بسمه تعالی

تمرین سری اول ساختمان داده ها و مبانی الگوریتم ها

نکات :
- پاسخ تمرین ها را تایپ شده یا اسکن شده در قالب pdf با نام HW1_StudentNumber.pdf ارسال نمایید.
- مهلت ارسال این سری تمرین تا شنبه ۱۴ مهر ساعت ۲۳:۵۵ است.

۱- توابع زیر را برحسب رشد مجانبی مرتب کنید. برابری ها را نیز ذکر کنید:

$\ln n, \log n, \sqrt{n}, n, n\log n, n^2, 2^n, n^{\frac{1}{n}}, n^{1/\log n}, \log_n 1000, 2^{\sqrt{\log n}}, \sqrt{2}^{\log n}, \log^{\sqrt{2}} n, n^{\sqrt{2}}, F_n, F_{\frac{n}{2}}, \log F_n, F_{\log n}$

تابع F(n) همان جمله nام دنباله فیبوناچی می باشد.

۲- تعداد جابجایی عناصر در مرتب سازی درجی در بهترین حالت ، بدترین حالت و میانگین چه تعداد است ؟

۳-صحت مرتب سازی حبابی

مرتب سازی حبابی (Bubble sort) یکی از روش های شناخته شده ی مرتب سازی است. مرتب سازی حبابی با جا به جایی عناصر مجاور نامرتب به صورت مکرر عمل مرتب سازی را انجام می دهد. الگوریتم مرتب سازی حبابی به صورت زیر می باشد.

```
Bubble_Sort ( A )
۱ For i ← ۱ to length [ A ]
۲ Do for j ← length [ A ] downto i + ۱
۳ Do if A [ j ] < A [ j - ۱ ]
۴ Then exchange A [ j ] ↔ A [ j - ۱ ]
```

الف) در این سوال درستی الگوریتم مرتب سازی حبابی را اثبات خواهیم کرد. برای اثبات درستی الگوریتم مرتب سازی حبابی ابتدا باید اثبات کرد که الگوریتم پایان می پذیرد. سپس باید اثبات کرد که رابطه ی $A[۱] \le A[۲] \le \cdots \le A[n]$ پس از اجرای الگوریتم برقرار است (در این رابطه n برابر اندازه ی آرایه A می باشد) . به نظر شما چه ویژگی دیگری باید اثبات گردد تا درستی الگوریتم مرتب سازی حبابی اثبات گردد.

ب) برای حلقه موجود در الگوریتم ثابت حلقه (Loop invariant) را به طور دقیق مشخص کنید و اثبات کنید که حلقه پایان پذیر است. برای مطالعه بیشتر در ارتباط با روش اثبات می توانید به کتاب مرجع درس مراجعه کنید.

ج) حال با توجه به ثابت حلقه ی تایین شده در قسمت قبل اثبات کنید که پس از پایان الگوریتم نامساوی $A[۱] \le A[۲] \le \cdots \le A[n]$ برقرار خواهد بود. برای مطالعه بیشتر در ارتباط با روش اثبات می توانید به کتاب مرجع درس مراجعه کنید.

د) پیچیدگی زمانی اجرای الگوریتم مرتب سازی حبابی در بدترین حالت چیست؟ الگوریتم مرتب سازی حبابی را با الگوریتم مرتب سازی درجی (Insertion sort)از نظر زمانی مقایسه کنید.

۴- شبه کد زیر که مربوط به الگوریتم odd-even sort است . شبیه به مرتب سازی حبابی می باشد.

```
function oddEvenSort(list) {
 function swap( list, i, j ){
  var temp = list[i];
  list[i] = list[j];
  list[j] = temp;
 }

 var sorted = false;
 while(!sorted)
 {
  sorted = true;
  for(var i = 1; i < list.length-1; i += 2)
  {
   if(list[i] > list[i+1])
   {
    swap(list, i, i+1);
    sorted = false;
   }
  }

  for(var i = 0; i < list.length-1; i += 2)
  {
   if(list[i] > list[i+1])
   {
    swap(list, i, i+1);
    sorted = false;
   }
  }
 }
}
```

الف) مثالی پیدا کنید که این الگوریتم بهتر از مرتب سازی حبابی عمل کند؟
ب) آیا مثالی وجود دارد که مرتب سازی‌حبابی بهتر از این مرتب سازی عمل کند؟