Write a program to implement Knapsack problem using brute-force design

technique and analyze its time efficiency. Knapsack Problem: Given n items of

known weights w1, w2, ..wn values v1, v2, ...vn and a knapsack of capacity B, find

the most valuable subset of items that fit into the knapsack.

|  |
| --- |
| **Description:** The exhaustive search approach to Knapsack problem leads to generating all the subsets of the set of n items given computing the total weight of each subset in order to identify feasible subsets and finding a subset of the largest value among them. |

|  |
| --- |
| **ALGORITHM** *Knapsack (W*[0*..n* − 1],*V[**0..n-1],B)*  //Determine the Knapsack for a capacity B  //Input: An array W[0..*n* − 1] of weights, V[0..*n* − 1] of values, capacity B  //Output: An array K[0..m-1] of Knapsack items  maxVal ← 0  while there is a unique bit array of size n  begin  0 in the bit array at position i implies element vi is absent in Knapsack  1 in the bit array at position i implies element vi is present in Knapsack  Add all the weights corresponding to 1 in the bit array  if total weight < B then  Add all the Values corresponding to 1 in the bit array  if total value > maxVal then  maxVal ← total value  end if  end if  end while |

|  |
| --- |
| **Time Complexity :**  The basic operation in this is comparison inside loop. This is repeated as many times as the iterations of the loop. The loop is repeated 2ntimes.    Hence, C(n)= θ(2n) |

|  |
| --- |
| **Program**  #include<stdio.h>  #include<math.h>  #include<stdlib.h>  int Knapsack(unsigned int \*w, unsigned int \*v, unsigned int n, unsigned int B)  {  unsigned int i,temp;  unsigned int maxVal=0, maxSequence=0;  unsigned int totalWeight, totalValue;  unsigned int index=0;  //Generate bit array  for(i=1;i<pow(2,n);i++)  {  //Compute total weight  temp = i;  totalWeight=totalValue=0;  index=0;  while(temp) {  if(temp & 0x1) {  totalWeight = totalWeight + w[index];  totalValue = totalValue + v[index];  }  index++;  temp = temp >> 1;  }  if(totalWeight <= B && totalValue > maxVal) {  maxVal = totalValue;  maxSequence = i;  }  }  return maxSequence;  }    void main() {  unsigned int \*v,\*w, i,n,knaps, B;  printf("Enter the number of elements ");  scanf("%d", &n);  v= (unsigned int \*)calloc(n, sizeof(unsigned int));  w = (unsigned int \*) calloc(n, sizeof(unsigned int));  printf("Please enter the weights");  for(i=0;i<n;i++)  scanf("%d",&w[i]);  printf("Please enter the values");  for(i=0;i<n;i++)  scanf("%d",&v[i]);    printf("Please enter the Knapsack capacity");  scanf("%d", &B);  knaps = Knapsack(w,v,n,B);  printf("Knapsack contains the following items \n");  i=0;  while(knaps) {  if(knaps & 0X1)  printf("item %u value %u", (i+1), v[i]);  i++;  knaps = knaps >> 1;  }    } |

|  |
| --- |
| **Sample Input and Output :**  Enter the number of elements 3  Please enter the weights1 2 4  Please enter the values2 4 8  Please enter the Knapsack capacity5  Knapsack contains the following items  5 item 1 value 2item 3 value 8 |