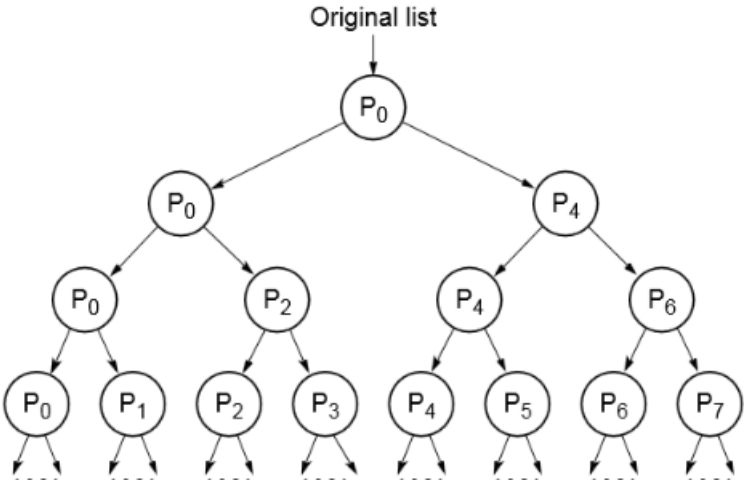
Paralel Algoritmalar

- Paralel algoritma oluşturmak için tipik adımlar
  - aynı anda hangi iş parçalarının gerçekleştirilebileceğini belirlemek
  - bağımsız işlemcilere bölme ve eşleme çalışması
  - bir programın giriş, çıkış ve ara verilerini dağıtmak
  - paylaşılan verilere erişimleri koordine etmek: çakışmaları önlemek
  - senkronizasyonu kullanarak doğru çalışma sırasını sağlamak

Paralel Algoritmalar

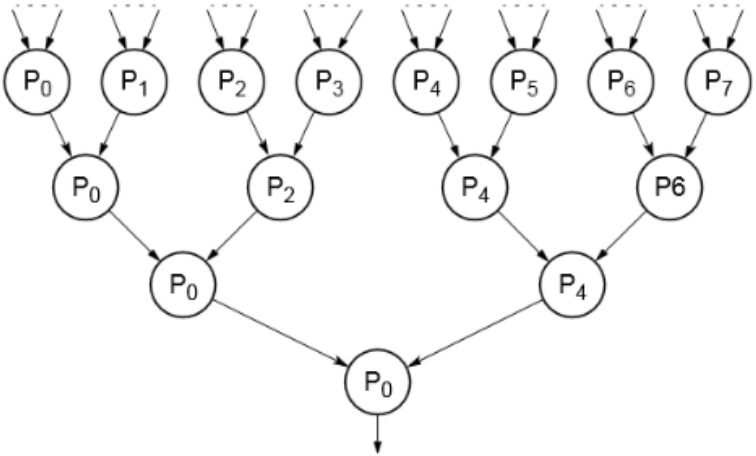
- Böl ve fethet!
  - problemin alt problemlere ayrılması
  - bölünme genellikle özyineleme yoluyla yapılır
  - alt problemlerden elde edilen çözümler daha sonra orijinal probleme çözüm vermek için birleştirilir

Divide



Kaynak

Conquer



Kaynak

Paralel Algoritmalar

- Sıralama
  - elemanların sırasız bir listesinin artan veya azalan sıraya göre düzenlenmesi işlemi,
  - en yaygın olarak incelenen algoritmik problemlerden biri olarak kabul edilir,
  - çok çeşitli yararlı uygulamalarda sıklıkla kullanılır.

Kaynak

Paralel QuickSort Algoritmaları

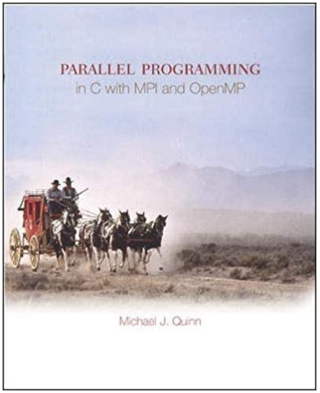
- Hızlı Sıralama
  - Böl ve fethet!
  - Listeyi daha küçük alt listelere bölerek sırasız bir sayı listesini artan bir sırayla sıralama

- Paralel hesaplamadaki ilerlemeyle birlikte birçok paralel sıralama algoritması araştırılmıştır.

Kaynak

Paralel QuickSort Algoritmaları

- Sıralı Quicksort Algoritması
- Paralel Hesaplama için Üç Hızlı Sıralama Algoritması
- Michael J. Quinn, MPI ve OpenMP ile C'de Paralel Programlama (Bölüm 14)



Kaynak

Sıralı Hızlı Sıralamanın Özeti

- Sıralı quicksort algoritması: özyinelemeli bir prosedür
  - Bir sayı listesi verildiğinde, bunlardan birini pivot olarak seçin
  - Referans değer olarak kabul edilen pivot
  - Listeyi iki alt listeye ayırın:
    - pivottan daha küçük sayılar içeren bir "düşük liste"
    - pivottan daha büyük sayılar içeren bir "yüksek liste"

Kaynak

Sıralı Hızlı Sıralamanın Özeti

- Düşük liste ve yüksek liste, kendilerini sıralamak için prosedürü özyinelemeli olarak tekrarlar
- Nihai sıralanmış sonuç, sıralanmış düşük listenin, pivotun ve sıralanmış yüksek listenin birleştirilmesidir

Kaynak

Unordered list of values

Kaynak

Sıralı Hızlı Sıralamanın Özeti

Choose pivot value

Kaynak

Sıralı Hızlı Sıralamanın Özeti

Low list
(< 17)

High list
(> 17)

Kaynak

Recursively
apply quicksort to low list

Kaynak

Recursively
apply quicksort to high list

Kaynak

Sorted list of values

Kaynak

Sıralı Hızlı Sıralama Örneği

- Bir sayı listesi verildiğinde:
  - {79, 17, 14, 65, 89, 4, 95, 22, 63, 11}
- Pivot olarak ilk sayı olan 79 'u seçin
  - Düşük liste {17, 14, 65, 4, 22, 63, 11} içerir
  - En üstteki listede {89, 95}

Kaynak

Sıralı Hızlı Sıralama Örneği

- Alt liste {17, 14, 65, 4, 22, 63, 11} için pivot olarak 17 'yi seçin
  - Düşük liste {14, 4, 11} içerir
  - Yüksek liste {65, 22, 63}
- Alt liste {14, 4, 11} için pivot olarak 14 'ü seçin
  - Düşük liste {4, 11} içerir
  - Yüksek liste boş (daha fazla özyinelemeye gerek yok)

Kaynak

Sıralı Hızlı Sıralama Örneği

- Alt liste {4, 11} için pivot olarak 4 'ü seçin
  - Düşük liste boş (daha fazla özyinelemeye gerek yok)
  - Yüksek liste {11} içerir (daha fazla özyinelemeye gerek yoktur)
- Alt liste {65, 22, 63} için pivot olarak 65 'i seçin
  - Düşük listede {22, 63}
  - Yüksek liste boş (daha fazla özyinelemeye gerek yok)

Kaynak

Sıralı Hızlı Sıralama Örneği

- Alt liste {22, 63} için pivot olarak 22 'yi seçin
  - Düşük liste boş (daha fazla özyinelemeye gerek yok)
  - Yüksek liste {63} içerir (daha fazla özyinelemeye gerek yoktur)
- {4, 11, 14, 17, 22, 63, 65} alt listenin sıralanmış sonucudur
  {17, 14, 65, 4, 22, 63, 11}

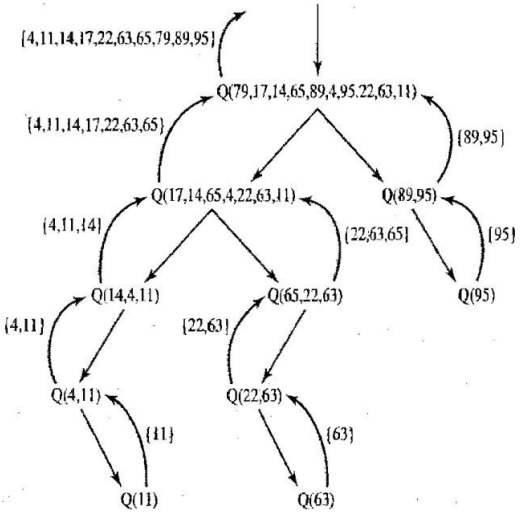
Kaynak

Sıralı Hızlı Sıralama Örneği

- Alt liste {89, 95} için pivot olarak 89 'u seçin
  - Düşük liste boş (daha fazla özyinelemeye gerek yok)
  - Yüksek liste {95} içerir (daha fazla özyinelemeye gerek yoktur)
- {89, 95} alt listenin sıralanmış sonucudur {89, 95}
- Nihai sıralama sonucu :{ 4, 11, 14, 17, 22, 63, 65, 79, 89, 95}

Kaynak

Sıralı Hızlı Tasnifin İllüstrasyonu



Kaynak : M. Quinn, *MPI ve OpenMP ile C'de Paralel Programlama*. McGraw Hill, 2004.

Sıralı Hızlı Sıralama için Gözlemler

- Hızlı sıralama, ortalama durumda anahtarların karşılaştırılmasına dayalı olarak genellikle en hızlı sıralama algoritması olarak kabul edilir

- Quicksort'un bazı doğal eşzamanlılıkları var
  - Düşük liste ve yüksek liste aynı anda kendilerini sıralayabilir

Kaynak

Sıralı Hızlı Sıralama Öznitelikleri

- N veri değerleri için zaman karmaşıklığı aşağıdaki gibidir
  - En iyi/Ortalama durum : O(N log N)
  - En kötü durum : $O(N^2)$
    - Düşük ve yüksek listeler her bölümleme adımında maksimum dengesiz olduğunda oluşur
    - Pivot değeri seçmek için örnekleme, en kötü durumu daha az olası hale getirebilir

Kaynak