Ampliación de la Programación

Envasadora

Backtracking

Envasadora: Backtracking

Resolución

En cada etapa k decidimos el envase en el que metemos el objeto k supuesto que ya tengamos envasados los k-1 objetos anteriores.

- **Test Solución**: Etapa *N* (cuando hayamos envasado todos los trabajos).
  - **Comprobaremos** entonces si hemos o no mejorado nuestra mejor solución (*Test Fracaso –No Fracaso-*, para la mejor solución).
- **Test Fracaso**: No podemos meter un objeto en un envase en el que no tengamos ya espacio.
- **Generación de Descendientes**: Cada objeto *k* podemos meterlo en uno de los envases utilizados, y también tenemos la posibilidad de **envasarlo en uno no utilizado** hasta el presente.

Envasadora: Backtracking

Resolución: Código

```
#include <stdio.h>
#define MAX 50
#define INFINITO 10000

typedef int vector[MAX];

typedef struct{
   int num;
   vector envases; //envases[i]= envase en el que metemos el objeto i
}Solucion;

void Entrada(int *N,int *C,vector pesos);
void Salida(Solucion Sol, int N);
void Envasadora(int k, int N, vector c,vector pesos,Solucion *Sol,Solucion *SolOptima);

void Envasadora2(int k,int N,vector c,vector pesos,Solucion *Sol,Solucion *SolOptima);

int maximo (int a, int b) {
   return ((a > b) ? a: b);
}
```

Envasadora: Backtracking

Resolución: Código

```
int main() {
   int N, C, i;
   Solucion Sol, SolOptima;
   vector pesos;   // pesos de los objetos
   vector c;       // contiene las capacidades de los envases

   Entrada(&N,&C,pesos);

   // Inicializaciones
   Sol.num = 0;
   SolOptima.num = INFINITO;
   // Al inicio todos los envases están vacíos: Capacidad C
   for(i=0; i<N; i++) c[i] = C;

   Envasadora(0,N,c,pesos,&Sol,&SolOptima);
   // Envasadora2(0,N,c,pesos,&Sol,&SolOptima);

   Salida(SolOptima,N);
   return 0;
}
```

Envasadora: Backtracking

Resolución: Código

```
void Entrada(int *N, int *C, vector pesos) {
   int i;
   printf("Introduzca el numero de objetos disponibles: "); scanf("%d",N);
   printf("Introduzca la capacidad de los envases: "); scanf("%d",C);
   printf("Introduzca el peso de los objetos :");
   for (i=0; i<*N; i++){
      printf("\nObjeto%d= ",i); scanf("%d",&pesos[i]);
      if (pesos[i] > *C){
         printf("El peso maximo es %d\n",*C); i--;
      }
   }
   printf("\n\n");
}

void Salida(Solucion Sol, int N) {
   int i;
   printf("La solucion tiene %d envases y es:\n\n",Sol.num+1);
   for (i=0; i<N; i++)
      printf("Objeto %d --> Envase %d\n",i,Sol.envases[i]);
}
```

Envasadora: Backtracking

Resolución

```
// En la etapa k almacenamos el objeto k en un envase, supuesto que ya hemos
// almacenado los k-1 objetos anteriores
void Envasadora (int k,int N,vector c,vector pesos,Solucion *Sol,Solucion *SolOptima) {
  int i, utilizados;

  // si la solucion es peor no seguimos explorando
  if (Sol->num < SolOptima->num)
    if (k == N) *SolOptima = *Sol; // actualizamos
    else {
      utilizados = Sol->num;
      // generamos descendientes
      for (i=0; i<=utilizados; i++)
        // Cogemos el objeto k y lo metemos si podemos en el envase i
        if (pesos[k] <= c[i]){
          c[i] -= pesos[k];
          Sol->envases[k] = i;
          Envasadora(k+1,N,c,pesos,Sol,SolOptima);
          // Deshacemos
          c[i]+=pesos[k];
        }
```

Envasadora: Backtracking

Resolución

```
        // Siempre tenemos la posibilidad de meterlo en un envase nuevo
        Sol->num++;
        Sol->envases[k] = Sol->num;   //metemos el objeto k en el envase Sol->num
        c[Sol->num] -= pesos[k];
        Envasadora(k+1,N,c,pesos,Sol,SolOptima);
        // Deshacemos
        c[Sol->num] += pesos[k];
        Sol->num--;
      } // else
} // fin Envasadora
```

Envasadora: Backtracking

Resolución 2

```
// En la etapa k almacenamos el objeto k en un envase, supuesto que ya hemos
// almacenado los k-1 objetos anteriores
void Envasadora2 (int k,int N,vector c,vector pesos,Solucion *Sol,Solucion *SolOptima) {
   int i, utilizados;
   // si la solucion es peor no seguimos explorando
   if (Sol->num < SolOptima->num)
      if (k == N) *SolOptima = *Sol; // Actualizamos
      else {
         utilizados = Sol->num;
         // generamos descendientes
         for (i=0; i<=utilizados+1; i++)
            // Cogemos el objeto k y lo metemos si podemos en el envase i
            if (pesos[k] <= c[i]){
               c[i] -= pesos[k];
               Sol->envases[k] = i;
               Sol->num = maximo (Sol->num,i);
               Envasadora2(k+1,N,c,pesos,Sol,SolOptima);
               //Deshacemos
               c[i] += pesos[k];
            }
         // Deshacemos, para liberar el envase nuevo
         Sol->num = utilizados;
      } // else
} // fin Envasadora2
```