ESERCIZIO: SEMINARIO

Implementazione Java di un bruteforcer di chiavi AES di 16 caratteri (da 0 a Integer.MAX_VALUE)

Marco Macrì (MAT. 220070)

Idea d’approccio

L’idea è di suddividere la ricerca della chiave su n thread scelti dal chiamante del metodo di bruteforcing.
Siccome si opera in un intervallo specifico, quello che ho pensato di fare è di progettare un modo per assegnare i sotto intervalli di ricerca ai thread.

Prima di affrontare il problema sollevato, serve sapere come adattare un Integer affinché sia compatibile con lo standard della chiave richiesta.

Come sappiamo essa dovrà essere composta da 16 cifre le cui «significative» dovranno essere degli Integer. Per farlo abbiamo bisogno di un Padding, ossia un’aggiunta di z 0 prima del valore dell’Integer.

Ho voluto fornire la classe di utilità «Metodi», che contiene i metodi necessari affinché l’intero meccanismo sia avviato da una sola chiamata, per esempio, in un main.

String dammiStringa(Integer intero)

Il metodo riceve un intero (Integer), ne calcola la lunghezza della relativa stringa e appende tante volte 0 e il toString() di Integer fino ad arrivare alla lunghezza finale di 16 caratteri.
Per fare ciò mi avvalgo di uno StringBuilder e mi assicuro che il valore dato in input dia idoneo, se così non fosse sollevo un'eccezione

```
public static String dammiStringa(Integer intero) {
   if (intero < 0) throw new IllegalArgumentException("Valore negativo");
   StringBuilder stringa = new StringBuilder();
   for(int i = 0; i< 16 - intero.toString().length(); i++, stringa.append("0"));
   stringa.append(intero);
   return stringa.toString();
}
```

void bruteForce(int nThread, File input, String indizio)

Il metodo riceve in input il numero di thread da utilizzare, il file da decifrare, e una striga indizio che dovrà essere ricercata dentro i files decifrati, se essa sarà presente, la chiave sarà stampata a schermo.
Creo un array di threads e calcolo i passi da fare per singolo thread mediante divisione intera. Tengo conto del resto, che delego all'ultimo thread.
Comincio a fare un ciclo sull'array incrementando il conto e popolando gli spazi…
Dopo mando in attesa il main di tutti i thread. Al termine dell'attività dei thread restituisco il tempo impiegato dalla ricerca.

```
public static void bruteForce(int nThread, File input, String indizio) throws Exception{
    Instant start = Instant.now();
    if(nThread <= 0) throw new IllegalArgumentException("Numero thread non valido!");
    if(input == null) throw new IllegalArgumentException("File == null!");
    BForcer[] bForcers = new BForcer[nThread];
    int passo = Integer.MAX_VALUE / nThread;
    int resto = Integer.MAX_VALUE % nThread;
    int conto = 0;
    for (int i = 0; i < nThread; i++) {
        if (i == nThread-1) bForcers[i] = new BForcer(input, conto, conto + passo + resto, i, indizio, bForcers);
        else {bForcers[i] = new BForcer(input, conto, conto + passo, i, indizio, bForcers);
        conto += passo;}
        bForcers[i].start();
    }
    for (int i = 0; i < nThread; i++) bForcers[i].join();
    Instant end = Instant.now();
    System.out.println(Duration.between(start, end).toMinutes());
}
```

void bruteForce(int nThread, File input, String indizio)

Il cuore è la classe BForcer che si occupa di provare tutte le chiavi da un inizio ad una fine. Il run(…) comprende un ciclo che tenta le chiavi nell'intervallo. Si mette in una stringa la possibile chiave e con l'aiuto della classe CryptoUtils decodifica il file.
Il passo successivo è di far iterare un BufferedReader sul file generato per ricercare l'indizio.
Il metodo interrompiTutti() interrompe tutti i thread tranne quello che sta eseguendo, esso verrà interrotto poi dal run.

```
public class BForcer extends Thread{
    private int inizio, fine, id;
    File input;
    String indizio;
    BForcer[] bForces;

    public BForcer(File input, int inizio, int fine, int id, String indizio, BForcer[] bForces) {
        super();
        this.input = input;
        this.inizio = inizio;
        this.fine = fine;
        this.id = id;
        this.indizio = indizio;
        this.bForces = bForces;
    }

    @Override
    public void run() {
        System.out.println("comincio thread "+ id + " da "+inizio+" a "+fine);
        for (int i = inizio; i < fine; i++) {
            String chiave = Metodi.dammiStringa(i);
            try {
                String outputName = "forced" + id + ".possible";
                CryptoUtils.decrypt(chiave, input, new File(outputName));
                BufferedReader br = new BufferedReader(new FileReader(outputName));
                String s = "";
                while (s != null) {
                    s = br.readLine();
                    if(s.contains(indizio)) {
                        System.out.println(" la chiave corretta è" + chiave + " il file decifrato è " + outputName);
                        interrompiTutti();
                    }
                }
            } catch (Exception e) {
            }
        }
    }
    private void interrompiTutti() {
        for(BForcer bf : bForces) if(bf != this) bf.interrupt();
        this.interrupt();
    }
}
```