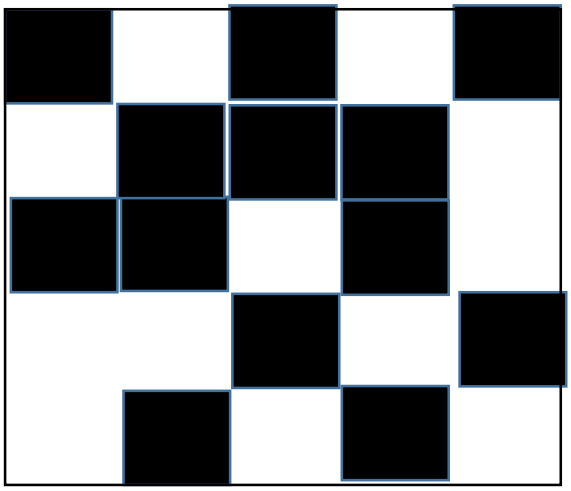
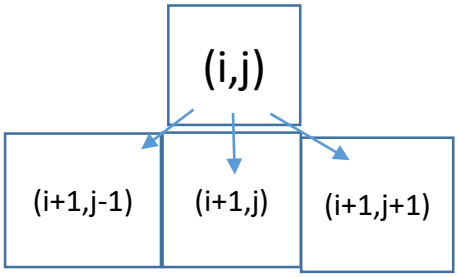
Esercizio a tempo n. 5 del 18/4/2019

Dopo aver letto m (m>0), il main alloca un array dinamico bool N[m*m] che verrà visto nell'esercizio come una matrice X[m][m]. Le caselle true di X sono caselle bianche (libere), mentre quelle false sono caselle nere (proibite). Si cerca un cammino che consiste solo di caselle bianche e che conduce dalla prima all'ultima riga di X. Vediamo un esempio con m=5.

Esempio. Sia X la matrice che segue. Esiste un cammino di caselle bianche che porta dalla prima riga all'ultima e che consiste delle seguenti caselle: (0,3) (1,4) (2,4) (3,3) (4,2)



Come si osserva da questo esempio, il cammino deve andare sempre avanti, cioè deve avere lunghezza pari al numero delle righe. Inoltre, se siamo nella casella (i,j), la prossima casella bianca può essere solo (i+1,j-1) o (i+1,j) oppure (i+1,j+1). Insomma da (i,j) il cammino può proseguire solo nei modi seguenti:



Ovviamente se j=0 o j=m-1, una delle 3 possibili mosse condurrebbe fuori da X, ma un semplice test può verificare se questo succede. Si osservi il cammino, (0,3) (1,4) (2,4) (3,3) (4,2) che conduce dalla prima all'ultima riga della figura precedente, passando solo per caselle bianche. Non è l'unico tale cammino. Anche (0,3) (1,4) (2,4) (3,3) (4,4) consiste solo di caselle bianche. Visto che ad ogni passo si deve passare alla riga successiva, le prime componenti delle coppie di tali cammini sono sempre 0,1,2,…,m-1 e quindi non servono. Basta ricordare solo le seconde componenti delle coppie e questo lo possiamo fare in un array di m posizioni. Quindi il primo cammino, (0,3) (1,4) (2,4) (3,3) (4,2), sarà rappresentato dall'array P=[3,4,4,3,2] e il secondo cammino da P=[3,4,4,3,4]. Diremo che un cammino i0…im-1 è più a sinistra di un altro j0..jm-1 quando:

a) o i0< j0

b) oppure i0=j0 e i1…im-1 è più a sinistra di j1..jm-1.

Insomma i cammini possono essere uguali fino ad un certo punto, ma dopo questo punto, il primo continua con un valore più piccolo (più a sinistra) dell'altro. Nel nostro esempio, [3,4,4,3,2] è più a sinistra di [3,4,4,3,4].

Si chiede di scrivere 2 funzioni ricorsive che cercano un cammino di celle bianche tra la prima e l'ultima riga e restituiscono true sse un tale cammino c'è e in questo caso restituiscono anche il cammino trovato in un array P. La prima funzione è *partenza* e deve soddisfare la seguente specifica:

PRE_p=(N ha m*m valori definiti, 0<=i<=m e P ha m elementi)

bool partenza(bool*N, int m, int i, int*P)

POST_p=(risponde true sse esiste un cammino di caselle bianche dalla prima all'ultima riga di N, vista come X[m][m], e che inizia in una casella tra i e m-1 della prima riga di X) &&(se risponde true allora P[0..m-1] è il cammino più a sinistra con queste proprietà)

La funzione *partenza*, per ogni possibile punto di partenza della prima riga, invoca un'altra funzione, che si chiama *trova*, anch'essa ricorsiva, che soddisfa la seguente specifica:

PRE_t=(N ha almeno m*m elementi definiti, 0<=i<m, j è definito e P ha m-i posizioni)

bool trova(bool* N, int m, int i, int j, int*P)

POST_t=(restituisce true sse in N vista come X[m][m] c'è un cammino dalla casella (i,j) alla riga m-1 composto da sole caselle bianche) &&(se la risposta è true, in P [0..m-i-1] c'è il cammino più a sinistra con queste proprietà)

Si osservi che il main dato invoca *partenza* e poi, in caso il cammino ci sia, lo stampa.

Correttezza:

Dimostrare induttivamente la correttezza della funzione *trova* rispetto a PRE_t e POST_t.