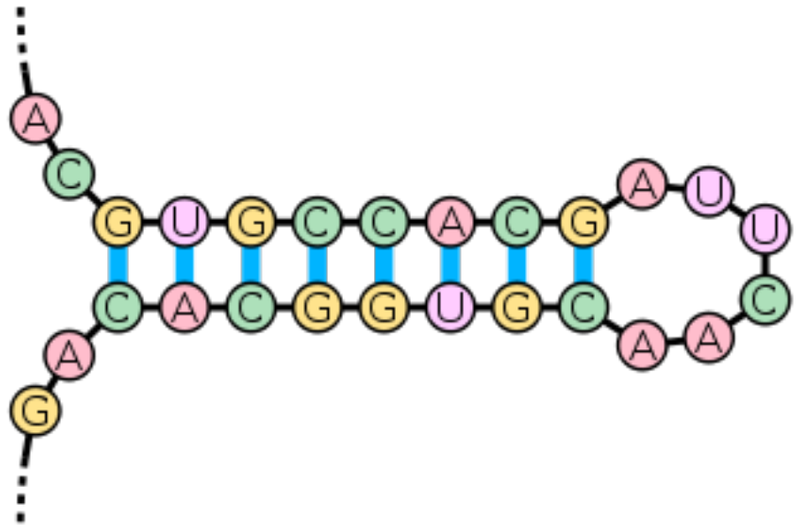
RNA Stem Loop Visualizer

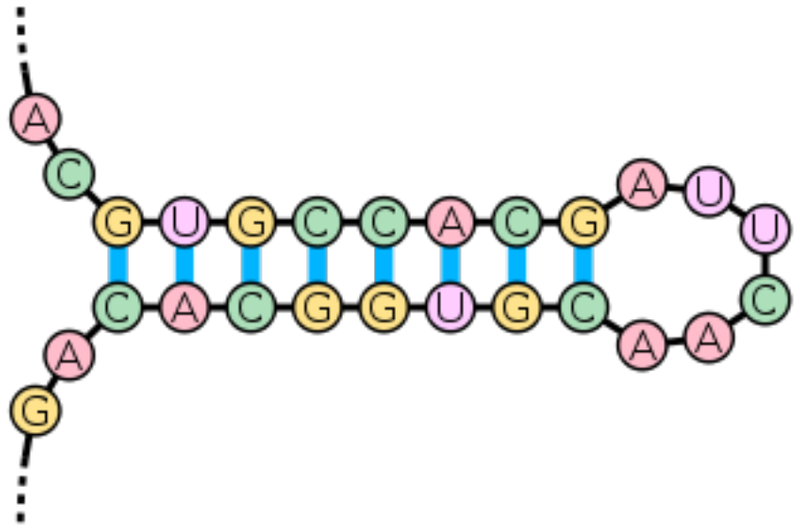
Project Machines & Berekenbaarheid

Fase II



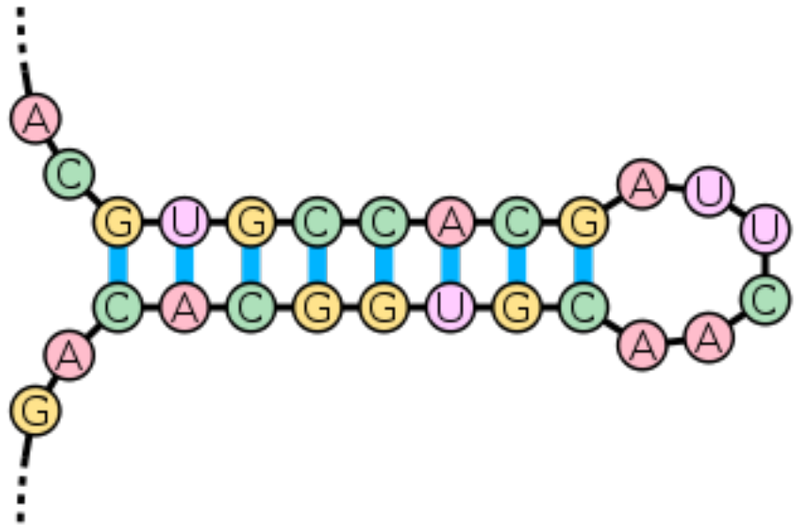
```cpp
/**
 * @enum ENucleotideType
 * @brief The different types of the nucleotides.
 */
enum ENucleotideType {
    kADENINE,
    kCYTOSINE,
    kGUANINE,
    kURACIL
};

/**
 * @class Nucleotide
 * @brief The nucleotide
 */
class Nucleotide {
public:
    /**
     * @brief Constructor.
     *
     * @param type Which type of nucleotide we are talking about.
     */
    Nucleotide(const ENucleotideType& type);

    /**
     * @brief Check whether this nucleotide match with another nucleotide.
     * @details Cytosine match with Guanine and Adenine with Uracil.
     *
     * @param that The other nucleotide you want to compare.
     *
     * @return True if they match, false otherwise.
     */
    bool match(const Nucleotide& that) const;

private:
    /**
     * @var fType
     * @brief The type of the nucleotide.
     */
    ENucleotideType fType;
};
```

Fase II



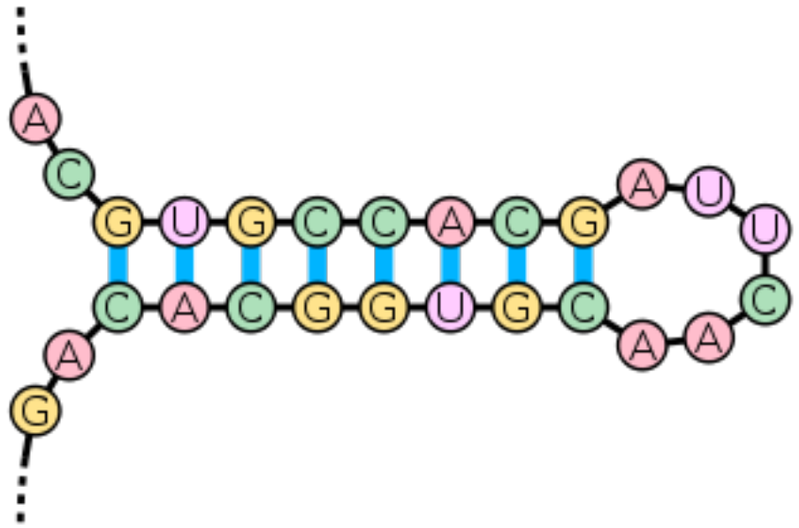
```cpp
/**
 * @typedef RNAString
 * @brief Represents an RNA sequence of nucleotides.
 */
typedef std::vector<Nucleotide*> RNAString;

/**
 * @class RNAParser
 * @brief The RNA sequence parser.
 * @details This is an ordinary RNA sequence parser. This does not do
 * additionally stuffs such as recognizing stem loops or something like that.
 * It just check whether the sequence of RNA is a valid one (i.e. there are
 * no wrong symbols in the sequence).
 */
class RNAParser {
public:
    /**
     * @brief Constructor.
     *
     * @param filename The name of the file to read from it.
     */
    RNAParser(const char* filename[]);

    /**
     * @brief Parse the sequence.
     * @details Since this is an ordinary RNA sequence parser, this does only
     * checking whether all symbols in the RNA sequence is valid (i.e. either
     * one of the A, C, G or U).
     *
     * @throws std::runtime_error When there's something wrong with parsing
     * or the RNA sequence in the file is not valid.
     *
     * @note If you call this method more than once, it will forgot the last
     * parsed RNA sequence and do it over again.
     */
    virtual void parse();

    /**
     * @brief Get the RNA string after parsing the sequence.
     *
     * @return The parsed RNA string.
     *
     * @note If you've parsed nothing you'll get an empty RNA string.
     */
    RNAString getRNAString() const;

    /**
     * @brief Get the configuration for the visualizer.
     *
     * @return TODO
     */
    virtual TODO getConfiguration() const;

private:
    /**
     * @var fRNASequence
     * @brief The parsed RNA string.
     */
    RNAString fRNASequence;
};
```

Fase II



```cpp
/**
 * @class RNAOneStemLoop
 * @brief Class responsible for parsing a basic stem loop.
 * @details A basic stem loop is something as:
 * @code
 *  G == C
 *  G == C
 *  C == G
 *  A == U
 * x      x
 *  x    x
 *   x  x
 *    xx
 * @endcode
 */
class RNAOneStemLoop : public RNAParser {
public:
    /**
     * @brief Constructor.
     *
     * @param filename The name of the file to read from it.
     */
    RNAOneStemLoop(const char* filename[]);

    /**
     * @brief Parse the sequence.
     * @details Since this is an ordinary RNA sequence parser, this does only
     * checking whether all symbols in the RNA sequence is valid (i.e. either
     * one of the A, C, G or U).
     *
     * @throws std::runtime_error When there's something wrong with parsing
     * or the RNA sequence in the file is not valid.
     *
     * @note If you call this method more than once, it will forgot the last
     * parsed RNA sequence and do it over again.
     */
    void parse();

    /**
     * @brief Get the indexes of the RNA string where the loop starts and
     * end respectively.
     */
    std::tuple<unsigned int, unsigned int> getLoopIndexes() const;

    /**
     * @brief Get the configuration for the visualizer.
     *
     * @return TODO
     */
    TODO getConfiguration() const;

private:
    /**
     * @var fBegin
     * @brief The index of the RNA string where the loop starts.
     */
    unsigned int fBegin;

    /**
     * @var fEnd
     * @brief The index of the RNA string where the loop ends.
     */
    unsigned int fEnd;
};
```

Fase II



```cpp
/**
 * @class RNAVisualizer
 * @brief Just a class that will visualize the RNA.
 */
class RNAVisualizer {
public:
    /**
     * @brief Just visualize the RNA sequence without any specials stuffs.
     *
     * @param TODO (see above)
     */
    void visualize(const RNAParser& p);
};
```

Uitbreidingen

- Input RNA sequence where the initial nucleotide is not necessarily the beginning of the stem loop.
- Support sequences with multiple stem loops.
- Input which elements should be in the loop, so you can track down specific stem loops.
- Use something fancier than ASCII to visualize the stem loop.
- Use LL or RL parser or Turing Machine instead of PDA.
- GUI