Ampliación de la Programación

2 Procesadores

Backtracking

2 Procesadores: Backtracking

Resolución

En cada etapa k el procesador en el que realizamos el trabajo k supuesto que ya tengamos decididos los k-1 trabajos anteriores.

- **Test Solución**: Etapa *N* (cuando hayamos tomado las decisiones para todos los trabajos).
  - **Comprobaremos** entonces si hemos o no mejorado nuestra mejor solución (*Test Fracaso –No Fracaso-*, para la mejor solución).
- **Test Fracaso**: No hay en la generación (utilizaremos un test fracaso como una mejora), siempre podemos hacer el trabajo *k* en uno de los dos procesadores.
- **Generación de Descendientes**: Para el trabajo *k* tenemos dos alternativas, hacerlo en un procesador o en otro.

2 Procesadores: Backtracking

Resolución

```
#include <stdio.h>
#define MAX 20
#define INFINITO 1000
#define TRUE 1
#define FALSE 0

  typedef struct {
    int tiempo;
    int trabajo [MAX]; // vector[i] es del procesador que hace el trabajo i
   } TipoSolucion;

  void ProcesadoresBack (int k, int tiempo0, int tiempo1,
                                TipoSolucion *sol, TipoSolucion *solOptima, int N, int Coste[2][MAX]);

  void Inicializa (TipoSolucion *solOptima);

  void Salida (TipoSolucion sol, int N, int Coste[2][MAX]);
  int maximo (int a, int b) {
    if (a < b) return b ;else return a;
  }
```

2 Procesadores: Backtracking

Resolución

```
int main () {

   int Coste [2][MAX]; // matriz de costes
   int N, i, j;
   TipoSolucion sol, solOptima;

   // entrada de datos
   printf("Introduzca el número de trabajos "); scanf("%d",&N);
   printf("\nDuración de los trabajos en el procesador 0 (separados por un espacio)\n");
   for (i=0; i<N; i++)
      scanf(" %d", &Coste[0][i]);
   printf("\nDuración de los trabajos en el procesador 1 (separados por un espacio)\n");
   for (i=0; i<N; i++)
      scanf(" %d", &Coste[1][i]);

   for (i= 0; i<2; i++) {
      for (j=0; j<N; j++)
         printf(" %4d",Coste[i][j]);
      printf("\n");
   }
```

2 Procesadores: Backtracking

Resolución

```
   Inicializa (&solOptima);

   ProcesadoresBack(0,0,0,&sol,&solOptima,N,Coste);

   Salida(solOptima,N,Coste);

   return 0;

} // main


   void Inicializa (TipoSolucion *solOptima) {
      solOptima->tiempo = INFINITO;
   }
```

2 Procesadores: Backtracking

Resolución

```
void ProcesadoresBack (int k, int tiempo0, int tiempo1,
                       TipoSolucion *sol, TipoSolucion *solOptima, int N, int Coste[2][MAX]) {

   if (k == N) { // Hemos terminado de construir la solución
     sol->tiempo = maximo (tiempo0,tiempo1);
     if (sol->tiempo < solOptima->tiempo) *solOptima = *sol;
   }
  else {
     // lo hace el procesador 0
     sol->trabajo[k] = 0;
     ProcesadoresBack (k+1,tiempo0+Coste[0][k],tiempo1,sol,solOptima,N,Coste);

     // lo hace el procesador 1
     sol->trabajo[k] = 1;
     ProcesadoresBack (k+1,tiempo0,tiempo1+Coste[1][k],sol,solOptima,N,Coste);
  }
}
```

2 Procesadores: Backtracking

Resolución

```
void Salida (TipoSolucion sol, int N, int Coste[2][MAX]) {
  int i;

  printf("\nTrabajos realizados por el procesador 0: ");
  for (i=0; i<N; i++)
    if (sol.trabajo[i] == 0) printf("%d  (+ %d ) ", i, Coste[0][i]);

  printf("\nTrabajos realizados por el procesador 1: ");
  for (i=0; i<N; i++)
    if (sol.trabajo[i] == 1) printf("%d  (+ %d ) ", i, Coste[1][i]);
  printf("\n\nEl coste de la solucion es: %d",sol.tiempo);

}
```

2 Procesadores: Backtracking

Resolución Mejorado

En cada etapa k decidimos el procesador en el que realizamos el trabajo k supuesto que ya tengamos asignados los k-1 trabajos anteriores.

- **Test Solución**: Etapa *N* (cuando hayamos tomado las decisiones para todos los trabajos).
  - **Comprobaremos** entonces si hemos o no mejorado nuestra mejor solución.
- **Test Fracaso**: Si la solución *S* que estamos construyendo "cuesta más" que la mejor que hemos construido hasta el momento, no seguimos construyendo *S*.
- **Generación de Descendientes**: Para el trabajo *k* tenemos dos alternativas, hacerlo en un procesador o en otro.

2 Procesadores: Backtracking

Resolución Mejorado

```
void ProcesadoresBack2 (int k, int tiempo0, int tiempo1,
                        TipoSolucion *sol, TipoSolucion *solOptima, int N, int Coste[2][MAX]) {

  sol->tiempo = maximo (tiempo0,tiempo1); // coste hasta este momento
  if (sol-tiempo < solOptima->tiempo) // vamos bien, si no FRACASO
    if (k == N)
      *solOptima = *sol;
    else {
      // lo hace el procesador 0
      sol->trabajo[k] = 0;
      ProcesadoresBack2 (k+1,tiempo0+Coste[0][k],tiempo1,sol,solOptima,N,Coste);

      // lo hace el procesador 1
      sol->trabajo[Etapa] = 1;
      ProcesadoresBack2 (k+1,tiempo0,tiempo1+Coste[1][k],sol,solOptima,N,Coste);
    }
}
```