., Benavent, E., & Martínez-Sykora, A. (2019). The Steiner Traveling Salesman Problem and its extensions. *European Journal of Operational Research*, 278(2), 615-628. doi:10.1016/j.ejor.2019.04.047

Runeson, P. (2006). A survey of unit testing practices. *IEEE Software*, 23(4), 22-29. doi:10.1109/MS.2006.91

Soundarajan, N., Joshi, S., & Ramnath, R. (2015). Collaborative and cooperative-learning in software engineering courses. Paper presented at the Proceedings of *the 37th International Conference on Software Engineering* - Volume 2, Florence, Italy.

Zamani, H., Schedl, M., Lamere, P., & Chen, C.-W. (2019). An Analysis of Approaches Taken in the ACM RecSys Challenge 2018 for Automatic Music Playlist Continuation. *ACM Trans. Intell. Syst. Technol.*, 10(5), 1-21. doi:10.1145/3344257