Solución del problema planteado en el proyecto#1

Planteamiento del problema:

Se tienen N imágenes, que tienen como característica un ancho, un alto y un carácter ASCII, que servirá para identificar la imagen dentro de la matriz(un arreglo de arreglos), se requiere que aplicando técnicas de backtracking se encuentre la mejor forma de organizar las imágenes dentro de la matriz de tal manera que el espacio usado sea lo mas reducido posible y que en el se encuentren todas las imágenes dadas.

Análisis del problema:

Este tipo de problema requiere que se implemente el esquema de backtracking de la solución optima con el criterio de menor dimensión.

Por lo tanto es necesario evaluar todas las soluciones parciales(las distintas formas en la que se oueden organizar todas imágenes) para tomar la que cumpla con la condición de ser la mejor según el criterio mencionado anteriormente.

Es decir se tienen N imágenes y lo que se necesita es encontrar un rectángulo donde se puedan insertar todas las imágenes y cuyas dimensiones sean las minimas.

Explicación de la solución

Mi forma de organizar las imágenes es en una matriz de 21 x 21 porque 1<= ancho<=3 1,<= alto<=3, 1<= N<=7,luego tengo dos alternativas para organizarlas, la primera es que luego que coloque la primera pruebo con insertar la segunda a la derecha de esta luego

saco la imagen que acabo de insertar y pruebo insertándola debajo de la anterior, apoyándome de la recursividad implemento esto haciéndolo para N imágenes, para lograr esto cree dos acciones INSERTAR_IMAGEN e insertar_IMAGEN_DEBAJO, acciones que me “actualizaran” las dimensiones de mi matriz a medida que iba insertando las imágenes en ella, esto me permitió proponer un caso base para mi algoritmo, voy comparando cada solución por sus dimesiones para obtener la menor de ellas.

Cree una clase imagen que permitió tener las características de estas en un objeto(ancho, alto,carácter), asi cree además un arreglo de objetos de esta clase para permutar el orden en que las iba insertando y asi probar todas las formas de organizarlas en mi matriz.

El resultado comparo cada solución y elijo la que contega las minimas dimensiones.

Limitaciones del sistema

Como comprueba todas las soluciones a medida que incremeta el N se hace lento

Algoritmo de la solucion en pseudoformal

```
class imagen

public
character  carácter;
integer  ancho;
integer alto;
public

public procedure rotar()
```

```
integer aux;
aux=ancho;
ancho=alto;
alto=aux;
end procedure

end class

array   imagenes[32] of imagen;

array imagenes_usadas[32] of Boolean;

Boolean encontro=false;

Boolean solucion=false;

integer posicionj=0;

integer posicioni=0;

ineteger menor=1000;

intege rN=0;

integer mejori=0;

integer mejorj=0;

integerpixeles=0;

array matriz [441][441] of carácter;

array mejor_solucion[441][441] of carácter;
```

```
procedure suma_dimensiones()

for i=0 to N do
pixeles=pixeles+(imagenes[i].ancho*imagenes[i].alto);
endfor

procedure mostrar()

write(mejorj);
 write(mejori);
 for i=0 to i<mejori do
 for j=0 to j<mejorj do
write(mejor_solucion[i][j]);

endprocedure



procedure guardar_solucion(int i,int j)

if i*j<menor

menor=i*j;
mejori=i;
mejorj=j;
for int i=0 to i<mejori do

for j=0 to j<mejorj do

mejor_solucion[i][j]=matriz[i][j];

 endfor
 endfor

function alternativa_valida(int i):boolean
```

```
If !imagenes_usadas[i]
return true;
endif
return false;
endfunction

procedure marcar_opcion(int i)

imagenes_usadas[i]=true;

endprocedure

procedure desmarcar_opcion(int i)

imagenes_usadas[i]=false;
endprocedure

function insertadas_todas():boolean

for i=0;to i<N do
 if!imagenes_usadas[i]
return false;
endif
endfor
return true;
endfunctiion

 procedure actualizari()
integer i=0;

integerj=0;

integer acum=0;
```

```
integer acum2=0;

boolean caracter=false;

while i!=21 do
while j!=21 && !caracter do
if matriz[i][j]!='#'

acum++;
caracter=true;
endif

j=j+1;
 endwhile
if acum!=0
acum2++;
endif
i=i+1;
j=0;
acum=0;
caracter=false;
endwhile
posicioni=acum2;
endprocedure

procedure actualizarj()

integer i=0;
integerj=0;
integer acum=0;
integer caracteres=0;
boolean caracter=false;
while i!=21 do
while j!=21 && !caracter do
if matriz[j][i]!='#'
```

```
acum++;
caracter=true;
endif
j++;
 endwhile

if acum!=0

caracteres=caracteres+1;
endif
acum=0;
j=0;
i=i+1;
caracter=false;
endwhile;
posicionj=caracteres;
endprocedure
procedure  inicializar()

for i=0 to i<21 do
for j=0 to j<21 do

matriz[i][j]='#';
mejor_solucion[i][j]='#';
 endfor
endfor
for j=0 to j<32 do

imagenes[j]=new imagen();
imagenes_usadas[j]=false;
endfor
endprocedure

procedure eliminar_imagen(imagen a_insertar)
```

```
integeri=0;
integer j=0;
integer aux1=0;
integer aux2=0;
boolean listo=false;
while i!=21 && !listo do
while j!=21 &&!listo do

if matriz[i][j]==a_insertar.caracter

listo=true;

integer fila=i;
 integer columna=j;
while aux1!=a_insertar.alto do
while aux2!=a_insertar.ancho do
matriz[fila][columna]='#';
columna=columna+1;
aux2=aux2+1;
endwhile
columna=j;
fila=fila+1;
aux2=0;
aux1=aux1+1
 endif
j=j+1;
endwhile
i=i+1;
j=0;
endwhile
actualizari();
actualizarj();


endprocedure
```

```
function hay_espacio(integer ancho,integer alto,integer j,integer i=0)

integer aux1=j;
integer ancho2=0;
integer alto2=0;
integer aux2=i;
while alto2!=alto do

 while ancho2!=ancho

 if matriz[aux2][aux1]!='#'


 return false;

 endif
 aux1++;
 ancho2++;
 endwhile
 aux1=j;
ancho2=0;
aux2=aux2+1;
 alto2=alto2+1;
 endwhile
 return true;
endprocedure

procedure einsertar_imagen_debajo(imagen a_insertar)

boolean encajo_pieza=false;
integer i=0;
integer parada=0;
integer auxiliar=posicioni;
integer filas=0;
```

```
integer columnas=0;
integer auxiliar_columnas=0;

while filas!=posicioni && !encajo_pieza do

while columnas!=posicionj && !encajo_pieza do


if hay_espacio(a_insertar.ancho,a_insertar.alto,columnas,filas)
auxiliar=filas;
auxiliar_columnas=columnas;
encajo_pieza=true;
 endif

columnas=columnas+1;
endwhiile

columnas=0;
filas++;
endwhile

integer respaldo=auxiliar:
integer columnas;
while i!=a_insertar.alto do

while parada!=a_insertar.ancho do

matriz[auxiliar][auxiliar_columnas]=a_insertar.caracter;

parada++;
auxiliar_columnas++;
endwhile

auxiliar_columnas=respaldo;
auxiliar++;
```

```
parada=0;
i=i+1;
endwhile
actualizari();
actualizarj();
endprocedure

procedure insertar_imagen(imagen a_insertar)

boolean encajo=false;
integer i=0;
integer relleno_alto=0
integer aux=0;
while i!=21 && !encajo
if hay_espacio(a_insertar.ancho,a_insertar.alto,i)

integer relleno_ancho=i;

while relleno_alto!=a_insertar.alto
while aux!=a_insertar.ancho
matriz[relleno_alto][relleno_ancho]=a_insertar.caracter;
relleno_ancho=relleno_ancho+1;
aux=aux+1;
 endwhile
 relleno_ancho=i;
 aux=0;
 relleno_alto=relleno_alto+1;
endwhile
 actualizari();
actualizarj();
encajo=true;
endif

i=i+1;
endwhile
```

```
endprocedure

procedure backtracking()

if insertadas_todas()

If posicioni*posicionj==pixeles

solucion=true;
endif
if posicioni*posicionj<menor
 guardar_solucion(posicioni,posicionj);
             endif


         else

        integer i=0;
       while i!=N && !solucion do

     If alternativa_valida(i)
 marcar_opcion(i);
insertar_imagen(imagenes[i]);
if posicioni*posicionj<menor
backtracking();
endif

eliminar_imagen(imagenes[i]);

if imagenes[i].ancho!=imagenes[i].alto

imagenes[i].rotar();
insertar_imagen(imagenes[i]);
if posicioni*posicionj<menor
```

```
backtracking();
 endif
eliminar_imagen(imagenes[i]);
 endif
if imagenes[i].ancho!=imagenes[i].alto

imagenes[i].rotar();
endif
  insertar_imagen_debajo(imagenes[i]);
ifposicioni*posicionj<menor

backtracking();
 endif
eliminar_imagen(imagenes[i]);

if imagenes[i].ancho!=imagenes[i].alto

imagenes[i].rotar();
insertar_imagen_debajo(imagenes[i]);
if posicioni*posicionj<menor

backtracking();
 endif
eliminar_imagen(imagenes[i]);
imagenes[i].rotar();
      endif
desmarcar_opcion(i);

endif
i++;
endwhile
endif
endprocedure

procedure main()
```

```
Inicializar();
read(N);
for i=0 to i<N do

read(imagenes[i].ancho);
read(imagenes[i].alto);
read(imagenes[i].carácter);
endfor
suma_dimensiones();
for i=0 to i<N do

marcar_opcion(i);
 insertar_imagen(imagenes[i]);
backtracking();
 eliminar_imagen(imagenes[i]);

if imagenes[i].ancho!=imagenes[i].alto

 imagenes[i].rotar();
 insertar_imagen(imagenes[i]);
 backtracking();
 eliminar_imagen(imagenes[i]);
 imagenes[i].rotar();
 endif

 desmarcar_opcion(i);
endfor
mostrar;
endprocedure
```