

**11/18/2020**

ALGORITHM DESIGN



Homework 3

Sort & Ch9



1. نشان دهید زمان اجرای الگوریتم مرتب سازی سریع هنگامی که آرایه ورودی شامل اجزای متمایزی باشد که بصورت نزولی مرتب شده­اند، برابر با خواهد بود.

|  |
| --- |
| میدانیم که کد quick sort بصورت زیر است:  quickSort(arr[], p, r)  if (p < r)  q = partition(arr, p, r  quickSort(arr, p, q - 1)  quickSort(arr, q + 1, r)  و برای تابع partition داریم:  partition (arr[], p, r)  pivot = arr[r]  i = (p - 1)  for (j = p; j <= r- 1; j++)  if (arr[j] < pivot)  i++  swap arr[i] and arr[j]  swap arr[i + 1] and arr[r])  return (i + 1)  اگر آرایه از همان ابتدا sort شده باشد (چه بصورت صعودی و چه نزولی) در قسمت partition ، عنصر pivot همواره در سر یا ته قرار خواهد گرفت. یعنی اگر آرایه ای n عنصری در اختیار داشته باشیم بعد از فراخوانی partition، در یک سمت pivot صفر عنصر و در سمت دیگر n-1 عنصر قرار میگیرد. یعنی:  پس در هر مرحله باید دوباره روی n-1 عنصر باقی مانده تابع partition که خود میباشد را فراخوانی کنیم. بنابراین میتوان به این نتیجه رسید که در چنین حالتی الگوریتم خواهد بود. |

1. توضیح دهید که چگونه می­توان عدد صحیح در بازه­ی تا را در زمان مرتب کرد.

|  |
| --- |
| میدانیم در الگوریتم radix sort ، زمان مصرف میشود. در این حالت برابر است با تعداد حالاتی که یک رقم میتواند داشته باشد. همچنین نیز تعداد ارقام اعدادی است که میخواهیم sort کنیم. با توجه به اینکه میخواهیم اعداد تا را بررسی کنیم، میتوان با این فرض که اعداد را در مبنای میبینیم گفت که قرار است اعدادی که مرتب میکنیم اعداد **دو رقمی در مبنای** هستند. (مثلا برای اگر اعداد ۰ تا ۹ را بنویسیم و آنها را به مبنای ۳ تبدیل کنیم به ترتیب برابر خواهند بود با **00** ، **01** ، **02** ، **10** ، **11** ، **12** ،**20**  ، **21** و **22**)  یعنی قرار است عدد رقمی که ارقام آن میتواند هستند را با radix sort مرتب کنیم. یعنی: |

1. در الگوریتم SELECT، ورودی ها در ابتدا به گروه­های 5 تایی تقسیم می­شوند. آیا درصورتی که ورودی­ها به دسته­های 7 تایی تقسیم شوند الگوریتم همچنان در زمان خطی کار می­کند؟

|  |
| --- |
| با تقسیم ورودی ها به ۷ قسمت،‌ گروه خواهیم داشت. حداقل نیمی از این گروه ها، حداقل ۴ عضو بزرگتر از میانه ی میانه های این ۷ قسمت دارند و همچنین میانه ی میانه ها حداقل از میانه های ۴ گروه از کوچکتر میباشد. میتوان به نوعی گفت که تقریبا میانه میانه ها از عناصر ورودی بزرگتر است و از عناصر ورودی کوچکتر میباشد. پس نمیتوان بیشتر از بار آن را بصورت بازگشتی فراخوانی کرد. بنابراین خواهیم داشت. حال باید نشان دهیم این رابطه در زمان خطی کار میکند. با فرض برای خواهیم داشت:  بنابراین این الگوریتم برای گروه های ۷ تایی نیز بصورت خطی کار میکند. |

1. فرض کنید می­خواهیم n عدد صحیح را که شامل log n عدد متمایز است مرتب کنیم. الگوریتمی با پیچیدگی زمانی (O(n log log n برای مرتب کردن مقایسه‌ای این اعداد طراحی کنید.

|  |
| --- |
| یگ binary search tree در نظر میگیریم که برای هر یک از گره های آن یک شمارنده وجود دارد. هنگام پیمایش آرایه برای هر یک از عناصر موجود یک بار روی این درخت عمل جستجو را انجام میدهیم.   * اگر گره مورد نظر پیدا شد، شمارنده آن را یکی افزایش میدهیم. * اگر گره در درخت موجود نبود آن را فرزندی برگی قرار میدهیم که به آن رسیده بودیم.(واضح است که شمارنده آن باید مقدار اولیه ۱ بگیرد)   قرار دادن مقادیر داخل درخت به این گونه است که گره متمایز در درخت خواهیم داشت. بنابراین ارتفاع درخت برابر خواهد بود با .  حال باید از روی این درخت ساخته شده، آرایه مرتب شده را تولید میکنیم. یعنی عناصر داخل درخت را از کوچک به بزرگ به تعداد تکرارشان (شمارنده متناظر) در آرایه با زمان جایگذاری میکنیم. |

1. فرض کنید که L = (x1, x2,...,xn) **فهرستی از nعدد** حقیقی باشد. **ترتیب اعداد مشخص نیست**. میگوییم عدد xدر فهرست L «**غالب**» است اگر **اکیدا بیشتر از 1/3 اعداد در فهرست L مساوی x باشند**. این ([لینک](https://www.geeksforgeeks.org/n3-repeated-number-array-o1-space)) الگوریتم قطعی مبتنی بر مقایسه زیر را برای پیدا کردن اعداد «غالب» در نظر بگیرید:

الف) درستی الگوریتم و کد آن را ثابت کنید. (می­توانید از loop invariantها کمک بگیرید)

ب) ثابت کنید که این الگوریتم از θ(n) مقایسه انجام می­دهد.

|  |
| --- |
| تکه کد این الگوریتم به شرح زیر است:  function(arr, n)  count1 = 0  count2 = 0  first = sys.maxsize  second = sys.maxsize  for (i = 0; 1 < n; i++)  if (first == arr[i]) count1 += 1  else if (second == arr[i]) count2 += 1  else if (count1 == 0)  count1 += 1  first = arr[i]  else if (count2 == 0):  count2 += 1  second = arr[i]  else  count1 -= 1  count2 -= 1  count1 = 0  count2 = 0  for (i = 0; 1 < n; i++)  if (arr[i] == first) count1 += 1  else if (arr[i] == second) count2 += 1  if (count1 > n / 3) return first  if (count2 > n / 3) return second  return -1  **الف)** میخواهیم اثبات کنیم که for اول، دو انتخاب مربوط به عنصر غالب را در صورت وجود آن درست انجام میدهد.   * **Loop Invariant:** در پیمایش آرایه هنگامی که در حال بررسی عنصر i هستیم دو عنصر نامزدی که بیشترین تکرار ها را در بازه arr[0] تا arr[i] دارند انتخاب شده اند. * **Initialization:** در ابتدا i برابر با ۰ است. در تکرار های ۰ و ۱ با توجه با مقادیر count در دو متغیر first و second برای اولین بار مقدار دهی میشوند. * **Maintenance:**  در هر مرحله دو کاندید و مقادیر تکرار هرکدام تا اینجا مشخص شده اند. با رفتن به پیمایش بعدی آرایه، یا مقادیر تکرار هر کاندید عوض میشوند یا حالتی وجود دارد که کاندید قبلی مقدار ۰ دارد و خود کاندید آپدیت میشود. * **Termination:** در آخرین مرحله تکرار که i با n-1 برابر است، دو نامزدی که در احتمال دارد عناصر غالب باشند مشخص شده اند و در ادامه میتوان چک کرد که آیا واقعا قالب هستند یا خیر.   **ب)** در کدی که در بالا آمده داریم:  function(arr, n)  ⫶  for (i = 0; 1 < n; i++)  ⫶  ⫶  for (i = 0; 1 < n; i++)  ⫶  در این کد که خلاصه شده ی کد قبل تر میباشد، میتوان گفت در کل ۲ حلقه for داریم که کل عناصر آرایه ی مورد نظر را طی میکنند و مقایسه هایی انجام میدهند. قسمت هایی که با علامت ⫶ به اختصار در آمده اند، اعمالی نظیر چک کردن شروط مخلتف (if…else…) ، مقدار دهی ها و همچنین return میباشند که هر کدام از آنها دارای هستند. بنابراین محتوای داخل حلقه های for نیز مقایسه دارند. پس میتوان گفت هر یک از حلقه ها داری مقایسه هستند. از طرفی میدانیم مقایسه های بعد از for دوم نیز همگی هستند. با توجه به تمام این موضوعات میتوان ادعا کرد: |

* مهلت ارسال تمرین ساعت 23:55 روز **جمعه 7** آذر­ماهمی­باشد.
* سوالات خود را می­توانید از طریق ایمیل از تدریسیاران بپرسید.
  + - * + parsa.abdollahi.pa@gmail.com
        + faridi.mina.1@gmail.com
* ارائه پاسخ تمرین به دو روش ممکن است:
  + 1. تایپ داخل همین فایل و ارائه فایل Pdf
    2. چاپ تمرین و پاسخ دهی به صورت دستنویس خوانا
* ارائه تمرین به روش اول شامل 10% نمره امتیازی می­گردد.
* فایل پاسخ تمرین را تنها با قالب **HW3-9531747.pdf** در مودل بارگزاری کنید.
* فایل زیپ ارسال **نکنید**.