$h(n) \le \Gamma_{n \to n'} + h(n')$

Paris 20	10	Reims 15	
Nancy 30	20	Reims 15	
Nancy 30 $\le$	10	Paris 20	✓
Nevers 35	20	Paris 20	
Orleans 55	38	Paris 20	
St.Malo 80	70	Paris 20	
Paris 20		Nancy 30	✓
Lyon 40	20	Nancy 30	
Roenne 35	5	~~Nancy~~ Nevers 35	
Paris 20	75	Nevers 35	
Limoges 80	60	Nevers 35	
Paris 20		Orleans 55	
Nantes ~~100~~ 95	55	Orleans 55	
Limoges 80	85	Orleans 55	
Nantes ~~100~~ 95	15	St.Malo 80	
Paris 20		St. Malo 80	
Brest 100	40	St. Malo 80	

$\forall n, n'$ n' sucesor de n.

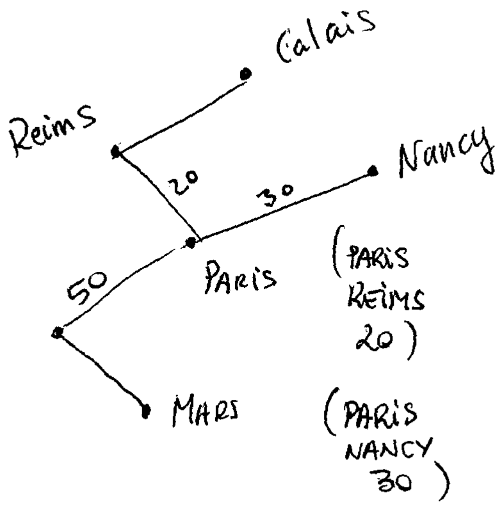
$h(n) \le \Gamma_{n \to n'} + h(n')$

Roenne 35	25	Lyon 40
Toulouse ~~110~~ 100	95	Lyon 40
Avignon 70	40	Lyon 40
Lyon 40	5	Roenne 35
Toulouse 100 ~~120~~	35	~~Roe~~ Limoges 80
Orleans 55		Limoges 80
Nevers 35		Limoges 80
St.Malo 80		Nantes 95
Brest 100	35	Nantes 95
Orleans 55		Nantes 95
Toulouse 100	80	Nantes 95
St. Malo 80		Brest 100
Nantes 95		Brest 100
Nantes 95		Toulouse 100
Limoges 80		Toulouse 100
Lyon 40	~~95~~	Toulouse 100
Marseille ~~200~~ 95	120	Toulouse 100
Lyon 40	20	Avignon 70
Marseille ~~200~~ 95	25	Avignon 70

Toulouse 100	120	Marseille 95
Avignon 70		Marseille 95

```
(defstruct
 (make-action   :name    'train-cost
                :origin  'Marseille
                :final   'Lyon
                :cost    15.0 )
```

```
(defparameter  *travel-fast*
 (make-problem
     :states  *cities*
     :initial-state  *initial*
 f(state)
```



```
navigate (state  edges  cfun  name  forbidden)

nav_aux(state  edge  cfun  name  forbidden)
   if  state = first(edge)  and
       second(edge)  not  in  forbidden
           make-action ( :name   = name
                         :origin = state
                         :final  = second (edge)
                         :cost   = cfun ( third (edge)))
```

```
f-goal-test (node  dest  mandatory)
    if  node-state (node)  in  dest
        and  null (eliminate (node  mandatory))
T
else
NIL

 if  node-parent  == NULL        CASO  BASE
     mandatory                   ELIMINATE

eliminate (node  mandatory)
else:
let
   m = mandatory \ node-state
in
   eliminate (node-parent, m

(and  ___ -
      ___  )
```

(1 ( ) ( ) ( ) . . . ( ))

AND

(expand lst) → expand ( )

(or-expand (rest lst))

algo

- (A B) ≡ A∧B [1]
- ((A) (B)) ≡ A∨B [2]
- A ≡ expand (A) [3]
- (!A) ≡ expand (!A) [4]

condición parada

ej: ((A ⇒ B)

opción 1:

(if (null lst)

NIL

opción 2:

(if (null (rest lst)

(expand (first ls

* depende de la función combinatoria

(dfun handle-and □ O)

rest

- ((A) (B) (C)) ≡ A∨B∨C (a)
- (A B C) ≡ A∧B∧C (b)
- A ≡ expand (A) (c)
- (!A) ≡ expand (!A) (d)

caso [1] (a)	caso [1] (b)	caso [1] (c)	caso [1] (d)
…-lstoflsts-merge [1] (a))	(double-and-merge [1] (b))	(append [1] (list (c)))	(append [1] (list (d)))
…so [2] (a)	caso [2] (b)	caso [2] (c)	caso [2] (d)
…le-or-merge [2] (a))	(lst-lstoflsts-merge (b) [2])	(elt-lstoflsts-merge (c) [2])	(elt-lstoflsts-merge (d) [2])
… [3] (a)	caso [3] (b)	caso [3] (c)	caso [3] (d)
…stoflsts-merge [3] (a))	(append (b) (list [3]))	(cons [3] (list (c)))	(cons [3] (list (d)))
… [4] (a)	caso [4] (b)	caso [4] (c)	caso [4] (d)
…stoflsts-merge [4] (a))	(append (b) (list [4]))	(cons (list (c)) (list [4]))	(cons [4] (list (d)))

el positivo!

((v) ( ) ( ) ( ) . . . ( ))

OR

$\underbrace{(\ )}_{\text{(expand lst)}}\ \underbrace{(\ )\ (\ )\ \ldots\ (\ )}_{\text{(or-expand (rest lst))}}$

expand()

algo

- (A B) ≡ A∧B $\boxed{1}$
- ((A) (B)) ≡ A∨B $\boxed{2}$
- A ≡ expand(A) $\boxed{3}$
- (!A) ≡ expand(!A) $\boxed{4}$

rest

- ((A) (B) (C)) ≡ A∨B∨C (a)
- (A B C) ≡ A∧B∧C (b)
- A ≡ expand(A) (c)
- (!A) ≡ expand(!A) (d)

Condición parada

ej: ((A ⇒ B))

opción 1: (if (null lst) NIL

opción 2: (if (null (rest lst)) (expand (first l

(*) depende de la función combinadora

(defun handle-or □ ())

caso $\boxed{1}$ (a)	caso $\boxed{1}$ (b)	caso $\boxed{1}$ (c)	caso $\boxed{1}$ (d)
(append (a) (list $\boxed{1}$))	(cons $\boxed{1}$ (list (b)))	(cons (list (c)) (list $\boxed{1}$))	(cons (d) (list $\boxed{1}$))
caso $\boxed{2}$ (a)	caso $\boxed{2}$ (b)	caso $\boxed{2}$ (c)	caso $\boxed{2}$ (d)
append $\boxed{2}$ (a))	(append $\boxed{2}$ (list b))	(cons (list (c)) $\boxed{2}$)	(append $\boxed{2}$ (list (d)))
aso $\boxed{3}$ (a)	caso $\boxed{3}$ (b)	caso $\boxed{3}$ (c)	caso $\boxed{3}$ (d)
ns (list $\boxed{3}$) (a))	(cons (list $\boxed{3}$) (list (b)))	(cons (list $\boxed{3}$) (list (list (c))))	(cons (list $\boxed{3}$) (list (d)))
so $\boxed{4}$ (a)	caso $\boxed{4}$ (b)	caso $\boxed{4}$ (c)	caso $\boxed{4}$ (d)
end (a) (list $\boxed{4}$))	(cons $\boxed{4}$ (list (b)))	(cons (list (c)) (list $\boxed{4}$))	(cons $\boxed{4}$ (list (d)))

$(\Rightarrow (\wedge\ P\ Q)\ R)\ (\Rightarrow (P\ (\vee\ (!Q)\ R))))$

$$\underbrace{[(P \wedge Q) \Rightarrow R]}_{A} \Leftrightarrow \underbrace{[P \Rightarrow (\neg Q \vee R)]}_{B}$$

$(A \Rightarrow B) \wedge (B \Rightarrow A)$ expand

$(\neg A \vee B) \wedge (\neg B \vee A) \equiv (\neg A \wedge \neg B) \vee (A \wedge B)$

$\cancel{[(P \wedge Q) \Rightarrow R] \wedge [P \Rightarrow (\neg Q \vee R)]}$

$$\underbrace{(\wedge (\Rightarrow (\wedge\ P Q)\ R)\ (\Rightarrow P\ (\vee\ (!Q)\ R)))}_{1^a \text{ parte}}\ (\wedge (\neg$$

1^{er} paso $\rightarrow$ pasar expresión a FNC

(a b) ($\neg$a $\neg$b)

$[\Leftrightarrow (\Rightarrow (\wedge\ P\ Q)\ R)\ (\Rightarrow P\ (\vee\ (!Q)\ R))\ ]$

bicondicional:

$[\vee((\Rightarrow(\wedge\ P\ Q)\ R) \wedge (\Rightarrow P\ (\vee\ (!Q))\ R))\ ((\neg(\Rightarrow(\wedge\ PQ)\ R) \wedge \neg(\Rightarrow P(\vee(!Q)$

$[\vee\underbrace{(\wedge(\Rightarrow(\wedge PQ)R)(\Rightarrow P(\vee(!Q)R)))}\ \underbrace{(\wedge(\neg(\Rightarrow(\wedge PQ)R))\ (\neg(\Rightarrow P(\vee(!Q)R))))}$

$(\Rightarrow(\wedge\ PQ)\ R)\ (\Rightarrow P(\vee(!Q)R))$

$\cancel{\neg[(\wedge PQ)\vee R]\ \neg[}$

$(\neg(\wedge PQ) \vee R)$ $\quad$ $(\neg P \vee (\neg Q \vee R))$

$[\vee\ \underbrace{(\neg(\wedge\ PQ))}\ \underbrace{R}]$ AND $[\vee\ \underbrace{(\neg P)}\ \underbrace{(\vee\ (\neg Q)\ R)}]$

$\neg P \vee \neg Q$
$\equiv$
$(\vee\ (!P)\ (!Q))$

$[()(P)]$ $\quad$ $[()(Q)]$

R

$[(R)()]$

$!P$

$[()(P)]$

$(\vee\ \underbrace{(!Q)}\ \underbrace{R})$

$!Q$ $\quad$ R

$[()(Q)]$ $\quad$ $[(R)()]$

AND

$(P\wedge Q \Rightarrow R) \wedge (P \Rightarrow (\neg Q \vee R))$

$$\boxed{\begin{matrix}(P\wedge Q) \Rightarrow R \\ P \Rightarrow \neg Q \vee R\end{matrix}}$$

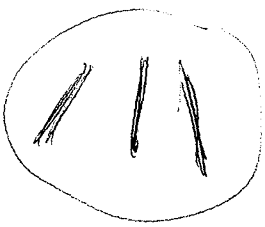
$\neg(P\wedge Q)$ $\quad$ R

$[()()]$

$\neg P$ $\quad$ $\neg Q$

$\neg P \vee \neg Q \vee R$

$\neg Q$ $\quad$ R



$[($

$(\neg P\vee\neg Q)\vee R$ $\quad$ $(\neg P\vee \neg Q \vee R)$

cond-expand

$$(\Rightarrow lst1\ lst2) \equiv (\vee\ (!lst1)\ (lst2))$$

ej: $A \Rightarrow B \equiv \neg A \vee B \longrightarrow (\Rightarrow A\ B) \equiv (\vee\ (!A)\ B)$

- caso: 2 listas: '(v A B) = lst1 '(v C D) = lst2

  (append (cons 'v (cons (cons '! (cons lst1 NIL)) NIL)) (cons lst2 NIL))

- caso: 1º literal y 2º lista literal = 'A lst2 = '(v A

  igual que arriba

- caso: 1º lista y 2º literal

  igual que arriba

- caso: ambos literales lst1 = 'A lst2 = 'B

bicond-expand

$$(\Leftrightarrow lst1\ lst2) \equiv (\wedge (\Rightarrow lst1\ lst2)\ (\Rightarrow lst2\ lst1))$$

ej: $A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A) \equiv (\neg A \vee B) \wedge (\neg B \vee A) \equiv (A \wedge B) \vee (\neg A \wedge \neg B)$

$(A \Rightarrow B)$ $\neg((A \vee B) \vee (C \Rightarrow D))$

append

$(A \Rightarrow B)$ → $\neg A \vee B$

$(A \vee B) \vee (C \Rightarrow D)$

or

expand

$\neg A$ | B

$(A \vee B)$ $(C \Rightarrow D)$ | $(A \vee B)$ $(C \Rightarrow D)$

A B $\neg C$ D | A B $\neg C$ D

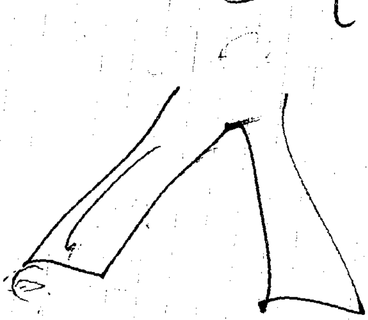
$(\neg A)$ (B)

$(\neg A)$ $(\vee$

$(\neg A\ (\vee\ A\ B))$

$(\neg A\ (A)(B))$

$[(\neg A\ A)$
$(\neg A\ B)\ \dots]$



~~$(\neg A\ A)$~~
$(A\ \neg A)$
$(B\ \neg A)$
$(\neg C\ \neg A)$
$(D\ \neg A)$

$(A\ B)$
$(B\ B)$
$(\neg C\ B)$
$(D\ B)$

[→]

~~tre~~ ~~eval~~

$\neg(A \Leftrightarrow B)$

expand
~~truth-tree~~ (expr)
append (~~truth-tree~~ expand
((

T

NIL

AND
append

expand (first lst)
expand (rest lst)

tree-truth

en expand tenemos caso base
de literal!

$[(\neg A \Rightarrow \neg B) \wedge ((A \Rightarrow B) \wedge (a \vee b))]$

$\boxed{\begin{array}{c} \neg A \Rightarrow \neg B \\ (A \Rightarrow B) \wedge (a \vee b) \end{array}}$

$\neg A \Rightarrow \neg B \rightarrow \neg\neg A \vee \neg B$

$\neg\neg A$	$\neg B$
$A \Rightarrow B$ ($\neg A \vee B \leftarrow A \Rightarrow B$)	$A \Rightarrow B$
$a \vee b$	$a \vee b$

$\neg A$: A, B — B: A, B — $\neg A$: A, B — B: A, B

$(A \quad \neg A \quad \neg\neg A)$ $\quad$ $(A \quad \neg A \quad \neg B)$

$(B \quad \neg A \quad \neg\neg A)$ $\quad$ $(B \quad A \quad \neg B)$

$\checkmark (A \quad B \quad \neg\neg A)$ $\quad$ $(A \quad B \quad \neg B)$

$\checkmark (B \quad B \quad \neg\neg A)$ $\quad$ $(B \quad B \quad \neg B)$

$$\boxed{1 - \frac{\sum_{1}^{n} x_i y_i}{\sqrt{\sum_{1}^{n} x_i^2}\ \sqrt{\sum_{1}^{n} y_i^2}}}$$

a =
b =
c

(1 2 3)
(3 2 1)

$$1 - \frac{10}{\sqrt{14 \cdot 14}} = 1 - \frac{10}{14} = \frac{4}{14}.$$

```
(defun ______ (x y)
   (cond  ~~(cond (= 0 (sq~~
      (= 0 (squares x)) 1)
      (= 0 (squares y)) 1)
      (T (- 1 (______)))))
(let ((simb val) (simb val)) (función))
```