Assignment 2 Design and analysis of algorithm

Name abdullah

class no p166033

Q1

#include<iostream>

using namespace std;

void removegarbage(int twodlist[8][8]){

int i,j;

for (i=8;i>=1;i--)

for (j=8;j>=1;j--){

twodlist[i][j] = 0;

}

}

int noofshortestpath(){

int twodlist[8][8];

int i ,j;

removegarbage(twodlist);

for (i=0;i<=7;i++){

for (j=0;j<=7;j++){

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

twodlist[i][j]=1;

else

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

cout << "Number of shortest paths form top right to "<<i<<"th row "<<j<<"th column is "<< twodlist[i][j]<<endl;

}

}

}

int main(){

cout << noofshortestpath();

system("pause");

}

The runing time is O(n^2)

Q2

#include<iostream>

using namespace std;

int maximum(int a, int b)

{

if(a > b)

return a;

else

return b;

}

int Wirecut(int profit[5], int length)

{

int array[length+1];

array[0] = 0;

int i, j;

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

{

int max = -1;

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

max = maximum(max, profit[j] + array[i-j-1]);

array[i] = max;

}

return array[length];

}

int main()

{

int profit[]={2,5,7,8,10},length = 6;

cout<<Wirecut(profit, length)<<endl;

system("pause");

}

Time Complexity of the above implementation is O(n^2)

Q#3

#include <iostream>

#include <cstring>

using namespace std;

void Display\_subString(char\* str, int low, int high)

{

for( int i = low; i <= high; ++i )

cout << str[i];

}

int Long\_palandrom(char \*str)

{

int maxlen = 1;

int start = 0;

int len = strlen(str);

int low, high;

for (int i = 1; i < len; ++i)

{

low = i - 1;

high = i;

while (low >= 0 && high < len && str[low] == str[high])

{

if (high - low + 1 > maxlen)

{

start = low;

maxlen = high - low + 1;

}

--low;

++high;

}

low = i - 1;

high = i + 1;

while (low >= 0 && high < len && str[low] == str[high])

{

if (high - low + 1 > maxlen)

{

start = low;

maxlen = high - low + 1;

}

--low;

++high;

}

}

cout<<"Longest palindrome substring is: ";

Display\_subString(str, start, start + maxlen - 1);

return maxlen;

}

int main()

{

char str[] = "racecar"<<endl;

cout<<"Length is: "<<Long\_palandrom(str)<<endl;

return 0;

}

Time complexity: O(n^2)

Q4

#include <iostream>

using namespace std;

int missingSearch(int array[], int size) {

int left = 0, right = size - 1;

int mid;

while ((right - left) > 1) {

mid = (left + right) / 2;

if ((array[left] - left) != (array[mid] - mid))

right = mid;

else if ((array[right] - right) != (array[mid] - mid))

left = mid;

}

return (array[mid] + 1);

}

int main()

{

int array[] = { 1, 2,3,4,5,7 };

int size = sizeof(array) / sizeof(array[0]);

cout << "Missing number :" << missingSearch(array, size)<<endl;

system("pause");

}

Time compleity O(log n).

Q#6

#include<iostream>

#include<stdio.h>

int count( int S[], int m, int n )

{

if (n == 0)

return 1;

if (n < 0)

return 0;

if (m <=0 && n >= 1)

return 0;

return count( S, m - 1, n ) + count( S, m, n-S[m-1] );

}

int main()

{

int i, j;

int arr[] = {1,3,5,7};

int m = sizeof(arr)/sizeof(arr[0]);

cout<<count(arr, m, 9));

getchar();

return 0;

}

Q5

#include <stdlib.h>

#include<iostream>

using namespace std;

int lis(int arr[], int n)

{

int \*lis, i, j, max = 0;

lis = (int\*)malloc(sizeof(int) \* n);

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

lis[i] = 1;

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

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

if (arr[i] > arr[j] && lis[i] < lis[j] + 1)

lis[i] = lis[j] + 1;

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

if (max < lis[i])

max = lis[i];

delete(lis);

return max;

}

int main()

{

int arr[] = { 10, 22, 9, 33, 21, 50, 41, 60 };

int n = sizeof(arr) / sizeof(arr[0]);

cout<<"Length of lis is"<< lis(arr, n)<<endl;

return 0;

}