Data Structure Lab

Created	@March 19, 2023 6:12 PM
Last Edited Time	@March 20, 2023 9:00 PM
By	**Borhan**<ASH2101008M>
Email	

From:

- Tamim Shahriar (DSA Playlist)
- Merge Sort Implementation (Simple Snippets)
- rand() function (GFG)

Generating Random Numbers

Generating Number from (1 to the N) : $rand()\%N + 1$

```
  #include<math.h>
...
  ll a[n];
  for(ll i=0; i<n; i++) a[i]=(rand()%100 + 1);
...
```

Selection Sort

Time Complexity : $O(n^2)$

```
#include<stdio.h>

#define ll int

int main(){
  ll n, lowest, t;
  scanf("%d", &n);

  ll a[n];
  for(ll i=0; i<n; i++) scanf("%d", &a[i]);

  /*
      // For generating random numbers in the array
      for(ll i=0; i<n; i++) a[i]=(rand()%100 + 1);
  */

  for(ll i=0; i<n; i++){
    lowest=i;

    for(ll j=i+1; j<n; j++){
      if(a[lowest] > a[j]) lowest=j;
    }
```

```
    t=a[lowest];
    a[lowest]=a[i];
    a[i]=t;
  }

  for(ll i=0; i<n; i++) printf("%d ", a[i]);
}
```

Bubble Sort

Time Complexity : $O(n^2)$

```
#include<stdio.h>

int main(){
  int n,t;
  scanf("%d", &n);

  int a[n];
  for(int i=0; i<n; i++) scanf("%d", &a[i]);

  /*
      // For generating random numbers in the array
      for(ll i=0; i<n; i++) a[i]=(rand()%100 + 1);
  */

  for(int i=0; i<n; i++){
    for(int j=0; j<n-i-1; j++){
      if(a[j] > a[j+1]){
        t=a[j];
        a[j]=a[j+1];
        a[j+1]=t;
      }
    }
  }

  for(int i=0; i<n; i++) printf("%d ", a[i]);
}
```

Insertion Sort

Time Complexity : $O(n^2)$

```
#include<stdio.h>

#define ll int

int main(){
  ll n,j,k,t;
  scanf("%d", &n);

  ll a[n];
  for(ll i=0; i<n; i++) scanf("%d", &a[i]);
  /*
      // For generating random numbers in the array
      for(ll i=0; i<n; i++) a[i]=(rand()%100 + 1);
  */
```

```
  for(ll i=1; i<n; i++){
    j= i-1;
    k=i;

    while((a[j] > a[k]) && j >= 0){
      t=a[k];
      a[k]=a[j];
      a[j]=t;

      j--;
      k--;
    }

  }

  for(ll i=0; i<n; i++) printf("%d ", a[i]);
}
```

Merge Sort

Time Complexity : $O(nlogn)$

```
#include<stdio.h>

void merge(int a[], int left, int mid, int right){
  int i = left;
  int j=mid+1;
  int k=0;
  int tmpArr[right-left+1];

  while(i<=mid && j <= right){
    if(a[i] < a[j]){
      tmpArr[k++]=a[i];
      i++;
    }else{
      tmpArr[k++]=a[j];
      j++;
    }
  }

  while(i<=mid) tmpArr[k++]=a[i++];
  while(j<=right) tmpArr[k++]=a[j++];

  for(int i=0; i<right-left+1; i++)
    a[left+i]=tmpArr[i];
}

void merge_sort(int a[], int left, int right){
  if(left>=right) return;

  merge_sort(a, left, (left+right)/2);
  merge_sort(a, ((left+right)/2)+1, right);

  merge(a, left, (left+right)/2, right);
}

int main(){
  int n;
  scanf("%d", &n);

  int a[n];
  for(int i=0; i<n; i++) scanf("%d", &a[i]);

  /*
```

```
      // For generating random numbers in the array
      for(ll i=0; i<n; i++) a[i]=(rand()%100 + 1);
  */

  merge_sort(a, 0, n-1);

  for(int i=0; i<n; i++) printf("%d ", a[i]);
}
```

Quick Sort by Recursion

Time Complexity : $O(nlogn)$

```
#include<stdio.h>
#include<math.h>

#define ll int

ll partition(ll a[],  ll low, ll high){
  ll pivot,i,j,t;

  pivot = a[high];
  for(i=low-1, j=low; j<high; j++){
    if(a[j] < pivot){
      i++;
      t=a[j];
      a[j]=a[i];
      a[i]=t;
    }
  }

  t=a[high];
  a[high]=a[i+1];
  a[i+1]=t;

  return (i+1);
}

void quickSort(ll a[], ll low, ll high){
  if(low >= high) return;

  ll p = partition(a, low, high);

  quickSort(a, low, p-1);
  quickSort(a, p+1, high);
}

int main(){
  ll n; scanf("%d", &n);
  ll a[n];
  for(ll i=0; i<n; i++){
    scanf("%d", &a[i]);
  }

  /*
      // For generating random numbers in the array
      for(ll i=0; i<n; i++) a[i]=(rand()%100 + 1);
  */

  quickSort(a, 0, (n-1));

  for(ll i=0; i<n; i++){
    printf("%d ", a[i]);
```

```
  }
}
```

Quick Sort By Stack

Time Complexity : $O(nlogn)$

```
#include<stdio.h>
#include<math.h>

#define ll int
#define MAX 100000

ll HIGH[MAX];
ll topHigh=-1;

ll LOW[MAX];
ll topLow=-1;

void highPush(ll data){
  if(topHigh == MAX-1){
    printf("Stack is full\n");
    return;
  }
  HIGH[++topHigh]=data;
}

int highPop(){
  if(topHigh == -1){
    printf("Stack is empty");
    return -1;
  }
  return HIGH[topHigh--];
}

void lowPush(ll data){
  if(topLow == MAX-1){
    printf("Stack is full");
    return;
  }
  LOW[++topLow]=data;
}

int lowPop(){
  if(topLow == -1){
    printf("Stack is empty");
    return -1;
  }
  return LOW[--topLow];
}


ll partition(ll a[],  ll low, ll high){
  ll pivot,i,j,t;

  pivot = a[high];
  for(i=low-1, j=low; j<high; j++){
    if(a[j] < pivot){
      i++;
      t=a[j];
      a[j]=a[i];
      a[i]=t;
```

```
    }
  }

  t=a[high];
  a[high]=a[i+1];
  a[i+1]=t;

  return (i+1);
}


int main(){
  ll n; scanf("%d", &n);
  ll a[n];

  for(ll i=0; i<n; i++){
    scanf("%d", &a[i]);
  }
  /*
      // For generating random numbers in the array
      for(ll i=0; i<n; i++) a[i]=(rand()%100 + 1);
  */

  lowPush(0); highPush(n-1);
  while(topHigh != -1 && topLow != -1){
    ll low=lowPop();
    ll high=highPop();

    ll p = partition(a, low, high);

    if(low < p-1){
      lowPush(low);
      highPush(p-1);
    }
    if(p+1 < high){
      lowPush(p+1);
      highPush(high);
    }

  }

  for(ll i=0; i<n; i++){
    printf("%d ", a[i]);
  }
}
```

Median of an array by Selection Sort

Time Complexity : $O(n^2)$

- If array_size/n is odd ⇒ middle element is median
- else ⇒ (summation of two middle element / 2)

```
#include<stdio.h>

#define ll int
```

```
int main(){
  ll n, lowest, t;
  scanf("%d", &n);

  ll a[n];
  for(ll i=0; i<n; i++) scanf("%d", &a[i]);

  for(ll i=0; i<n; i++){
    lowest=i;

    for(ll j=i+1; j<n; j++){
      if(a[lowest] > a[j]) lowest=j;
    }

    t=a[lowest];
    a[lowest]=a[i];
    a[i]=t;
  }

  if(n%2==0)
    printf("%0.2f", (a[(n)/2]+a[(n-1)/2])*1.000/2);
  else
    printf("%d", a[n/2]);
}
```

Median of an array using Random Numbers

```
#include<stdio.h>
#include<math.h>

#define ll int

int main(){
  ll n, lowest, t;
  scanf("%d", &n);


  ll a[n];
  // Generating random number from (1-100)
  for(ll i=0; i<n; i++) a[i]=(rand()%100 + 1);


  for(ll i=0; i<n; i++){
    lowest=i;

    for(ll j=i+1; j<n; j++){
      if(a[lowest] > a[j]) lowest=j;
    }

    t=a[lowest];
    a[lowest]=a[i];
    a[i]=t;
  }

  printf("The sorted array:\n");
  for(ll i=0; i<n; i++) printf("%d ", a[i]);
  printf("\n");

  printf("Median of the array: ");
  if(n%2==0)
    printf("%0.2f", (a[(n)/2]+a[(n-1)/2])*1.000/2);
  else
    printf("%d", a[n/2]);
}
```

Inserting

```
#include<stdio.h>
#include<math.h>

#define ll int

int main(){
  ll max_size_array,n;
  printf("Maximum size of array and n (the number of element you want to insert into the array at first)");
  scanf("%d %d", &max_size_array, &n);

  ll a[max_size_array];
  for(ll i=0; i<n; i++) scanf("%d", &a[i]);

  ll current_element=n;

  ll t;
  printf("Number of element you want to insert:");
  scanf("%d", &t);
   printf("\n");
  ll position, data;
  //position starts from 1,2,... but  array index starts from 0
  // so we must do (position-1)
  printf("Position (position <= current_element+1) and  data\n");
  while(t--){

     scanf("%d %d", &position, &data);
     if(current_element>=max_size_array){
      printf("Overflow.");
      break;
     }

     // current element starts from 1, but index from 0
     for(ll i=current_element-1; i>=(position-1); i--){
      a[i+1]=a[i];
     }

     a[position-1]=data;
     current_element++;

     for(ll i=0; i<current_element; i++){
      printf("%d ", a[i]);
     }
     printf("\n");
  }

}
```

Deleting

```
#include<stdio.h>
#include<math.h>

#define ll int

int main(){
  printf("Enter size of array:\n");
```

```
  ll n;
  scanf("%d", &n);

  ll a[n];
  for(ll i=0; i<n; i++){
    a[i]=rand()%100 + 1;
  }

  printf("The real array:\n");
  for(ll i=0; i<n; i++){
    printf("%d ", a[i]);
  }
  printf("\n");


  ll t;
  printf("Number of test cases or elements you wanna delete\n");
  scanf("%d", &t);

  ll position;
  printf("Enter the position (position <= n) element you wanna delete\n");
  while(t--){
    scanf("%d", &position);

    if(n <= 0) {
      printf("Underflow");
      break;
    }

    for(ll i=position-1; i<n-1;i++){
      a[i]=a[i+1];
    }
    n--;
    for(ll i=0; i<n; i++) printf("%d ", a[i]);
    printf("\n");
  }

}
```

Linear Searching

```
#include<stdio.h>
#include<math.h>

#define ll int

int main(){
  printf("Enter size of array & elements:\n");
  ll n;
  scanf("%d", &n);

  ll a[n];
  for(ll i=0; i<n; i++){
    scanf("%d", &a[i]);
  }

  printf("\nNumber of queries\n");
  ll t, data; scanf("%d", &t);

  printf("\nSearch value\n");
  while(t--){
```

```
    scanf("%d", &data);
    ll ans=-1;

    for(ll i=0; i<n; i++){
      if(a[i]==data){
        ans=i;
        break;
      }
    }

    if(ans==-1) printf("Not found\n");
    else printf("Found at %d (0 bases indexing)", ans);
  }

}
```

Binary Search Algorithm

```
#include<stdio.h>
#include<math.h>

#define ll int

int main(){
  printf("Enter size of array & elements:\n");
  ll n;
  scanf("%d", &n);

  ll a[n];
  for(ll i=0; i<n; i++){
    scanf("%d", &a[i]);
  }

  printf("\nNumber of queries\n");
  ll t, data; scanf("%d", &t);

  printf("\nSearch value\n");
  while(t--){

    scanf("%d", &data);
    ll ans=-1;

    ll i=0, j=n-1;
    ll mid;
    while(i<=j){
      mid = (i+j)/2;
      if(a[mid] == data){
        ans = mid;
        break;
      }else if(a[mid] > data){
        j=mid-1;
      }else{
        i=mid+1;
      }
    }

    if(ans==-1) printf("Not found\n");
    else printf("Found at %d (0 base)\n", ans);
  }

}
```