자료구조 및 알고리즘

정렬 알고리즘 구현 및 성능 분석

과 목 명	자료구조및알고리즘
담당교수	최 상 원 교수님
제 출 자	202015402/정민준 (전자공학과)
제 출 일	2023/12/14(목) 저녁7시

과제 내용

- **Size가 NX1인 X에 대해서, X의 원소들이 가장 작은 값에서 가장 큰 값으로 순서대로 정렬된 NX1의 Y를 구하는 함수를 구현하고, 이를 main.m에서 함수로 호출하여 결과를 출력하도록 구현하시오.**
  - main.m
  - Function 구현:
    - 퀵정렬: quickSort(X,N).m
    - 삽입정렬: insertSort(X,N).m
    - 셸정렬: shellSort(X,N).m
    - 기수정렬: radixSort(X,N).m
    - 제안하는 정렬: proposedSort(X,N).m

- **과제 채점 비중: 기존 알고리즘 구현 완성도 (70%), 제안하는 정렬 알고리즘 구현 완성도 (30%)**
  - 단, 제안하는 정렬 알고리즘은 기존 알고리즘보다 성능이 좋아야 인정됨.

- **제출 방법: 본 PPT 파일과 MATLAB source file을 하나로 압축하여 제출**

정렬 알고리듬 구현 현황

- 본 과제 수행을 통해 구현한 정렬 알고리즘 현황을 표시

정렬 알고리즘	핵심 코드 부분	구현 여부 (O, X로 표시)
퀵정렬	알고리즘 핵심 코드#1 (캡쳐본) 알고리즘 핵심 코드#2 (캡쳐본)	O

알고리즘 핵심 코드#1 (캡쳐본)

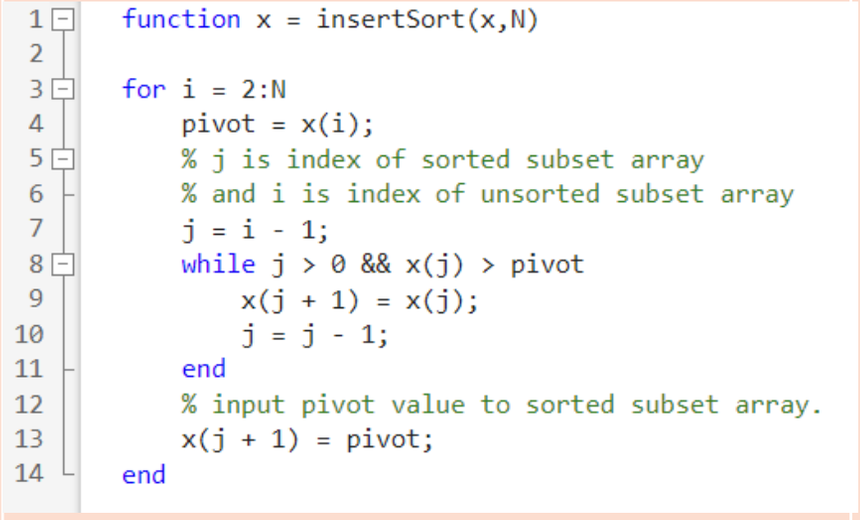
```
function [x] = quickSort(x,b,e)      % b= begin, e = end
if b<e
    % devide and Conquer
    [x1,p]=partition(x,b,e);         % devide ( return array and pivot )
    x2=quickSort(x1,b,p-1);          % recursive structure
    x3=quickSort(x2,p+1,e);
    x=x3;
end
end

```

알고리즘 핵심 코드#2 (캡쳐본)

```
function [a,p] = partition(a,L,R)
low = L;
high = R;
%set pivot as L (fisrt index in input array)
pivot = a(L);

%loop until low index cross high index
while low<high
    while low<=R&&a(low)<=pivot
        low=low+1;
    end
    while high>=L&&a(high)>pivot
        high = high-1;
    end
    if low<high
        swap = a(low);
        a(low) = a(high);
        a(high) = swap;
    end
end
% after low index cross high index
swap = a(L);
a(L) = a(high);
a(high) = swap;
%return pivot index
p = high;
end
```

정렬 알고리듬 구현 현황

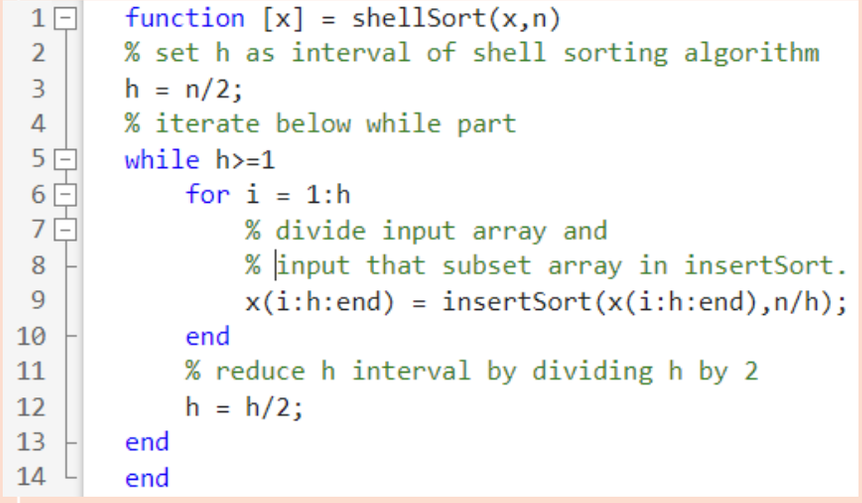
- 본 과제 수행을 통해 구현한 정렬 알고리즘 현황을 표시

정렬 알고리즘	핵심 코드 부분	구현 여부 (O, X로 표시)
삽입정렬		O

```
function x = insertSort(x,N)

for i = 2:N
    pivot = x(i);
    % j is index of sorted subset array
    % and i is index of unsorted subset array
    j = i - 1;
    while j > 0 && x(j) > pivot
        x(j + 1) = x(j);
        j = j - 1;
    end
    % input pivot value to sorted subset array.
    x(j + 1) = pivot;
end
```

정렬 알고리듬 구현 현황

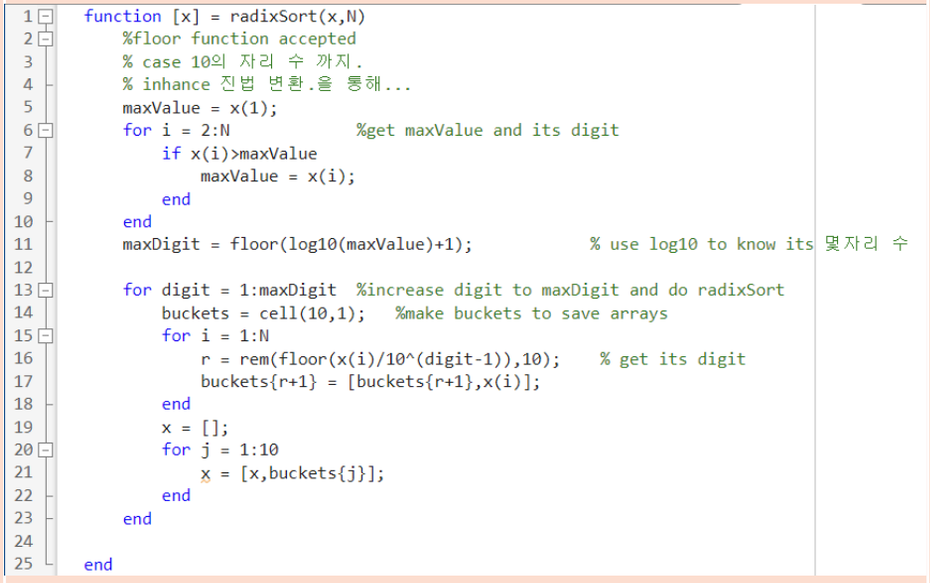
- 본 과제 수행을 통해 구현한 정렬 알고리즘 현황을 표시

정렬 알고리즘	핵심 코드 부분	구현 여부 (O, X로 표시)
셸정렬		O

```
function [x] = shellSort(x,n)
% set h as interval of shell sorting algorithm
h = n/2;
% iterate below while part
while h>=1
    for i = 1:h
        % divide input array and
        % input that subset array in insertSort.
        x(i:h:end) = insertSort(x(i:h:end),n/h);
    end
    % reduce h interval by dividing h by 2
    h = h/2;
end
end
```

정렬 알고리듬 구현 현황

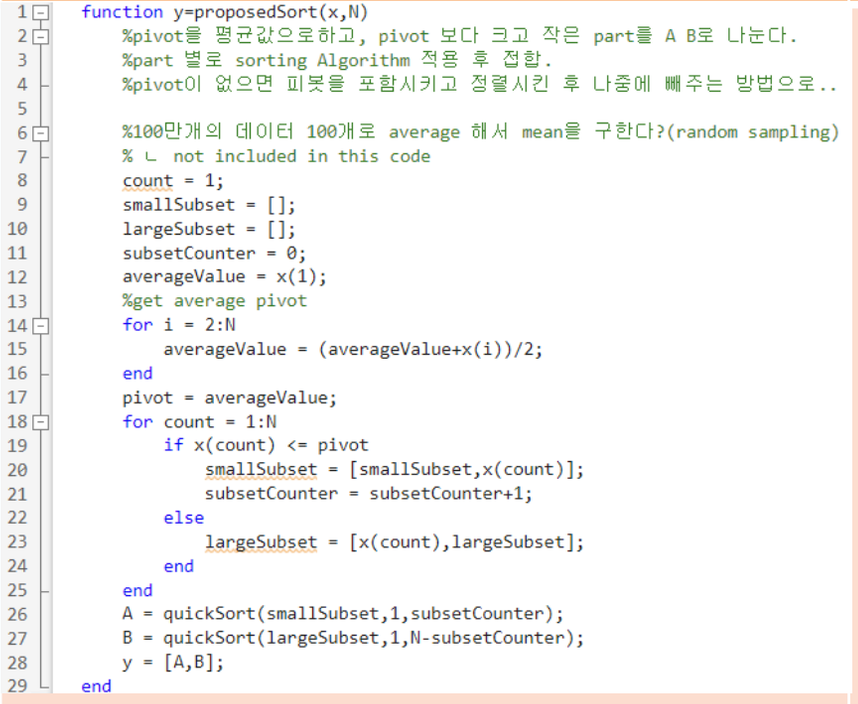
- 본 과제 수행을 통해 구현한 정렬 알고리즘 현황을 표시

정렬 알고리즘	핵심 코드 부분	구현 여부 (O, X로 표시)
기수 정렬		O

```
function [x] = radixSort(x,N)
    %floor function accepted
    % case 10의 자리 수 까지.
    % inhance 진법 변환.을 통해...
    maxValue = x(1);
    for i = 2:N             %get maxValue and its digit
        if x(i)>maxValue
            maxValue = x(i);
        end
    end
    maxDigit = floor(log10(maxValue)+1);           % use log10 to know its 몇자리 수

    for digit = 1:maxDigit  %increase digit to maxDigit and do radixSort
        buckets = cell(10,1);   %make buckets to save arrays
        for i = 1:N
            r = rem(floor(x(i)/10^(digit-1)),10);    % get its digit
            buckets{r+1} = [buckets{r+1},x(i)];
        end
        x = [];
        for j = 1:10
            x = [x,buckets{j}];
        end
    end

end
```

정렬 알고리듬 구현 현황

본 과제 수행을 통해 구현한 정렬 알고리즘 현황을 표시

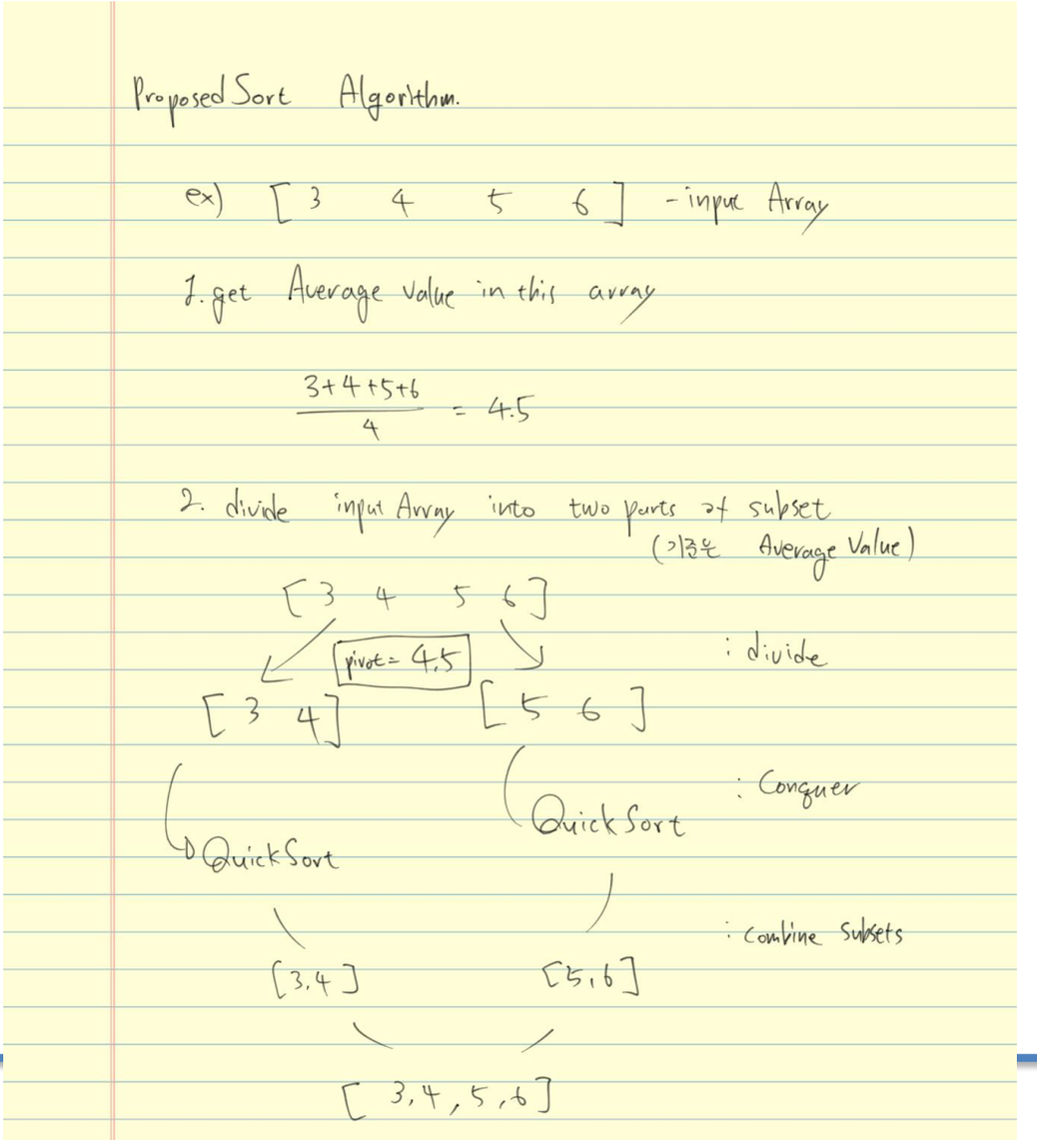
정렬 알고리즘	핵심 코드 부분	구현 여부 (O, X로 표시)
제안하는 정렬		O

```
function y=proposedSort(x,N)
    %pivot을 평균값으로하고, pivot 보다 크고 작은 part를 A B로 나눈다.
    %part 별로 sorting Algorithm 적용 후 접합.
    %pivot이 없으면 피봇을 포함시키고 정렬시킨 후 나중에 빼주는 방법으로..

    %100만개의 데이터 100개로 average 해서 mean을 구한다?(random sampling)
    % ㄴ not included in this code
    count = 1;
    smallSubset = [];
    largeSubset = [];
    subsetCounter = 0;
    averageValue = x(1);
    %get average pivot
    for i = 2:N
        averageValue = (averageValue+x(i))/2;
    end
    pivot = averageValue;
    for count = 1:N
        if x(count) <= pivot
            smallSubset = [smallSubset,x(count)];
            subsetCounter = subsetCounter+1;
        else
            largeSubset = [x(count),largeSubset];
        end
    end
    A = quickSort(smallSubset,1,subsetCounter);
    B = quickSort(largeSubset,1,N-subsetCounter);
    y = [A,B];
end
```

제안하는 정렬 기법

제안하는 정렬 기법을 그림이나 단계 별 정렬 과정 예제 등을 활용하여 설명하시오.

```
function y=proposedSort(x,N)
    %pivot을 평균값으로하고, pivot 보다 크고 작은 part를 A B로 나눈다.
    %part 별로 sorting Algorithm 적용 후 접합.
    %pivot이 없으면 피봇을 포함시키고 정렬시킨 후 나중에 빼주는 방법으로..

    %100만개의 데이터 100개로 average 해서 mean을 구한다?(random sampling)
    % ㄴ not included in this code
    count = 1;
    smallSubset = [];
    largeSubset = [];
    subsetCounter = 0;
    averageValue = x(1);
    %get average pivot
    for i = 2:N
        averageValue = (averageValue+x(i))/2;
    end
    pivot = averageValue;
    for count = 1:N
        if x(count) <= pivot
            smallSubset = [smallSubset,x(count)];
            subsetCounter = subsetCounter+1;
        else
            largeSubset = [x(count),largeSubset];
        end
    end
    A = quickSort(smallSubset,1,subsetCounter);
    B = quickSort(largeSubset,1,N-subsetCounter);
    y = [A,B];
end
```



정렬 알고리듬 성능 분석

- 각 알고리듬을 다음 시뮬레이션 시나리오에 입각하여 시뮬레이션 돌린 뒤 결과표를 작성하시오.

Simulation environments

1. Iteration 1e3
2. Size of random array(vector)
   2^9, 2^10, 2^11, 2^12, 2^13
3. Max of random array
   = Size of random array(vector)

My analysis of Sorting Algorithm Performance:
ShellSort algorithm is powerful in any size of random vector.
RadixSort (in the Decimal System) &MyProposedSort(devide and conquer) (both over than 2^13 size) is better than lower size vector.