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

Adı – Soyadı – Numarası:

Soru 1: Diziye indeks ile erişim hızlıdır. Bu erişim hızının nedeni nedir? Dizinin herhangi bir elemanına erişimin O(1) olması ne anlama gelir?

Diziye indeks ile erişimin hızlı olmasının nedeni, dizinin bellekte ardışık (contiguous) olarak saklanmasıdır. Diziler, bellekte sıralı bir şekilde yerleştirilir, yani her bir eleman bir öncekinden hemen sonra gelir. Bu düzen, belirli bir elemanın adresine ulaşmayı hızlı hale getirir.

Diziye erişim işlemi O(1) zaman karmaşıklığına sahiptir, çünkü:

Doğrudan Bellek Erişimi: Dizilerde her elemanın bellekteki konumu sabittir. Bir elemanın bellekteki adresine, dizinin başlangıç adresi ile elemanın indeksinin çarpımıyla doğrudan ulaşılabilir. Örneğin, dizinin başlangıç adresine bir "offset" (kayma) eklenerek ilgili elemana hızlıca erişilebilir. Bu nedenle, herhangi bir eleman için önceki elemanları taramak gerekmez. Erişim zamanının sabit olması, erişimin O(1) olduğunu gösterir.

Sabit Süreli Erişim: O(1), "sabit zaman" anlamına gelir. Yani, dizi elemanına erişmek için gereken süre, dizinin boyutuna bakılmaksızın her zaman aynıdır. Eleman sayısı arttıkça harcanan zaman artmaz. Bu, özellikle büyük dizilerde büyük bir avantajdır.

Soru 2: Bağlı listelerde bellekte her bir düğüm için ekstra bir alan ayrılır. Bu bellek yönetimi diğer veri yapılarıyla karşılaştırıldığında hangi avantajlar ve dezavantajları sunar?

Bağlı listeler, bellek yönetimi açısından bazı avantajlar ve dezavantajlar sunar.

Avantajlar:

Dinamik Bellek Kullanımı: Bağlı listeler, belleği dinamik olarak ayırır. Yani, yalnızca ihtiyaç duyulan bellek miktarı kadar yer ayırılır ve eleman eklemek ya da çıkarmak, belleği sürekli yeniden düzenlemeyi gerektirmez. Dizilerde, boyut önceden belirlenmiş olduğundan, dizinin büyütülmesi veya küçültülmesi gerektiğinde fazla bellek harcanabilir veya bellek yetersiz kalabilir. Bağlı listelerde, her düğüm yalnızca bir sonraki düğümün adresini (veya iki yönlü listelerde bir önceki ve sonraki düğüm adreslerini) tutar. Bu, bellek kullanımını daha verimli kılar.

Esneklik ve Yüksek Performans: Bağlı listelerde, eleman eklemek veya silmek O(1) zaman karmaşıklığına sahiptir, çünkü sadece düğümlerin bağlantıları değiştirilir. Bu, dizilere göre büyük bir avantajdır, çünkü dizilerde bir eleman eklemek veya silmek O(n) zaman alabilir (özellikle dizinin başında veya ortasında).

Dezavantajlar:

Ekstra Bellek Kullanımı: Her düğüm, veriyi tutmanın yanı sıra, bir veya iki ek işaretçi (pointer) tutar. Bu, her düğüm için ekstra bellek gerektirir. Örneğin, bir bağlı listede her düğüm, bir veri elemanı ve bir sonraki düğümü işaret eden bir işaretçi içerirken, bir çift yönlü bağlı listede her düğüm, bir önceki ve bir sonraki düğümü işaret eden iki işaretçi tutar. Bu ekstra işaretçiler, veri yapısının bellek tüketimini artırır.

Bellek Erişim Hızı: Bağlı listelerde, her düğüme erişim O(n) zaman alır çünkü bir elemandan diğerine geçmek için sırasıyla düğümler arasında ilerlemek gerekir. Yani, bir elemanın bellekteki fiziksel konumu ardışık değildir. Bu, dizilerdeki sabit zamanlı (O(1)) erişim hızına göre daha yavaştır. Bellek erişimlerinin daha düzensiz olması, CPU önbelleği açısından verimsiz olabilir, çünkü bellek erişimi ardışık değil ve bellekte düğümler arasında büyük mesafeler olabilir.

Yönetim Karmaşıklığı: Bağlı listelerde düğümlerin işaretçilerini güncellemek ve yönetmek, daha fazla karmaşıklık getirebilir. Yeni düğümler eklerken veya mevcut düğümleri silerken, işaretçilerin doğru şekilde ayarlanması gerekir.

Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 6.11.2024

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

Soru 3: Dairesel bağlı liste (circular linked list) ile tek yönlü bağlı liste arasındaki farkları açıklayınız. Dairesel liste üzerinde gezinirken listenin sonuna gelindiği nasıl anlaşılır?

Dairesel bağlı liste (circular linked list) ile tek yönlü bağlı liste (singly linked list) arasındaki başlıca farklar, listenin yapısı ve elemanlara nasıl erişildiğiyle ilgilidir.

Tek Yönlü Bağlı Liste (Singly Linked List):

Bu tür listelerde her düğüm, veriye ve bir sonraki düğümün adresine (işaretçisine) sahiptir.

Listenin sonundaki düğümün next işaretçisi null değerini tutar, yani son düğüm, başka eleman olmadığını belirtir.

Listeyi baştan sona doğru gezmek mümkündür; ancak listenin sonuna ulaşıldığında, işlem durur çünkü next işaretçisi null'a işaret eder.

Son düğüm, next işaretçisi null olan düğümdür. Yani, gezintiye başlamak ve bitirmek için null işaretçisi kullanılabilir.

Dairesel Bağlı Liste (Circular Linked List):

Dairesel bağlı listelerde, her düğümde veri ve bir sonraki düğümün adresi bulunur, fakat son düğümün next işaretçisi null yerine ilk düğümün adresine işaret eder. Yani, son düğümün next işaretçisi, başa dönerek bir döngü oluşturur.

Bu yapıda, son düğümün "son" olduğunu anlamak yerine, listenin sonuna veya herhangi bir düğümün sonuna gelindiğinde, listenin başına geri dönülecek şekilde döngüsel bir gezinti yapılır.

Dairesel listelerde bir giriş noktası (head pointer) bulunur ve bu giriş noktası, döngüsel gezintiyi anlamak için kullanılabilir. Eğer başa döner ve aynı düğüm tekrar görülürse, listenin sonuna gelindiği anlaşılır.

Soru 4: Aşağıda verilen sınıf tanımında, dizi yapısı kullanılarak Yığıt (Stack) veri yapısı geliştirmek için gerekli ekle (push) metodunu yazınız.

```
public class YigitDiziGosterimi {

	private int kapasite;
	private int tepe;
	private int[] dizi;

	public void ekle(int deger) {
		if (tepe == kapasite - 1) {
			System.out.println("Yığın dolu. " + deger + " eklenemedi.");
		} else {
			dizi[++tepe] = deger;
			System.out.println(deger + " yığına eklenmiştir.");
		}
	}
}
```

Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 6.11.2024

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

Soru 5: Aşağıda verilen sınıf tanımında, bağlı liste (linkedlist) yapısı kullanılarak Kuyruk (Queue) veri yapısı geliştirmek için gerekli cikar (dequeue) metodunu yazınız.

```
public class KuyrukBagliListeGosterimi<E> {

	private TekYonluDugum<E> bas;
	private TekYonluDugum<E> son;
	private int boyut;

	public E cikar() {
		// Eğer kuyruk boşsa, hata ver
		if (boyut == 0) {
			throw new IllegalStateException("Kuyruk boş");
		}

		// Öğe çıkarma işlemi: Baş düğümdeki veriyi al
		E veri = bas.veri;

		// Baş düğümünü bir sonraki düğüme taşıyarak öğeyi kuyruktan çıkar
		bas = bas.sonraki;

		// Kuyruktaki öğe sayısını azalt
		boyut--;

		// Çıkarılan öğeyi döndür
		return veri;
	}
}
```

Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 6.11.2024