LCS problem

Write a program to find longest common subsequence using dynamic programming approach.

#include<stdio.h>

int max(int a, int b);

/\* Returns length of LCS for X[0..m-1], Y[0..n-1] \*/

int lcs( char \*X, char \*Y, int m, int n )

{

int L[m+1][n+1];

int i, j;

/\* Following steps build L[m+1][n+1] in bottom up fashion. Note

that L[i][j] contains length of LCS of X[0..i-1] and Y[0..j-1] \*/

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

{

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

{

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

L[i][j] = 0;

else if (X[i-1] == Y[j-1])

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

else

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

}

}

return L[m][n];

}

int max(int a, int b)

{

return (a > b)? a : b;

}

int main()

{

char X[] = "AGGTAB";

char Y[] = "GXTXAYB";

int m = strlen(X);

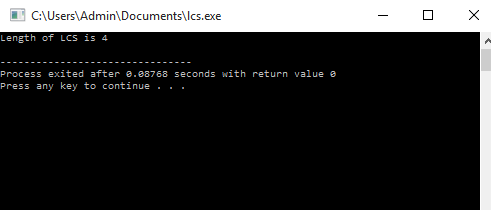
int n = strlen(Y);

printf("Length of LCS is %d\n", lcs( X, Y, m, n ) );

return 0;

}

Output:-



01KNAPSACK PROBLEM

Write a program to solve 01 knapsack problem using dynamic programming approach .

// A Dynamic Programming based solution for 0-1 Knapsack problem

#include<stdio.h>

int max(int a, int b) { return (a > b)? a : b; }

int knapSack(int W, int wt[], int val[], int n)

{

int i, w;

int K[n+1][W+1];

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

{

for (w = 0; w <= W; w++)

{

if (i==0 || w==0)

K[i][w] = 0;

else if (wt[i-1] <= w)

K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w]);

else

K[i][w] = K[i-1][w];

}

}

return K[n][W];

}

int main()

{

int val[] = {60, 100, 120};

int wt[] = {10, 20, 30};

int W = 50;

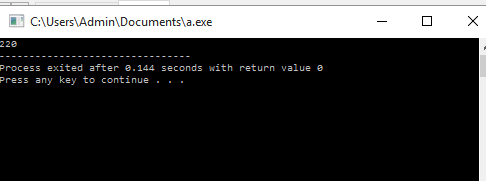
int n = sizeof(val)/sizeof(val[0]);

printf("%d", knapSack(W, wt, val, n));

return 0 ;

}

Output:-



FLOYD –WARSHELL ALGORITHM

Write a program to find all pair shortest path using dynamic programming approach.

//program for finding all pair shortest path using dynamic programming

#include<stdio.h>

#define V 4

#define INF 99999

void printSolution(int dist[][V]);

void floydWarshell (int graph[][V])

{

int dist[V][V], i, j, k;

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

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

dist[i][j] = graph[i][j];

for (k = 0; k < V; k++)

{

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

{

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

{

if (dist[i][k] + dist[k][j] < dist[i][j])

dist[i][j] = dist[i][k] + dist[k][j];

}

}

}

printSolution(dist);

}

void printSolution(int dist[][V])

{

int i,j;

printf ("Following matrix shows the shortest distances"

" between every pair of vertices \n");

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

{

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

{

if (dist[i][j] == INF)

printf("%7s", "INF");

else

printf ("%7d", dist[i][j]);

}

printf("\n");

}

}

int main()

{

int graph[V][V] = { {0, 5, INF, 10},

{INF, 0, 3, INF},

{INF, INF, 0, 1},

{INF, INF, INF, 0}

};

floydWarshell(graph);

return 0;

}

Output:-

