LECCION 10.- ALGORITMOS VORACES
(continuación)

EJEMPLO 3.- PLANIFICACIÓN DE TAREAS

Problema de secuenciación de tareas o trabajos que tienen un plazo definido para realizarse.

Consideraciones

- Tenemos un procesador y n tareas disponibles
- Todas las tareas requieren una unidad de tiempo para ejecutarse (1 seg). Cada tarea además tiene:
  - b_i: beneficio que se obtiene si se ejecuta la tarea i
  - d_i: Plazo máximo de ejecución de la tarea i
    ↳ la tarea i sólo podrá ejecutarse si se hace en un instante igual o anterior a d_i.

En general puede que no sea posible ejecutar todas las tareas.

OBJETIVO: dar una planificación de las tareas a ejecutar $(s_1, s_2, \ldots, s_m)$ de forma que se maximice el beneficio total. $b_{TOTAL} = \sum_{i=1..m} b_{s_i}$ s_i: representa la tarea que se ejecuta en el instante i

PROBLEMAS A RESOLVER

1) ¿Qué tareas se ejecutan?
2) ¿En qué orden se ejecutan?

EJEMPLO $n=4$ $b=(100, 10, 15, 27)$ $d=(2, 1, 2, 1)$

b: beneficios de las tareas

d: plazos de ejecución de las tareas: ej: la tarea 1 debe ejecutarse en el segundo 2 o antes

TIEMPO T	1	2
TAREA	1	3
b	100	15
d	2	2

$B_{TOTAL} = 115$

T	1	2
S	4	3
b	27	15
d	1	2

$B_{TOTAL} = 42$

T	1	2
S	(1)	(9)
b	100	27
d	(2)	(1)

NO FACTIBLE

T	1	2
S	4	1
b	27	100
d	1	2

$B_{TOTAL} = 127$

LECCION 10: AL6. VORACES

(Planificación de Tareas continuación)

ALGORITMO 1 (Trivial) - Comprobar todos los posibles órdenes de las tareas y quedarse con la mejor (y que sea factible). Equivale a buscar todas las posibles permutaciones. Un coste inferior es $\Omega(n!)$

ALGORITMO 2: TÉCNICA VORAZ

1) Empezamos con un planificación sin tareas y en cada paso vamos añadiendo una

2) La solución está formada por el conjunto de tareas seleccionadas junto con el orden de ejecución de las mismas
Así $S = (s_1, s_2, \dots, s_m)$ es una solución donde s_i representa la tarea que se ejecuta en el instante i.

3) FUNCION de SELECCION: de entre los candidatos restantes escoger aquel que produzca mayor beneficio

4) FUNCION FACTIBLE: Comprueba que d_i (plazo de la tarea escogida en el instante i es menor o igual al instante que se ejecuta.

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <fstream>
using namespace std;
struct tarea{
  int beneficio;
  int d;
  int key;
};
istream & operator >>(istream&is, tarea &t){
  is>>t.key>>t.beneficio>>t.d;
  return is;
}


bool compare(const tarea &t1,const tarea &t2){
    return t1.beneficio>t2.beneficio;
}

istream & operator >>(istream&is, vector<tarea> &T){
  tarea t;
  while (is>>t){
     T.push_back(t);
  }
  return is;
}

bool Factible(const vector<tarea> &T,vector<int> &S,int nueva,int k){
  int n=S.size();
  vector<int> Saux(n,-1);
  int i =0;
  while (i<k && (i+1)<T[nueva].d){
      Saux[i]=S[i];
      i++;
  }
  Saux[i]=nueva;
  while(i<k){
      Saux[i+1]=S[i];
      i++;
  }
  i=0;
  while (i<=k && T[Saux[i]].d>=(i+1)) i++; //Si i no avanza hasta el
                                           //final significa que no es factible
  if (i>k) {
    S=Saux;
    return true;
  }
  else return false;
}

int Planificacion(const vector<tarea> &T ,vector<int> &S){
  int tiempos=S.size();
  int n_tareas=T.size();
  int i=0,puestas=0;
  int bene_total=0;
  while (i<n_tareas && puestas<tiempos){
          int candidata=i;
          if (Factible(T,S,candidata,puestas)){
            bene_total+=T[candidata].beneficio;
            puestas++;
          }
```



```
          i++;
  }
  return bene_total;
}

int main( int argc, char* argv[]){
  if(argc !=3){
      cout<<"Los parametros son:"<<endl;
      cout<<"1.-El fichero con las tareas"<<endl;
      cout<<"2.-El maximo tiempo"<<endl;
  }
  int max_time=atoi(argv[2]);
  ifstream f(argv[1]);
  if (!f){
      cout<<"No puedo abrir" <<argv[1]<<endl;
      return 0;
  }
  vector<tarea> T;
  f>>T;
  sort(T.begin(),T.end(),compare);

  vector<int> S(max_time,-1);
  int Bt=Planificacion(T, S);
  cout<<"Beneficio Total " <<Bt<<endl;
  cout<<"Los tiempos y tareas que se ejecutan:"<<endl;
  int i=0;
  while (i<max_time && S[i]!=-1){
    cout<<"Tiempo: "<<i+1<< " Tarea :"<<T[S[i]].key<< " "<<T[S[i]].beneficio<<"d="<<
T[S[i]].d<<endl;
    i++;
  }



}
```

LECCION 10 : ALG. VORACES

Caracteristicas de las Técnicas Voraces

- **Problemas NP-Completos**: Son problemas que no se pueden resolver en tiempo polinomial. Su solución exacta puede requerir órdenes factoriales o exponenciales. (Explosión Combinatoria)

  → **OBJETIVO** en estos problemas → obtener soluciones (aproximaciones a la exacta) buenas en un tiempo de ejecución moderado o rápido

- ALG. de APROXIMACION.- Garantizan una solución más o menos buena (que responda a una aprox. a la solución optimal).

- → ALG. HEURÍSTICOS.- Son ALG. de Aproximación que se basan en el conocimiento intuitivo o experto del programador sobre cierto problema.

  se dividen en:
  - ALG. VORACES
  - ALG. GENÉTICOS
  - ALG. PROBABILISTICOS.

Todos ellos se centran en un estudio muy detallado de la FUNCIÓN de SELECCIÓN