1. **“main.py” DOSYASI:**

**1.1.** Arama işlemlerinin birleştirildiği dosya. “dfs.py”, “bfs.py”, “ucs.py”yi içe aktarır ve fonksiyonlarına erişim sağlar.

**1.2.** “controller” değişkeni: Klavyeden girilen sayıya “input()” metoduyla atanıp, girilen sayının if-else yapısı ile kontrol edilmesini sağlayan değişken. Kontrol sonucunda 3 arama tipinden birine gidilmesinin sağlanmasında görev alır.

**1.3.** “x”,”y,” değişkenleri: x, PacMan’in başlayacağı başlangıç düğümünün, y ise PacMan’in gitmesi gereken bitiş düğümünün bilgilerini üstünde tutan değişkendir. “input()” metodu kullanılarak kullanıcıdan giriş alınması sağlanır ve if-else yapısı ile aynı olup olmadığı kontrol edilir.

**“dfs.py”, “bfs.py” ve “ucs.py” dosyaları:**

**2. ORTAK YÖNLERİ:**

**2.1.** PacMap adlı harita: El ile girilmiş değerlerden oluşan, 10 x 10’luk bir harita. PacMap’de girilen her değerin ilk kısmı onun koordinatlarını, ikinci kısmı ise o kısmın durumunu üstünde tutar. İkinci kısımda “X”ler duvarları, boşluklar gidilebilecek yolları, “Y” PacMan’in yemesi gereken yemeği ve “C” ise PacMan’i temsil eder.

**2.2.** PacDict adlı sözlük: El ile girilmiş değerlerden oluşan, PacMap’deki boşlukları düğüm kabul eden ve bu düğümlerin komşu olanlarını birbirine bağlayan sözlük. Depth First Search, Breadth First Search ve Uniform Cost Search gibi algoritmalar, arama yaparken bu **“GRAF”** tarzındaki sözlüğü kullanırlar. İlk kısım, anahtar sözcük, ikinci kısım ise anahtar sözcüğün değeri olarak tanımlanmıştır. İkinci kısımlar, birden fazla değer içerebilmesi açısından (1 düğüm 1’den fazla düğüme bağlı olabileceği için) liste şeklinde belirtilmişlerdir.

**2.3.** MapDrawer() adlı metot: Bu metot, PacMap’in güncel halinin ekrana yazdırılmasını sağlar. Bekleme (güncelleme) süresi olarak 0.5 (yarım) saniye kullanılır. Bu kısımda, bekleme süresi için “time.py” adlı modül içe aktarılıp, “time.sleep()” metodu kullanılmıştır.

1. **“**i” değişkeni: MapDrawer() metodunda, PacMap’te gezebilmek için kullanılan sayaç.

**2.4.** PacFinder() adlı metot: PacMan’in, arama algoritmalarına ait dosyalardaki arayıcı fonksiyonu çağıran ve PacMan’in güncel halini belirleyen fonksiyon. Aynı zamanda PacMan’in hangi düğümden hangi düğüme gideceği bilgisini de kullanıcıdan istenilen “x,y” bilgisi ile edinir. Böylelikle arama algoritmalarına düğüm bilgisini gönderir.

1. “a” değişkeni: Kullanıcı tarafından girilen “x,y” değerlerine göre PacMap’de PacMan’i ve yemi yer aldıkları düğüme yerleşmelerini sağlayan döngüde PacMap içinde gezinmek için kullanılan sayaç.
2. “list” adlı liste: Arama algoritmalarının geri dönüş değerlerini üstünde tutan liste.
3. “x,y,z” değişkenleri: PacMan’in lokasyonlarını güncelleyen döngülerde kullanılan sayaç değerleri.
4. **ORTAK OLMAYAN YÖNLERİ:**

**“dfs.py”:**

* 1. “dfs()” adlı metot: Derinlik öncelikli arama (Depth First Search) algoritmasını içeren fonksiyon. Graf bilgisini, başlangıç düğümünü ve bitiş düğümünü ister.

1. “stack” adlı yığın veri yapısı: Gerçekleştirimde kullanılan yığın. Fonksiyonun başlangıç kısmında, başlangıç düğüm değerini listeye ekleyerek tanımlanmaktadır. (başlangıç durumu olduğu için).
2. “visited” adlı set: Set türünde, ziyaret edilen düğümleri üstünde tutan yapı.
3. “node” adlı değişken: O andaki düğümün bilgisini içeren değişken.
4. “path” adlı liste: PacMan’in gitmesi gereken yolu tutan liste.
5. “pop()” adlı metot: Yığının son elemanını döndüren fonksiyon.
6. “add()” adlı metot: “visited” adlı set türü yapıya değer eklemek için kullandığımız fonksiyon.
7. “append()” adlı metot: Yığının sonuna eleman eklemek için kullandığımız fonksiyon.
8. “childnode” adlı değişken: O anki düğümün bir sonraki komşu düğümünün değerini tutan değişken.

**“bfs.py”:**

**3.2. “**bfs()” adlı metot: Genişlik öncelikli arama (Breadth First Search) algoritmasını içeren fonksiyon. Graf bilgisini, başlangıç düğümünü ve bitiş düğümünü ister.

1. “queue” adlı kuyruk veri yapısı: Gerçekleştirimde kullanılan kuyruk veri yapısı.
2. “append()” adlı metot: Listelerin ve kuyruğun sonuna eleman eklemesi yapmak için kullandığımız fonksiyon.
3. “path” adlı liste: PacMan’in gitmesi gereken yolu tutan liste.
4. “node” adlı değişken: Düğüm bilgisini tutan değişken. “path” adlı listenin en son değerine eşitlenir. Eğer bitiş düğümüyse yol bilgisini geriye döndürülmesini sağlar.
5. “new\_path” adlı liste: Hedef düğüme ulaşana kadar ara adımlarda kullanılan geçici liste. Değer olarak “path” adlı listeyi alır ve kuyruğun sonuna atanır.
6. “childnode” adlı değişken: O anki düğümün bir sonraki komşu düğümünün değerini tutan değişken.

**“ucs.py”:**

* 1. “find\_all\_paths” adlı metot: PacMan’in yeme ulaşması için yapabileceği tüm hamleleri belirleyen fonksiyon. Graf, başlangıç düğümü ve bitiş düğümü bilgilerini ister. Ayriyeten başlangıç olarak “path” adlı listenin boş olmasını sağlar.

1. “path” adlı liste: PacMan’in yeme gitmesi için gidebileceği yol bilgisini tutan liste.
2. “paths” adlı liste: PacMan’in yeme gitmesi için gidebileceği tüm yol bilgilerini tutan liste. “path”den farkı, sadece anlık yol bilgisini tutmaz, tüm yol bilgilerini tutar. “path” ise sadece 1 yol bilgisini tutar.
3. “newpaths” adlı liste: Yol bilgisinin yazdırılmasında kullanılan ara liste.
4. “newpath” adlı değişken: “newpaths” adlı listenin elemanlarını anlık olarak tutan değişken.
5. “append()” adlı metot: “paths” adlı listenin en sonuna eleman eklemesi yapmak için kullandığımız fonksiyon.
   1. **“**PacFinder()” adlı metottaki farklılığı: Uniform Cost Search algoritmasını tam olarak işletebilmek için geriye dönülen tüm yolların en az maliyetli olanını (burada en kısayı kullandık çünkü her düğüm arası mesafe eşit, o yüzden en kısa yolu almak aynı işlevi görmesini sağlayacaktır.) bulmasını sağlayan algoritmamız bulunmaktadır.
6. “shortest” adlı liste: Uniform Cost Search için kullandığımız liste. For döngüsü ile geri dönüş değerini alan “list” adlı listedeki tüm elemanları “i” adlı değişkene atıyoruz sırasıyla ve if-else yapısı ile en kısa olanını belirleyip “shortest” adlı listeye yazdırıyoruz. Son olarak da “shortest” adlı listeyi ekrana yazdırıyoruz.