```
328   * START HERE: DO NOT CHANGE THE CODE ABOVE THIS POINT
329   *
330   */
331      /* 2.1. Allocate and fill sequence */
332      char *sequence = (char *)malloc( sizeof(char) * seq_length );
333      if ( sequence == NULL ) {
334          fprintf(stderr,"\n-- Error allocating the sequence for size: %lu\n", seq_length );
335          MPI_Abort( MPI_COMM_WORLD, EXIT_FAILURE );
336      }
337      random = rng_new( seed );
338      generate_rng_sequence( &random, prob_G, prob_C, prob_A, sequence, seq_length);
339
```

riga 332 : la malloc viene eseguita da ogni rank, se 'n' è il numero di caratteri nella sequenza, ogni rank alloca n caratteri, anche se nella generazione della sequenza, userà solo la sua porzione.

ognuno genera la sua porzione di sequenza, al termine una collettiva MPI_Allgather condividerà fra tutti i rank la sequenza globale. Questa dovrà essere acceduta in sola lettura, quindi da qui in poi non sarà necessario condividerla fra i thread.

```
359      int *seq_matches;
360      seq_matches = (int *)malloc( sizeof(int) * seq_length );
361      if ( seq_matches == NULL ) {
362          fprintf(stderr,"\n-- Error allocating aux sequence structures for size: %lu\n", seq_length );
363          MPI_Abort( MPI_COMM_WORLD, EXIT_FAILURE );
364      }
```

bisognerà al termine del programma eseguire una MPI_Allreduce su 'seq_matches'

esempio: { rank1 : [1 -1 1 2 -1] cast => [1 0 1 2 0] ; rank2 : [-1 1 1 1 1] [0 1 1 1 1] } = All_reduce => [1 1 2 3 1]