

2020 FEB

1) a) La mejor estrategia es DyV porque este problema es una búsqueda binaria en la que el número a buscar es el índice del centro de la división.

b) Función buscaPuntoFijo(~~sint~~ p: entero[], p: entero, q: entero)
: booleano {

Si $q < p$ Devuelve falso;

$$m := (p+q)/2;$$

if ($m = v[m]$) Devuelve verdadero;

if ($m < v[m]$) Devuelve buscaPuntoFijo(v, p, m-1);

* Devuelve buscaPuntoFijo(v, m+1, q);

}

$$c) T(n) = T(n/2) + c_2 \quad n \geq 1$$

$$T(0) = c_1$$

$$T(n) - T(n/2) = c_2 \Rightarrow T_m - T_{m-1} = c_2 \Rightarrow Ec: (x-1)(x-1)$$

$$T_m = c_1 \cdot 1^m + c_2 \cdot m \cdot 1^m // T(n) = c_1 + c_2 \cdot \log n \Rightarrow T(n) \in O(\log n)$$

2) Para mejorar la ratiocinación que encuentra este algoritmo el método que más efecto es ordenar por el ratio beneficio / peso. Otros criterios son: ordenar por mayor beneficio o por peso.

En este ejemplo, el algoritmo va a encontrar la combinación óptima $P[2, 3, 4] \cup b[1, 2, 5] \rightarrow (1, 0, 1) M=6$
Pero en el caso $P[5, 3, 3] \cup b[11, 6, 6]$ con $M=6$ encuentra $(1, 0, 0)$ cuando la óptima es $(0, 1, 1)$

Función mochila (n : entera, M : entera, p : entera[], b : entera[]): booleano[] {

x : booleana [];

Para $i = 1$ hasta n { $x[i] = \text{falso}$; } $S \leftarrow \emptyset$

Para $i = 1$ hasta $n-1$ {

Para $j = 1$ hasta $n-i$ {

Si $b[i]/p[i] < b[i+1]/p[i+1]$ {

$\text{swap}(b[i], b[i+1]);$

$\text{swap}(p[i], p[i+1]);$

} {

$i = 1;$

while($i \leq n$ y $M > 0$) { } [salir]

Si ($M \geq p[i]$) { } [factible + reelección]

$x[i] = \text{verdadero};$ } [insertar]

$M = M - p[i];$

} {

$i = i + 1;$

Devaluar $x;$

}

b)

Evaluación recurrente:

$$V[i, j] = \max(V[i-1][i], V[i-1][j-p[i]] + b[i])$$

$$\& V[0, j] = 0 \text{ y } V[i, 0] = 0 \text{ y } V[i, j] = -\infty \text{ si } i < 0 \text{ o } j < 0$$

La evaluación se compone recomienda de abajo a arriba la columna de mayo para y añadiendo y cambiando de columna conforme el beneficio cambia. Ya que si cambia es que el elemento se reelección.

Eulerian Maehila (n : entera, M : entera, p : entera[], b : entera[]): backtracking

V : entera [$n+1$, $M+1$];

Para $i=0$ hasta n { $V[i, 0] = 0$ }

Para $j=0$ hasta M { $V[0, j] = 0$ }

Para $i=1$ hasta n {

Para $j=1$ hasta M {

dejor = $V[i-1, j]$;

Si $m - p[i] < 0$

$V[i, j] = \text{dejor}$;

otra {

cayer = $V[i-1, j] - p[i] + b[i]$;

~~si cayer > dejor~~ $V[i, j] = \max(\text{cayer}, \text{dejor})$;

~~variazca~~

}

}

}

$j = M$;

x : entera [n]

Para $i = n-1$ hasta 0 para -1

Si ($V[i, j] < V[i+1, j]$) {

$x[i] = \text{verdadero}$;

$j = j - p[i]$;

}

Devuelve x ;

}

Recorrido
Salieron

\diagdown	0	1	2	3	4	5	6	capacidad de los mochiles
0	0	0	0	0	0	0	0	
1	0	0	0	0	0	11	11	
2	0	0	0	6	6	11	11	
3	0	0	6	6	6	11	12	

↓ elemento que se intenta agregar

c)

Función Machila (n : entera, M : entera, P : entera[], b : entera[]): booleano [] {
 x: booleano[];
 machila ($n, M, P, b, x, 1, 0$);
 Devalver (x);
 }

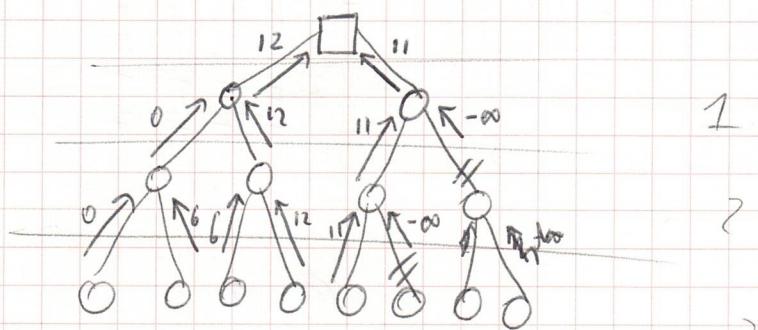
Función Machila (n : entera, M : entera, P : entera[], b : entera[],
 x: booleano[], c: entero, B: entera): entera {
 si ($M < 0$)
 Devalver - ∞ ;
 else si ($c > n$
 Devalver B ;

cayer = Machila ($n, M - P[c], P, b, x, c + 1, B - b[c]$);
 deyo = Machila ($n, M, P, b, x, c + 1, B$);

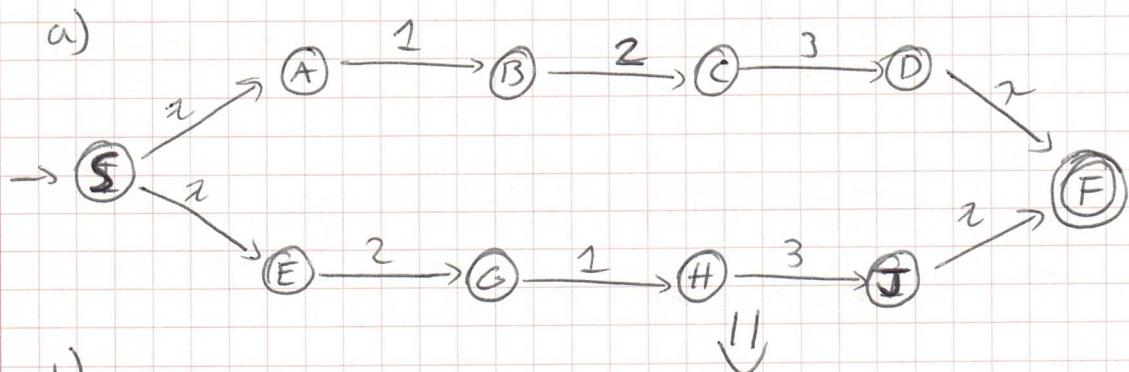
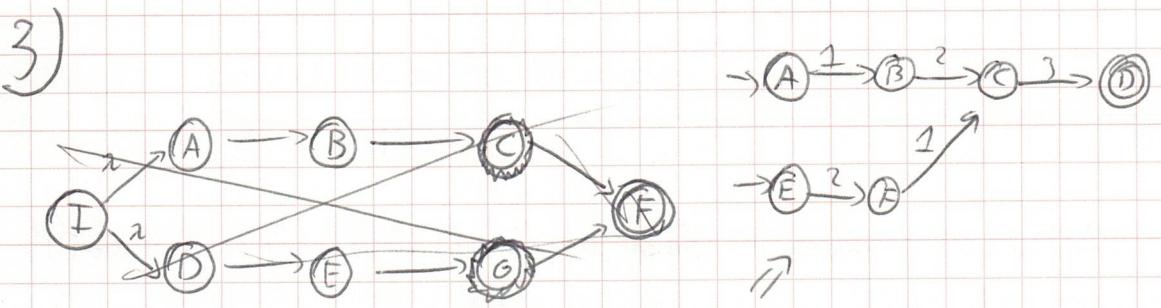
si cayer > deyo {
 x[c] = verdadero;
 Devalver cayer;
 }
 si no {
 x[c] = falso;
 Devalver deyo;

}

// = Poda
 por exceder M

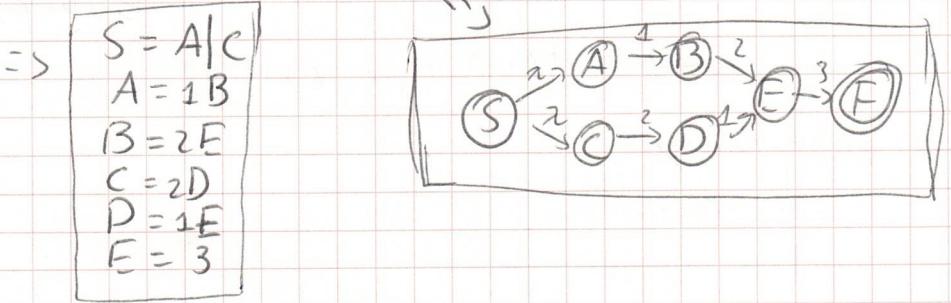


x	0	0	0	1	1	1	1
0	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
B	0	3	3	12	11	-∞	-∞



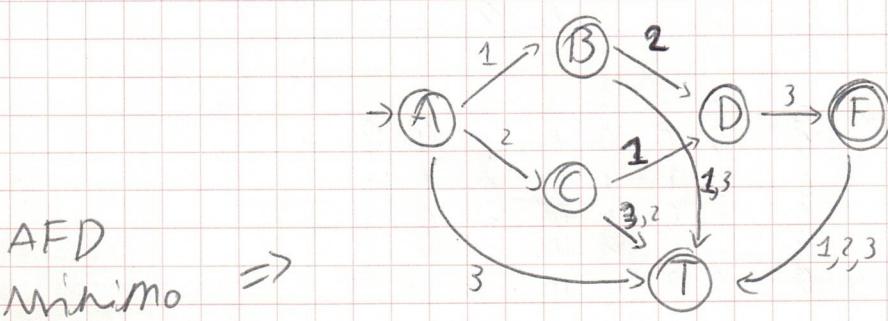
b)

$$\begin{aligned}
 S &= A | D \\
 A &= 1B \\
 B &= 2C \\
 C &= 3 \\
 D &= 2E \\
 E &= 1G \\
 G &= 3
 \end{aligned}$$

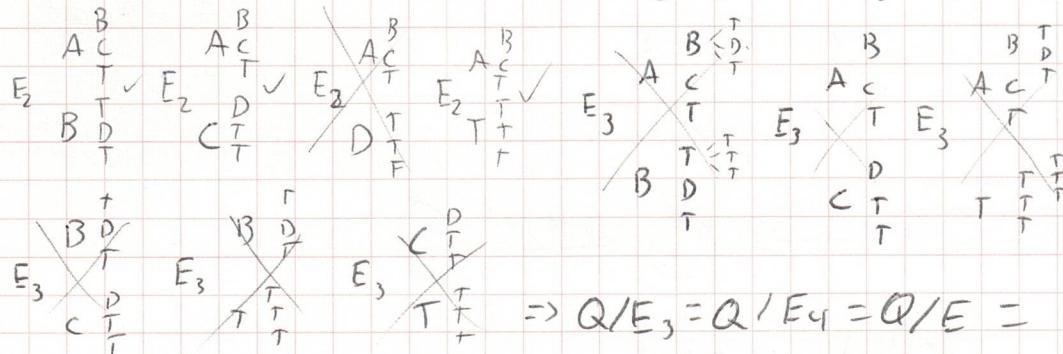


c)

	1	2	3	-
$\rightarrow(A)$	SAC	B	D	-
(B)	B	B	-	E
(C)	D	D	E	-
(D)	E	E	-	-
*(F)	F	F	-	F
(T)				

$$\Rightarrow \begin{array}{c|ccc}
 & 1 & 2 & 3 \\
 \hline
 A & B & C & I \\
 B & T & D & T \\
 C & D & T & T \\
 D & T & F & F \\
 *F & T & T & T \\
 T & T & T & T
 \end{array}$$


$$Q/E_1 = [\{A, B, C, D, T\}, \{F\}] \Rightarrow Q/E_2 = [\{A, B, C, T\}, \{D\}, \{F\}]$$



$$\Rightarrow Q/E_3 = Q/E_4 = Q/E = [\{A\}, \{B\}, \{C\}, \{D\}, \{T\}, \{F\}]$$

El efecto no es reducible.

4) a)

$$S \rightarrow AS'$$

$$S' \rightarrow =AS'|z$$

$$A \rightarrow B \quad \cancel{A'}$$

$$A' \rightarrow +BA' - BA' | z$$

$$B \rightarrow (S)|a|b$$

	Primera	Siguiente
S	(, a, b	\$,)
S'	z, =	\$,)
A	(, a, b	=, \$,)
A'	z, +, -	=, \$,)
B	(, a, b	+, -, =, \$,)

	+	-	=	()	a	b	\$	*
S				AS'					
S'			=AS'		z				
A				BA'		BA'	BA'		
A'	+BA'	-BA'			z				
B				(S)		a	b		

No hay conflictos Primera-Primera

$$\text{Primera}(=) \cap \text{Primera}(z) = \emptyset$$

$$\text{``} (+BA') \cap \text{``} (-BA') \cap \text{Primera}(z) = \emptyset$$

$$\text{``} ((S)) \cap \text{``} (a) \cap \text{``} (b) = \emptyset$$

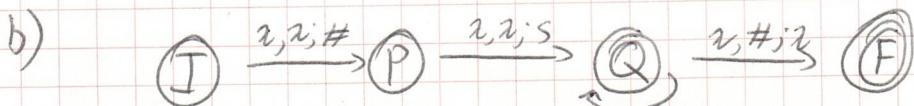
No hay conflictos Primera siguiente

Siguiente(s') \cap Primera($=AS'$) = \emptyset

Siguiente(s') \cap Primera($+BA'$) = \emptyset

Siguiente(s') \cap Primera($-BA'$) = \emptyset

Es una gramática LL1 válida.



ESTADO / PILA	Entada
I	$a=b\$$
P	$a=b\$$
Q	$\#S$
Q	$\#S'A$
Q	$\#S' A' B$
Q	$\#S' A' a$
Q	$\#S' A'$
Q	$\#S'$
Q	$\#S' A=$
Q	$\#S' A$
Q	$\#S' A' B$
Q	$\#S' A b$
Q	$\#S' A$
Q	$\#S'$
Q	$\#$
F	\$

$+,-;z$
 $=,-;z$
 $=,=;z$
 $(,(;z$
 $),);z$
 $a,a;a$
 $b,b;b$
 $,,$
 $z,S;AS$
 $z,S';AS'$
 $z,S';z$
 $z,A;BA'$
 $z,A';+BA'$
 $z,A';-BA'$
 $z,A';z$
 $z,B;(S)$
 $z,B;a$
 $z,B;b$

ACEPTADA

c) La tabla la puse en el A y no me cuenta.*

y el análisis de $a=b\$$ es exactamente igual que en el b) sea en la columna Estado.

Procedimiento analisis(L){

Aplor(' #');

Apaplos
Aplor(S); ← "5"

Leer(SLA); ←

Mientras NO pila_vacia {

Si cima_pila_es_terminal { ← tener car un car

Si cima_pila = SLA {

Dereplor(SLA);

Leer(SLA);

} fina error();

} fina si cima_pila_es_ma_tamal {

tmp = Tabla[cima_pila, SLA];

Si tmp <> vacia {

dereplor(cima_pila);

aplor(tmp);

} fina error();

}

Si cima_pila == '#' {

Dereplor('#');

// Cadena aceptada

} fina error();

↳ error_untactica.

}

d)

Programa Principal

SLA = leer();

S();

Si SLA $\neq \$$
 emerg();

}

Procedimiento reader (SLA Tamaño){

Si SLA == Tamaño

 leer();
 the emerg();

}

Procedimiento S {

A()

S'()

}

Procedimiento S'{

core SLA of

'=': reader('=!');

A()

S'()

'\$', ')':

the: emerg();

}

Procedimiento A'{

core SLA of

'+' : reader('+'');

'-' : reader(''-');

'\$'; '='; ')':

the: emerg();

}

Procedimiento A{

B();

A();

}

Procedimiento B{

core SLA of

'(' : reader('(');

S();

reader(')');

'a' : reader('a');

'b' : reader('b');

}

Principal |

↓

S

↓

A

↓

B

↓

'a'

↓

A'

↓

'b'

↓

S'

↓

'='

Resuelto