Dynamic Programming

Prepared by Jakir Hasan
SUST CSE'18
18/11/21

Content

0-1 Knapsack Problem

Statement

Given a set of n items with each item i having a positive weight wi and a positive benefit bi. Goal is to choose items with maximum total benefit but with weight at most w.

Example

id	1	2	3	4	
weight	2	3	4	5	
value	3	4	5	6	

Solution

i/w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

Implementation

```
#include <bits/stdc++.h>
using namespace std;

struct item
{
    int id;
    int weight;
    int profit;
};


int main()
{
    int n, i, j, weight, profit, capacity;
    item items[1000];
    int container[100][100];

    cin>> n >> capacity;

    for (i = 1; i <= n; i++)
    {
        cin>> weight >> profit;

        items[i].id = i;
        items[i].weight = weight;
        items[i].profit = profit;
    }

    for (i = 0; i <= n; i++)
        container[i][0] = 0;

    for (j = 0; j <= capacity; j++)
        container[0][j] = 0;

    for (i = 1; i <= n; i++)
    {
        weight = items[i].weight;
        profit = items[i].profit;

        for (j = 1; j <= capacity; j++)
        {
```

```
            if (weight <= j)
            {
                container[i][j] = max(container[i-1][j],
container[i-1][j-weight] + profit);
            }
            else
            {
                container[i][j] = container[i-1][j];
            }
        }
    }

    cout<< "Maximum profit is " << container[n][capacity] << "\n";

    return 0;
}

/*
4 5

2 3
3 4
4 5
5 6
*/
```

@Credit: Slides of Masum Sir

Coin Change Problem

Statement
Change amount A into as few coins as possible, when we have n coin denominations.

Example

A = 7
Denominations = [1, 2, 4, 7]

#di/amount	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	1	1	2	2	3	3	4
3	0	1	1	2	1	2	2	3
4	0	1	1	2	1	2	2	1

Implementation

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int n, i, j, coins[10000], target;
    int arr[100][100];

    cin>> n >> target;

    for (i = 1; i <= n; i++)
        cin>> coins[i];

    for (i = 0; i <= n; i++)
        arr[i][0] = 0;

    for (j = 0; j <= target; j++)
        arr[0][j] = INT_MAX;

    for (i = 1; i <= n; i++)
    {
        int coin_value = coins[i];
```

```
        for (j = 1; j <= target; j++)
        {
            if (j >= coin_value)
            {
                arr[i][j] = min(arr[i-1][j], arr[i][j-coin_value] + 1);
            }
            else
            {
                arr[i][j] = arr[i-1][j];
            }
        }
    }

    printf("Minimum number of coins to change amount %d is %d\n", target,
arr[n][target]);

    return 0;
}

/*
4 7
1 2 4 7
*/
```

@Credit: Slides of Masum Sir

Longest Common Subsequence Problem (LCS)

A subsequence of a sequence/string S, is obtained by deleting symbols from S. For example: the following are some subsequences of president: pred, sdn, predent

LCS Problem
Given two sequences X = {x1, x2, ……, xn} and Y = {y1, y2, ……, yn}, find a maximum length common subsequence of X and Y.

Example

X = “ABCBDAB”
Y = “BDCABA”

		0	1	2	3	4	5	6
		yi	B	D	C	A	B	A
0	xi	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	1	1	2	2	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4

Implementation

```
#include <bits/stdc++.h>
using namespace std;

char directions[100][100];
vector<char> subsequence;

void print_LCS(string X, int i, int j);

int main()
{
    string X, Y;
    int n, m, i, j, arr[100][100], p, q;

    cin>> X >> Y;

    m = X.size();
    n = Y.size();
```

```
    for (i = 0; i <= m; i++)
        arr[i][0] = 0;

    for (j = 0; j <= n; j++)
        arr[0][j] = 0;

    for (i = 1; i <= m; i++)
    {
        for (j = 1; j <= n; j++)
        {
            p = i-1;
            q = j-1;

            if (X[p] == Y[q])
            {
                arr[i][j] = arr[i-1][j-1] + 1;
                directions[i][j] = 'D';
            }
            else if (arr[i-1][j] > arr[i][j-1])
            {
                arr[i][j] = arr[i-1][j];
                directions[i][j] = 'U';
            }
            else
            {
                arr[i][j] = arr[i][j-1];
                directions[i][j] = 'L';
            }
        }
    }

    print_LCS(X, m, n);
    printf("Maximum length of common subsequence of %d\n", arr[m][n]);
    cout<< "Longest common subsequence is: ";

    for (i = 0; i < subsequence.size(); i++)
        cout<< subsequence[i];
    cout<< "\n";

    return 0;
}
```

```
void print_LCS(string X, int i, int j)
{
    if (i == 0 or j == 0)
        return;

    if (directions[i][j] == 'D')
    {
        print_LCS(X, i-1, j-1);
        subsequence.push_back(X[i-1]);
    }
    else if (directions[i][j] == 'U')
        print_LCS(X, i-1, j);
    else
        print_LCS(X, i, j-1);

}

/*
ABCBDAB
BDCABA
*/
```

@Credit: Slides of Masum Sir

Matrix Chain Multiplication Problem

Problem
Given a sequence of matrices A1, A2, …., An, with Ai of dimension mi * ni, insert parentheses to minimize the total number of scalar multiplications.

Example

Matrix	Dimension
A1	30 * 20
A2	20 * 15

A3	15 * 5
A4	5 * 10
A5	10 * 5
A6	5 * 10

Solution

	1	2	3	4	5	6
1	0	9000	4500	6000	5125	6500
2		0	1500	2500	2125	3000
3			0	750	625	500
4				0	250	500
5					0	500
6						0

Implementation

```
#include <bits/stdc++.h>
#define INF INT_MAX
using namespace std;

int main()
{
    int n, i, j, k, arr[100][100], length, dimensions[10000], product, len;
    cin>> len;

    for (i = 0; i < len; i++)
        cin>> dimensions[i];

    n = len-1;

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

```
        arr[i][i] = 0;

    for (length = 1; length <= n-1; length++)
    {
        for (i = 1; i <= n-length; i++)
        {
            j = i + length;
            arr[i][j] = INF;

            for (k = i; k <= j-1; k++)
            {
                product = arr[i][k] + arr[k+1][j] + dimensions[i-1] *
dimensions[k] * dimensions[j];

                if (product < arr[i][j])
                    arr[i][j] = product;
            }
        }
    }

    cout<< "Minimum number of multiplications is " << arr[1][n] << "\n";

    return 0;
}

/*
7
30 20 15 5 10 5 10
*/
```

@Credit: Slides of Masum Sir