Kocaeli Üniversitesi
Bilgisayar Mühendisliği Bölümü

Programlama aboratuvarı II
Proje 1

GEZGİN ROBOT PROJESİ

Ahmad Alhomsi
ID:210201140
tarikuka.aa@gmail.com

ÖZET

Bu rapor Programlama Laboratuvarı II dersi 1. Projesinin çözümünü açıklamak üzere hazırlanmıştır. Bu proje Java diliyle yapılmıştır. Öncelikle "Swing" i kullanarak ızgarayı yapılmıştır. Belirli kurallara göre hareket eden bir robotun önündeki engelleri aşarak istenen hedefe ulaşmasını sağlayan bir oyun tasarlanması beklenmektedir. Oyunda iki tane problemin çözülmesi gerekmektedir. Problemlerin çözümü için nesneye yönelik programlama ve veri yapıları bilgilerinin kullanılması beklenmektedir. Amaç: Proje gerçekleştirimi ile nesneye yönelik programlama ve veri yapıları bilgisi pekiştirilmesi ve problem çözme becerisinin gelişimi amaçlamaktadır. robotu ızgara (grid) üzerinde verilen hedefe engellere takılmadan en kısa sürede ve en kısa yoldan ulaştırmanız beklenmektedir. Robotu tüm ızgarayı değil, yalnızca gerekli yolları gezerek hedefe ulaşmasını sağlamaktadır.

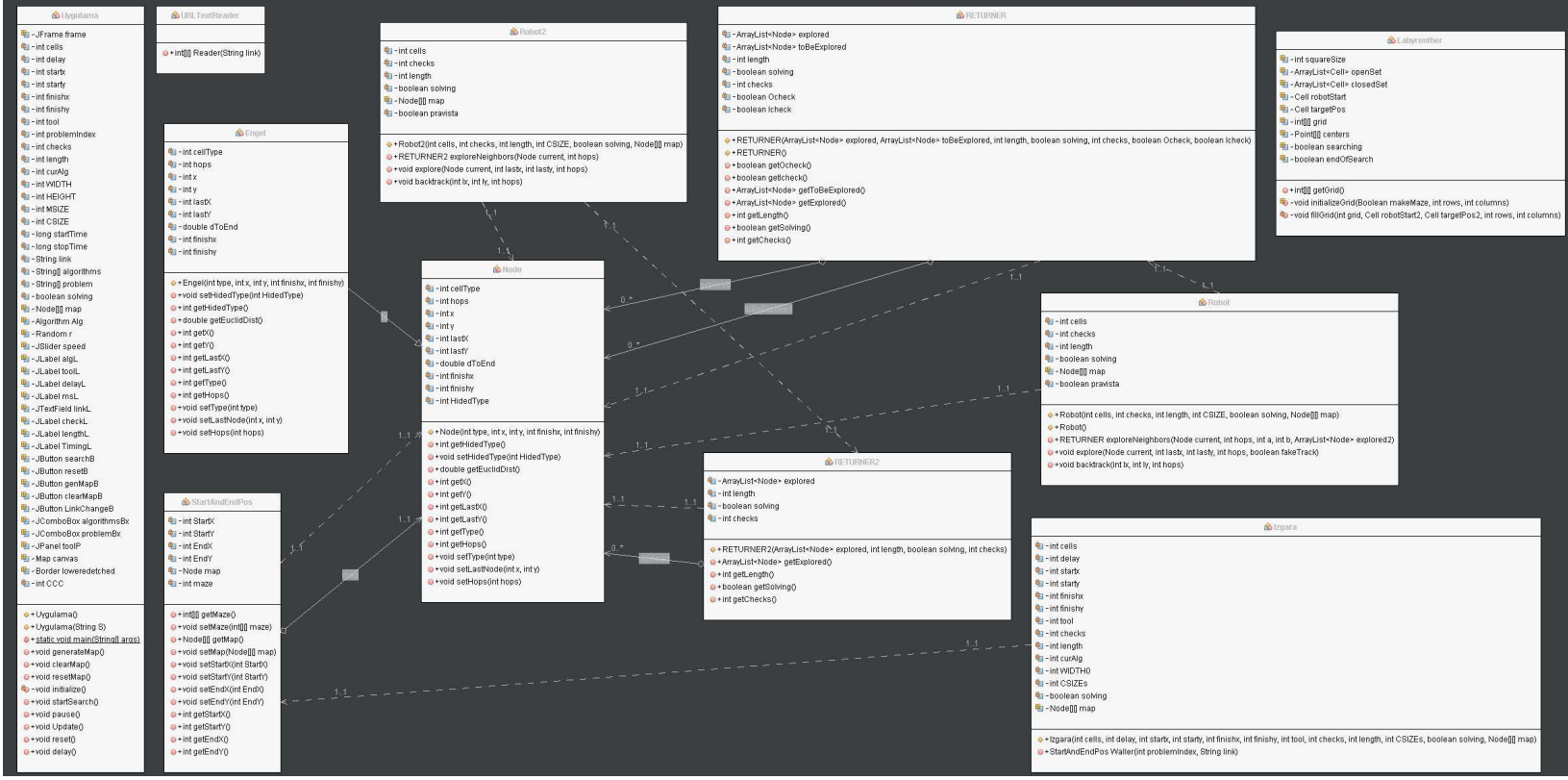
GİRİŞ

Problem 1 de Izgara boyutu, engel sayısı ve engellerin konum bilgileri içeriği matris biçimindeki bir text dosyasından alınır “http://bilgisayar.kocaeli.edu.tr/prolab2/url1.txt” ve “http://bilgisayar.kocaeli.edu.tr/prolab2/url2.txt” linklerinden okunur. Problem 1 seçildikten sonra ve generate maze tuşuna basıldıktan sonra otomatik olarak erişilerek URL text dosyasındaki tasarıma göre ızgara ve engel yapısı oluşturulur. Engeller birbirinden farklı tipteki nesnelerden oluşabilir. (Verilecek text dosyasındaki 0 değeri engelsiz yollara; 1, 2, 3 değerleri ise üç farklı tipteki nesne için engelleri temsil edilmektedir. Birbirinden farklı sayıda karesel alan işgal eden bu üç engel nesnesinden 1 değerli nesne yalnızca 1 karelik alan 2 değerine sahip nesneler yanyana 2 kare içeren maksimum 2x2 lik; 3 değerine sahip nesneler ise yan yana 3 kare içeren maksimum 3x3 lük kare alana yerleştirmektedir.)

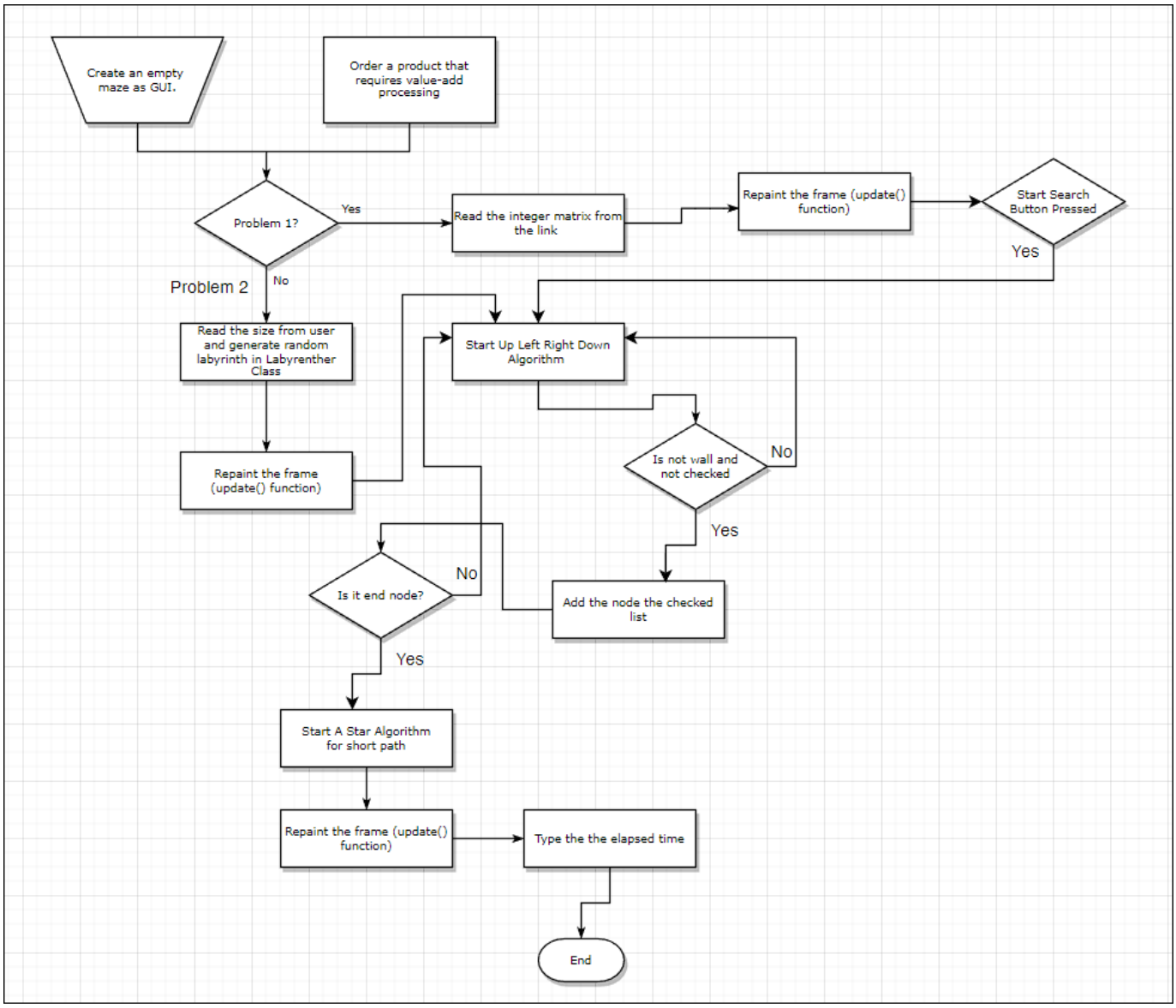
Robotun başlangıç ve hedef noktaları ızgara üzerindeki uygun (engel veya duvar içermeyen) karelere rastgele belirlenmelidir. Robot başlangıçta tüm ızgara dünyasını bilmemelidir, sadece bir adım sonraki kareleri görebilmelidir. Her adımda robotun öğrenmediği kareler bulutlu (kapalı) olarak gösterilmeli, öğrenilen kareler ise açılarak ilgili karelerde bulunan nesneye göre (engel, duvar, yol, vs.) belirmektedir. Tüm bu bilgiler doğrultusunda, robotun hedefe en kısa sürede ulaşabileceği en kısa yol, adım adım ızgara üzerinde gösterilmektedir. Robotun daha önce geçtiği yerler belli olacak şekilde her adımda yol üzerinde iz bırakması gerekmektedir. Hedefe ulaşıldığında ise başlangıç noktasından hedef konuma giden robota göre en kısa yol ızgara üzerinde ayrıca çizdirilmektedir. Geçen toplam süre (sn cinsinden) ve kaç kare üzerinden geçildiği bilgileri ekranda gösterilmektedir.
Problem 2 de robotu labirentteki çıkış noktasına ulaşmaktadır. Kullanıcı tarafından istenilen boyutlarda bir ızgara oluşturmaktadır. Izgara üzerine 1 nolu tipte engeller yerleştirilerek labirent oluşturulmalıdır. Labirent içerisinde mutlaka çıkışa ulaşamayan yollar bulunmaktadır. Labirentin giriş ve çıkış noktaları dörtgen ızgaranın herhangi çapraz 2 köşesi olarak belirlenmelidir. Robot başlangıçta labirenti bilmemelidir. Labirentte yanlış girilen bir yol algılandığında robotun doğru olarak tespit ettiği en son konuma giderek buradan itibaren yol aramaya devam etmektedir. Tüm bu bilgiler doğrultusunda, robotun çıkışa ulaşmak için izlediği yol adım adım ızgara üzerinde gösterilmektedir.
Her adımda robotun daha önce geçtiği yollar üzerinde iz bırakması gerekmektedir. Robot hedefe ulaştığında giriş noktasından çıkış noktasına giden yol ızgara üzerinde çizilmektedir. Geçen toplam süre (sn cinsinden), kaç kare üzerinden geçildiği bilgileri ekranda gösterilmektedir.

YÖNTEM

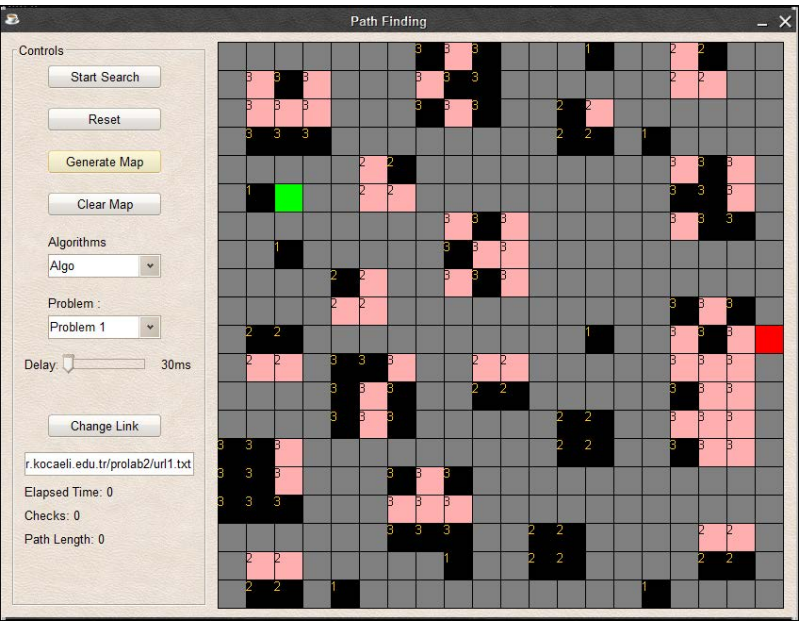
İlk olarak sınıf tanımları şu şekilde:
Robot Sınıfı: Robot sadece yukarı-aşağı ya da sağ-sol doğrultusunda hareket edebilmeli, çapraz yönde hareket etmemektedir. Robot ızgara dünyasında bulunduğu konumdan sadece bir birim sonrası ile ilgili bilgileri görebilir. Haritanın tümünü görmemektedir.
Izgara Sınıfı: Bu sınıfta problem 1 için ızgara tasarımı verilen url adresindeki text dosyasına göre oluşturulurken problem 2 için ızgara kullanıcıdan alınacak boyut bilgisine göre oluşturulmaktadır.
Engel Sınıfı: Engel sınıfında problem 1 için üç farklı tipteki nesneler kullanarak engellerin oluşturulması url adresindeki text dosyasına göre yapılırken problem 2 için labirent oluşumu 1 nolu tek bir nesne türü kullanılarak uygulama içerisinde rastgele oluşturulmaktadır.
Uygulama Sınıfı: Uygulama içerisinde robotun problem 1 ve problem 2 deki hedefe ulaşma süresi, kaç kare üzerinden geçildiği gibi bilgilerin tutulduğu ve ekranda gösterilmesi fonksiyonlarını sağlamaktadır.
Bu projedeki algoritma Up-Left-Right-Down şekilde gider ve son olarak hedef ulaştığında en kısa yol çizdirmek için A Star Algoritması çalışmaktadır.



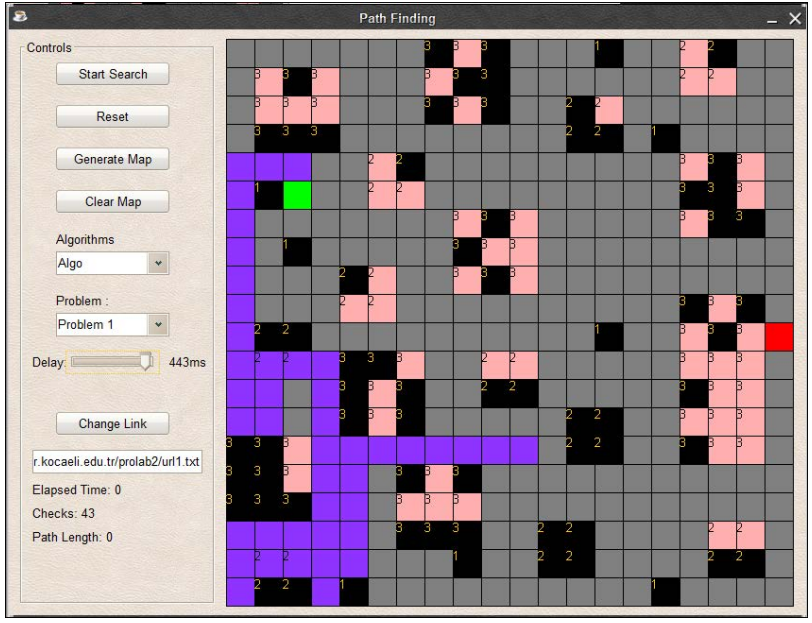
UML Diagram



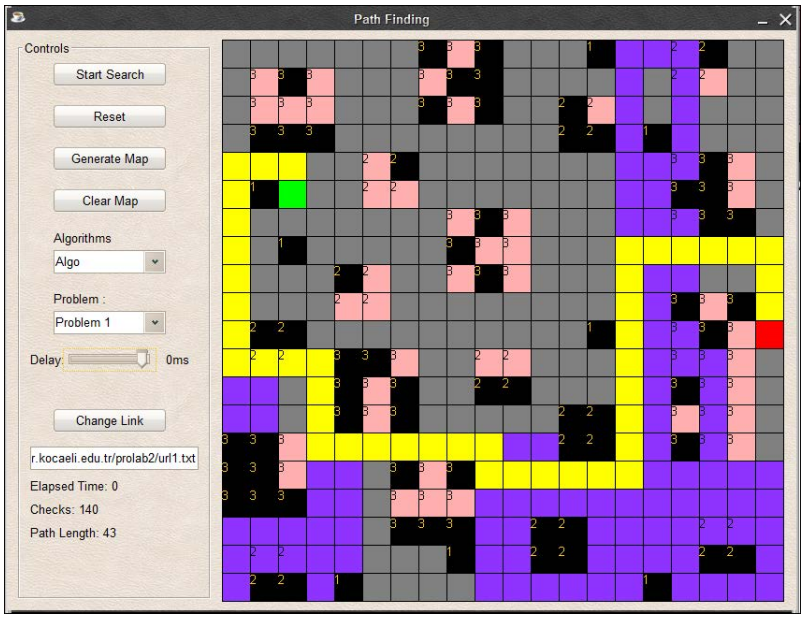
Flow Chart



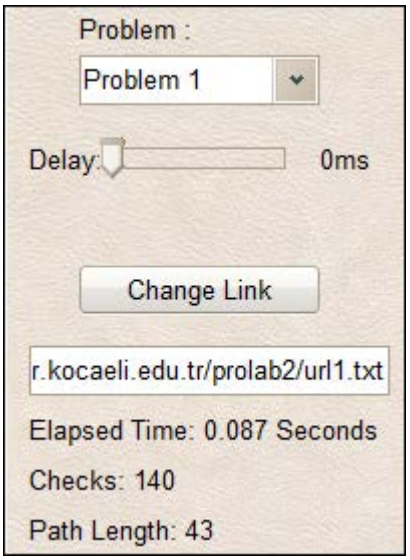
1 Problem 1



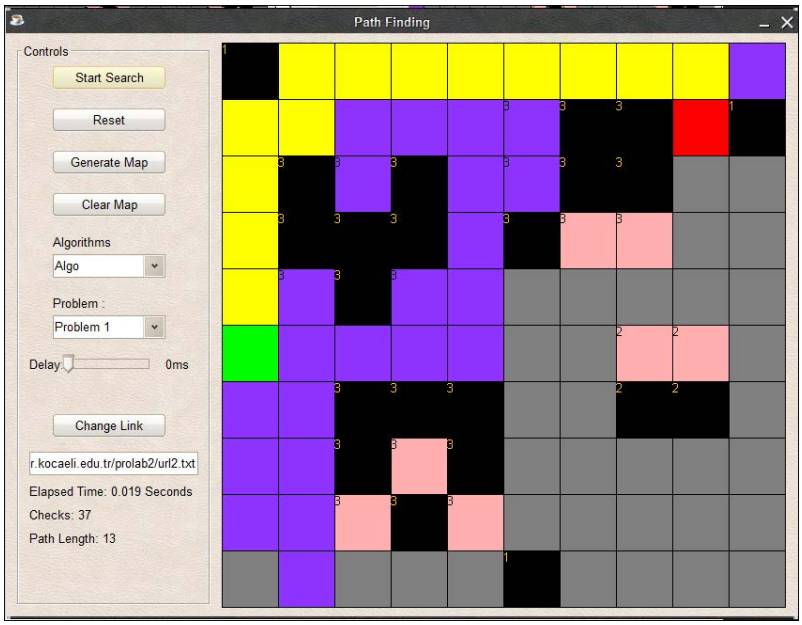
2 Problem 1



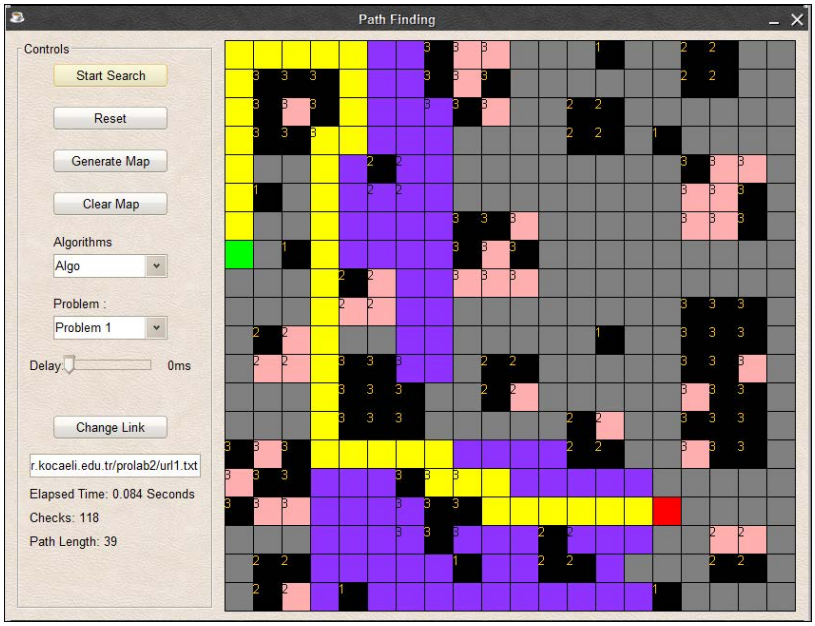
3 Problem 1



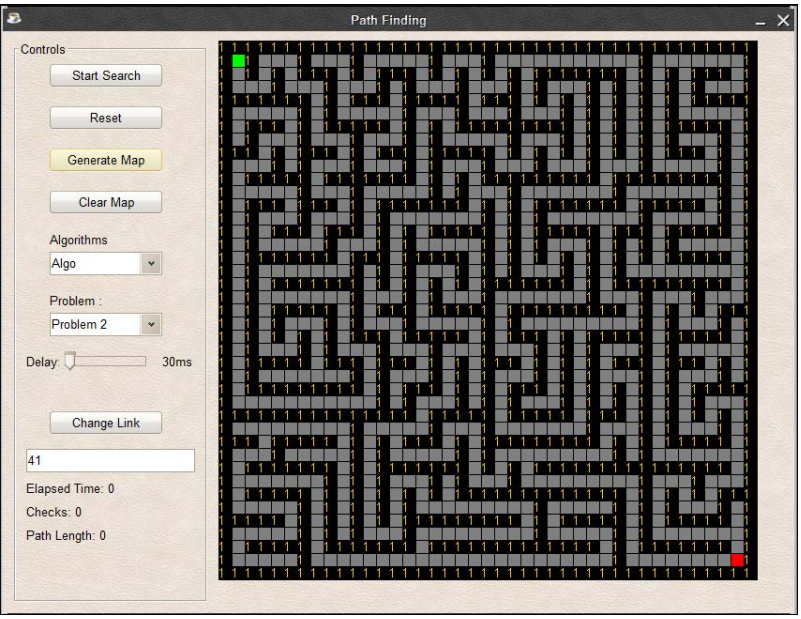
4 Side Panel



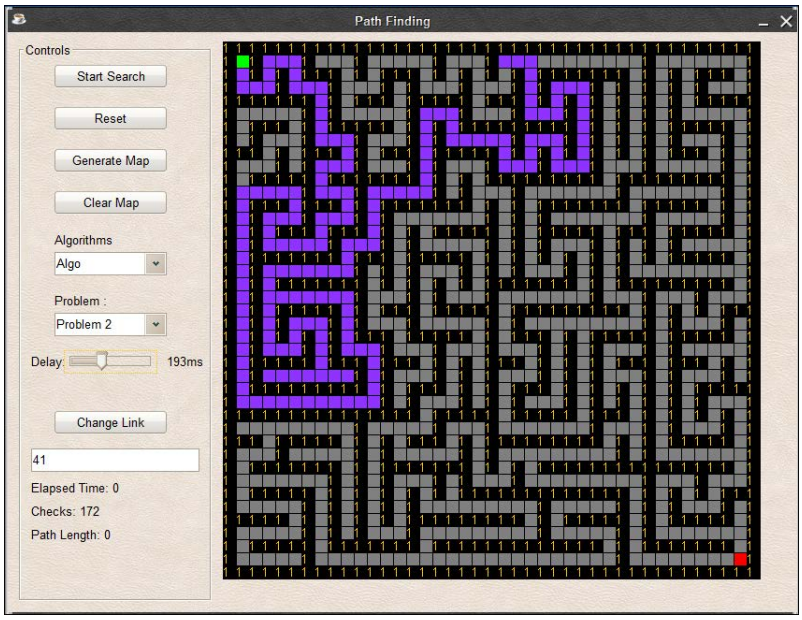
5 Problem 1



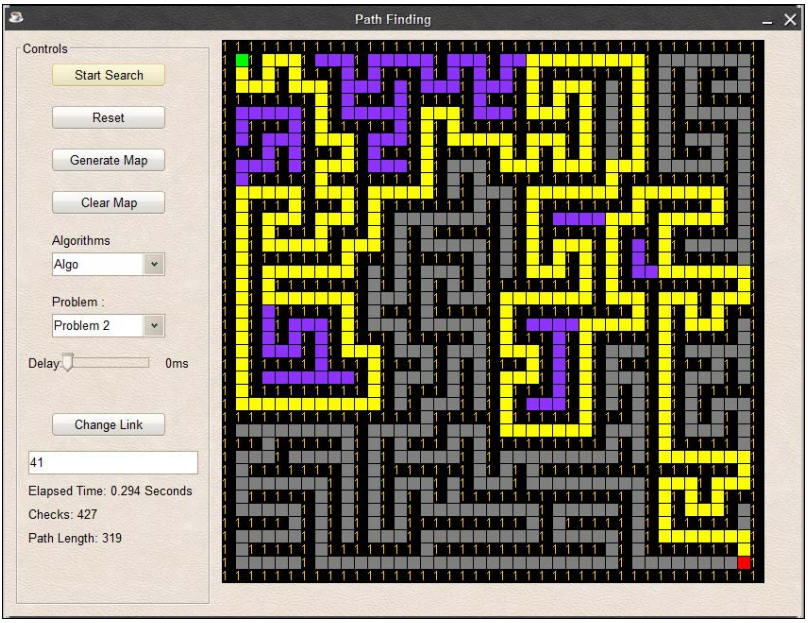
6 Problem 1



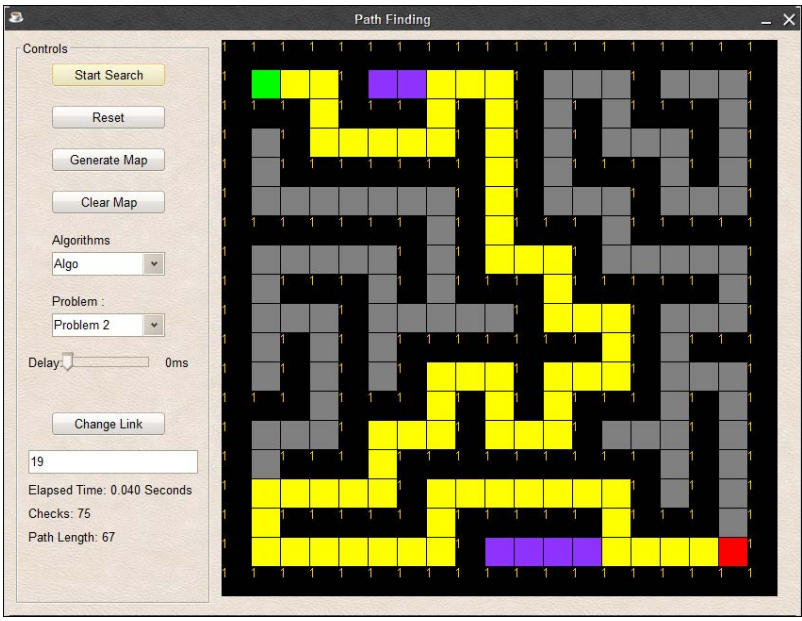
7 Problem 2



8 Problem 2



9 Problem 2



10 Problem 2

KAYNAKÇA

REFERENCES

[1] https://www.youtube.com/watch?v=0ol_PptA7rM&t=17s

[2] https://www.geeksforgeeks.org/priority-queue-set-1-introduction/

[3] https://www.programiz.com/dsa/priority-queue

[4] https://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html

[5] https://github.com/mgeee35/easyUML

[6] https://www.geeksforgeeks.org/binary-heap/

[5] https://www.sanfoundry.com/c-program-implement-heap/

[6] https://www.youtube.com/watch?v=HqPJF2L5h9Ut=1335s

[9] https://www.youtube.com/watch?v=wptevk0bshY

[7] https://www.youtube.com/watch?v= U1AJZQxYTU

[8] https://www.codesansar.com/c-programming/recursive-function.htm

[12] https://www.tutorialspoint.com/cprogramming/c recursion.htm

[13] https://www.programiz.com/c-programming/c-recursion

[14] https://www.youtube.com/watch?v=kepBmgvWNDw

[15] Book Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology by Dan Gusfield

[16] https://www.youtube.com/watch?v=ngCos392W4w

[17] https://www.youtube.com/watch?v=ggk7HbcnLG8

[18] https://www.youtube.com/watch?v=QWDCKPEWSWot=425s

[19] https://www.youtube.com/watch?v=cv7CY8UmFL0

[20] https://www.geeksforgeeks.org/readwrite-structure-file-c/

[21] https://cboard.cprogramming.com/c-programming/77351-writing-array-struct-file.html

[22] https://courses.cs.washington.edu/courses/cse373/17wi/lectures/priority-queues.pdf

[23] http://www.cs.cornell.edu/courses/cs2110/2015fa/L17-PriorityQueuesAndHeaps/cs2110PqueuesHeaps.pdf

[24] https://www.geeksforgeeks.org/priority-queue-using-binary-heap/

[25] https://runestone.academy/runestone/books/published/pythonds/Trees/BinaryHeapImplementation.html