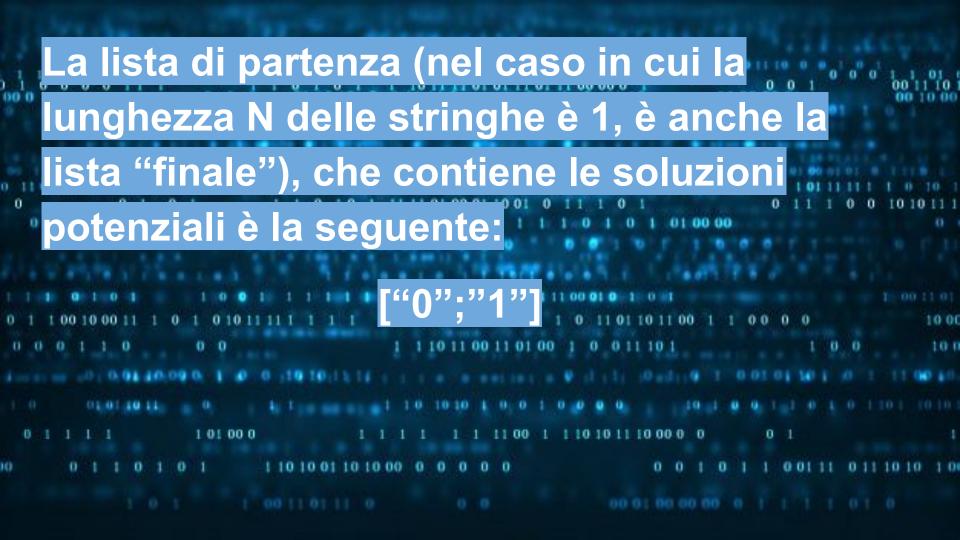


Si consideri un insieme finito S di stringhe di lunghezza N sull'alfabeto {0,1,2}. Determinare, se esiste, una stringa x di lunghezza N sull'alfabeto {0,1} tale che per ogni stringa s∈S d(s,x) >0, dove d è la distanza di

Hamming. Si risolva il problema utilizzando

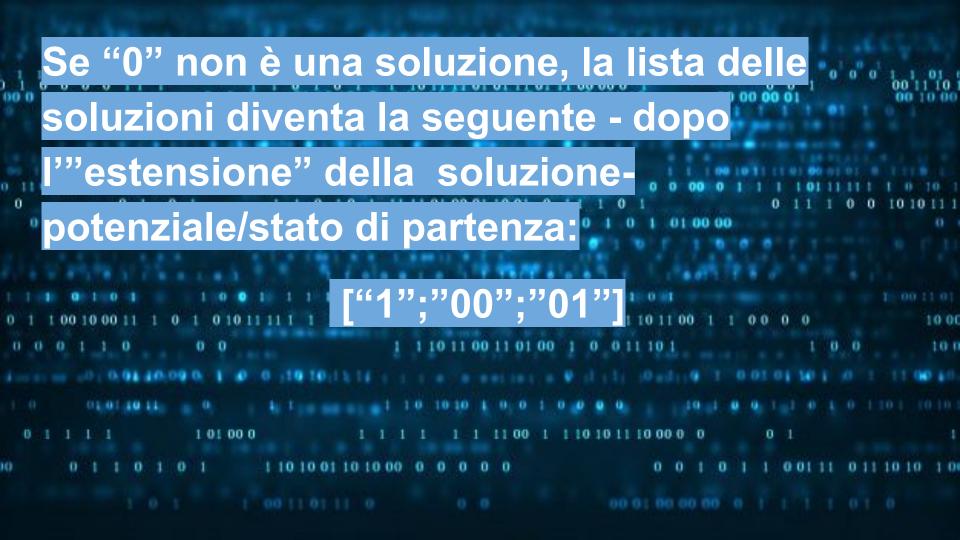
una ricerca in ampiezza.

# Distanza di Hamming è il numero di posizioni in cui due stringhe di bit/caratteri della stessa lunghezza hanno bit/caratteri diversi: HammingDistance ("<u>00</u>10","<u>11</u>10") = 2



Se la lista contiene delle stringhe di lunghezza 1, per ogni soluzione potenziale viene calcolato il numero di volte in cui appare nella lista S. Se almeno uno dei due non vi compare mai, allora c'è una soluzione.

Se la lista contiene delle stringhe di lunghezza maggiore di 1, per ogni soluzione potenziale di lunghezza Y<=N viene calcolato il numero di volte in cui appare nella lista S', dove S' è composta dalle sottostringhe di S ciascuna di lunghezza Y. Se almeno una delle soluzioni potenziali non vi compare mai, allora c'è una soluzione.



#### exception ThereIsNoSolution;;

let hittingString stringList =

if stringList = [] then raise

ThereIsNoSolution (\*...\*)

1 01 00 0

l'argomento della funzione principale è una lista di stringhe

> se la lista è vuota solleviamo l'eccezione: non ci sono soluzioni

10.00



#### let stringLength = String.length

calcoliamo la lunghezza N di ciascuna stringa valutando la lunghezza della prima stringa nella

(List.hd stringList) in (\*...\*)

1 01 00 0

#### let filteredStringList = List.filter

#### (fun x -> not (String.contains x '2'))

#### stringList in (\*...\*)

considerato che la distanza di hamming di una stringa composta da 0 e 1 dalla stringa che contiene un 2 sarà sempre maggiore di zero, filtriamo la lista originale, in modo tale che contenga le stringhe in cui non c'è un 2 funzione che specifica il modo in cui la lista originale viene filtrata

#### if filteredStringList = [] then String.make

stringLength 'x' (\*...\*)

1 01 00 0

se nella lista originale c'erano soltanto le stringhe contenenti almeno un 2, allora qualsiasi stringa composta solo da 0 e 1, di lunghezza N (qui <stringLength>), è una soluzione valida al nostro problema



#### let lengthFilteredList = List.length

filteredStringList in (\*...\*)

ci serve contare il numero di stringhe nella lista che è stata filtrata (questo numero verrà utilizzato in seguito, per capire se abbiamo trovato la soluzione)

#### let partialWeightedSolution subLength =

#### begin fun x ->

1 01 00 0



la funzione prende come argomento un numero che indica la lunghezza delle sottostringhe da ricavare dalla lista originale filtrata

la funzione prende come argomento una soluzione potenziale - che, come vedremo adesso, viene confontata con le sottostringhe "originali"

#### let subsubList = List.map

(fun y -> String.compare x y = 0)

la funzione confronta una specifica soluzione potenziale x con una sottostringa "originale"

# (List.map (fun x -> String.sub x 0 subLength)

la funzione che crea la lista delle sottostringhe di lunghezza <subLength> a partire dalla lista originale

#### filteredStringList) in (\*...\*)

creiamo una lista che registra, per ogni indice i-esimo, se la sottostringa i-esima è uguale o meno alla soluzione potenziale x

#### (x, List.length (List.filter (fun z -> z)

#### subsubList)) (\*...\*)

questa coppia (x, num) serve per capire, in seguito, se x è una soluzione o meno

1 01 00 0

<subsubList> è stata appena calcolata (vedi slide precedente) e serve a calcolare, qui, il numero di volte che la soluzione parziale x è comparsa nella lista delle sottostringhe originali della stessa lunghezza di x

### let extendSolution =

1 01 00 0

questa funzione serve per 

potenziale/stato-non-finale 11 01 10 11 00 1 1 0 0 0 0

#### let rec aux step solutionsList =

if step > stringLength then

1 01 00 0

questa funzione ausiliaria serve per trovare la soluzione: <step> è un parametro che registra la lunghezza della soluzione potenziale che stiamo analizzando

raise ThereIsNoSolution (\*...\*)

se abbiamo iniziato ad analizzare le soluzioni potenziali di lunghezza superiore a quella delle stringhe originali (N), vuol dire che le soluzioni non ci sono else

#### let heaD = List.hd solutionsList in

let taiL = List.tl solutionsList in (\*...\*)

<taiL> è la coda della lista contenente le soluzioni potenziali

1 01 00 0

<heaD> è la stringa che si trova in testa alla lista delle soluzioni potenziali 10.00

### if snd (partialWeightedSolution step heaD) = 0

then (\*...\*)

1 01 00 0

calcoliamo la coppia (x, num) - spiegata in precedenza - e se num risulta uguale a 0 allora abbiamo trovato la soluzione

#### if step < stringLength then

let difference = stringLength - step in

heaD^(String.make difference 'x')

else heaD (\*...\*)

se la soluzione parziale che stiamo valutando risulta essere una soluzione al problema, allora, se ha la lunghezza N (qui <stringLength>) uguale a quella delle stringhe originali allora forniamo quella all'utente, altrimenti ...

... altrimenti aggiungiamo dei restanti 'x' alla soluzione che abbiamo trovato 10.00

10 0



#### let extension = extendSolution heaD in

aux (String.length (List.hd taiL))

#### (taiL@extension) (\*...\*)

se la soluzione parziale che stiamo valutando non risulta essere una soluzione al problema, allora richiamiamo la stessa funzione sull'elemento successivo nella lista delle soluzioni parziali

estendiamo la soluzione parziale con un 0 e un 1 e aggiungiamo entrambe le stringhe in coda alla lista delle soluzioni parziali

#### in aux 1 ["0";"1"]

1 01 00 0

la ricerca della soluzione parte con la chiamata della funzione ausiliaria dove il primo argomento - che all'inizio è 1 - indica che prima si analizzano le sottostringhe/soluzioni-parziali di lunghezza 1

il secondo argomento è la lista delle soluzioni parziali di partenza, che man mano vengono estese con 0 e 1

```
val hittingString : string list -> String.t = <fun>
                                                                               il risultato della compilazione
  # hittingString lista;;
                                                                               avvenuta con succeso del
  Exception: ThereIsNoSolution.
                                                                               programma "hittingString"
# let lista = ["2"];;
  val lista : string list = ["2"]
                                                                              prove d'uso (lista è stata
  # hittingString lista;;
                                                                              dichiarata precedentemente
  - : String.t = "x"
                                                                              uguale alla lista vuota)
  # let lista = ["0"];;
 |val lista : string list = ["0"]
  # hittingString lista;;
  - : String.t = "1"
  # let lista = ["01";"10"];;
  val lista : string list = ["01"; "10"]
# hittingString lista;;
                                                                                                             10.00
  - : String.t = "00"
  # let lista = ["01";"10";"00"];;
  val lista : string list = ["01"; "10"; "00"]
  # hittingString lista;;
  - : String.t = "11"
                                                                  1 10 10 11 10 00 0 0
                                                                                          0
  # let lista = [];;
o val lista : 'a list = []
  # hittingString lista;;
  Exception: ThereIsNoSolution.
```

