به نام خدا

|  |  |  |  |
| --- | --- | --- | --- |
| مدرس: مظفر بگ محمدي | دانشگاه ايلام | ترم اول سال تحصيلي 87-86 | سري چهارم تمرينات ساختمان داده |

1. صف: چگونه مي توان با استفاده از متدهاي استاندارد صف و بدون استفاده از پشته، ترتيب عناصر يک صف را معکوس کرد. براي حل مساله مي توانيد از چندين صف استفاده کنيد.
2. پشته و صف: توضيح دهيد چگونه مي توان عناصر يک پشته را وارد يک پشته ديگر نمود به نحوي که ترتيب عناصر پشته دوم و اول يکسان باشند. مي توانيد از يک صف کمکي براي حل مساله استفاده کنيد.
3. پشته: در مسئله برجهای هانوی، n حلقه که دارای شعاعهای 1 تا n هستند به ترتیب نزولی روی یک میله قرا دارند. دو میله خالی نیز وجود دارند. هدف مسئله انتقال حلقه ها به میله سوم است به طوری که ترتیب حلقه ها تغییر نکند. در ضمن در هر حرکت مجاز به انتقال یک حلقه به میله های دیگر هستید به طوری که هیچگاه یک حلقه بزرگتر روی یک حلقه کوچکتر قرار نگیرد. فرض کنید که شما مجاز به استفاده از یک پشته آرایه ای با اندازه محدود (m) هستید. در این حالت می توانید از هر کدام از میله ها یک حلقه به پشته اضافه کنید و برعکس. توضیح دهید که چگونه می توان با استفاده از این پشته مسئله برجهای هانوی را برای n های بزرگتر از m حل کرد.
4. پشته: توضیح دهید که چگونه با پشته می توان بزرگترین مقسوم علیه مشترک دو عدد دلخواه را پیدا کرد.
5. پشته وصف: توضيح دهيد که چگونه مي توان توسط دو پشته، يک صف درست کرد.
6. مرتب سازی: آرایه ی (7,8,2,5,9,11,1) را با استفاده از روشهای مرتب سازی درجی و ادغام مرتب کنید. پس از انجام هر مرحله آرایه ی حاصل را نشان دهید.
7. مرتب سازی: آرایه ی فوق را با استفاده از مرتب سازی سریع مرتب کنید. برای اینکار از کد زیر استفاده کنید. تمام مراحل را نشان دهید. دقت کنید که کد partition با آنچه سر کلاس گفتیم متفاوت است. برای فهمیدن کد به لینک <http://www.algolist.net/Algorithms/Sorting/Quicksort> مراجعه کنید.

**int** partition(**int** arr[], **int** left, **int** right)

{

      **int** i = left, j = right; **int** tmp;

      **int** pivot = arr[(left + right) / 2];

      **while** (i <= j) {

            **while** (arr[i] < pivot)

                  i++;

            **while** (arr[j] > pivot)

                  j--;

            **if** (i <= j) {

                  tmp = arr[i];

                  arr[i] = arr[j];

                  arr[j] = tmp;

                  i++;

                  j--;

            }

      };

      **return** i;

}

**void** quickSort(**int** arr[], **int** left, **int** right) {

      **int** index = partition(arr, left, right);

      **if** (left < index - 1)

            quickSort(arr, left, index - 1);

      **if** (index < right)

            quickSort(arr, index, right);

}