*CARGO LOADING WITH PRIORITIES*

NISHANT GARG – 16BCE0088 (G1)

ANUSHA - 16BIS0113 (G2)

UTSAV -16BME0503 (G1)

ABSTRACT:

The container loading problem is one of the most difficult cutting and packing (mathematical optimization) problems that aims at optimal use of resources. Container loading is a very important part of supply chain management and efficient packing of containers is important to ensure reduction of unnecessary costs and satisfy customers. There are a lot of real life constraints that have to be considered while computing an algorithm for these type of problems, related to the container, cargo items, load bearing, etc. This project considers a type of cargo constraints.

**INTRODUCTION:**

The container loading problem represents a type of cutting and packing problems. They can be interpreted as geometric assignment problems where three dimensional small items – the cargo, have to be packed into a larger three-dimensional container.

This type of problems has to be solved daily in real life for a variety of different situations. Because of its importance, the container loading problem has been researched on extensively and a lot of solutions have been proposed.

In real life scenarios, there are a lot of constraints that come into play when the loading has to be efficient, like- maximum space utilization, conditions on orientation of boxes, weight constrictions, loading priorities, multi drop, etc. Most researches have focused on maximum volume utilization of container.

In the problem considered in this project, we accept cargo boxes with priorities. The aim is the arrangement should be such that the boxes with higher priority are placed in more accessible positions than the boxes with lower priority.

**PROBLEM DESCRIPTION:**

We assume a dry storage container of dimensions 5mx3mx5m, in which we need to pack cargo boxes.

INPUTS:

* The number of boxes ‘n’
* The length(len), breadth(brd), height(ht) and priority(pt) of each box.

BOUNDARY CONDITIONS:

* The dimensions of the cargo boxes should not exceed the dimensions of the container
* The total volume of the cargo cannot exceed the total volume of the container.

**ALGORITHM:**

1. Assume the size of the container as 5mx3mx5m
2. Accept the number of cargo boxes as ‘n’
3. Initialize arrays len[n], brd[n], ht[n], and pt[n] for length, breadth, height and priority of boxes.
4. Initialize box[n][8][3] to store coordinates of placed boxes
5. In a loop, from i=0 to n-1,
   1. Accept the dimensions in ‘x’, ‘y’, ‘z’
      1. Store max(x,y,z) in len[i] and min(x,y,z) in brd[i], and the other dimension in ht[i]
      2. Condition- len[i]<5 and brd[i]< 3. If this condition is not satisfied, exit.
   2. Accept priority in pt[i]
   3. Compute total volume in ‘volume’. If volume>75, INVALID- exit.
6. Sort all data according to priority pt[], such that least priority is in the beginning of the array
7. Among boxes with same priority, arrange in descending order of brd[]
8. In a loop from i=0 to n,
   1. Chose depth of a layer as depth=brd[i]
   2. Let ‘layvol’ be the total volume of the layer and ‘volfill’ be total volume occupied in the layer
   3. Place box i such that it occupies maximum space along the y axis
   4. After placing the box, store its coordinates in 3-dimensional array box[i][8][3], and add its volume to volfill
   5. Assuming a box is not being placed, push it to the end of the list of boxes with same priority and continue with the next box
   6. If one horizontal layer is complete, go to box[][][] and extract available space on z axis. Repeat from 8(c)
   7. If volfill>=layvol, then create another layer in front and repeat from 8(a).
   8. Continue till i>=n, i.e. all boxes are placed.
9. Print the array box[][][] in nested loops, from box number i+1, I from 0 to n-1.

**ALGORITHM DESCRIPTION:**

This algorithm is very close to the packing procedure proposed by David Pisinger in a paper published by European Journal of Operational Research in 2002, titled “Heuristics for the container loading problem”. The layer building approach has been adopted as described by the paper, and the rule applied for choosing layer depth is maximum space occupation along y axis. The algorithm proposed in this project is greedy to some extent, in the sense that the solution varies a lot on the type of cargo dimensions entered, the number of cargo boxes to be packed and it may not always guarantee the best result.

**DRAWBACKS/ ASSUMTIONS TO ALGORITHM EFFICIENCY:**

* This algorithm works efficiently for a weakly heterogenous or homogenous set of cargos, i.e. most cargo boxes have similar dimensions, or for a small number of cargo boxes with varying dimensions.
* The algorithm prioritises the priority for the arrangement in a greedy way. The solution derived will satisfy the conditions- the boxes to be placed in a way such that higher priority boxes are easily accessible, the packing is done. However, even though the solution is one of the right solutions, it may not always be the best.
* For a small number of cargo, because the accessibility is prioritised above everything else, there may seem to be a waste of space.

**CODE:**

//to load a container of dimensions 5x3x5 cubic metres

import java.io.\*;

class clp

{int n;

int len[],brd[],ht[],pt[];

int box[][][];

int spx,spy,spz;

clp()

{//default constructor

}

clp(int n)

{this.n=n;

len=new int[n];

brd=new int[n];

ht=new int[n];

pt=new int[n];

spx=0;

spy=0;

spz=0;

box=new int[n][8][3];

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

{for(int j=0;j<8;j++)

{for(int k=0;k<3;k++)

box[i][j][k]=0;

}

}

}

public void accept()throws IOException

{BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

int i=0;

int volume=0;

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

{System.out.println("Enter length, breadth and height");

int x=Integer.parseInt(br.readLine());

int y=Integer.parseInt(br.readLine());

int z=Integer.parseInt(br.readLine());

int sum=x+y+z;

x=(int)Math.max((int)Math.max(x,y),(int)Math.max(x,z));

y=(int)Math.min((int)Math.min(x,y),(int)Math.min(x,z));

z=sum-(x+y);

if(x>5 || y>3)//INVALID INPUT

{System.out.println("Invalid dimension");

System.exit(0);

}

len[i]=x;

brd[i]=y;

ht[i]=z;

System.out.println("Enter priority");

pt[i]=Integer.parseInt(br.readLine());

volume+=(x\*y\*z);

if(volume>75)//INVALID INPUT

{System.out.println("Total volume exceeds the container volume.");

System.exit(0);

}

}

}//end accept()

//ARRANGING CARGO

public void sortpt()

{for(int i=0;i<n-1;i++)

{int in=i;

for(int j=i+1;j<n;j++)

{if(pt[j]<pt[in])

in=j;

}

int t=pt[i];

pt[i]=pt[in];

pt[in]=t;

t=len[i];

len[i]=len[in];

len[in]=t;

t=brd[i];

brd[i]=brd[in];

brd[in]=t;

t=ht[i];

ht[i]=ht[in];

ht[in]=t;

}

}

//sorting according to brd

public void sortbrd()

{for(int i=0,x=0;i<n-1;i++)

{if(pt[i]!=pt[i+1])

{ for(int j=x;j<=i;j++)

{ int in=j;

for(int z=j+1;z<=i;z++)

{if(brd[z]>brd[in])

in=z;

}

int t=pt[j];

pt[j]=pt[in];

pt[in]=t;

t=len[j];

len[j]=len[in];

len[in]=t;

t=brd[j];

brd[j]=brd[in];

brd[in]=t;

t=ht[j];

ht[j]=ht[in];

ht[in]=t;

}

x=i+1;

}

}

}

public void actarr()//actually arranging

{int j=0, depth=0;

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

{depth=brd[i];

int layvol=depth\*5\*3;

int volfill=0;

while(volfill<layvol&&j<n&&spx<3)

{int ret=0;

if(len[j]<depth)

ret=place(1,j,i);

else if(brd[i]<depth)

ret=place(2,j,i);

else

ret=place(3,j,i);

if(ret==1)//box number j has been placed

{volfill+=ht[j]\*brd[j]\*len[j];

j+=1;

}

i=j-1;

}

if(spx>=3)

{ spz=spz+box[j-1][4][2]-box[j-1][0][2];

spx=0;

}

if(volfill>=layvol)

{spz=0;

spx=0;

spy=spy+depth;

}

}

}

public int place(int a, int j,int i)//place box number j

{if(a==1)//placing length wise

{ if((5-spz)>ht[j])

{box[j][0][0]=spx;

box[j][0][1]=spy;

box[j][0][2]=spz;

box[j][1][0]=spx+brd[j];

box[j][1][1]=spy;

box[j][1][2]=spz;

box[j][2][0]=spx+brd[j];

box[j][2][1]=spy+len[j];

box[j][2][2]=spz;

box[j][3][0]=spx;

box[j][3][1]=spy+len[j];

box[j][3][2]=spz;

box[j][4][0]=spx;

box[j][4][1]=spy;

box[j][4][2]=spz+ht[j];

box[j][5][0]=spx+brd[j];

box[j][5][1]=spy;

box[j][5][2]=spz+ht[j];

box[j][6][0]=spx+brd[j];

box[j][6][1]=spy+len[j];

box[j][6][2]=spz+ht[j];

box[j][7][0]=spx;

box[j][7][1]=spy+len[j];

box[j][7][2]=spz+ht[j];

spx=spx+brd[j];

return 1;

}

else if((5-spz)>brd[j])

{box[j][0][0]=spx;

box[j][0][1]=spy;

box[j][0][2]=spz;

box[j][1][0]=spx+ht[j];

box[j][1][1]=spy;

box[j][1][2]=spz;

box[j][2][0]=spx+ht[j];

box[j][2][1]=spy+len[j];

box[j][2][2]=spz;

box[j][3][0]=spx;

box[j][3][1]=spy+len[j];

box[j][3][2]=spz;

box[j][4][0]=spx;

box[j][4][1]=spy;

box[j][4][2]=spz+brd[j];

box[j][5][0]=spx+ht[j];

box[j][5][1]=spy;

box[j][5][2]=spz+brd[j];

box[j][6][0]=spx+ht[j];

box[j][6][1]=spy+len[j];

box[j][6][2]=spz+brd[j];

box[j][7][0]=spx;

box[j][7][1]=spy+len[j];

box[j][7][2]=spz+brd[j];

spx=spx+ht[j];

return 1;

}

else

{int t=j;

while(t<n-1)

{if(pt[t]!=pt[t+1])

break;

t++;

}

int t1=pt[j];

int t2=ht[j];

int t3=brd[j];

int t4=len[j];

int x=j;

for(x=j;x<t;x++)

{pt[x+1]=pt[x];

ht[x+1]=ht[x];

brd[x+1]=brd[x];

len[x+1]=len[x];

}

pt[x]=t1;

ht[x]=t2;

brd[x]=t3;

len[x]=t4;

return 0;

}

}//end a==1

if(a==2)

{ if((5-spz)>ht[j])

{box[j][0][0]=spx;

box[j][0][1]=spy;

box[j][0][2]=spz;

box[j][1][0]=spx+len[j];

box[j][1][1]=spy;

box[j][1][2]=spz;

box[j][2][0]=spx+len[j];

box[j][2][1]=spy+brd[j];

box[j][2][2]=spz;

box[j][3][0]=spx;

box[j][3][1]=spy+brd[j];

box[j][3][2]=spz;

box[j][4][0]=spx;

box[j][4][1]=spy;

box[j][4][2]=spz+ht[j];

box[j][5][0]=spx+len[j];

box[j][5][1]=spy;

box[j][5][2]=spz+ht[j];

box[j][6][0]=spx+len[j];

box[j][6][1]=spy+brd[j];

box[j][6][2]=spz+ht[j];

box[j][7][0]=spx;

box[j][7][1]=spy+brd[j];

box[j][7][2]=spz+ht[j];

spx=spx+len[j];

return 1;

}

else if((5-spz)>len[j])

{box[j][0][0]=spx;

box[j][0][1]=spy;

box[j][0][2]=spz;

box[j][1][0]=spx+ht[j];

box[j][1][1]=spy;

box[j][1][2]=spz;

box[j][2][0]=spx+ht[j];

box[j][2][1]=spy+brd[j];

box[j][2][2]=spz;

box[j][3][0]=spx;

box[j][3][1]=spy+brd[j];

box[j][3][2]=spz;

box[j][4][0]=spx;

box[j][4][1]=spy;

box[j][4][2]=spz+len[j];

box[j][5][0]=spx+ht[j];

box[j][5][1]=spy;

box[j][5][2]=spz+len[j];

box[j][6][0]=spx+ht[j];

box[j][6][1]=spy+brd[j];

box[j][6][2]=spz+len[j];

box[j][7][0]=spx;

box[j][7][1]=spy+brd[j];

box[j][7][2]=spz+len[j];

spx=spx+ht[j];

return 1;

}

else//push to end of same priority

{int t=j;

while(t<n-1)

{if(pt[t]!=pt[t+1])

break;

t++;

}

int t1=pt[j];

int t2=ht[j];

int t3=brd[j];

int t4=len[j];

int x;

for(x=j;x<t;x++)

{pt[x+1]=pt[x];

ht[x+1]=ht[x];

brd[x+1]=brd[x];

len[x+1]=len[x];

}

pt[x]=t1;

ht[x]=t2;

brd[x]=t3;

len[x]=t4;

}

}//end a==2

if(a==3)//a==3

{ if((5-spz)>brd[j])

{box[j][0][0]=spx;

box[j][0][1]=spy;

box[j][0][2]=spz;

box[j][1][0]=spx+len[j];

box[j][1][1]=spy;

box[j][1][2]=spz;

box[j][2][0]=spx+len[j];

box[j][2][1]=spy+ht[j];

box[j][2][2]=spz;

box[j][3][0]=spx;

box[j][3][1]=spy+ht[j];

box[j][3][2]=spz;

box[j][4][0]=spx;

box[j][4][1]=spy;

box[j][4][2]=spz+brd[j];

box[j][5][0]=spx+len[j];

box[j][5][1]=spy;

box[j][5][2]=spz+brd[j];

box[j][6][0]=spx+len[j];

box[j][6][1]=spy+ht[j];

box[j][6][2]=spz+brd[j];

box[j][7][0]=spx;

box[j][7][1]=spy+ht[j];

box[j][7][2]=spz+brd[j];

spx=spx+len[j];

return 1;

}

else if((5-spz)>len[j])

{box[j][0][0]=spx;

box[j][0][1]=spy;

box[j][0][2]=spz;

box[j][1][0]=spx+brd[j];

box[j][1][1]=spy;

box[j][1][2]=spz;

box[j][2][0]=spx+brd[j];

box[j][2][1]=spy+ht[j];

box[j][2][2]=spz;

box[j][3][0]=spx;

box[j][3][1]=spy+ht[j];

box[j][3][2]=spz;

box[j][4][0]=spx;

box[j][4][1]=spy;

box[j][4][2]=spz+len[j];

box[j][5][0]=spx+brd[j];

box[j][5][1]=spy;

box[j][5][2]=spz+len[j];

box[j][6][0]=spx+brd[j];

box[j][6][1]=spy+ht[j];

box[j][6][2]=spz+len[j];

box[j][7][0]=spx;

box[j][7][1]=spy+ht[j];

box[j][7][2]=spz+len[j];

spx=spx+brd[j];

return 1;

}

else//push to end of same priority

{int t=j;

while(t<n-1)

{if(pt[t]!=pt[t+1])

break;

t++;

}

int t1=pt[j];

int t2=ht[j];

int t3=brd[j];

int t4=len[j];

int x;

for(x=j;x<t;x++)

{pt[x+1]=pt[x];

ht[x+1]=ht[x];

brd[x+1]=brd[x];

len[x+1]=len[x];

}

pt[x]=t1;

ht[x]=t2;

brd[x]=t3;

len[x]=t4;

}

}//end a==3

return 0;

}

public void print()

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

{System.out.println("Box number "+(i+1));

for(int j=0;j<8;j++)

{for(int k=0;k<3;k++)

{System.out.print(box[i][j][k]+" ");

}

System.out.println();

}

}

}

public static void main()throws IOException

{BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the number of cargo boxes");

int n=Integer.parseInt(br.readLine());

clp obj=new clp(n);

obj.accept();

obj.sortpt();

obj.sortbrd();

obj.actarr();

obj.print();

}

}

**SAMPLE INPUT/OUTPUT:**

Enter the number of cargo boxes

7

Enter length, breadth and height

1

1

1

Enter priority

1

Enter length, breadth and height

1

1

1

Enter priority

2

Enter length, breadth and height

1

1

1

Enter priority

3

Enter length, breadth and height

1

1

1

Enter priority

4

Enter length, breadth and height

1

1

1

Enter priority

5

Enter length, breadth and height

2

1

1

Enter priority

2

Enter length, breadth and height

2

1

2

Enter priority

3

Box number 1

0 0 0

1 0 0

1 1 0

0 1 0

0 0 1

1 0 1

1 1 1

0 1 1

Box number 2

1 0 0

2 0 0

2 1 0

1 1 0

1 0 1

2 0 1

2 1 1

1 1 1

Box number 3

2 0 0

4 0 0

4 1 0

2 1 0

2 0 1

4 0 1

4 1 1

2 1 1

Box number 4

0 0 1

1 0 1

1 1 1

0 1 1

0 0 2

1 0 2

1 1 2

0 1 2

Box number 5

1 0 1

3 0 1

3 2 1

1 2 1

1 0 2

3 0 2

3 2 2

1 2 2

Box number 6

0 0 2

1 0 2

1 1 2

0 1 2

0 0 3

1 0 3

1 1 3

0 1 3

Box number 7

1 0 2

2 0 2

2 1 2

1 1 2

1 0 3

2 0 3

2 1 3

1 1 3

**ACKNOWLEDGEMENTS:**

This project has been a humble attempt on our part to solve one of the most challenging questions researchers face while solving real life situations- optimizing and considering all forms of criteria’s. We would like to thank our Professor Bhoominathan P for giving us this opportunity and providing his valuable guidance in solving the problem. We would also like thank the authors whose research papers guided us in understanding the problem better, and everyone else who helped us with it.

Thank you very much.