**NIT Kurukshetra**

**Department of computer engineering**

**Algorithms Design (Pr)**

**IT-311**

**Name : Abhinandan**

**Roll No: 1130704**

**Section: IT-5**

**Experiment 1 (Quick and Heap sort)**

**Program 1: Implement the ascending and desending order using Quick Sort.**

#include<bits/stdc++.h>

using namespace std;

int part(int arr[],int low,int high)

{

int pivot=low;

int i=low,j=high;

while(i<j)

{

while(i<high&&arr[i]<=arr[pivot])

i++;

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

j--;

if(i<j)

{

int temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

int temp=arr[pivot];

arr[pivot]=arr[j];

arr[j]=temp;

return j;

}

void quicksort(int arr[],int low,int high)

{

if(low<high)

{

int i=part(arr,low,high);

quicksort(arr,low,i-1);

quicksort(arr,i+1,high);

}

}

int arr[11000];

int main()

{

int i,j,k;

int n;

scanf("%d",&n);

for(i=0;i<n;i++)

{

scanf("%d",&arr[i]);

}

quicksort(arr,0,n-1);

for(i=0;i<n;i++)

{

printf("%d ",arr[i]);

}

return 0;

}

**Program 2: Implement Quick Sort with duplicate numbers in the given array/elements.**

#include<bits/stdc++.h>

using namespace std;

int part(int arr[],int low,int high)

{

int pivot=low;

int i=low,j=high;

while(i<j)

{

while(i<high&&arr[i]>=arr[pivot])

i++;

while(arr[j]<arr[pivot])

j--;

if(i<j)

{

int temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

int temp=arr[pivot];

arr[pivot]=arr[j];

arr[j]=temp;

return j;

}

void quicksort(int arr[],int low,int high)

{

if(low<high)

{

int i=part(arr,low,high);

quicksort(arr,low,i-1);

quicksort(arr,i+1,high);

}

}

int arr[11000];

int main()

{

int i,j,k;

int n;

scanf("%d",&n);

for(i=0;i<n;i++)

{

scanf("%d",&arr[i]);

}

quicksort(arr,0,n-1);

for(i=0;i<n;i++)

{

printf("%d ",arr[i]);

}

return 0;

}

**Program 3: Finding kth minimum and maximum element in Heap.**

#include<bits/stdc++.h>

using namespace std;

int arr[11000];

//max\_heap

void percolate\_down(int arr[],int n,int idx)

{

if(idx==n)

return;

int l,r;

l=idx\*2+1;

r=idx\*2+2;

int max\_val=arr[idx];

int max\_idx=idx;

if(l<n&&arr[l]>max\_val)

{

max\_idx=l;

max\_val=arr[l];

}

if(r<n&&arr[r]>max\_val)

{

max\_idx=r;

max\_val=arr[r];

}

if(max\_idx!=idx)

{

int temp=arr[idx];

arr[idx]=arr[max\_idx];

arr[max\_idx]=temp;

percolate\_down(arr,n,max\_idx);

}

}

void heapify(int arr[],int n)

{

int i,j;

for(i=n/2-1;i>=0;i--)

{

percolate\_down(arr,n,i);

}

}

int extract\_max(int arr[],int &n)

{

int ans=arr[0];

arr[0]=arr[n-1];

n--;

percolate\_down(arr,n-1,0);

return ans;

}

int main()

{

int i,j,k;

int n;

scanf("%d",&n);

for(i=0;i<n;i++)

{

scanf("%d",&arr[i]);

}

heapify(arr,n);

for(i=0;i<n;i++)

cout<<arr[i]<<" ";

cout<<endl;

cout<<"Enter value k : ";

scanf("%d",&k);

for(i=0;i<k-1;i++)

{

extract\_max(arr,n);

}

cout<<extract\_max(arr,n)<<endl;

return 0;

}

**Program 4: Delete kth indexed element in Min heap and Max heap.**

#include<bits/stdc++.h>

using namespace std;

int arr[11000];

//max\_heap

void percolate\_down(int arr[],int n,int idx)

{

if(idx==n)

return;

int l,r;

l=idx\*2+1;

r=idx\*2+2;

int max\_val=arr[idx];

int max\_idx=idx;

if(l<n&&arr[l]>max\_val)

{

max\_idx=l;

max\_val=arr[l];

}

if(r<n&&arr[r]>max\_val)

{

max\_idx=r;

max\_val=arr[r];

}

if(max\_idx!=idx)

{

int temp=arr[idx];

arr[idx]=arr[max\_idx];

arr[max\_idx]=temp;

percolate\_down(arr,n,max\_idx);

}

}

void delete\_index(int arr[],int &n,int idx)

{

arr[idx]=arr[n-1];

n--;

percolate\_down(arr,n,idx);

}

void heapify(int arr[],int n)

{

int i,j;

for(i=n/2-1;i>=0;i--)

{

percolate\_down(arr,n,i);

}

}

int extract\_max(int arr[],int &n)

{

int ans=arr[0];

arr[0]=arr[n-1];

n--;

percolate\_down(arr,n-1,0);

return ans;

}

int main()

{

int i,j,k;

int n;

scanf("%d",&n);

for(i=0;i<n;i++)

{

scanf("%d",&arr[i]);

}

heapify(arr,n);

for(i=0;i<n;i++)

cout<<arr[i]<<" ";

cout<<endl;

cout<<"Enter index k : ";

scanf("%d",&k);

delete\_index(arr,n,k);

for(i=0;i<n;i++)

cout<<arr[i]<<" ";

return 0;

}

**Experiment 2 (Dynamic Programming)**

**Program 1: Implement the matrix chain multiplication problem.**

#include<bits/stdc++.h>

using namespace std;

int arr[101000];

int mcm(int n)

{

int dp[n][n];

int i, j, k, L, q;

for (i = 1; i < n; i++)

dp[i][i] = 0;

for (L=2; L<n; L++)

{

for (i=1; i<=n-L+1; i++)

{

j = i+L-1;

dp[i][j] = INT\_MAX;

for (k=i; k<=j-1; k++)

{

q = dp[i][k] + dp[k+1][j] + arr[i-1]\*arr[k]\*arr[j];

if (q < dp[i][j])

dp[i][j] = q;

}

}

}

return dp[1][n-1];

}

int main()

{

int i,j,k;

int n;

scanf("%d",&n);

for(i=0;i<n;i++)

{

scanf("%d",&arr[i]);

}

printf("Minimum required multiplications %d\n",mcm(n));

return 0;

}

**Program 2 : Find the Longest palindromic subsequence.**

#include<bits/stdc++.h>

using namespace std;

int longestps(char str[])

{

int n = strlen(str);

int i, j, gap;

int dp[n][n];

for (i = 0; i < n; i++)

dp[i][i] = 1;

for (gap=2; gap<=n; gap++)

{

for (i=0; i<n-gap+1; i++)

{

j = i+gap-1;

if (str[i] == str[j] && gap == 2)

dp[i][j] = 2;

else if (str[i] == str[j])

dp[i][j] = dp[i+1][j-1] + 2;

else

dp[i][j] = max(dp[i][j-1], dp[i+1][j]);

}

}

return dp[0][n-1];

}

char str[10100];

int main()

{

int i,j,k;

scanf("%s",str);

printf("Required length of lps : %d\n",longestps(str));

return 0;

}

**Program 3 : Find the longest common substring.**

#include<bits/stdc++.h>

using namespace std;

char str1[1010];

char str2[1010];

int n,m;

int dp[1010][1010];

int lcs()

{

int ans = 0;

for (int i=0; i<=m; i++)

{

for (int j=0; j<=n; j++)

{

if (i == 0 || j == 0)

dp[i][j] = 0;

else if (str1[i-1] == str2[j-1])

{

dp[i][j] = dp[i-1][j-1] + 1;

ans = max(ans, dp[i][j]);

}

else dp[i][j] = 0;

}

}

return ans;

}

int main()

{

int i,j,k;

scanf("%s",str1);

scanf("%s",str2);

m=strlen(str1);

n=strlen(str2);

printf("lcs is : %d",lcs());

return 0;

}