Durante lo sviluppo del progetto non sono stati riscontrati particolari problemi per quanto riguarda al modalità *giocatore singolo*. Una volta giunti alla *challenge* finale sono insorti i limiti dell’algoritmo sviluppato sino ad allora, ovvero l’incapacità di risoluzione dei vicoli ciechi. Per questo abbiamo preso spunto dal metodo di risoluzione di problemi simili come l’algoritmo di Dijkstra però per non cambiare completamente il codice sviluppato sino ad allora la risoluzione del problema in linea generale ci è stata fornita all’interno delle *challenge*. Partendo da questo spunto abbiamo sviluppato l’algoritmo preso in esame.

La maggiore difficoltà incontrata è stato il *debugging* del primo programma da noi sviluppato di modesta complessità, ciò ha richiesto indubbiamente il maggiore sforzo.

**FUNZIONAMENTO DEL PROGRMMA**

*GIOCATORE SINGOLO:*

inizialmente si alloca la matrice da leggere in input, passandola come puntatore alla funzione “*matrix\_player”* poi la matrice viene modificata a seconda delle mosse disponibili scelte dal giocatore. La coda è creata con una struct comprendente: due vettori che salvano la posizione di riga e colonna di ogni punto della coda, e la lunghezza della coda. Poi i valori salvati nella struct scorrono in avanti ad ogni mossa in modo da visualizzare a schermo la coda desiderata

*IA:*

L’algoritmo *IA,* dopo l’allocazione della matrice in input, calcola le direzioni preferite verticali e orizzontali controllando la posizione dell’arrivo. Poi si da una priorità alle direzioni da seguire, se la distanza verticale dal giocatore all’arrivo è maggiore di quella orizzontale, la direzione verticale avrà una priorità maggiore, e viceversa. Successivamente è stata data una decrescente importanza alle mosse da intraprendere nel seguente ordine: 1 Se possibile vai sull’arrivo ci va. 2 Se possibile vai su un “$” o un “T”. 3 Se possibile vai nella direzione preferita con priorità più elevata. 4 Se possibile vai nella direzione preferita con priorità minore 5 Vai nella posizione da cui non sei arrivato.

Una volta eseguita la mossa dell’*IA* controlla che non siano avvenute ripetizioni all’interno del vettore che controlla le mosse fatte, se si riscontra una ripetizione entra in funzione *pattern*, una funzione di riconoscimento dei pattern, se il pattern viene riconosciuto si modifica la variabile globale del pattern. Poi viene salvata la posizione in cui si è formato un pattern, e se la pedina dovesse ripresentare un pattern nella stessa posizione si cambierebbe la variabile globale opposta.

Parlando infine della suddivisione del lavoro, Simone ha iniziato stendendo la funzione main, il sistema di lettura della matrice, la modalità *giocatore singolo,* e le funzioni di manipolazione degli array usate: nella rappresentazione della coda e nella *IA*.

Bruno ha da subito iniziato con lo sviluppo dell’algoritmo implementato nella *IA*, modificandolo a seconda di quanto richiesto dalle *challenge,* lo sviluppo delle funzioni di riconoscimento dei pattern.