**Experiment-6**

**Title: Dynamic Programming-I**

1. **Implement 0/1 Knapsack problem using Dynamic Programming**

Code:

#include<stdio.h>

int main()

{

int i,n,wt,r,c,temp,k;

printf("Enter the number of elements of the array: ");

scanf("%d",&n);

printf("Enter the total weight of the bag: ");

scanf("%d",&wt);

int item[n],value[n],weight[n],matrix[n+1][wt+1];

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

{

printf("\n\n");

printf("Enter the item number, value and weight of the %d item\n",i+1);

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

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

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

}

for(r=0;r<n+1;r++)

{

for(c=0;c<wt+1;c++)

{

matrix[r][c]=0;

}

}

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

{

for(c=1;c<wt+1;c++)

{

if(weight[r-1]>c)

{

matrix[r][c]=matrix[r-1][c];

}

else if(weight[r-1]<=c)

{

k=c-weight[r-1];

temp=value[r-1]+matrix[r-1][k];

if((matrix[r-1][c])>=(temp))

{

matrix[r][c]=matrix[r-1][c];

}

else

{

matrix[r][c]=temp;

}

}

}

}

printf("\n\n");

for(r=0;r<n+1;r++)

{

for(c=0;c<wt+1;c++)

{

printf("%d\t",matrix[r][c]);

}

printf("\n");

}

return 0;

}

1. **The order in which we parenthesize the product affects the number of simple arithmetic operations needed to compute the product, or the *efficiency* .(using matrix chain multiplication)**

Code:

#include<stdio.h>

#include<limits.h>

int MatrixChainOrder(int p[], int i, int j)

{

if(i == j)

return 0;

int k;

int min = INT\_MAX;

int count;

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

{

count = MatrixChainOrder(p, i, k) + MatrixChainOrder(p, k+1, j) + p[i-1]\*p[k]\*p[j];

if (count < min)

min = count;

}

return min;

}

int main()

{

int n,i;

printf("Enter the number of elements of array: ");

scanf("%d",&n);

int arr[n];

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

{

printf("Enter the %d element of the array: ",i+1);

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

}

printf("Minimum number of multiplications is %d ", MatrixChainOrder(arr, 1, n-1));

return 0;

}