

ALGORITMOS VORACES

GENERAL: Dadas entradas verificando condiciones en búsqueda de solución

Complejidad: $\text{IP}(x)$

¿Desventajas?

- No óptimo global
- Tiempo $\text{IP}(x)$
- No reconsideran

¿Cuando?

- Buscamos sol
- Fácil implementación
- No óptimo global

Mochila / objetos → óptimo.

Pseudo código:

FuncióN VORAZ (Datos) dev: ConjuntoDatos

Declarar una variable para copiar los datos, nos declaramos el conjunto de datos que será solución a vacío.

while (! candidatosVacio) \wedge (! Solucion (S))

Cogemos candidato de la copia, lo guardamos en un auxiliar y lo quitamos de la copia candidatos

if (factible (S, Aux))

Añadimos la solución al conjunto solución

Función Solucion (S): Depende del problema, en general es satisfacer la condición de fin si se hubiera antes de quitar el candidato

Función Factible (S, Aux): Comprueba si añadir el objeto a soluciones es posible, obtener el mejor objeto ...

ALGORITMOS VORACES

CAMBIO :

ESQUEMA :

CANDIDATOS : Valores de las monedas del sistema monetario

SOLUCION : Conjunto monedas suman el cambio a devolver

AUXILIAR : Suma de las monedas escogidas en un momento dado

FACTIBLE : Leger moneda mas alta que no supere el importe

Pseudo código :

Funcion Cambio (cambio , valoresMonedas)

Añadimos en ValoresMonedas los candidatos a cambio , y nos ayudamos de un auxiliar para saber cuanto llevamos.

for (ValoresMon. long)

while (i ≤ ValMon.BegInt \wedge aux != cambio)

Pasarnos al siguiente valor del sistema monetario . y sumamos el cambio devuelto la moneda

No obtenemos optima , solo solución.

MOCILA:

ESQUEMA:

CANDIDATOS: Todos los objetos a meter

SOLUCIÓN: Alcanzar el peso maximo de la mochila

FATIBLE: Discreto: El objeto introducido es el mas ligero (max obj) etc...

Continuo: Objeto con mejor ratio peso/valor.

Si lo cogemos por orden decreciente = OPTIMO

Pseudocódigo:

Funcióñ Mochila (candidatos, pesoMax, sol)

Hacemos copia de los candidatos y un auxPeso que lleve la cuenta de los metodos

while (pesoAux != pesoMax \wedge !Candi(Cop vacio))

cogemos el mejor objeto, el que menos peso ejemplo

if (pesoAux + pesoObj < pesoMax)

lo añadimos a la solución y se pesoAux le sumamos el del objeto

else Borrmos el objeto de Copia.Candidatos

Devolvemos la solución

ALGORITMOS VORACES

TIEMPO:

CANDIDATOS: Todas las actividades a escoger

SOLUCION: Depende del problema, minimizar tiempo espera, hacer las maximas actividades, maximizar tiempo de ejecución ...

FACTIBLE: Actividades que podemos realizar antes de agotar el tiempo

Pseudocódigo:

Función Tiempo (candi, tiempo, sol)

Hacemos una copia de los candidatos y guardamos lo que sea la solución

while (!CopiaCandVacia \wedge \neg Solucion)

Nos guardamos el candidato en un auxiliar

if (aux Factible*)

Solucion le añadimos el candidato Aux

else Borramos el candidato auxiliar

Devolvemos la solución

*

La función de factible depende mucho del problema, nos dice si nos pasamos del tiempo, si cumple las restricciones etc...

VORACES GRAFOS

PRIM: Obtenemos el arbol de recubrimiento minimo, desde un vertice arbitrario

Complejidad: $\Theta(\max|\text{Vert}| |\text{Arist}|) \stackrel{\text{cf.}}{\Rightarrow} \Theta(|A| \cdot \log |V|)$

Pseudocódigo:

Funcion Prim (candidatos)

Creamos un diccionario de visitados de longitud candidatos

for (numCand.)

if (!visitado[i])

PriorityQueue Cola = Priority Queue

Porremos el nodo a visitado y añadimos a la cola el nodo que estamos evaluando

while (!cola vacia.)

Nos guardamos en un aux el cola.poll

if (!visitado [aux.destino])

Añadimos a la cola el aux.destino y ponemos a visitado

grafo = List[ArrayList]

