به نام یکتای هستی بخش

دانشگاه تهران، دانشکده‌ی مهندسی برق و کامپیوتر

ساختمان داده‌ها و الگوریتم‌ها، نیمسال اول، سال تحصیلی ۹۴-۹۳

تمرین شماره ۴ مهلت تحویل: ، ساعت ۲۳:۵۵

الگوریتم‌هایی که برای سوالات زیر ارائه می‌دهید باید از بهترین مرتبه‌ی زمانی باشد.

۱.داده‌های زیر را به روش‌های merge sort، heap sort، quick sort و insertion sort مرتب کنید و مراحل آن‌ها را به صورت گام به گام و مختصر توضیح دهید. در مورد stable بودن الگوریتم‌ها نیز بحث کنید.

۴ ۱۲ ۴ ۱ ۲ ۳ ۱۲ ۹ ۲ ۱

برای مشاهده روش مرتب سازی می‌توانید به سایت http://visualgo.net/sorting.html مراجعه کنید.

پایداری الگوریتم‌ها:

Merge sort	Heap sort	Quick sort	Insertion sort
بلی	**خیر**	**خیر (در حالت عادی)**	**بلی**

۲.توضیح دهید در چه حالتی insertion sort بهتر از merge sort عمل خواهد کرد.

در حالتی که داده‌ها همه از پیش مرتب باشند. هزینه‌ی insertion sort برابر O(n) خواهد ولی هزینه‌ی merge sort تغییری نخواهد کرد.

۳.تعداد n عدد در آرایه ای داریم. می‌خواهیم با کمترین هزینه تمامی اعدادی که قبل از عددی کوچکتر از خودشان قرار دارند را محاسبه کنیم. الگوریتمی مبتنی بر merge sort برای این کار ارائه دهید. سعی کنید الگوریتم جواب را به صورت inplace[1] ارائه دهید.

مانند merge sort اعداد را از وسط به دو دسته تقسیم می کنیم. حال می دانیم که تعداد عناصری که جلوتر از عددی کوچکتر از خودشان آمده‌اند برابر اعدادی است که در دسته راست جلوتر از عددی کوچکتر از خودشان آمده‌اند به علاوه اعدادی که در دسته چپ این گونه اند. به علاوه اعدادی که در دسته سمت چپ از عددی که در دسته سمت راست است کوچک تر است. حال برای دسته راست و چپ به صورت بازگشتی این کار را انجام می دهیم. و برای ادغام آن‌ها

[1]- الگوریتم inplace الگوریتمی است که در آن به حافظه‌ی اضافی زیادی نیاز نیست. به عبارتی الگوریتمی که مرتبه‌ی حافظه‌ی آن O(1) باشد، inplace است.

نیز مانند merge sort عمل می‌کنیم با این تفاوت که هر عددی که از دسته راست می‌آید به تعداد اعداد باقی مانده در دسته سمت چپ به تعداد نابجایی‌ها اضافه می کنیم.

۴.در یک آرایه به طول n عددی وجود دارد که بیشتر از n/2 ام بار در این آرایه تکرار شده است. الگوریتمی ارائه دهید که این عدد را پیدا کند. الگوریتم شما باید از مرتبه‌ی زمانی O(n) و مرتبه‌ی حافظه O(1) باشد.

روی اعداد الگوریتم quick sort را انجام می دهیم. با این تفاوت که هر دفعه دسته‌ی کوچکتر را دور می ریزیم و الگوریتم را روی دسته‌ی بزرگتر انجام می‌دهیم. تا جایی که تمام عناصر موجود در آرایه یکی شوند. این عنصر، عنصر مورد نظر است.

۵.تعداد n عنصر در k لیست پیوندی داریم. که هر لیست به صورت صعودی مرتب شده است و اولین عنصر آن کوچک‌ترین عنصر است. الگوریتمی ارائه دهید که با بهترین مرتبه بتواند یک لیست پیوندی صعودی از عناصر این k دسته بسازد.

یک heap از لیست‌های پیوندی درست می‌کنیم. که ملاک مقایسه در آن عنصر سر لیست است. حال عنصر سر لیست پیوندی‌ای که سر heap است را بر می‌داریم و بر اساس عنصر اولی جدید downheap می کنیم. هر وقت لیستی خالی شد آن را حذف می کنیم. هزینه این الگوریتم برابر n بار downheap کردن است که برابر می شود با $n \log k$.

۶.اثبات کنید مرتبه ی تمامی الگوریتم‌های مرتب سازی مبتنی بر مقایسه در حوزه‌ی اعداد حقیقی حداقل O(nlogn) است.

کلیه حالاتی که یک دنباله از اعداد می‌توانند به خود بگیرند برابر است با !n حالت. با هر مقایسه‌ای که بین دو عدد انجام دهیم می‌توانیم این حالات را نصف کنیم. (دو حالت مرتب و غیر مرتب برای دو عدد وجود دارد.) پس با انجام $\log n!$ مقایسه می توان به یک حالت رسید. طبق قواعد ریاضی داریم $\log n! = \theta(n \log n)$.

۷.دنباله ای به طول n که شامل تمامی اعداد ۱ تا n است را با قطعه کد زیر مرتب می‌کنیم. (کد مربوط به merge sort است.)

Floorبه معنای کف و علامت .. به معنای بازه است.

```
function merge_sort(arr):
    n = arr.length()
```

```
    if n <= 1:
        return arr

    // arr is indexed 0 through n-1, inclusive
    mid = floor(n/2)

    first_half = merge_sort(arr[0..mid-1])
    second_half = merge_sort(arr[mid..n-1])
    return merge(first_half, second_half)

function merge(arr1, arr2):
    result = []
    while arr1.length() > 0 and arr2.length() > 0:
        if arr1[0] < arr2[0]:
            print '1' // for debugging
            result.append(arr1[0])
            arr1.remove_first()
        else:
            print '2' // for debugging
            result.append(arr2[0])
            arr2.remove_first()

    result.append(arr1)
    result.append(arr2)
    return result
```

الگوریتمی ارائه دهید که با گرفتن n و دنباله‌ی اعداد چاپ شده در الگوریتم (در قسمت مقایسه دو دستور print وجود دارد.) دنباله‌ی اصلی را تولید نماید.

به عنوان مثال برای دنباله‌ای به طول ۲ و دنباله‌ی چاپی ۱ باید لیست زیر تولید شود

[1, 2]

یا برای دنباله‌ای به طول ۴ و دنباله‌ی چاپی ۱۲۲۱۲ باید لیست زیر تولید شود.

[2, 4, 3, 1]

مرتبه‌ی الگوریتم شما باید از O(nlogn) باشد.

جواب سوال را می‌توانید در لینک زیر مشاهده کنید.

http://united-coders.com/christian-harms/facebook-hackercup-2012-round-1-with-merge-sort/