GİRESUN ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
VERİ YAPILARI DERSİ TELAFİ SINAVI

Adı – Soyadı – Numarası:

Soru 1: Parametre olarak verilen bir dizide en çok tekrar eden elemanın değerini ve tekrar etme sayısını konsola yazdıran aşağıdaki fonksiyonu tamamlayınız.

```
void modHesapla(int[] dizi) {

	int enCokTekrarEden = dizi[0];  // En cok tekrar eden sayi, ilk eleman ile baslatiliyor
	int enCokTekrarSayisi = 1;      // Ilk elemanin tekrar sayisi 1
	int mevcutTekrarEden = dizi[0]; // Su anda kontrol edilen en cok tekrar eden sayi
	int mevcutTekrarSayisi = 1;    // Bu sayinin tekrar sayisi

	for (int i = 1; i < dizi.length; i++) {
		if (dizi[i] == dizi[i - 1]) {
			// Eger iki ardil eleman esit ise, mevcut sayinin tekrar sayisini arttir
			mevcutTekrarSayisi++;
		} else {
			// Farkli bir eleman bulundu, mevcut sayiyi degistir
			mevcutTekrarSayisi = 1;
			mevcutTekrarEden = dizi[i];
		}

		// En cok tekrar eden sayiyi ve tekrar sayisini guncelle
		if (mevcutTekrarSayisi > enCokTekrarSayisi) {
			enCokTekrarSayisi = mevcutTekrarSayisi;
			enCokTekrarEden = mevcutTekrarEden;
		}
	}

	System.out.println("Mod: " + enCokTekrarEden + " (Tekrar: " + enCokTekrarSayisi + ")");
}
```

Soru 2: Aşağıda verilen DaireselBagliListe sınıfı için listenin tüm elemanlarını konsola yazdıran fonksiyonu tamamlayınız.

```
public class DaireselBagliListe {

	private TekYonluDugum<Integer> ilk; // Dairesel bagli listenin ilk dugumu
	private TekYonluDugum<Integer> son; // Dairesel bagli listenin son dugumu

	public void yazdir() {
		if (ilk == null) {
			System.out.println("Bos Dairesel Bagli Liste"); // Liste bossa bildir
			return;
		}

		TekYonluDugum<Integer> simdiki = ilk; // Baslangic dugumu ile basla
		System.out.print("Dairesel Bagli Liste: ");
		do {
			System.out.print(simdiki.veri + " -> "); // Dugumun verisini yazdir
			simdiki = simdiki.sonraki; // Bir sonraki dugume gec
		} while (simdiki != ilk); // Basa donene kadar donguyu devam ettir
		System.out.println("Basa Don"); // Donguyu tamamladiginda basa donuldugunu belirt
	}
}
```

Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 9.12.2024

GİRESUN ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
VERİ YAPILARI DERSİ TELAFİ SINAVI

Soru 3: Aşağıda verilen sınıf dizi veriyapısı kullanarak yığıt veriyapısını gerçeklemektedir. Verilen ekle ve cikar fonksiyonlarını tamamlayınız.

```
public class YigitDiziGosterimi {

    private int kapasite; // Yiginin maksimum kapasitesini belirtir.
    private int tepe; // Yiginin tepe elemanini isaret eder.
    private int[] dizi; // Yiginin elemanlarini saklayan dizi.

    public void ekle(int deger) {
        if (tepe == kapasite - 1) {
            // Ekleme yapilamazsa bilgi mesaji yazdirir.
            System.out.println("Yigin dolu. " + deger + " eklenemedi.");
        } else {
            dizi[++tepe] = deger; // Yeni degeri tepeye ekle ve tepeyi arttir.
            System.out.println(deger + " yigina eklenmistir.");
        }
    }
    public int cikar() {
        if (tepe == -1) {
            // Cikarma yapilamazsa bilgi mesaji yazdirir ve belirleyici bir deger dondurur.
            System.out.println("Yigin bos. Cikarma yapilamaz.");
            return -1;
        } else {
            // Tepedeki elemani cikar ve tepeyi azalt.
            int cikarilanDeger = dizi[tepe--];
            System.out.println(cikarilanDeger + " yigindan cikarilmistir.");
            return cikarilanDeger;
        }
    }
}
```

Soru 4: Dizi (Array) ve Bağlı liste (Linked list) veri yapılarının birbirlerine karşı avantaj ve dezavantajlarından bahsediniz.

Dizi (Array) Avantajları: Dizilerde elemanlara indeksleme yöntemiyle O(1) zaman karmaşıklığında erişilebilir. Doğrudan adresleme sayesinde gerçekleşir. Dizi elemanları bellekte ardışık olarak tutulduğu için daha az bellek harcar. Dizi yapısı sabittir, bu nedenle kullanımı ve uygulanması daha basittir.

Dezavantajları: Dizi oluşturulurken boyutunun önceden belirlenmesi gerekir. Gereğinden fazla bellek ayırılırsa israf olur, yetersiz yer ayırılırsa hata alınır. Ortaya veya başa bir eleman eklemek/silmek, elemanların kaydırılmasını gerektirir ve O(n) zaman alır. Dizi boyutu sonradan değiştirilemez.

Bağlı Liste (Linked List) Avantajları: Bağlı listeler, eleman ekleme/silme işlemlerinde boyutlarını dinamik olarak değiştirebilir, bellek kullanımı daha verimlidir. Listenin başına, sonuna veya ortasına eleman eklemek/silmek oldukça kolay ve O(1) veya O(n) zaman alır (konuma bağlı olarak). Elemanlar bellekte ardışık olmayan yerlerde tutulabilir, bu da belleğin daha esnek kullanılmasını sağlar.

Dezavantajları: Bir elemanı bulmak için baştan başlayarak tüm listeyi dolaşmak gerekebilir. Rastgele erişim mümkün değildir ve O(n) zaman alır. Her eleman için bir veri ve bir de işaretçi (pointer) saklanır. Bu nedenle daha fazla bellek tüketir. Bağlı listenin uygulanması dizilere göre daha karmaşıktır ve hata yapma olasılığı yüksektir (örneğin, işaretçi hataları).

Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 9.12.2024

GİRESUN ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
VERİ YAPILARI DERSİ TELAFİ SINAVI

Soru 5: Yığıt (Stack) ve Kuyruk (Queue) veri yapıları birbirlerinin yerine kullanılabilir mi? Kullanım alanlarına örnek veriniz. Birbirlerine karşı avantaj ve dezavantajlarından bahsediniz.

Yığıt ve kuyruk, temel veri yapılarıdır ve birbirlerinden farklı işlevlere sahiptirler.

Yığıt (Stack): Son giren, ilk çıkar (LIFO - Last In, First Out) prensibiyle çalışır.

Kuyruk (Queue): İlk giren, ilk çıkar (FIFO - First In, First Out) prensibiyle çalışır.

Yığıt (Stack) Kullanım Alanları: Programlama dillerinde çağrı yığınları (call stack) kullanılarak, fonksiyonların sıralı bir şekilde çağrılması ve geri dönmesi sağlanır. Metin editörlerinde yapılan işlemleri geri almak için yığıt kullanılır. Kod yazarken açma ve kapama parantezlerinin dengeli olup olmadığını kontrol etmek için kullanılır. Çizge veya ağaç yapılarında derinlik öncelikli arama algoritmasında yığıt kullanılır.

Kuyruk (Queue) Kullanım Alanları: Yazıcı işlemleri, işlemci görevleri gibi iş sırasının düzenli bir şekilde tutulması gerektiği durumlarda kullanılır. Çizge veya ağaç yapılarında genişlik öncelikli arama algoritmasında kuyruk kullanılır. Veri paketlerinin bir sırayla işlendiği ağ sistemlerinde (örneğin, veri yönlendirme). Banka, bilet kuyruğu gibi ilk giren kişinin öncelikli olduğu durumlarda.

Birbirlerinin Yerine Kullanılabilir mi?

Yığıt ve kuyruk, farklı prensiplere dayandıkları için birbirlerinin yerine doğrudan kullanılamazlar. Örneğin: Undo/Redo işlemleri: Bu işlemler sıralı bir şekilde geriye gidilmesi gerektiğinden yığıt gerektirir. Kuyruk kullanılması bu işlevi sağlayamaz. İş sırası: İlk gelenin öncelikli olduğu bir işleme kuyruk gerekir; yığıt, sıralamayı tersine çevireceği için uygun değildir. Ancak, belirli bir problemi çözmek için bu veri yapılarından biri diğerine dönüştürülebilir. Örneğin, kuyruk özelliklerini taklit etmek için iki yığıt kullanılabilir. Ancak bu, çözümün performansını düşürebilir.

Yığıtın Avantajları ve Dezavantajları: Basit ve hızlıdır. LIFO prensibi, geri alma işlemleri gibi durumlar için idealdir. Sadece yığının en üstündeki elemana erişilebilir. FIFO gerektiren durumlarda yetersizdir.

Kuyruğun Avantajları ve Dezavantajları: FIFO düzeni sayesinde sıralı işlemler için uygundur. Ekleme ve çıkarma işlemleri baştan ve sondan kolayca yapılır. Rastgele erişim mümkün değildir. Öncelik sırası gerektiren durumlarda yeterli olmaz (öncelikli kuyruk kullanılabilir).

Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 9.12.2024