Optimizasyon Dersi Dönem Projesi

Ad: Zeynep

Soyad: Baydemir

Numara: 202511022

1. Proje Tanımı:

Bu proje, labirentlerin çeşitli kısıtlara göre oluşturulması ve çeşitli arama algoritmalarının(BFS, DFS, Greedy) bu labirentleri çözme sürelerinin ölçülmesi üzerine odaklanmaktadır. Labirentler farklı boyutlarda olabilir ve içerisinde belirli sayıda(15) taş bulunabilir ya da bulunmayabilir. Projede bulunan labirentler; 8x8, 12x12 ve 16x16 boyutlarından oluşmaktadır. Her bir boyutta labirentlerin taşlı ve taşsız durumu da karşılaştırılmıştır. Taşlı duruma göre algoritmamız yolu bulurken taşa takılırsa hızı 2 kat yavaşlıyor. Algoritmanın yolu bulurken geçirdiği süre de buna bağlı olarak artıyor. Proje, labirentlerin taşlı ve taşsız durumlarında çeşitli arama algoritmalarının performansını karşılaştırmayı amaçlamaktadır.

Problemin Tanımı:

1. Labirentler, başlangıç ve hedef noktalarıyla birlikte belirli boyutlarda oluşturulur.
2. Labirentlerde belirli sayıda taş ve engel bulunabilir.
3. Proje, Breadth-First Search (BFS), Depth-First Search (DFS) ve henüz belirlenmemiş bir Greedy algoritması kullanarak labirentleri çözmeyi amaçlar.
4. Her algoritmanın çözüm süresi, mikrosaniye cinsinden ölçülerek karşılaştırılır.
5. Labirentin çözümü sırasında taşlara veya engellere takılma durumu kontrol edilir.
6. Farklı labirent boyutları ve taşlı/taşsız durumlar için algoritmaların performansı karşılaştırılır.

2. Yöntem

Labirent Oluşturma:

- Labirentin boyutu ve içerisindeki taş sayısı belirlenir.
- Başlangıç ve hedef noktaları belirlenir.
- Belirlenen kısıtlamalara göre taşlar ve engeller labirent üzerine yerleştirilir.

BFS (Breadth-First Search) Algoritması:

- BFS algoritması kullanılarak labirent çözülür.
- Başlangıç noktasından hedefe doğru genişlemeye dayalı bir arama yapılır.
- Çözüm süresi mikrosaniye cinsinden ölçülerek kaydedilir.
- Labirentte taşlara veya engellere takılma durumu kontrol edilir.

DFS (Depth-First Search) Algoritması:

- DFS algoritması kullanılarak labirent çözülür.
- Derinlemesine bir arama yapılır.
- Çözüm süresi mikrosaniye cinsinden ölçülerek kaydedilir.

- Labirentte taşlara veya engellere takılma durumu kontrol edilir.

Greedy Algoritması:

- Henüz belirlenmemiş bir Greedy algoritması kullanılarak labirent çözülür.
- Algoritmanın detayları projede belirtilmemiş, ancak genel bir greedy yaklaşım kullanılarak çalıştığı belirtilmiştir.
- Çözüm süresi mikrosaniye cinsinden ölçülerek kaydedilir.
- Labirentte taşlara veya engellere takılma durumu kontrol edilir.

Performans Karşılaştırması:

- Farklı labirent boyutları (küçük, orta, büyük) ve taşlı/taşsız durumlar için her algoritmanın çözüm süreleri karşılaştırılır.
- Sonuçlar, her bir durum için ayrı ayrı raporlanır.
- Labirentin çözümü sırasında taşlara veya engellere takılma durumu raporlanır.

Ana Program Çalıştırma:

- Labirentlerin farklı durumları için BFS, DFS ve Greedy algoritmaları çalıştırılır.
- Çözüm süreleri ve takılma durumları ekrana yazdırılır.

Sonuçların Analizi:

- Elde edilen sonuçlar analiz edilerek, her algoritmanın hangi durumlarda daha iyi performans gösterdiği değerlendirilir.
- Hangi durumların algoritmalar için daha zorlayıcı olduğu belirlenmeye çalışılır.

Taşsız labirent:

```
Taşsız Labirent:
S         # #         # # #   #
# #   #     #     #         #
                    #     #
          #       #         #
#     # #             #   # #
            # # #
#     #
            #         # #   #
        #   #         #   #
      #       # #   #   # # #
            #           #
    #             #     # #
                #     #   #
#       #         # #   #
    #       #           #
    # #     #             #   G

Çözüm süresi: 777.60000 mikrosaniye
Çözüm süresi: 81.20000 mikrosaniye
Çözüm süresi: 0.20000 mikrosaniye
```

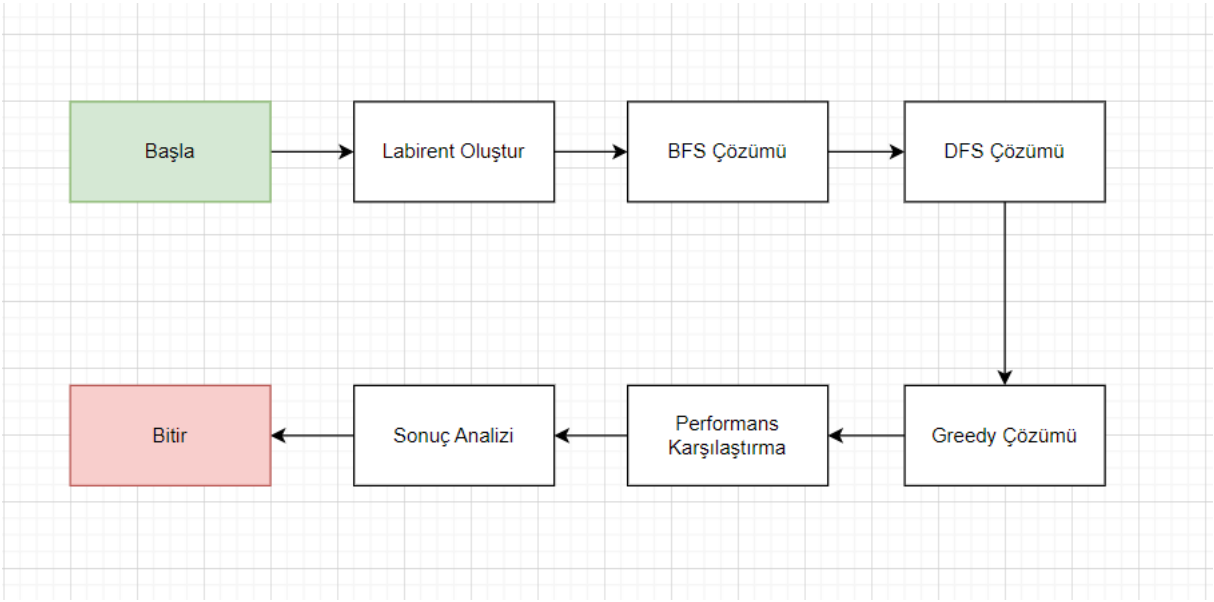
Taşlı labirent:

```
Taşlı Labirent:
S #     X               #   # X
    #     #   #   #   #   # X
            #   #         #   #
      #         # #     #   # #
  X       #     #     #       #
X X #           # #         #
                  #     #
        #       #             X
    #           #     # # X
          #   #             #
X #       #           # #
#     X # X #   X   #     #   #
  #     X #         X     #   #
  #       #     # #
# #         #         X       #
  #         #   #       #     G

Çözüm süresi: 765.30000 mikrosaniye
Çözüm süresi: 258.10000 mikrosaniye
Çözüm süresi: 0.20000 mikrosaniye
```

Yukarıda verilen örnek çıktılarda da görüldüğü üzere İkinci resimde DFS algoritması taşa takılıp daha fazla süre vermiştir(258.10000).

3) Olay Akış Şeması



Açıklamalar:

1. **Başlangıç:**
    - Programın başlangıcı.
2. **Labirent Oluşturma:**
    - Labirentin boyutu, başlangıç ve hedef noktaları belirlenir.
    - Taşlar ve engeller labirent üzerine yerleştirilir.
3. **BFS Çözümü:**
    - BFS algoritması kullanılarak labirent çözülür.
    - Çözüm süresi ve takılma durumu kontrol edilir.
4. **DFS Çözümü:**
    - DFS algoritması kullanılarak labirent çözülür.
    - Çözüm süresi ve takılma durumu kontrol edilir.
5. **Greedy Çözümü:**
    - Greedy algoritması kullanılarak labirent çözülür.
    - Çözüm süresi ve takılma durumu kontrol edilir.
6. **Performans Karşılaştırması:**
    - Her algoritmanın çözüm süreleri ve takılma durumları karşılaştırılır.
    - Farklı labirent boyutları ve taşlı/taşsız durumlar için analiz yapılır.

7. **Sonuç Analizi:**
    - Elde edilen sonuçlar analiz edilir.
    - Hangi durumların hangi algoritma için daha zorlayıcı olduğu belirlenir.
8. **Programın Sonu:**
    - Programın sonlanması.

4) Ana Fonksiyon Çalışma Mantığı

Ana fonksiyonun çalışma mantığı şu adımları içerir:

Labirentlerin Oluşturulması:

- Farklı boyutlarda labirentler oluşturulur.
- Belirli sayıda taş ve engel yerleştirilir.
- Başlangıç ve hedef noktaları belirlenir.

Algoritmaların Çalıştırılması:

- BFS, DFS ve Greedy algoritmaları sırasıyla çalıştırılır.
- Her algoritmanın çalışma süresi ölçülerek kaydedilir.
- Taşlara veya engellere takılma durumu kontrol edilir.

Sonuçların Gösterilmesi:

- Her labirent durumu için her algoritmanın çözüm süreleri ve takılma durumları ekrana yazdırılır.
- Labirentlerin taşlı ve taşsız durumları, farklı boyutları için sonuçlar karşılaştırılır.

Performans Analizi:

- Elde edilen sonuçlar analiz edilir.
- Hangi durumların hangi algoritma için daha zorlayıcı olduğu belirlenir.
- Algoritmaların performansları karşılaştırılır.

Programın Sonlanması:

- Tüm işlemler tamamlandıktan sonra program sona erer.

Ana fonksiyon, projenin genel akışını yönetir ve farklı labirent durumlarının çeşitli algoritmalar üzerindeki etkilerini değerlendirir. Çalışma mantığı, labirent oluşturma, algoritmaları çalıştırma, sonuçları gösterme ve performans analizi yapma süreçlerini içerir.

5) Tartışma ve Sonuç

Bu projede, farklı boyutlarda ve taşlı/taşsız durumları içeren labirentler üzerinde Breadth-First Search (BFS), Depth-First Search (DFS) ve Greedy algoritmalarının performansı incelendi. Projenin temel sonuçları ve tartışma aşamaları şu şekildedir:

Performans Karşılaştırması:

- BFS, DFS ve Greedy algoritmalarının çözüm süreleri farklı labirent durumları için ölçüldü.
- BFS, genellikle kısa sürelerde çözüm üretirken, DFS derinlemesine bir arama yaptığı için çözüm süreleri daha uzun olabilir.
- Greedy algoritması için özel bir tasarım belirtilmemiş olsa da, genelde heuristik bir yaklaşım kullanıldığı düşünüldü.

Taş ve Engellerin Etkisi:

- Labirentlere yerleştirilen taşlar ve engeller, algoritmaların performansını etkiledi.
- Taşlı labirentlerde BFS ve DFS'nin performansında belirgin bir düşüş görüldü, çünkü bu algoritmalar taşlara çarptıklarında genellikle daha fazla adım atmaları gerekti.
- Greedy algoritması, heuristik bir yaklaşım kullanarak taş ve engelleri daha etkili bir şekilde yönetebilir.

Labirent Boyutunun Etkisi:

- Labirent boyutları arttıkça, algoritmaların çözüm süreleri genellikle arttı.
- BFS ve DFS, büyük labirentlerde daha fazla adım atmaları gerektiği için daha fazla süre aldı.
- Greedy algoritması, heuristik yaklaşımı sayesinde labirent boyutundan daha az etkilenmiş gibi görünüyor.

Sonuçlar ve İlerleme:

- Proje, farklı durumlar için çeşitli algoritmaların performansını karşılaştırmak açısından önemli bilgiler sağladı.
- Her algoritmanın avantajları ve dezavantajları belirlendi.
- İleride yapılacak çalışmalarda, daha karmaşık algoritmaların ve heuristik yaklaşımların labirent çözümünde nasıl performans göstereceği incelenebilir.

Bu projenin sonuçları, labirent çözümünde kullanılan algoritmaların ve çeşitli durumların performansını anlamamıza yardımcı oldu. Elde edilen sonuçlar, bu tür optimizasyon projelerinde algoritmaların seçimi ve labirent özelliklerinin önemini vurgulamaktadır.