Università degli studi di Urbino

Dipartimento di Scienze Pure e Applicate

Corso di Laurea in Informatica Applicata

# Progetto di Programmazione Logica e Funzionale

Studenti

Tommaso Maccaroni
Matricola 321621

Angelo Labbruzzo
Matricola 319928

Anno Accademico 2024/2025 - Sessione estiva
Docente: Prof. Marco Bernardo

# Indice

**1 Specifica del Problema** **2**

**2 Analisi del Problema** **3**
2.1 Dati in Ingresso del Problema . . . . . 3
2.2 Dati in Uscita del Problema . . . . . 3
2.3 Relazioni Intercorrenti tra i Dati del Problema . . . . . 3

**3 Progettazione dell'Algoritmo** **4**
3.1 Scelte di Progetto . . . . . 4
3.2 Passi dell'Algoritmo . . . . . 4

**4 Implementazione dell'Algoritmo** **5**
4.1 Implementazione in Haskell . . . . . 5
4.2 Implementazione in Prolog . . . . . 9

**5 Testing** **13**
5.1 Testing del Programma Haskell . . . . . 13
5.2 Testing del Programma Prolog . . . . . 23

# 1 Specifica del Problema

Scrivere un programma Haskell e un programma Prolog che acquisiscano da tastiera una scheda del Futoshiki in formato $N \times N$ e le eventuali disuguaglianze tra le celle. Lo scopo del programma è quello di risolvere il Futoshiki, trovando dove è possibile una scheda nella quale su ogni riga e colonna ogni valore, che va da 1 a $N$, appare esattamente una volta e le disuguaglianze vengono rispettate.

# 2 Analisi del Problema

## 2.1 Dati in Ingresso del Problema

I dati in ingresso sono una scheda del Futoshiki in formato $N \times N$ e le relative disuguaglianze presenti.

## 2.2 Dati in Uscita del Problema

Il dato in uscita, quando esiste una soluzione, è una scheda del Futoshiki in formato $N \times N$ risolta.

## 2.3 Relazioni Intercorrenti tra i Dati del Problema

La scheda del Futoshiki è definita in formato $N \times N$, ogni cella può contenere un numero compreso tra 1 e $n$.

Una configurazione valida del Futoshiki deve soddisfare i seguenti vincoli:

**Vincoli di unicità**:

- Ogni riga deve contenere tutti i numeri da 1 a $n$ esattamente una volta.
- Ogni colonna deve contenere tutti i numeri da 1 a $n$ esattamente una volta.

**Vincoli di disuguaglianza**:

- Tra alcune celle adiacenti (orizzontalmente o verticalmente) sono presenti simboli di disuguaglianza ($<$, $>$).
- Questi vincoli devono essere rispettati: se tra la cella $(i, j)$ e la cella $(i, j + 1)$ è presente il simbolo "$<$", allora il valore nella cella $(i, j)$ deve essere strettamente minore del valore nella cella $(i, j + 1)$.

**Vincoli iniziali**:

- Alcune celle possono essere già riempite con valori predefiniti che non possono essere modificati

Formalmente, una soluzione $S$ del Futoshiki è una funzione $S$ :

$$\{1,\ldots,\text{n}\} \times \{1,\ldots,n\} \to \{1,\ldots,n\}$$

tale che:

1. $\forall i \in \{1,\ldots,n\} : |\{S(i,j) : j \in \{1,\ldots,n\}\}| = n$ (unicità per riga)
2. $\forall j \in \{1,\ldots,n\} : |\{S(i,j) : i \in \{1,\ldots,n\}\}| = n$ (unicità per colonna)
3. $\forall (i,j),(i',j') \in \text{Vincoli} : S(i,j) \odot S(i',j')$ dove $\odot$ è l'operatore di confronto specificato nel vincolo
4. $\forall (i,j) \in \text{CellePreriempite} : S(i,j) = \text{ValoreIniziale}(i,j)$

dove Vincoli rappresenta l'insieme delle coppie di celle adiacenti con i relativi operatori di disuguaglianza, e CellePreriempite rappresenta l'insieme delle celle che hanno un valore iniziale fisso. Il problema consiste nel trovare una soluzione $S$ che soddisfi tutti questi vincoli simultaneamente, se tale soluzione esiste.

# 3 Progettazione dell'Algoritmo

## 3.1 Scelte di Progetto

Per risolvere una scheda del Futoshiki abbiamo utilizzato l'algoritmo di backtracking al posto di un algoritmo euristico, in quanto il primo garantisce sempre la restituzione di una soluzione quando questa esiste. Questo accade perché l'algoritmo di backtracking, a differenza di quello euristico, esplora tutte le possibili combinazioni, tuttavia il suo costo computazionale cresce in maniera esponenziale rispetto a quello euristico, che riduce significativamente il numero di percorsi da esplorare.

L'acquisizione della scheda e delle sue disuguaglianze, quando presenti, avviene in due tempi. Questo è stato fatto per facilitare la gestione dei dati all'interno del programma.

Durante l'acquisizione della scheda, le celle vuote vengono inserite come 0, questo per permetterci di definire la cella vuota che altrimenti sarebbe difficile da inserire.

La scheda è rappresentata come una struttura dati bidimensionale allocata dinamicamente in quanto non si conosce a priori la dimensione di essa.

Le disuguaglianze vengono rappresentate come 2 coppie di interi, le quali corrispondono alle coordinate di 2 celle, e un vincolo di disuguaglianza che definisce la relazione tra le celle.

La soluzione viene stampata sotto forma di tabella senza mostrare le disuguaglianze.

## 3.2 Passi dell'Algoritmo

I passi dell'algoritmo per risolvere il problema sono i seguenti:

1. Acquisizione di una scheda del Futoshiki sottoforma di struttura dati bidimensionale.
2. Acquisizione delle disuguaglianze del Futoshiki, ognuna delle quali rappresentata come 2 coppie di interi legate da un vincolo di disuguaglianza.
3. Esecuzione dell'algoritmo di backtracking:
    - Caso Base: si verifica che la scheda sia completa e che vengano rispettati i vincoli di unicità e di disuguaglianza. In caso affermativo, il Futoshiki è considerato corretto e viene restituito come soluzione sottoforma di tabella completa.
    - Caso Induttivo:
        (a) si individuano le celle vuote, cioè con valore corrispondente a 0.
        (b) si aggiorna il valore nelle celle, inserendone uno nuovo compreso tra 1 e la dimensione della scheda.
        (c) si genera una nuova scheda da validare, si verifica quindi che non siano presenti duplicati nelle righe e nelle colonne, e che anche le disuguaglianze siano verificate.
        (d) nel caso la verifica fallisca si procede in maniera ricorsiva creando una nuova scheda.
4. Terminazione:
    - Al termine dell'algoritmo, se esiste una soluzione, questa viene stampata come una tabella, in caso contrario si segnala che la soluzione non esiste.

# 4 Implementazione dell'Algoritmo

## 4.1 Implementazione in Haskell

**File sorgente futoshiki.hs:**

```haskell
-- ##############################################################
-- #        Corso di Programmazione Logica e Funzionale         #
-- #       Progetto per la sessione estiva A.A. 2024/2025       #
-- #                    di Tommaso Maccaroni                    #
-- #                      Matricola:321621                      #
-- #                     e Angelo Labbruzzo                     #
-- #                      Matricola:319928                      #
-- #                    Anno di corso: terzo                    #
-- ##############################################################

{-
    Specifica: Scrivere un programma Haskell e un programma Prolog che acquisiscano da tastiera
    una scheda del Futoshiki in formato NxN e le eventuali disuguaglianze tra le celle.
    Lo scopo del programma    quello di risolvere il Futoshiki, trovando dove    possibile una
    scheda nella quale su ogni riga e colonna ogni valore, che va da 1 a N, appare esattamente
    una volta e le disuguaglianze vengono rispettate.
-}

module Main where

-- Caricamento delle funzioni necessarie per operazioni con le matrici.
import Data.Matrix (Matrix , prettyMatrix , ncols , nrows , (!) , getRow , getCol , transpose ,
    fromLists , setElem, toLists , toList)

-- Caricamento della funzione per convertire un vettore in una lista di elementi.
import qualified Data.Vector as V (toList)

-- Caricamento della funzione per eliminare i duplicati da una lista di elementi.
import Data.List (nub)

{- Dato per le posizioni nella griglia. -}
type Posizione       = (Int, Int)

{- Definizione di un tipo di dato per un vincolo di disuguaglianza. -}
data Ordinamento     = Maggiore | Minore | Errore deriving (Show, Eq)

{- Dato per un vincolo tra due posizioni. -}
type Disuguaglianza  = (Posizione, Ordinamento, Posizione)

{- Dato per un intero Futoshiki. -}
type Futoshiki       = (Int,Matrix Int, [Disuguaglianza])

{- Funzione principale del programma. -}
main :: IO ()
main = do
        putStrLn "Il seguente programma determina la soluzione del Futoshiki in base alla       "
        putStrLn "scheda data e le disuguaglianze da verificare.                                "
        putStrLn "Inserire la scheda della dimensione che vuoi basta che sia del tipo NxN.      "
        putStrLn "Ecco un esempio di una scheda 3x3 dove i valori con 0 rappresentano il vuoto: "
        putStrLn " 0 2 0 <Invio>                                                                "
        putStrLn " 0 3 0 <Invio>                                                                "
        putStrLn " 0 0 0 <Invio>                                                                "
        putStrLn "Premere invio nella riga vuota per terminare l'input.                         "
        scheda <- leggiSchedaConValidazione
        putStrLn "Per inserire le disuguaglianze inserirle nel seguente formato 'x1 y1 c x2 y2' "
        putStrLn "Ecco un esempio di un inserimento di disuguaglianze:                          "
        putStrLn " 1 1 > 1 2 <Invio>                                                            "
        putStrLn " 2 2 < 2 3 <Invio>                                                            "
        putStrLn "Premere invio nella riga vuota per terminare l'input.                         "
        disuguaglianze <- leggiDisuguaglianzeConValidazione (nrows scheda)
        putStrLn "Futoshiki da risolvere:"
        putStrLn (prettyMatrix scheda)
        let soluzioni = risolviFutoshiki (nrows scheda, scheda, disuguaglianze)
        mostraRisultato soluzioni
    where
        mostraRisultato soluzioni
            | null soluzioni = putStrLn "La soluzione non esiste."
            | otherwise = do
                        putStrLn "Soluzione trovata:"
                        putStrLn (prettyMatrix (head soluzioni))
```

```haskell

{- Funzione che fa la validazione stretta della scheda controllando che essa sia scritta in
   maniera conforme per essere risolta. -}
leggiSchedaConValidazione :: IO (Matrix Int)
leggiSchedaConValidazione = do
                            putStrLn "Inserire la scheda:  "
                            righe <- leggiScheda []
                            processaRighe righe
    where
        processaRighe [] = do
                           putStrLn "Errore: Scheda vuota!"
                           leggiSchedaConValidazione
        processaRighe righe = validaRighe righe
        validaRighe righe
            | not (all (== head lunghezze) lunghezze) = erroreQuadrata
            | head lunghezze /= length righe = erroreQuadrata
            | otherwise = validaMatrice (fromLists (reverse righe))
            where
                lunghezze = map length righe
                erroreQuadrata = do
                                 putStrLn "Errore: La scheda deve essere quadrata (NxN)!"
                                 leggiSchedaConValidazione
        validaMatrice matrice
            | validaSchedaIniziale matrice = return matrice
            | otherwise = do
                          putStrLn "Errore: La scheda contiene valori non validi!"
                          leggiSchedaConValidazione

{- Funzione che legge la tabella del Futoshiki da voler risolvere. -}
leggiScheda :: [[Int]] -> IO[[Int]]
leggiScheda scheda = do
                     stringa <- getLine
                     leggiScheda' stringa scheda
    where
        leggiScheda' stringa scheda
            | null stringa = return scheda
            | otherwise = do
                          let riga = map read (words stringa) :: [Int]
                          leggiScheda (riga:scheda)

{- Funzione che controlla che nella scheda non ci siano valori che non potrebbero esserci.
   - l'argomento    la scheda del Futoshiki. -}
validaSchedaIniziale :: Matrix Int -> Bool
validaSchedaIniziale scheda = let valori = toList scheda
                                  dimensione = nrows scheda
                              in all (\x -> x >= 0 && x <= dimensione) valori

{- Funzione che fa la validazione stretta delle disuguaglianze controllando che esse siano
   scritte in maniera conforme alle regole e alla scheda.
   - l'argomento    la dimensione della scheda per eseguire i dovuti controlli. -}
leggiDisuguaglianzeConValidazione :: Int -> IO [Disuguaglianza]
leggiDisuguaglianzeConValidazione d = do
                                      putStrLn " Inserire le disuguaglianze: "
                                      disuguaglianze <- leggiDisuguaglianze []
                                      validaDidugaglianze disuguaglianze
    where
        validaDisuguaglianze [] = return []
        validaDidugaglianze dis
            | not (all validaDisuguaglianza dis) = do
                                                   putStrLn "Errore: le posizioni o gli operatori
                                                       non sono corretti"
                                                   leggiDisuguaglianzeConValidazione d
            | any stessaCella dis = do
                                    putStrLn "Errore: una disuguaglianza e' sulla stessa cella"
                                    leggiDisuguaglianzeConValidazione d
            | not (all celleAdiacenti dis) = do
                                             putStrLn "Errore: una disuguaglianza non punta su una
                                                 cella adiacente "
                                             leggiDisuguaglianzeConValidazione d
            | otherwise = return dis
        validaDisuguaglianza ((r1,c1), ord, (r2,c2)) =
            controllaRange r1 && controllaRange c1 && controllaRange r2 && controllaRange c2 &&
            (ord == Maggiore || ord == Minore)
        controllaRange p =
            p >= 1 && p <= d
        stessaCella ((r1,c1), _, (r2,c2)) =
            r1 == r2 && c1 == c2
        celleAdiacenti ((x1, y1),_, (x2, y2)) =
            (abs (x1 - x2) == 1 && y1 == y2) || (abs (y1 - y2) == 1 && x1 == x2)
```

```haskell

{- Funzione che legge le disuguaglianze del Futoshiki da dover verificare. -}
leggiDisuguaglianze :: [Disuguaglianza] -> IO [Disuguaglianza]
leggiDisuguaglianze disuguaglianze = do
                                     stringa <- getLine
                                     leggiDisuguaglianza stringa disuguaglianze
    where
        leggiDisuguaglianza stringa disuguaglianze
            | null stringa = return disuguaglianze
            | length (words stringa) /= 5 = do
                                            putStrLn "Errore: formato non valido. Inserire
                                                esattamente 5 elementi "
                                            leggiDisuguaglianze disuguaglianze
            | otherwise = do
                          let [x1, y1, c, x2, y2] = words stringa
                          let ord = gestisciOrd (head c)
                          let disuguaglianza = ((read x1, read y1),ord,(read x2, read y2))
                          leggiDisuguaglianze (disuguaglianza : disuguaglianze)
        gestisciOrd ord
            | ord == '>' = Maggiore
            | ord == '<' = Minore
            | otherwise  = Errore

{-  Funzione che determina la soluzione del Futoshiki.
    - l'argomento    il Futoshiki. -}
risolviFutoshiki :: Futoshiki  -> [Matrix Int]
risolviFutoshiki (dimensione, scheda, disuguaglianze)
    | not (validaScheda scheda disuguaglianze) = []
    | verificaCompletezza scheda = [scheda]
    | otherwise = do
                  let (riga, colonna) = trovaCellaVuota scheda
                  let nUsati = numeriUsati scheda riga colonna
                  nDisponibili <- rimuovi [1 .. dimensione] nUsati
                  let nuovaScheda = setElem nDisponibili (riga,colonna) scheda
                  risolviFutoshiki (dimensione, nuovaScheda , disuguaglianze)

{- Funzione che determina se una scheda del Futoshiki sia completa quindi che tutti i numeri al
   suo interno siano diversi da zero.
   - l'argomento    la scheda del Futoshiki. -}
verificaCompletezza :: Matrix Int -> Bool
verificaCompletezza scheda = all (/= 0) (toList scheda)

{- Funzione che trova nella scheda la prima cella vuota.
    - l'argomento    la scheda del Futoshiki. -}
trovaCellaVuota :: Matrix Int -> Posizione
trovaCellaVuota scheda
    | null celle = (-1,-1)
    | otherwise = head celle
    where
        celle = [(i, j) | i <- [1 .. nrows scheda], j <- [1 .. ncols scheda], scheda ! (i, j) == 0]

{- Funzione che trova tutti i numeri gi  presenti nella riga e colonna selezionata.
   - il primo argomento    la scheda del Futoshiki.
   - il secondo argomento    il numero della riga.
   - il terzo argomento    il numero della colonna. -}
numeriUsati :: Matrix Int -> Int -> Int -> [Int]
numeriUsati scheda riga colonna = let numeriInRiga = V.toList (getRow riga scheda)
                                      numeriInColonna = V.toList (getCol colonna scheda)
                                      in nub (filter (/= 0) (numeriInRiga ++ numeriInColonna))

{- Funzione che rimuove da una lista tutti gli elementi di un'altra lista.
   - il primo argomento    la lista da cui rimuovere gli elementi.
   - il secondo argomento    la lista degli elementi da rimuovere. -}
rimuovi :: [Int] -> [Int] -> [Int]
rimuovi x []      = x
rimuovi x (y:ys)  = rimuovi (filter (/= y) x) ys

{- Funzione che controlla se la scheda rispetta tutte le disuguaglianze e
   che non ci siano duplicati nelle righe e nelle colonne.
   - il primo argomento    la scheda del Futoshiki.
   - il secondo argomento    lista di tutte le disuguaglianze da controllare. -}
validaScheda :: Matrix Int -> [Disuguaglianza] -> Bool
validaScheda scheda disuguaglianze = controllaDisuguaglianze scheda disuguaglianze &&
                                     controllaDuplicati scheda &&
                                     controllaDuplicati (transpose scheda)
    where
        controllaDuplicati scheda = all (\riga -> let rigaFiltrata = filter (/= 0) riga
                                      in nub rigaFiltrata == rigaFiltrata) (toLists scheda)

```

```haskell
{- Funzione che controlla se la scheda rispetta tutte le disuguaglianze.
   - il primo argomento    la scheda del Futoshiki.
   - il secondo argomento    lista di tutte le disuguaglianze da controllare -}
controllaDisuguaglianze :: Matrix Int-> [Disuguaglianza] -> Bool
controllaDisuguaglianze scheda []     = True
controllaDisuguaglianze scheda (d:ds) = controllaDisuguaglianza scheda d && controllaDisuguaglianze
    scheda ds
    where
        controllaDisuguaglianza scheda ((r1,c1), ord, (r2,c2)) = let val1 = scheda ! (r1,c1)
                                                                     val2 = scheda ! (r2,c2)
                                                                     in val1 == 0 || val2 == 0 ||
                                                                        gestisciOrd ord val1 val2
        gestisciOrd Minore v1 v2 = v1 < v2
        gestisciOrd Maggiore v1 v2 = v1 > v2
```

Listing 1: Codice principale Haskell

## 4.2 Implementazione in Prolog

**File sorgente futoshiki.pl:**

```
/*#############################################################
  #        Corso di Programmazione Logica e Funzionale        #
  #       Progetto per la sessione estiva A.A. 2024/2025      #
  #                  di Tommaso Maccaroni                     #
  #                    Matricola:321621                       #
  #                   e Angelo Labbruzzo                      #
  #                    Matricola:319928                       #
  #                  Anno di corso: terzo                     #
  #############################################################*/

/*
    Specifica: Scrivere un programma Haskell e un programma Prolog che acquisiscano da tastiera
    una scheda del Futoshiki in formato NxN e le eventuali disuguaglianze tra le celle.
    Lo scopo del programma    quello di risolvere il Futoshiki, trovando dove    possibile una
    scheda nella quale su ogni riga e colonna ogni valore, che va da 1 a N, appare esattamente
    una volta e le disuguaglianze vengono rispettate.
*/

/* Predicato principale del programma. */
main :-
    write('Il seguente programma determina la soluzione del Futoshiki in base alla scheda  '), nl,
    write('data e le disuguaglianze da verificare.                                         '), nl,
    write('Inserire la scheda della dimensione che vuoi basta che sia del tipo NxN.        '), nl,
    write('Ecco un esempio di una scheda 3x3 dove i valori con 0 rappresentano il vuoto:   '), nl,
    write('[[0,0,0],[0,0,0],[0,0,0]]. <Invio>                                              '), nl,
    write('Si ricorda il punto finale prima dell\'invio e si raccomanda di seguire questa  '), nl,
    write('convenzione per evitare comportamenti anomali del programma.                    '), nl,
    leggiSchedaValidata(Scheda),
    write('Per inserire le disuguaglianze inserirle nel seguente formato \'x1 y1 c x2 y2\' '), nl,
    write('Ecco un esempio di un inserimento di disuguaglianze                             '), nl,
    write('[[1,1,<,1,2],[1,2,>,1,3]]. <Invio>                                              '), nl,
    write('Si ricorda il punto finale prima dell\'invio e si raccomanda di seguire questa  '), nl,
    write('convenzione per evitare comportamenti anomali del programma.                    '), nl,
    length(Scheda, D),
    leggiDisuguaglianzeValidate(D, Disuguaglianze),
    write('Futoshiki da risolvere:'), nl,
    stampaScheda(Scheda),
    risolviERispondi(Scheda, Disuguaglianze, D).

/* Predicato che serve per acquisire la scheda e a validarla. */
leggiSchedaValidata(Scheda) :-
    write('Inserire la scheda: '),
    read(SchedaInput),
    processaSchedaInput(SchedaInput,Scheda).

/* Predicato che serve a processare la scheda.
   - Il parametro    la scheda del Futoshiki da processare.
   Caso 1: Quando la scheda inserita    valida, il predicato unifica le due schede.
   Caso 2: Quando la scheda inserita non    valida, il predicato stampa un messaggio d'errore
           e la fa reinserire.*/
processaSchedaInput(SchedaInput, Scheda) :-
    validaSchedaInput(SchedaInput),
    !,
    Scheda = SchedaInput.
processaSchedaInput(_, Scheda) :-
    write('Errore: La scheda deve essere quadrata (NxN) e contenere solo valori 0-D!'), nl,
    leggiSchedaValidata(Scheda).

/* Predicato che serve a validare la scheda precedentemente processata
   - Il parametro    la scheda del Futoshiki da validare. */
validaSchedaInput(Scheda) :-
    is_list(Scheda),
    Scheda \= [],
    length(Scheda, D),
    validaRighe(Scheda, D).

/* Predicato che serve a validare le righe della scheda.
    - Il primo parametro    la scheda del Futoshiki da validare.
    - Il secondo parametro    la dimensione dalla scheda. */
validaRighe([], _).
validaRighe([Riga|Resto], D) :-
    is_list(Riga),
    length(Riga, D),
    validaRiga(Riga, D),
```

```
75      validaRighe(Resto, D).
76
77  /* Predicato che serve a validare una riga.
78     - Il primo parametro    la riga da validare.
79     - Il secondo parametro    la dimensione della scheda. */
80  validaRiga([], _).
81  validaRiga([X|Resto], D) :-
82      integer(X),
83      X >= 0,
84      X =< D ,
85      validaRiga(Resto, D).
86
87  /* Predicato che serve per acquisire le disuguaglianze e a validarle.
88      - Il parametro    la dimensione della scheda. */
89  leggiDisuguaglianzeValidate(D, Disuguaglianze) :-
90      write('Inserire le disuguaglianze: '),
91      read(DisInput),
92      processaDisuguaglianzeInput(DisInput, D, Disuguaglianze).
93
94  /* Predicato che serve a processare le disuguaglianze.
95     - Il primo parametro sono le disuguaglianze da processare.
96     - il secondo parametro    la dimensione della scheda.
97     Caso 1: Quando le disuguaglianze inserite sono valide, il predicato unifica le disuguaglianze
98     Caso 2: Quando le disuguaglianze inserite non sono valide, il predicato stampa
99             un messaggio d'errore e le fa reinserire. */
100 processaDisuguaglianzeInput(DisInput, D, Disuguaglianze) :-
101     validaDisuguaglianzeInput(DisInput, D),
102     !,
103     Disuguaglianze = DisInput.
104 processaDisuguaglianzeInput(_, D  ,Disuguaglianze) :-
105     write('Errore nelle disuguaglianze! Verificare:'), nl,
106     write('- Formato corretto: [[R1,C1,Op,R2,C2], ...]'), nl,
107     write('- Posizioni valide (1 a '), write(D ), write(')'), nl,
108     write('- Operatori > o <'), nl,
109     write('- Celle adiacenti'), nl,
110     write('- Celle diverse'), nl,
111     leggiDisuguaglianzeValidate(D, Disuguaglianze ).
112
113 /* Predicato che serve a verificare che il dato in esame sia una lista di disuguaglianze.
114     - Il primo parametro sono le disuguaglianze da validare.
115     - il secondo parametro    la dimensione della scheda. */
116 validaDisuguaglianzeInput(Disuguaglianze, D ) :-
117     is_list(Disuguaglianze),
118     validaDisuguaglianze(Disuguaglianze, D ).
119
120 /* Predicato che serve a validare tutte le disuguaglianze.
121     - Il primo parametro sono le disuguaglianze da validare.
122     - il secondo parametro    la dimensione della scheda. */
123 validaDisuguaglianze([], _).
124 validaDisuguaglianze([Dis|Resto], D ) :-
125     validaDisuguaglianza(Dis, D ),
126     validaDisuguaglianze(Resto, D ).
127
128 /* Predicato che serve a validare una singola disuguaglianza.
129     - Il primo parametro    la disuguaglianza da validare.
130     - il secondo parametro    la dimensione della scheda. */
131 validaDisuguaglianza([R1,C1,Op,R2,C2], D ) :-
132     R1 >= 1, R1 =< D ,
133     C1 >= 1, C1 =< D ,
134     R2 >= 1, R2 =< D ,
135     C2 >= 1, C2 =< D ,
136     member(Op, [>, <]),
137     celleAdiacenti(R1, C1, R2, C2),
138     \+ (R1 = R2, C1 = C2).
139
140 /* Predicato che serve a controllare se due celle sono adiacenti.
141     - Il primo parametro    la colonna del primo valore.
142     - il secondo parametro    la riga del primo valore.
143     - Il terzo parametro    la colonna del secondo valore.
144     - il quarto parametro    la riga del secondo valore.
145     Caso 1: Quando le celle si trovano sulla stessa colonna,
146             il predicato controlla che siano adiacenti.
147     Caso 2: Quando le celle si trovano sulla stessa riga,
148             il predicato controlla che siano adiacenti.*/
149 celleAdiacenti(R1, C1, R2, C2) :-
150     (abs(R1 - R2) =:= 1, C1 =:= C2).
151 celleAdiacenti(R1, C1, R2, C2) :-
152     (abs(C1 - C2) =:= 1, R1 =:= R2).
153
```

```
/* Predicato che serve per stampare una scheda.
   - Il parametro    la scheda del Futoshiki da stampare. */
stampaScheda(Scheda) :-
    maplist(stampaRiga, Scheda), nl.

/* Predicato che serve per stampare una riga della scheda.
   - Il parametro    la riga della scheda da stampare. */
stampaRiga(Riga) :-
    maplist(stampaCella, Riga), nl.

/* Predicato che serve per stampare una cella della scheda.
   - Il parametro    il valore della scheda da stampare. */
stampaCella(Valore) :-
    write(Valore), write(' ').

/* Predicato che serve a risolvere il Futoshiki e a rispondere di conseguenza.
   - Il primo parametro    la scheda del Futoshiki da risolvere.
   - Il secondo parametro sono le disuguaglianze che devono essere rispettate.
   - Il terzo parametro    la dimensione della scheda.
   Caso 1: Se dai dati inserite    possibile generare una soluzione, il predicato la stampa.
   Caso 2: Se dai dati inserite non    possibile generare una soluzione,
           il predicato stampa l'insuccesso del programma. */
risolviERispondi(Scheda, Disuguaglianze, D) :-
    risolviFutoshiki(Scheda, Disuguaglianze, D, SoluzioneScheda),
    !,
    write('Soluzione trovata:'), nl,
    stampaScheda(SoluzioneScheda).
risolviERispondi(_, _, _) :-
    write('La soluzione non esiste.'), nl.

/* Predicato che serve per determinare la soluzione del Futoshiki.
   - Il primo parametro    la scheda del Futoshiki da risolvere.
   - Il secondo parametro sono le disuguaglianze del Futoshiki da controllare.
   - il terzo parametro    la dimensione della scheda. */
risolviFutoshiki(Scheda, Disuguaglianze, D , Soluzione) :-
    completaScheda(Scheda, D , SchedaCompleta),
    validaSoluzione(SchedaCompleta, Disuguaglianze, D ),
    Soluzione = SchedaCompleta.

/* Predicato che serve per completare la scheda del Futoshiki.
   - Il primo parametro    la scheda del Futoshiki da completare.
   - il secondo parametro    la dimensione della scheda. */
completaScheda([], _, []).
completaScheda([Riga|Resto], D , [RigaCompleta|RestoCompleto]) :-
    completaRiga(Riga, D , RigaCompleta),
    completaScheda(Resto, D , RestoCompleto).

/* Predicato che serve per completare la riga del Futoshiki.
   - Il primo parametro    la riga del Futoshiki da completare.
   - il secondo parametro    la dimensione della scheda. */
completaRiga([], _, []).
completaRiga([0|Resto], D , [X|RestoCompleto]) :-
    between(1, D , X),
    completaRiga(Resto, D , RestoCompleto).
completaRiga([X|Resto], D , [X|RestoCompleto]) :-
    X \= 0,
    completaRiga(Resto, D , RestoCompleto).

/* Predicato che controlla se la scheda rispetta tutte le disuguaglianze e
   che non ci siano duplicati nelle righe e nelle colonne.
   - Il primo parametro    la scheda del Futoshiki da validare.
   - Il secondo parametro sono le disuguaglianze da verificare.
   - Il terzo parametro    la dimensione della scheda*/
validaSoluzione(Scheda, Disuguaglianze, D ) :-
    controllaDuplicati(Scheda, D ),
    trasponi(Scheda, SchedaTraspos),
    controllaDuplicati(SchedaTraspos, D ),
    controllaDisuguaglianze(Scheda, Disuguaglianze).

/* Predicato che controlla se nella scheda ci sono duplicati.
   - Il primo parametro    la scheda del Futoshiki da controllare.
   - Il secondo parametro    la dimensione della scheda */
controllaDuplicati([], _).
controllaDuplicati([Riga|Resto], D ) :-
    controllaInRiga(Riga, D ),
    controllaDuplicati(Resto, D ).

/* Predicato che controlla se nella riga ci sono duplicati.
    - Il primo parametro    la riga della scheda del Futoshiki da controllare.
```

```
    - Il secondo parametro    la dimensione della scheda */
controllaInRiga(Riga, D ) :-
    sort(Riga, RigaOrdinata),
    generaLista(D , ListaCompleta),
    RigaOrdinata = ListaCompleta.

/* Predicato che genera una lista di numeri da 1 a N.
   - Il parametro    il valore N. */
generaLista(1, [1]).
generaLista(N , Lista) :-
    N > 1, N1 is N - 1,
    generaLista(N1, ListaParziale),
    append(ListaParziale, [N], Lista).

/* Predicato che traspone la scheda del Futoshiki.
   - Il parametro    la scheda del Futoshiki da risolvere. */
trasponi([], []).
trasponi([[]|_], []).
trasponi(Matrice, [Prima|Resto]) :-
    trasponiRiga(Matrice, Prima, MatriceResto),
    trasponi(MatriceResto, Resto).

/* Predicato che traspone la riga della scheda del Futoshiki.
   - Il primo parametro    la scheda del Futoshiki da risolvere.
   - il secondo parametro    la riga della scheda da transporre. */
trasponiRiga([], [], []).
trasponiRiga([[H|T]|Resto], [H|PrimaCol], [T|MatriceResto]) :-
    trasponiRiga(Resto, PrimaCol, MatriceResto).

/* Predicato che controlla se la scheda rispetta tutte le disuguaglianze.
   - Il primo parametro    la scheda del Futoshiki.
   - Il secondo parametro sono le disuguaglianze da verificare. */
controllaDisuguaglianze(_, []).
controllaDisuguaglianze(Scheda, [Dis|Resto]) :-
    controllaDisuguaglianza(Scheda, Dis),
    controllaDisuguaglianze(Scheda, Resto).

/* Predicato che controlla se la scheda rispetta una singola disuguaglianza.
   - Il primo parametro    la scheda del Futoshiki.
   - Il secondo parametro    la disuguaglianza da verificare. */
controllaDisuguaglianza(Scheda, [R1,C1,Op,R2,C2]) :-
    ottieniValore(Scheda, R1, C1, Val1),
    ottieniValore(Scheda, R2, C2, Val2),
    applicaOperatore(Op, Val1, Val2).

/* Predicato che serve ad ottenere il valore in una determinata posizione della scheda.
   - Il primo parametro    la scheda del Futoshiki.
   - Il secondo parametro    la riga in cui si trova il valore.
   - Il terzo parametro    la colonna in cui si trova il valore.*/
ottieniValore(Scheda, R, C, Valore) :-
    nth1(R, Scheda, Riga),
    nth1(C, Riga, Valore).

/* Predicato che applica l'operatore inserito alla disuguaglianza
   - Il primo parametro    l'operatore della disuguaglianza.
   - Il secondo parametro    il primo valore della disuguaglianza.
   - Il terzo parametro    il secondo valore della disuguaglianza.
   Caso 1: Se l'operatore inserito    ">", il predicato lo applica alla disuguaglianza
   Caso 2: Se l'operatore inserito    "<", il predicato lo applica alla disuguaglianza. */
applicaOperatore(>, V1, V2) :-
    V1 > V2.
applicaOperatore(<, V1, V2) :-
    V1 < V2.
```

Listing 2: Codice principale Prolog

# 5 Testing

## 5.1 Testing del Programma Haskell

Figura 1: Test nr. 1

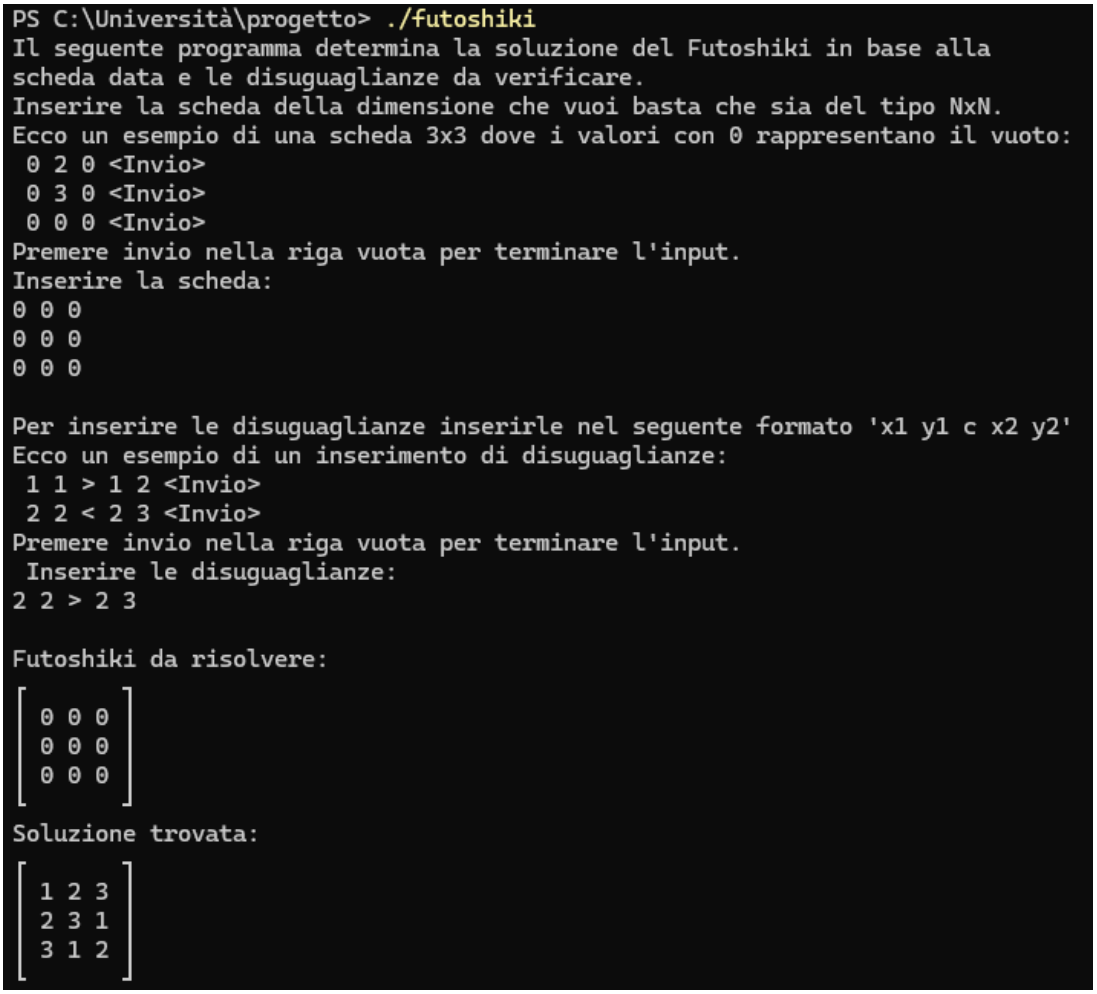

Figura 2: Test nr. 2

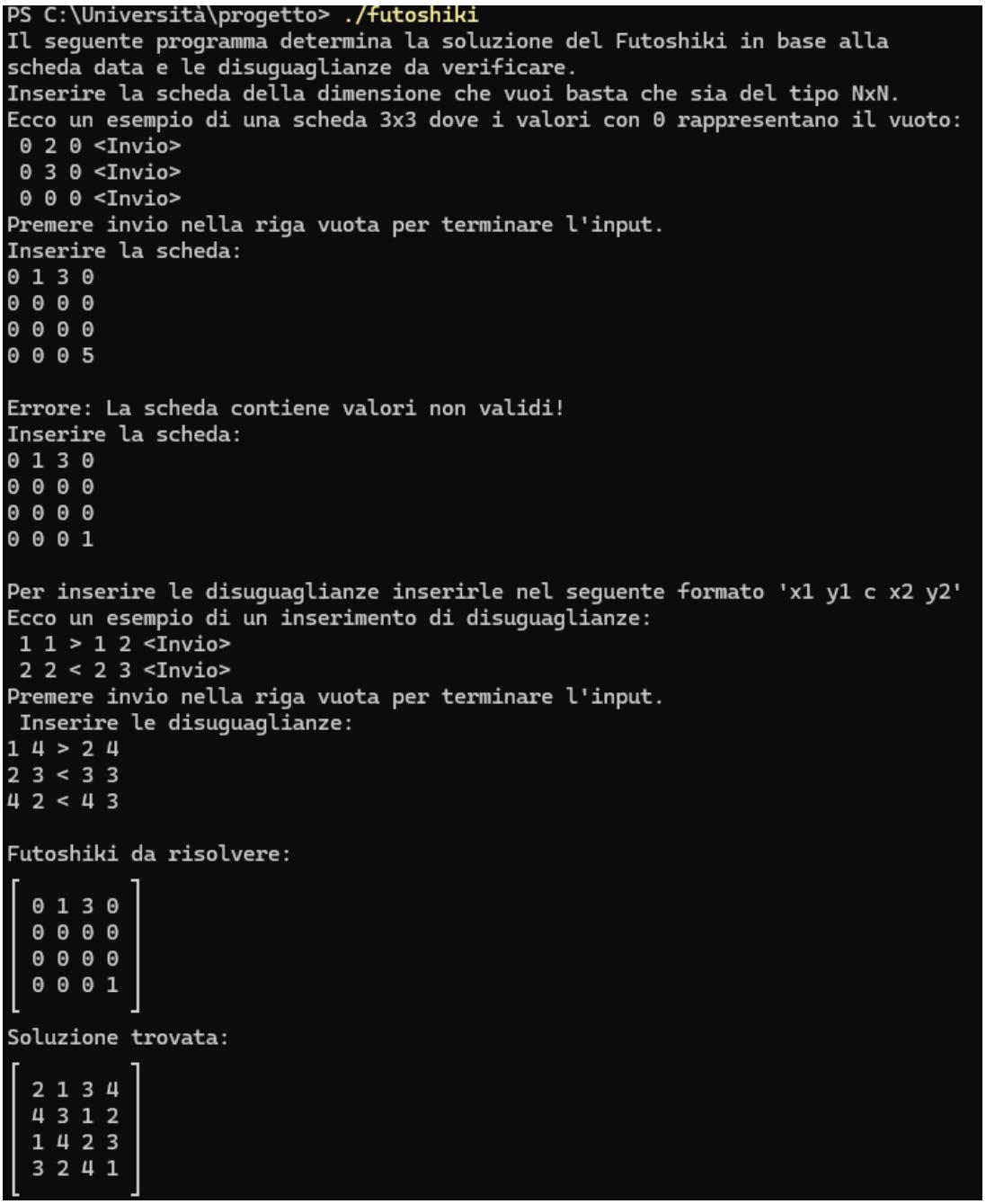

Figura 3: Test nr. 3

```
PS C:\Università\progetto> ./futoshiki
Il seguente programma determina la soluzione del Futoshiki in base alla
scheda data e le disuguaglianze da verificare.
Inserire la scheda della dimensione che vuoi basta che sia del tipo NxN.
Ecco un esempio di una scheda 3x3 dove i valori con 0 rappresentano il vuoto:
 0 2 0 <Invio>
 0 3 0 <Invio>
 0 0 0 <Invio>
Premere invio nella riga vuota per terminare l'input.
Inserire la scheda:
4 0 0 0
0 0 2 0
0 0 0 0
0 0 0

Errore: La scheda deve essere quadrata (NxN)!
Inserire la scheda:
4 0 0 0
0 0 2 0
0 0 0 0
0 0 0 1

Per inserire le disuguaglianze inserirle nel seguente formato 'x1 y1 c x2 y2'
Ecco un esempio di un inserimento di disuguaglianze:
 1 1 > 1 2 <Invio>
 2 2 < 2 3 <Invio>
Premere invio nella riga vuota per terminare l'input.
 Inserire le disuguaglianze:
2 2 > 2 3

Futoshiki da risolvere:
┌         ┐
│ 4 0 0 0 │
│ 0 0 2 0 │
│ 0 0 0 0 │
│ 0 0 0 1 │
└         ┘
Soluzione trovata:
┌         ┐
│ 4 1 3 2 │
│ 1 3 2 4 │
│ 2 4 1 3 │
│ 3 2 4 1 │
└         ┘
```

Figura 4: Test nr. 4

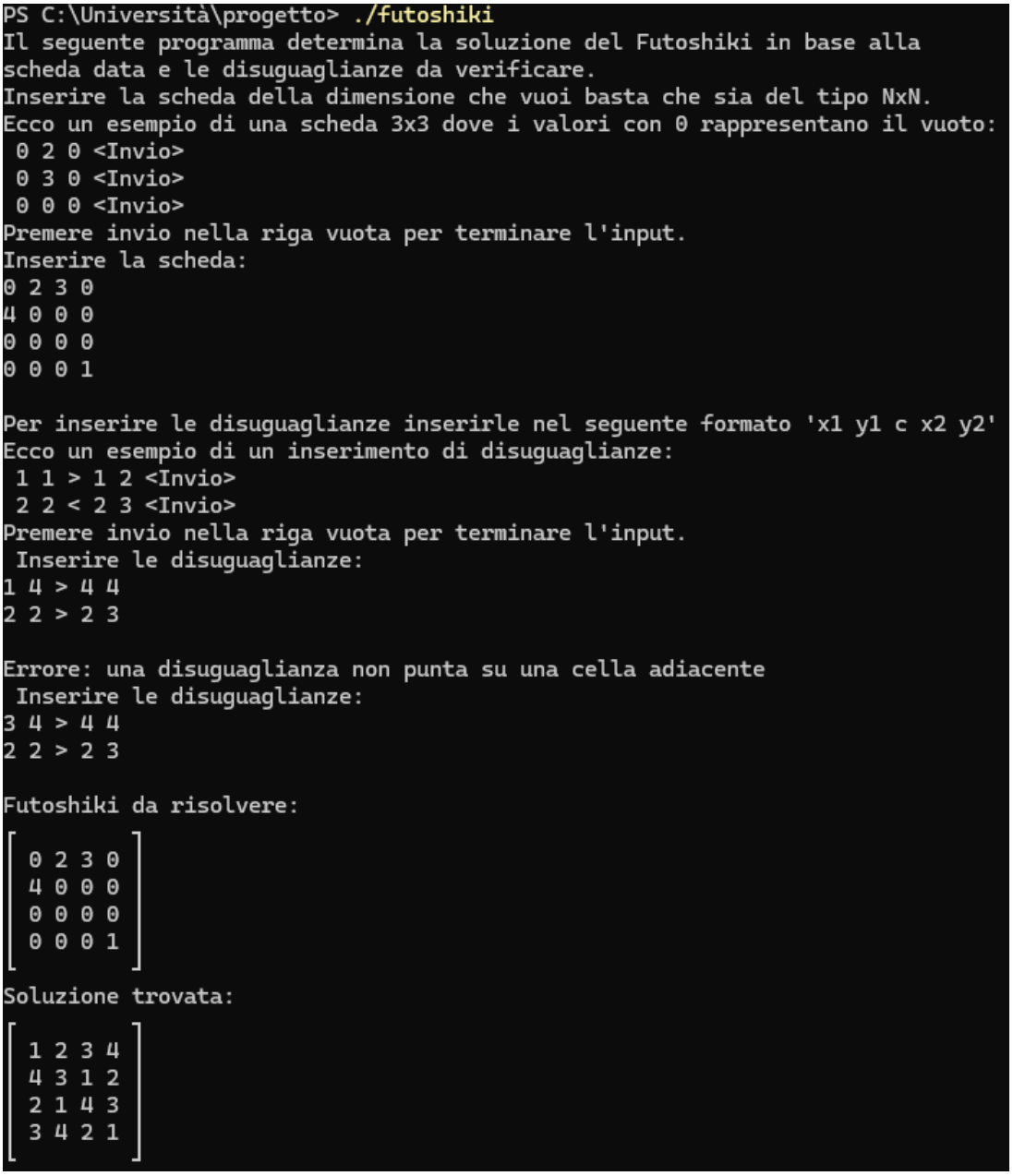

Figura 5: Test nr. 5

```
PS C:\Università\progetto> ./futoshiki
Il seguente programma determina la soluzione del Futoshiki in base alla
scheda data e le disuguaglianze da verificare.
Inserire la scheda della dimensione che vuoi basta che sia del tipo NxN.
Ecco un esempio di una scheda 3x3 dove i valori con 0 rappresentano il vuoto:
 0 2 0 <Invio>
 0 3 0 <Invio>
 0 0 0 <Invio>
Premere invio nella riga vuota per terminare l'input.
Inserire la scheda:
2 0 3 0 0
0 4 2 5 3
0 5 0 0 0
4 3 0 1 2
5 0 2 0 4

Per inserire le disuguaglianze inserirle nel seguente formato 'x1 y1 c x2 y2'
Ecco un esempio di un inserimento di disuguaglianze:
 1 1 > 1 2 <Invio>
 2 2 < 2 3 <Invio>
Premere invio nella riga vuota per terminare l'input.
 Inserire le disuguaglianze:
1 1 < 1 2
1 1 > 1 2

Futoshiki da risolvere:
┌           ┐
│ 2 0 3 0 0 │
│ 0 4 2 5 3 │
│ 0 5 0 0 0 │
│ 4 3 0 1 2 │
│ 5 0 2 0 4 │
└           ┘
La soluzione non esiste.
```

Figura 6: Test nr. 6

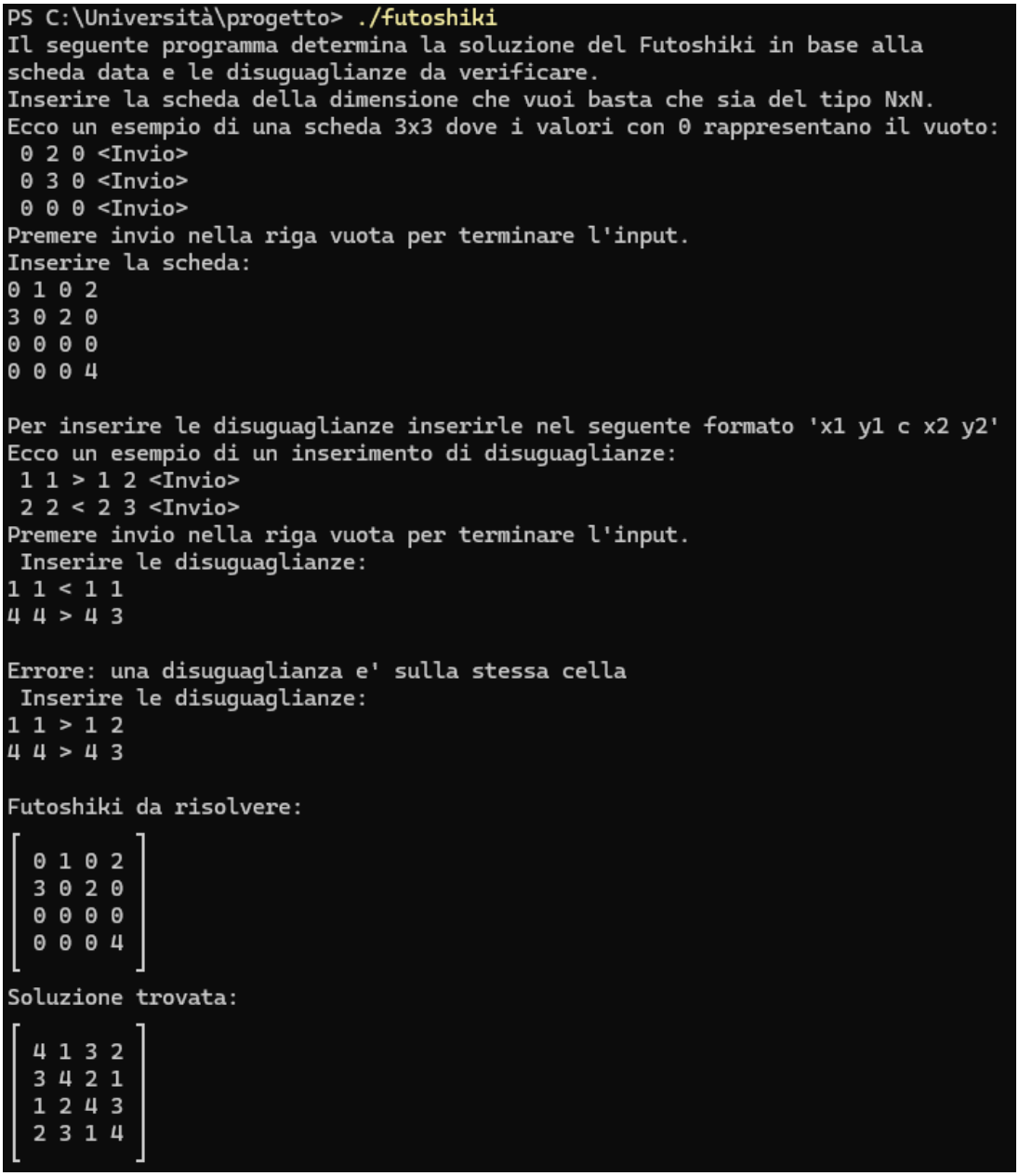

Figura 7: Test nr. 7

```
PS C:\Università\progetto> ./futoshiki
Il seguente programma determina la soluzione del Futoshiki in base alla
scheda data e le disuguaglianze da verificare.
Inserire la scheda della dimensione che vuoi basta che sia del tipo NxN.
Ecco un esempio di una scheda 3x3 dove i valori con 0 rappresentano il vuoto:
 0 2 0 <Invio>
 0 3 0 <Invio>
 0 0 0 <Invio>
Premere invio nella riga vuota per terminare l'input.
Inserire la scheda:
2 0 3 0 1
0 1 0 0 5
3 0 1 5 4
5 3 0 1 0
0 4 0 0 3

Per inserire le disuguaglianze inserirle nel seguente formato 'x1 y1 c x2 y2'
Ecco un esempio di un inserimento di disuguaglianze:
 1 1 > 1 2 <Invio>
 2 2 < 2 3 <Invio>
Premere invio nella riga vuota per terminare l'input.
 Inserire le disuguaglianze:

Futoshiki da risolvere:
⎡ 2 0 3 0 1 ⎤
⎢ 0 1 0 0 5 ⎥
⎢ 3 0 1 5 4 ⎥
⎢ 5 3 0 1 0 ⎥
⎣ 0 4 0 0 3 ⎦
Soluzione trovata:
⎡ 2 5 3 4 1 ⎤
⎢ 4 1 2 3 5 ⎥
⎢ 3 2 1 5 4 ⎥
⎢ 5 3 4 1 2 ⎥
⎣ 1 4 5 2 3 ⎦
```

Figura 8: Test nr. 8

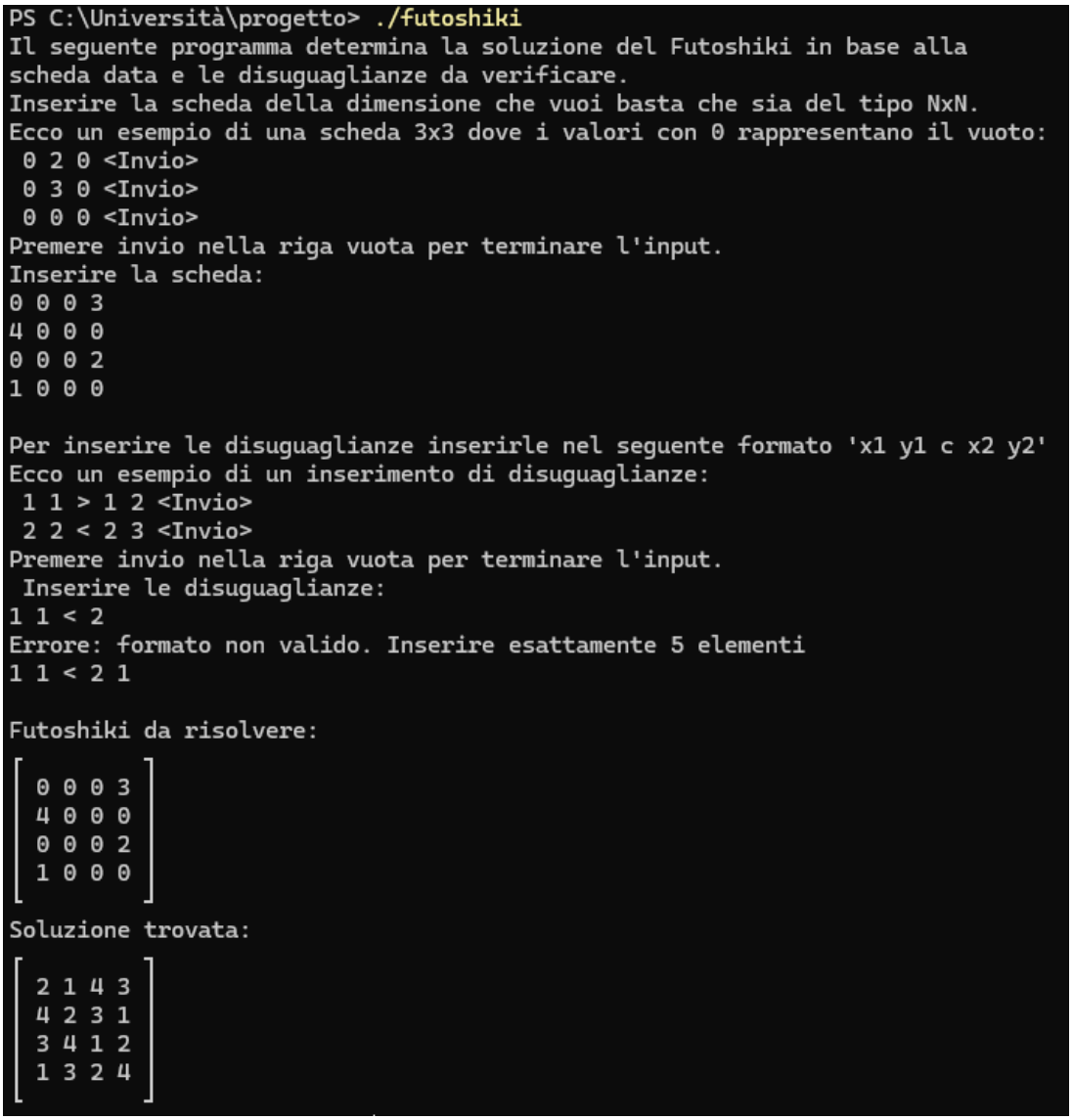

Figura 9: Test nr. 9

Figura 10: Test nr. 10

```
PS C:\Università\progetto> ./futoshiki
Il seguente programma determina la soluzione del Futoshiki in base alla
scheda data e le disuguaglianze da verificare.
Inserire la scheda della dimensione che vuoi basta che sia del tipo NxN.
Ecco un esempio di una scheda 3x3 dove i valori con 0 rappresentano il vuoto:
 0 2 0 <Invio>
 0 3 0 <Invio>
 0 0 0 <Invio>
Premere invio nella riga vuota per terminare l'input.
Inserire la scheda:
0 0 0 3
0 0 0
0 0 0 2
1 0 0 0

Errore: La scheda deve essere quadrata (NxN)!
Inserire la scheda:
0 0 0 3
0 0 0 0
0 0 0 2
1 0 0 0

Per inserire le disuguaglianze inserirle nel seguente formato 'x1 y1 c x2 y2'
Ecco un esempio di un inserimento di disuguaglianze:
 1 1 > 1 2 <Invio>
 2 2 < 2 3 <Invio>
Premere invio nella riga vuota per terminare l'input.
 Inserire le disuguaglianze:
1 1 > 3 3

Errore: una disuguaglianza non punta su una cella adiacente
 Inserire le disuguaglianze:
1 1 > 1 2

Futoshiki da risolvere:
┌         ┐
│ 0 0 0 3 │
│ 0 0 0 0 │
│ 0 0 0 2 │
│ 1 0 0 0 │
└         ┘
Soluzione trovata:
┌         ┐
│ 2 1 4 3 │
│ 3 4 2 1 │
│ 4 3 1 2 │
│ 1 2 3 4 │
└         ┘
```

## 5.2 Testing del Programma Prolog

Figura 11: Test nr. 1

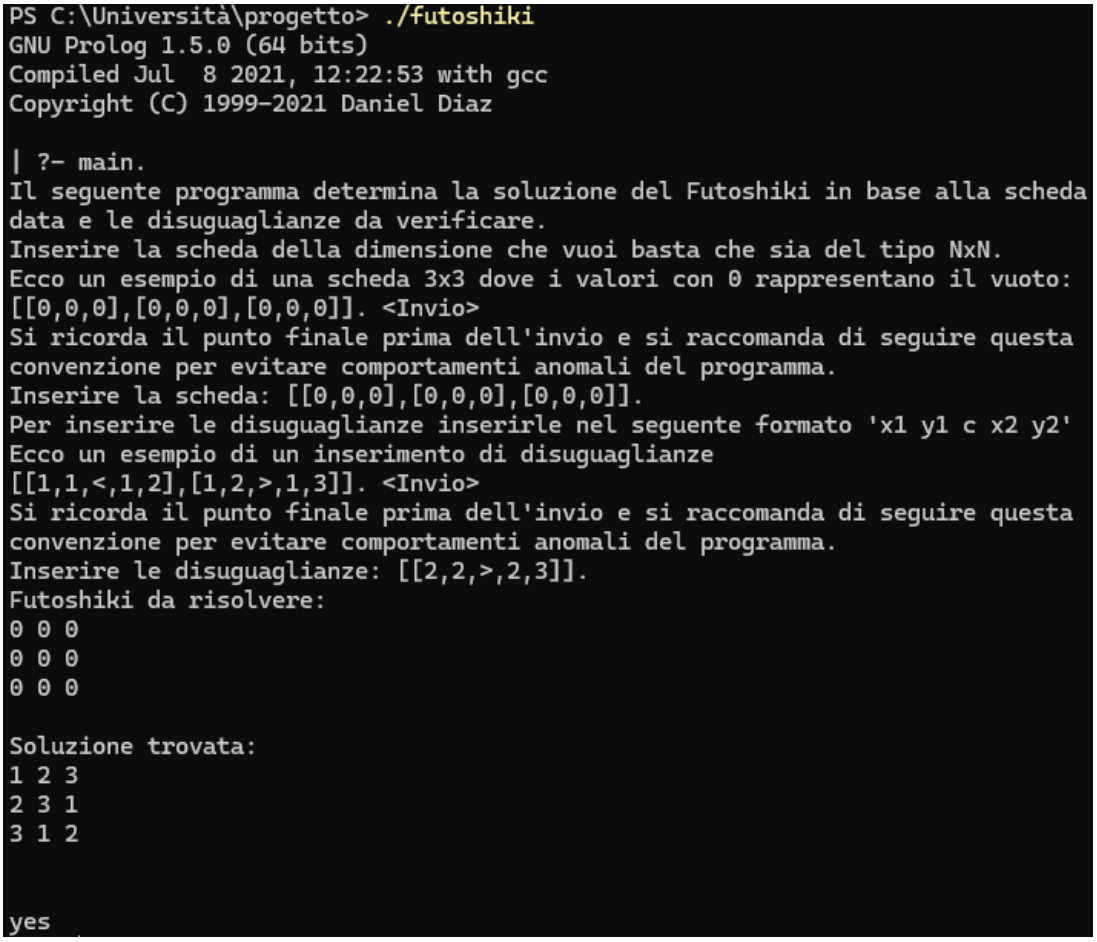

Figura 12: Test nr. 2

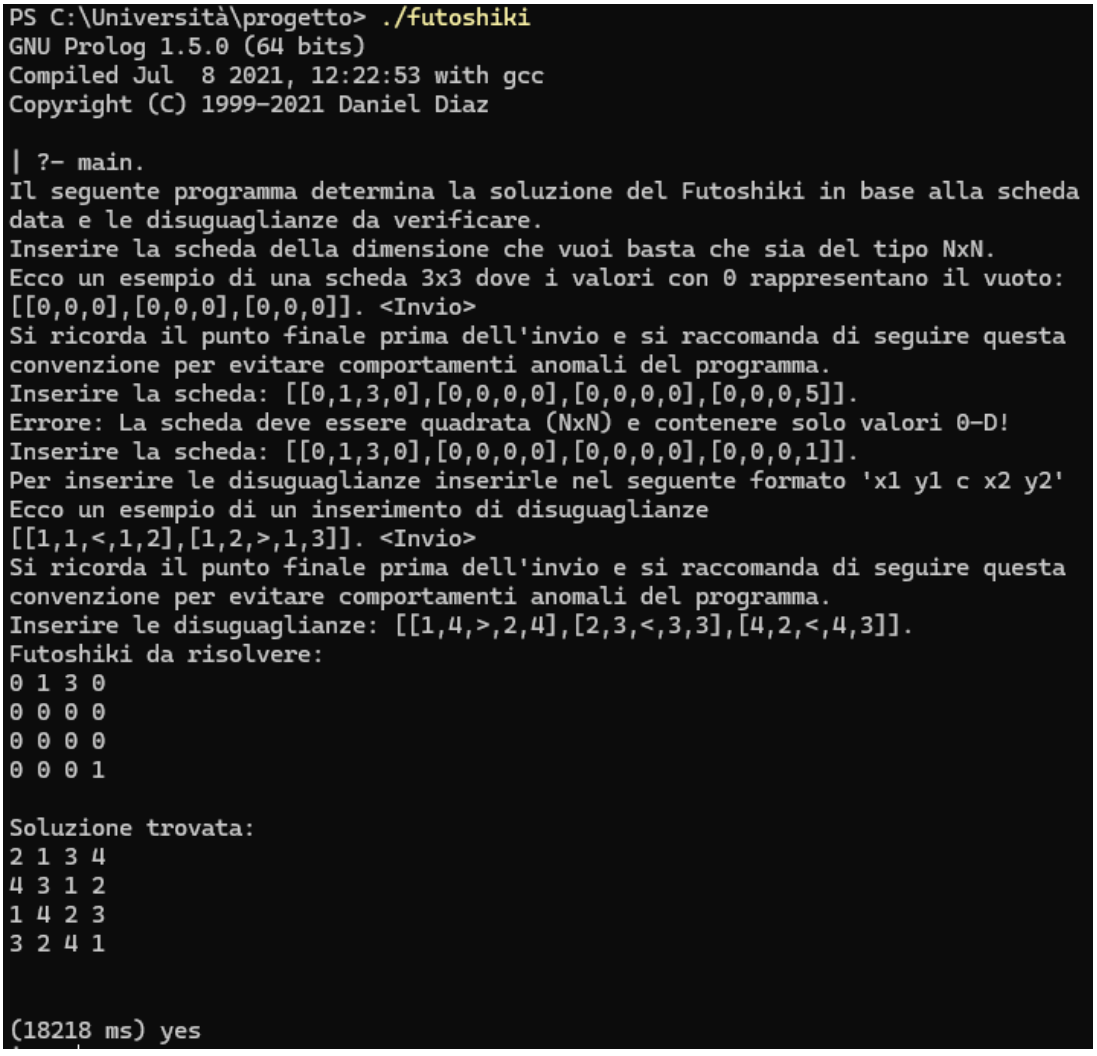

Figura 13: Test nr. 3

Figura 14: Test nr. 4

```
PS C:\Università\progetto> ./futoshiki
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- main.
Il seguente programma determina la soluzione del Futoshiki in base alla scheda
data e le disuguaglianze da verificare.
Inserire la scheda della dimensione che vuoi basta che sia del tipo NxN.
Ecco un esempio di una scheda 3x3 dove i valori con 0 rappresentano il vuoto:
[[0,0,0],[0,0,0],[0,0,0]]. <Invio>
Si ricorda il punto finale prima dell'invio e si raccomanda di seguire questa
convenzione per evitare comportamenti anomali del programma.
Inserire la scheda: [[0,2,3,0],[4,0,0,0],[0,0,0,0],[0,0,0,1]].
Per inserire le disuguaglianze inserirle nel seguente formato 'x1 y1 c x2 y2'
Ecco un esempio di un inserimento di disuguaglianze
[[1,1,<,1,2],[1,2,>,1,3]]. <Invio>
Si ricorda il punto finale prima dell'invio e si raccomanda di seguire questa
convenzione per evitare comportamenti anomali del programma.
Inserire le disuguaglianze: [[1,4,>,4,4],[2,2,>,2,3]].
Errore nelle disuguaglianze! Verificare:
- Formato corretto: [[R1,C1,Op,R2,C2], ...]
- Posizioni valide (1 a 4)
- Operatori > o <
- Celle adiacenti
- Celle diverse
Inserire le disuguaglianze: [[3,4,>,4,4],[2,2,>,2,3]].
Futoshiki da risolvere:
0 2 3 0
4 0 0 0
0 0 0 0
0 0 0 1

Soluzione trovata:
1 2 3 4
4 3 1 2
2 1 4 3
3 4 2 1


(2062 ms) yes
```

Figura 15: Test nr. 5

```
PS C:\Università\progetto> ./futoshiki
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- main.
Il seguente programma determina la soluzione del Futoshiki in base alla scheda
data e le disuguaglianze da verificare.
Inserire la scheda della dimensione che vuoi basta che sia del tipo NxN.
Ecco un esempio di una scheda 3x3 dove i valori con 0 rappresentano il vuoto:
[[0,0,0],[0,0,0],[0,0,0]]. <Invio>
Si ricorda il punto finale prima dell'invio e si raccomanda di seguire questa
convenzione per evitare comportamenti anomali del programma.
Inserire la scheda: [[2,0,3,0,0],[0,4,2,5,3],[0,5,0,0,0],[4,3,0,1,2],[5,0,2,0,4]].
Per inserire le disuguaglianze inserirle nel seguente formato 'x1 y1 c x2 y2'
Ecco un esempio di un inserimento di disuguaglianze
[[1,1,<,1,2],[1,2,>,1,3]]. <Invio>
Si ricorda il punto finale prima dell'invio e si raccomanda di seguire questa
convenzione per evitare comportamenti anomali del programma.
Inserire le disuguaglianze: [[1,1,<,1,2],[1,1,>,1,2]].
Futoshiki da risolvere:
2 0 3 0 0
0 4 2 5 3
0 5 0 0 0
4 3 0 1 2
5 0 2 0 4

La soluzione non esiste.

(36015 ms) yes
```

Figura 16: Test nr. 6

```
PS C:\Università\progetto> ./futoshiki
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- main.
Il seguente programma determina la soluzione del Futoshiki in base alla scheda
data e le disuguaglianze da verificare.
Inserire la scheda della dimensione che vuoi basta che sia del tipo NxN.
Ecco un esempio di una scheda 3x3 dove i valori con 0 rappresentano il vuoto:
[[0,0,0],[0,0,0],[0,0,0]]. <Invio>
Si ricorda il punto finale prima dell'invio e si raccomanda di seguire questa
convenzione per evitare comportamenti anomali del programma.
Inserire la scheda: [[0,1,0,2],[3,0,2,0],[0,0,0,0],[0,0,0,4]].
Per inserire le disuguaglianze inserirle nel seguente formato 'x1 y1 c x2 y2'
Ecco un esempio di un inserimento di disuguaglianze
[[1,1,<,1,2],[1,2,>,1,3]]. <Invio>
Si ricorda il punto finale prima dell'invio e si raccomanda di seguire questa
convenzione per evitare comportamenti anomali del programma.
Inserire le disuguaglianze: [[1,1,<,1,1],[4,4,>,4,3]].
Errore nelle disuguaglianze! Verificare:
- Formato corretto: [[R1,C1,Op,R2,C2], ...]
- Posizioni valide (1 a 4)
- Operatori > o <
- Celle adiacenti
- Celle diverse
Inserire le disuguaglianze: [[1,1,>,1,2],[4,4,>,4,3]].
Futoshiki da risolvere:
0 1 0 2
3 0 2 0
0 0 0 0
0 0 0 4

Soluzione trovata:
4 1 3 2
3 4 2 1
1 2 4 3
2 3 1 4


(2093 ms) yes
```

Figura 17: Test nr. 7

Figura 18: Test nr. 8

Figura 19: Test nr. 9

Figura 20: Test nr. 10