j→0	1	2	3
1	1	Ø	1
Ø	1	1	1
Ø	1	Ø	1
2	1	0	3

[2,1]

Ø = Paredes
1 = Libre
2 = entrada
3 = salida
4 = visitado.

i = filas [2,1]

j = columnas.

Para j++ arriba

if (tabla[-i] = 1 Ø)

Para moverse

Ratón: [3][Ø]

Tabla [i][j]Ø tabla[i]<4

Succes = False

Para (cada candidato en c) {

si (succes == true))

{return true}

si es factible {

si el candidato j = Ø

{

si el candidato j = 2

{

si el candidato j = 4

{ si el candidato [i][j]

si i > 0 y j > Ø

si i

2,3

3 2

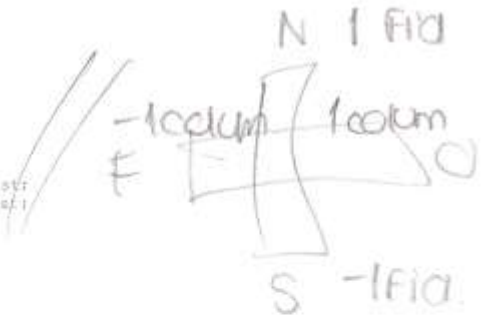
R.

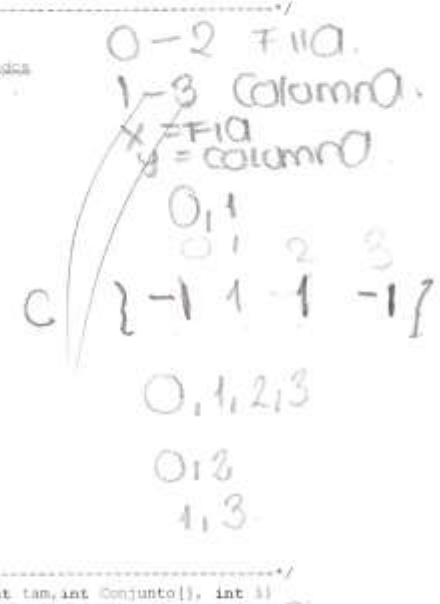
Insertion+

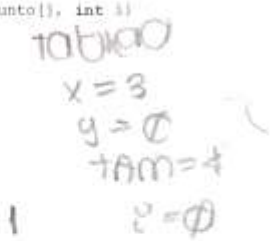
```
}
    printf("El valor se encuentra.\n\t");
    printf("***%d****\n",res);
}
else{
    printf("El valor NO se encuentra.\n");
}

return 0;
}
```

1 2.5

```
78      else
79          {
80              this -> last -> next = n;
81              n -> prev = this -> last;
82              this -> last = n;
83              this -> last -> next = this -> first;
84              this -> first -> prev = this -> last;
85          }
86
87      return TRUE;
88
89  }
90
91
92
93  /*-------------------------------------------------------------------------*/
94  Bool DLL_RemoveBack (DLL *this, int *x, int *y)
95  {
96      if (this -> first != this -> last) //2 o + Nodos
97      {
98          *y = this -> last -> Coord_y;
99          *x = this -> last -> Coord_x;
100
101         Node *tmp = this -> last -> prev;
102
103         deleteNode (this -> last);
104
105         this -> last = tmp;
106         this -> last -> next = this -> first;
107         this -> first -> prev = this -> last;
108
109         return TRUE;
110     }
111     else if (this -> empty == FALSE) // Un Nodo
112     {
113         *x = this -> last -> Coord_x;
114         *y = this -> last -> Coord_y;
115         deleteNode (this -> last);
116         this -> first = this -> last = NULL;
117         this -> empty = TRUE;
118         return TRUE;
119     }
120     else (return FALSE;)
121 }
122
123 /*-------------------------------------------------------------------------*/
124 Bool Factible(int tablero[][TAM],int x, int y, int tam,int Conjunto[], int i)
125 {
126     printf("\nI en factible: %d", i); //control
127     if(
128 {
129         printf("\nI igual: %d", i);
130         if((x+Conjunto[i] >= 0) & (x + Conjunto[i] <= tam))
131         {
132             if((y >= 0) & (y <= tam))
133             {
134                 int j;
135                 j = tablero[x+Conjunto[i]][y];
136
137                 if(j != 0)
138                 {
139                     if(j != 3)
140                     {
141                         if( j != 4)
142                         {
143                             return TRUE;
144                         }
145                         else (return FALSE;)
146                     }
147                     else (return FALSE;)
148                 }
149                 else (return FALSE;) //tercero
150             } //segundo
151             else(return FALSE;)
152         }
```



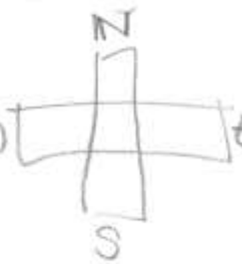




Laberinto (tabla0, entrada, salida, pila)
{
Si (entrada >∅ y entrada<4) {
Si (salida>∅ y salida<4) {
se crea la lista (miPila)
Bool res = bt(   )
if (res =TRUE) {
a- see (   ) ; }
else { // print (No hubo solución) }
}
}

Driver Program
entrada [2][3]
salida [1][1]

candidatos
Norte
Sur
Este
oeste



Norte y sur se mueve en colum
oeste y este se mueve en filas.

entrada = (x, y).

Norte = tabla [i][j++]
leos ?
Suer = tabla [i][j--]
leos ?
Oeste = tatbla [i--][j]
conjunto
Este = [tabla[i++][j]] {Norte, Este, Sur, oeste}

o

→ Conjunto

Conjunto[4] - {j++, i++, j--, i--}.

Conjunto[4] - {1, 1, -1, -1}.

bt(

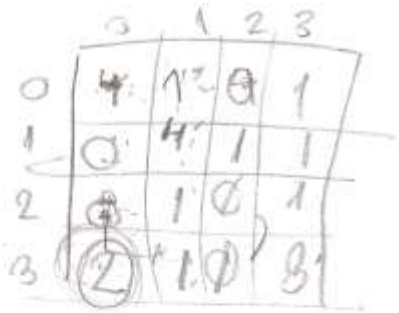
{

succes = false

para (i=0; i<4; i++) {

si (succes == true) { return true; }

si (tabla[ ][TAM], conjunto[ ], x, y, tam)



0 = paredes
1 = libre

Ratón = entrada

↳ entra a

Nodo * first

Nodo * last

[3,0]
x y
[2,0]

insert

Nodo

factible(tablero[ ][ ], x, y, tam, conjunto, i)

Salcoción (tablero[ ][ ], x, y)

num1 = 2
num2 = 3

si 3 == 0
agregar a
si 3<0
2-1, 3+1
2+1, 3-1

FIFO

if (|2|2| = 0)
num = = 1

11 - 2

12
12 / 2 = 6
6 - 2 = 4

6

7 4
3 2
4 1

-11
9

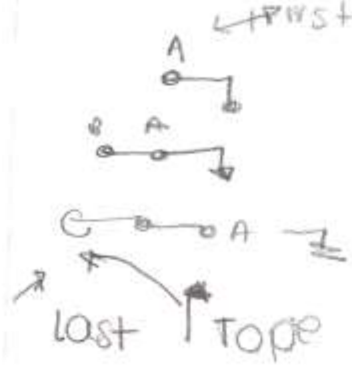
Lista
Pila
Cola

pila push

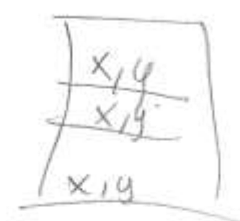
Definir hacia donde se inicia

Lista enlazada.

Insert → RemoveBack
InsertBack → Remove



Push
Pop
Peek



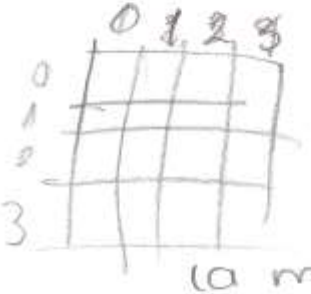
bt (C, P) → (S, Bool)

C = {  }

Para (todas las candidatas) {

≈

si (es factible) {



Si se sale
−1

Si se sale
1

la marcamos como visitada.
la metemos a la pila.

Si (es solución) {

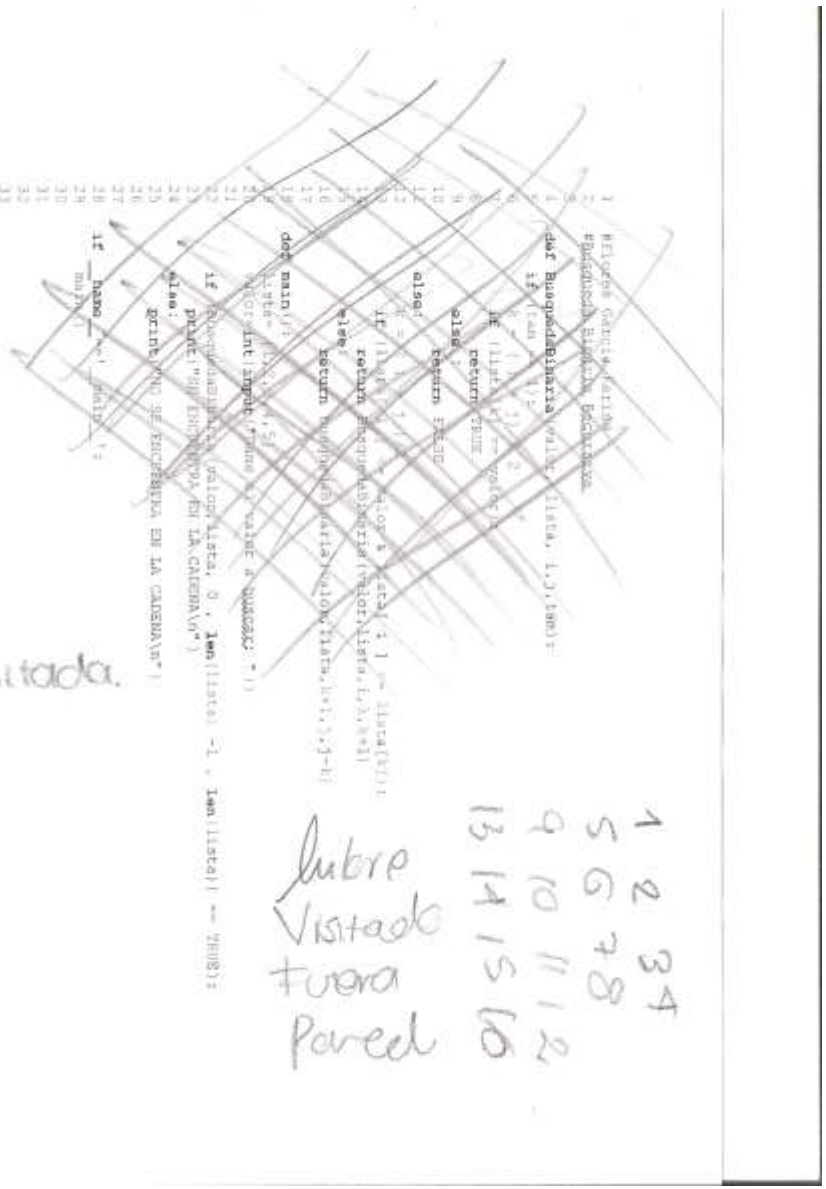
éxito = TRUE
success

Devuelve = TRUE

}

Si no hubo solución {

existe = bt (C, P)



→ BACKTRACKING

```
bt (C[], tablero[ ][ ], x, y, tam                    )
      success = False
  para (i=0; i<tam, i++)
    Si (Factible (tablero[ ][ ], x, y, tam, conjunto[i], i) ==
          TRUE )
    — Si i==1 OR i==3        NORTE Y SUR
        tablero [x + conjunto[i]] [y] = = 4
      Push ( tablero [x+conjunto[i]] [y] )
    — Sino Si i==2 OR i=4

      Push (tablero [x] [y + conjunto[i]] )
Registro → tablero [x] [y+conjunto [i] = 4

     * PARA Solución *
      Si (i==1 OR i==3) {
          si (solución (tablero [x+conjunto[i]] [y] = = TRUE )
             {
              success = TRUE
               Return = TRUE
            }
      elso if (i== 2 OR i = = 4)
          si (solución (tablero [x] [y+conjunto[i]] = = TRUE)
             success TRUE
```

tablero[x][y] = 4 → tablero[2][0] = 4
DLL insert (this, x, y) → insert (2, 0)

```
        if (EsSolucion(tablero,x,y)==TRUE) // CASO BASE
        {
            printf("\nentra caso solucion"); // control
            success = TRUE;
            return TRUE;
        }
        else
        {
            printf("\n\nEntra al else de fac");
            
            printf("\n\nAumento de i: %d", i);
            success = BackTracking(this, Conjunto, tablero, x, y, tam,i);
            if(success == FALSE)
            {
                DLL_RemoveBack (this,&X,&Y);
                tablero[x][y] = tablero[X][Y];
            }
        }
    }
    else //i es 2 o 4
    {
        y = y + Conjunto[i];
        tablero[x][y] = 4;
        DLL_Insert (this, x, y);

        if (EsSolucion(tablero,x,y)==TRUE) // CASO BASE
        {
            success = TRUE;
            return TRUE;
        }
        else
        {
            printf("\n\tAumento de i2: %d", i);
            success = BackTracking(this, Conjunto, tablero, x, y, tam, i);

            if(success == FALSE)
            {
                DLL_RemoveBack (this,&X,&Y);
                tablero[x][y] = tablero[X][Y];
            }
        }
      } //else
    } // if factible
  } //if del i
  return success;
}

/*-------------------------------------------------------------------------*/
Bool laberinto(DLL* this,int Conjunto[],int tablero[][TAM], int x, int y, int tam)
{
    printf("Entra a parte de laberinto");

    printf("\n x es : %d ", x);
    printf("\nTamaño es %d", tam);
    if((x >= 0) & (x <= tam))
    {
        printf("Y es: %d", y);
        if((y >= 0) & (y <= tam))
        {
            // Crea el conjunto solucion
            printf("\nEntra al if");
            Bool res;


            res = BackTracking(this, Conjunto,tablero,x,y,tam,i);
            printf("\nEntra a back ");
            if(res == TRUE)
            {
                return TRUE;
            }
            else {return FALSE;}

        }
```

si // i = 1 i = 3 columna
y = y + Conjunto[i] = 0 + 1
y = 1
tablero[3][1] = 4
Inserta (3, 1)
tablero[x][y] = 4 → tablero, 3, 1

tablero[x, y] = 4

Conjunto { 1, 1, -1, -1 }
x = 3 y = 0

i = 0

```
155     else
156     {
157         //return FALSE;
158         printf("\nI par: %d", i);
159         if((x >= 0) & (x <= tam))
160         {
161             if((y+Conjunto[i] >= 0) & (y+Conjunto[i] <= tam))
162             {
163                 int j;
164                 j = tablero[x][y+Conjunto[i]];
165
166                 if(j != 0)
167                 {
168                     if(j != 3)
169                     {
170                         if( j != 4)
171                         {
172                             return TRUE;
173                         }
174                         else {return FALSE;}
175                     }
176                     else {return FALSE;}
177                 }
178                 else {return FALSE;} //tercero
179             } //segundo
180             else{return FALSE;}
181         }
182         else{return FALSE;} // primer if
183
184     }
185 }
186
187
188 /*-------------------------------------------------------------------------*/
189
190 Bool EsSolucion(int tablero[][TAM],int x, int y)
191 {
192     int j = tablero[x][y];
193     int salida = tablero[3][3];
194     printf("\n\nLa salida es : %d", salida);
195     if (j == salida)
196     {
197         return TRUE;
198     }
199     else{
200       return FALSE;
201     }
202 }
203
204 /*-------------------------------------------------------------------------*/
205 Bool BackTracking(DLL* this, int Conjunto[],int tablero[][TAM], int x, int y, int tam,
    int i)
206 {
207     int X, Y;
208     Bool success;
209     success = FALSE;
210
211     printf("\nEntra a recursivo %d", i);
212     printf("\nEntra a BackTracking\n");
213     printf("\nx: %d ",x);
214     printf("\ny: %d", y);
215
216 for (j=0 ; j<4)
217
218     if (i < tam)
219     {
220         if (success == TRUE){return TRUE;}
221
222         printf("\nEl valor de i: %d", i); //control
223         if(Factible(tablero,x,y,tam,Conjunto,j)==TRUE)
224         {
225             printf("\n\nCumple con las");
226 i=0     if(i==1 || i==3)
227             {
228                 x = x + Conjunto[i];
```

i == 1 || i == 3. x = 3 y = 0

1 -> x >= 0

j = tablero[x][y+conjunto[i]]
tablero[3][1] = 1

tablero =
x = 3 y = 1
j = tablero[3][1] = 1

1 x = 3 y = 0
i = 0
tam = 4

j = 1

tablero, 3, 0, 4, C, 0

tablero, 3, 0, 4, C, 1