Dominic Ginter

CS 477

Homework #6

2. Let dHarmless[1…n] store optimal values if the laser is first used at second i in dHarmless[i]. If a laser is used at second i, then min(xi, f(i – j)) additional drones harmless if j represents the day the laser was previously used such that 0 <= j < i. Now we since we want the most optimal choice of all, we have to check all the values of j. So, dHarmless[i] = (from j = 0 to i – 1)findMax(findMin(xi, f(i-j)) + dHarmless[j])

#include <iostream>

using namespace std;

int findMin(int x, int y) //helper function to find min value

{

if (x < y) //Want min so if x < y return x....

{

return x;

}

return y; //Else return y

}

int findMax(int x, int y)

{

if(x > y)

{

return x;

}

return y;

}

void printTable(int x[], int f[], int drones[])

{

cout << "Second ";

for(int i = 0; i < 4; i++)

{

cout << i + 1 << " ";

}

cout << "\n- - - - - - - - - - - - - - - -|" << endl;

cout << "xI |";

for(int i = 0; i < 4; i++)

{

if ( i == 3) //hard code for table format

{

cout << " ";

}

cout << x[i] << " ";

}

cout << "\nfI |";

for(int i = 0; i < 4; i++)

{

cout << f[i] << " ";

}

cout << "\nSub |";

for(int i = 0; i < 4; i++)

{

cout << drones[i] << " ";

}

cout << "\n- - - - - - - - - - - - - - - -|" << endl;

}

int main()

{

int n = 4;

int xI[n] = {1, 10, 10, 1}; // Drone arrivals from table. first value at index 0 is a default #

int fI[n] = {1, 2, 4, 8}; // Function from table. first value at index 0 is a default #

int dHarmless[n]; // This will store number of harmless drones for each second

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

{

dHarmless[i] = findMin(xI[i], fI[i]); // Laser used for the first time

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

{

dHarmless[i] = findMax(findMin(xI[i], fI[i- j - 1]) + dHarmless[j], dHarmless[i]); //

}

}

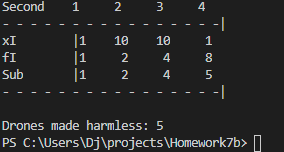
printTable(xI, fI, dHarmless);

cout << "\nDrones made harmless: " << dHarmless[n - 1] << endl; //Print out optimal solution.

return 0;

}

Output:



#include <iostream>

int tempArr[n];

int mat[n][n];

int xI[n] = {1, 10, 10, 1}; // Drone arrivals from table. first value at index 0 is a default #

int fI[n] = {1, 2, 4, 8}; // Function from table. first value at index 0 is a default #

int dHarmless[n]; // This will store number of harmless drones for each second

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

{

dHarmless[i] = findMin(xI[i], fI[i]); // Laser used for the first time

tempArr[i] = 0;

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

{

temp = findMax(findMin(xI[i], fI[i - j - 1]) + dHarmless[j], dHarmless[i]);

if(temp > dHarmless[i])

{

dHarmless[i] = temp;

tempArr[n] = j + 1;

tempArr[n - 1] = i + 1;

}

}

}

printTable(xI, fI, dHarmless);

temp = n - 1;

cout << "Laser used on days ";

cout << tempArr[n] << " and ";

cout << tempArr[n - 1];

cout << endl;

cout << "\nDrones made harmless: " << dHarmless[n - 1] << endl; //Print out optimal solution.

return 0;

}

#include <iostream>

#include <utility>

using namespace std;

int findMin(int x, int y) //helper function to find min value

{

if (x < y) //Want min so if x < y return x....

{

return x;

}

return y; //Else return y

}

int findMax(int x, int y)

{

if(x > y)

{

return x;

}

return y;

}

void printTable(int x[], int f[], int drones[])

{

cout << "Second ";

for(int i = 0; i < 4; i++)

{

cout << i + 1 << " ";

}

cout << "\n- - - - - - - - - - - - - - - -|" << endl;

cout << "xI |";

for(int i = 0; i < 4; i++)

{

if ( i == 3) //hard code for table format

{

cout << " ";

}

cout << x[i] << " ";

}

cout << "\nfI |";

for(int i = 0; i < 4; i++)

{

cout << f[i] << " ";

}

cout << "\nSub |";

for(int i = 0; i < 4; i++)

{

cout << drones[i] << " ";

}

cout << "\n- - - - - - - - - - - - - - - -|" << endl;

}

int main()

{

int n = 4;

int temp;

int tempArr[n];

int mat[n][n];

int xI[n] = {1, 10, 10, 1}; // Drone arrivals from table. first value at index 0 is a default #

int fI[n] = {1, 2, 4, 8}; // Function from table. first value at index 0 is a default #

int dHarmless[n]; // This will store number of harmless drones for each second

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

{

dHarmless[i] = findMin(xI[i], fI[i]); // Laser used for the first time

tempArr[i] = 0;

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

{

temp = findMax(findMin(xI[i], fI[i - j - 1]) + dHarmless[j], dHarmless[i]);

if(temp > dHarmless[i])

{

dHarmless[i] = temp;

tempArr[n] = j + 1;

tempArr[n - 1] = i + 1;

}

}

}

printTable(xI, fI, dHarmless);

cout << endl;

cout << "\nDrones made harmless: " << dHarmless[n - 1] << endl; //Print out optimal solution.

return 0;

}

d)

#include <iostream>

#include <utility>

using namespace std;

int findMin(int x, int y) //helper function to find min value

{

if (x < y) //Want min so if x < y return x....

{

return x;

}

return y; //Else return y

}

int findMax(int x, int y)

{

if(x > y)

{

return x;

}

return y;

}

void printTable(int x[], int f[], int drones[])

{

cout << "Second ";

for(int i = 0; i < 4; i++)

{

cout << i + 1 << " ";

}

cout << "\n- - - - - - - - - - - - - - - -|" << endl;

cout << "xI |";

for(int i = 0; i < 4; i++)

{

if ( i == 3) //hard code for table format

{

cout << " ";

}

cout << x[i] << " ";

}

cout << "\nfI |";

for(int i = 0; i < 4; i++)

{

cout << f[i] << " ";

}

cout << "\nSub |";

for(int i = 0; i < 4; i++)

{

cout << drones[i] << " ";

}

cout << "\n- - - - - - - - - - - - - - - -|" << endl;

}

int main()

{

int n = 4;

int temp;

int tempArr[n];

int mat[n][n];

int xI[n] = {1, 10, 10, 1}; // Drone arrivals from table. first value at index 0 is a default #

int fI[n] = {1, 2, 4, 8}; // Function from table. first value at index 0 is a default #

int dHarmless[n]; // This will store number of harmless drones for each second

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

{

dHarmless[i] = findMin(xI[i], fI[i]); // Laser used for the first time

tempArr[i] = 0;

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

{

temp = findMax(findMin(xI[i], fI[i - j - 1]) + dHarmless[j], dHarmless[i]);

if(temp > dHarmless[i])

{

dHarmless[i] = temp;

tempArr[n] = j + 1;

tempArr[n - 1] = i + 1;

}

}

}

printTable(xI, fI, dHarmless);

temp = n - 1;

cout << "Laser used on days ";

cout << tempArr[n] << " and ";

cout << tempArr[n - 1];

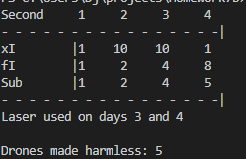
cout << endl;

cout << "\nDrones made harmless: " << dHarmless[n - 1] << endl; //Print out optimal solution.

return 0;

}

Output

:

2. True. If X and Y both begin with ‘A’, then ‘A’ then every subsequence of X and Y will also start with A as A is used as a prefix to the longest subsequence.
3. True because when 2 strings have the same last character, then that last character is removed when finding the LCS, the removed character is then appended back at the end to form the longest subsequence.