C++/Python Algoritma Problemi – Park Yeri Bulma Sistemi

Proje Amacı:

Bu projede, bir park alanındaki boş park yerlerini algılayarak, araçlar için en uygun park yerini bulan bir algoritma geliştirilmesi beklenmektedir. Algoritma, park alanını bir grid yapısında simüle ederek, kullanıcı için en uygun park yerini öneren bir çözüm sunacaktır.

Proje Gereksinimleri

1. **Park Alanı Düzeni:**
    - Park alanı, bir **grid (matris)** şeklinde temsil edilecektir. Grid boyutu belirli bir ölçekte (örneğin 5x5, 10x10) olabilir.
    - Grid üzerinde her bir hücreyi temsil eden **boş (0)** ve **dolu (1)** değerleri olacaktır.
2. **En Uygun Park Yerini Bulma:**
    - Algoritma, **en yakın boş park yerini** bulacak şekilde tasarlanmalıdır.
    - Mesafe hesaplama, **Manhattan Mesafesi** (yani, yatay ve dikey mesafelerin toplamı) ile yapılabilir.
    - Algoritma, **kullanıcı tarafından belirlenen (parametre olarak geçilen) bir başlangıç noktasına** en yakın boş park yerini bulacak ve çıktıyı verecektir.
3. **Algoritmanın Çalışması:**
    - Kullanıcı, park alanındaki boş park yerlerini görebilecek, algoritmanın en uygun park yerini bulmasını istediği başlangıç noktasını belirleyebilecektir.
    - Algoritma, grid üzerinde tüm boş park yerlerine olan mesafeyi hesaplayarak en kısa mesafeyi bulacak ve bu park yerini kullanıcıya bildirecektir.

****Algoritma Önerileri**

- Breadth-First Search (BFS)
- Dijkstra's Algoritması

4. **Çıktı ve Görselleştirme:**
    - **Metin Çıktısı:** Algoritma, park alanındaki en uygun boş park yerinin koordinatlarını ve bu park yerine olan mesafeyi metin olarak verecektir. Örneğin, "En uygun park yeri: (3, 4), Mesafe: 5 birim" şeklinde bir mesaj ile kullanıcı bilgilendirilecektir.
    - **Görselleştirme:** Park alanı ve en uygun park yeri, bir grafik (grid) üzerinde görselleştirilecektir. Park alanı bir grid (matris) şeklinde gösterilir. Grid üzerinde:

Boş park yerleri yeşil renkle,

Dolu park yerleri kırmızı renkle,

En uygun park yeri ise mavi renkle

belirtilecektir.

Teknolojiler ve Araçlar

- **Diller:** Python veya C++
- **Veritabanı / Veri Yapıları:** Grid yapısını modellemek için 2D diziler veya listeler kullanılabilir.
- **Görselleştirme Araçları (Opsiyonel):**
    - Python için: matplotlib, Plotly, veya pygame
    - C++ için: Console-based görselleştirme

Dikkat Edilmesi Gerekenler

1. **Algoritmanın Verimliliği:**
    - Park yerlerinin mesafelerini hesaplamak için kullanılan algoritma, en kısa mesafeyi **doğru ve hızlı** şekilde bulmalıdır.
    - Bu süreçte kullanılan algoritmaların zaman karmaşıklığı önemlidir. BFS veya Dijkstra gibi algoritmalar verimli çözümler sunabilir.
2. **Veri Yapıları ve Kod Organizasyonu:**
    - Algoritmanın ve park alanının verimli bir şekilde yönetilebilmesi için uygun veri yapıları kullanılmalıdır.
    - *Clean Code* prensiplerine dikkat edilmelidir. Kodun tekrar kullanılabilir ve bakımı kolay olması için doğru fonksiyonlar ve isimlendirmeler kullanılmalıdır. Özellikle, isimlendirme formatının tutarlı olması, kodun okunabilirliğini etkileyen önemli bir faktör olarak göz önünde bulundurulmalıdır.
3. **Test Senaryoları:**
    - Farklı park alanı düzenleri (tamamen dolu, tamamen boş, rastgele dağılmış park yerleri) üzerinde testler yapılmalıdır.
    - Kullanıcının park yeri için yaptığı taleplere karşı doğru sonuçlar üretildiği doğrulanmalıdır.

Proje Teslimi

- **GitHub Linki:** Proje tamamlandığında, GitHub linki üzerinden teslim edilmelidir.

- **Sunum:** Proje sonuçları ve algoritmanın nasıl çalıştığı ile ilgili bir sunum hazırlanacaktır. Sunumda algoritmanın doğruluğu ve test senaryoları grid üzerinden anlatılmalıdır.