자료구조론 CC343_2207

Reading assignment 13

경기대학교 컴퓨터공학부

201511837 이상민

Review Questions

1. Which technique of searching an element in an array would you prefer to use and in which situation?

① Sequential Search: In this, the list or array is traversed sequentially and every element is checked. For example: 선형 search

② Interval Search: These algorithms are specifically designed for searching in sorted data-structures. These type of searching algorithms are much more efficient than Linear Search as they repeatedly target the center of the search structure and divide the search space in half. For Example: 바이너리 search

2. Define sorting. What is the importance of sorting?

데이터를 기반으로 데이터를 추출할 때마다 정렬이 중요해진다.

만약에 대량의 데이터의 경우 정렬 목록이 필요할 때마다 정렬을 수행하는 대신 적절한 데이터 구조에 투자하여 순서가 지정된 각 속성에 인덱스를 생성하고 저장하게 될 경우 특정 데이터를 찾는 시간도가 줄게됩니다.

3. What are the different types of sorting techniques? Which sorting technique has the least worst case?

비교 기반 분류

배열 요소를 비교하여 정렬된 배열을 찾기 위해 배열 요소를 서로 비교합니다.

버블 종류 및 삽입 종류

평균 및 최악의 경우 시간 복잡도: n^2

최상의 사례 시간 복잡성: 어레이가 이미 정렬된 경우.

최악의 경우: 어레이가 역 방향으로 정렬된 경우.

- ✓ 선택 정렬: 고, 평균 및 최악의 경우 시간 복잡도: 데이터 분포와 독립적인 n^2.
- ✓ 병합 정렬: 적, 평균 및 최악의 경우 시간 복잡도: 데이터 분포와 독립적인 nlogn.
- ✓ 힙 정렬: 적, 평균 및 최악의 경우 시간 복잡도: 데이터 분포와 독립적인 nlogn.
- ✓ 빠른 정렬: 관계가 반복되는 분열과 정복의 접근이다.

비경쟁적 분류: 비 비교 기반 정렬에서는 배열 요소를 서로 비교하여 정렬된 배열을 찾지 않습니다.

- ✓ 기수 정렬: 최적, 평균 및 최악의 경우 시간 복잡도: 여기서 k는 배열 요소의 최대 자릿수입니다.
- ✓ 개수 정렬: 최적, 평균 및 최악의 경우 시간 복잡도: n+k. 여기서 k는 카운트 배열의 크기입니다.
- ✓ 버킷 정렬: 최적 및 평균 시간 복잡도: n+k. 여기서 k는 버킷 수입니다. 최악의 경우 시간 복잡도: 모든 요소가 동일한 버킷에 속하는 경우 n^2.

4. Explain the difference between bubble sort and quick sort. Which one is more efficient?

빠른 정렬은 빠른 정렬의 버블정렬보다 더 좋은 방법입니다. 우리는 피벗 요소를 선택하여 지정된 세트를 두세트로 나눈 다음 각 세트를 정렬하고 마지막으로 병합하는 것보다 더 좋은 방법입니다. 버블 종류에서는 첫번째 요소를 선택하고 지정된 요소를 사용하여 세트의 다음 요소를 확인하여 세트를 정렬합니다. 그러나 빠른 정렬의 최악의 경우 시간 복잡도는 O(logn)이고 버블은 O(n^2)입니다.

5. Sort the elements 77, 49, 25, 12, 9, 33, 56, 81 using

(a) insertion sort: 9 12 25 33 49 56 77 81

(b) selection sort: 9 12 25 33 49 56 77 81

(c) bubble sort: 9 12 25 33 49 56 77 81

(d) merge sort: 9 12 25 33 49 56 77 81

(e) quick sort: 9 12 25 33 49 56 77 81

(f) radix sort: 9 12 25 33 49 56 77 81

(g) shell sort: 9 12 25 33 49 56 77 81

6. Compare heap sort and quick sort.

quicksort의 가장 직접적인 경쟁자는 Heapsort이다. Heapsort는 일반적으로 quicksort보다 다소 느리지만, 최악의 실행 시간은 항상 O(nlogn)입니다. Quicksort는 일반적으로 더 빠르지만, 잘못된 사례가 감지될 때 Heapsort로 전환되는 introsort변형을 제외하고는 최악의 경우 성능이 남아 있습니다. 만약 Heapsort가 필요할 것이라고 미리 알고 있다면, introsort가 그것으로 바뀌기를 기다리는 것보다 그것을 직접 사용하는 것이 더 빠를 것이다.

7. Quick sort shows quadratic behaviour in certain situations. Justify.

Example : 1 2 3 4 5

선택된 피벗은 1, 피벗의 오른쪽에는 4개의 요소가 있고 왼쪽에는 요소가 없습니다. 동일한 논리를 반복 적용하고 선택한 피벗을 각각 2,3,4,5로 적용하면, 우리는 이러한 종류의 작업이 가능한 최악의 시기에 수행되는 상황에 도달했습니다.

입력이 잘 배치된 경우 Quicksort의 성능이 우수하다는 것이 권장되고 입증되었습니다.

더욱이, 종류의 선택은 보통 입력 영역에 대한 명확한 지식에 달려 있다. 예를 들어, 만약 입력이 크다면, 외부 메모리를 사용할 수 있는 외부 정렬이라고 불리는 것이 있습니다. 입력 크기가 매우 작으면 병합 종류를 선택할 수 있지만 추가 메모리를 사용하기 때문에 중간 및 큰 입력 세트는 선택할 수 없습니다. Quick정렬의 주요 장점은 "in-place"의미이며, 추가 메모리가 입력 데이터에 사용되지 않는다는 것입니다. 서류상 최악의 경우 시간은 O(n^2)이지만 여전히 널리 선호되고 사용되고 있다. 이로 인해 입력 세트에 대한 지식과 선호도에 따라 분류 알고리즘이 바뀔 수 있다는 것을 알 수 있다.

8. If the following sequence of numbers is to be sorted using quick sort, then show the iterations of the sorting process.

42, 34, 75, 23, 21, 18, 90, 67, 78

Step1: 42(Pivot), 34(low), 75, 23, 21, 18, 90, 67, 78(high)

Step2: 42(Pivot), 34, 75(low), 23, 21, 18, 90, 67(high), 78

Step3: 42(Pivot), 34, 67, 23(low), 21, 18, 90(high), 75, 78

Step4: 42(Pivot), 34, 67, 23, 21(low), 18(high), 90, 75, 78

Step5: 42(Pivot), 34, 67, 23, 18, 21, 90, 75, 78

Step6: 18, 34, 67, 23, 42(Pivot), 21, 90, 75, 78

42(Pivot)을 기준으로 양 옆으로 나누어서 각자의 Quick sort를 진행한다.

Step7: 18, 34, 67, 23, 42, 21, 75, 90, 78

Step8: 18, 34, 67, 23, 42, 21, 75, 90, 78

Step9: 18, 34, 23, 42, 21, 67, 75, 78, 90

Step10: 18, 23, 34, 21, 42, 67, 75, 78, 90

Step11: 18, 23, 21, 34, 42, 67, 75, 78, 90

Step12: 18, 21, 23, 34, 42, 67, 75, 78, 90

9. Sort the following sequence of numbers in descending order using heap sort.

42, 34, 75, 23, 21, 18, 90, 67, 78

answer: 18, 21, 23, 34, 42, 67, 75, 78, 90

10. A certain sorting technique was applied to the following data set,

45, 1, 27, 36, 54, 90

After two passes, the rearrangement of the data set is given as below:

1, 27, 45, 36, 54, 90

Identify the sorting algorithm that was applied.

answer: quick sort

11. A certain sorting technique was applied to the following data set,

81, 72, 63, 45, 27, 36

After two passes, the rearrangement of the data set is given as below:

27, 36, 81, 72, 63, 45

Identify the sorting algorithm that was applied.

answer: shell sort

12. A certain sorting technique was applied to the following data set,

45, 1, 63, 36, 54, 90

After two passes, the rearrangement of the data set is given as below:

1, 45, 63, 36, 54, 90

Identify the sorting algorithm that was applied.

answer: bubble sort

13. Write a recursive function to perform selection sort.

```
#include <stdio.h>
void selection(int [], int, int, int, int);
int main()
{
    int list[30], size, temp, i, j;
    printf("Enter the size of the list: ");
    scanf("%d", &size);
    printf("Enter the elements in list:₩n");
    for (i = 0; i < size; i++)
    {
        scanf("%d", &list[i]);
    }
    selection(list, 0, 0, size, 1);
    printf("The sorted list in ascending order is₩n");
    for (i = 0; i < size; i++)
    {
        printf("%d  ", list[i]);
    }
    return 0;
}
```

```
void selection(int list[], int i, int j, int size, int flag)
{
    int temp;

    if (i < size - 1)
    {
        if (flag)
        {
            j = i + 1;
        }
        if (j < size)
        {
            if (list[i] > list[j])
            {
                temp = list[i];
                list[i] = list[j];
                list[j] = temp;
            }
            selection(list, i, j + 1, size, 0);
        }
        selection(list, i + 1, 0, size, 1);
    }
}
```

14. Compare the running time complexity of different sorting algorithms.

Name	Best	Avg	Worst
삽입정렬	n	n^2	n^2
선택정렬	n^2	n^2	n^2
버블정렬	n^2	n^2	n^2
셸 정렬	n	$n^{1.5}$	n^2
퀵 정렬	$n\log_2 n$	$n\log_2 n$	n^2
힙 정렬	$n\log_2 n$	$n\log_2 n$	$n\log_2 n$
병합정렬	$n\log_2 n$	$n\log_2 n$	$n\log_2 n$

15. Discuss the advantages of insertion sort.

The insertion sorts repeatedly scans the list of items, each time inserting the item in the unordered sequence into its correct position.

The main advantage of the insertion sort is its simplicity. It also exhibits a good performance when dealing with a small list. The insertion sort is an in-place sorting algorithm so the space requirement is minimal. The disadvantage of the insertion sort is that it does not perform as well as other, better sorting algorithms. With n-squared steps required for every n element to be sorted, the insertion sort does not deal well with a huge list. Therefore, the insertion sort is particularly useful only when sorting a list of few items.

Multiple-choice Questions

1. (b)
2. (d)
3. (c)
4. (d)
5. (b)
6. (a)
7. (d)
8. (a)
9. (c)
10. (d)
11. (d)

True or False

1. False
2. False
3. False
4. False
5. True
6. False
7. True
8. False
9. True
10. True
11. True

Fill in the Blanks

1. Sorted array
2. $O(n)$
3. The process of arranging values in a predetermined order.
4. Merge sort / heap sort/ quick sort
5. External sorting
6. Bubble sort
7. $O(n^2)$
8. Merge sort/ quick sort
9. $O(n \log n)$
10. $O(n.k)$
11. $O(n \log n)$
12. Pivot element