**SCALETTA:**

* Copertina;
* Indice:
  + Strategia di Implementazione;
  + Raccolta ed Elaborazione Dati;
  + Approfondimenti;
* Strategia di Implementazione;
  + Idee scartate;
* Raccolta ed Elaborazione Dati;
  + Scrittura e uso di profiler.py;
  + Tabelle, grafici (**N.B:** riferire OS e possibilmente specifiche tecniche della macchina su cui sono stati effettuati i test) …
  + Studio di lenTuple, range / size …
  + Breve commento su ognuno dei risultati ottenuti
* Approfondimenti.
  + GitHub (?);
  + Argparse, random, plotter …
  + Qualsiasi cosa possa allungare il brodo …

**NOTE:**

* Ho usato font diversi in base a ciò che si scrive:
  + Codice: Courier New
  + Math:

Strategia di Implementazione

Per adempiere alle richieste presenti nella Traccia, è stata adottata la strategia di implementazione descritta di seguito:

* Implementazione di una variante dell’algoritmo QuickSort (**QS**), chiamata QuickSelectSort (**QSS**), che adopera un algoritmo di selezione per calcolare il pivot attorno a cui partizionare la lista di input;
* Implementazione di una variante dell’algoritmo Select (**S**), chiamata SampleMedianSelect (**SMM**), che estrae l’elemento desiderato k partizionando attorno a un pivot calcolato chiamando la funzione SampleMedian (**SM**).

Tali varianti sono state implementate nel file QSS\_module.py.

In fondo alla sezione è dedicato uno spazio per le strategie scartate.

Implementazione di QSS

A differenza del QS descritto nel modulo sorting.Sorting, QSS accetta come secondo parametro select {0, 1, 2}, il quale determina la variante di S che verrà invocata per calcolare il pivot attorno a cui verrà partizionata la lista da ordinare. Il vero e proprio lavoro di ordinamento è svolto da recursiveQuickSelectSort (**RQSS**), che rappresenta il nucleo ricorsivo. A quest’ultima sono passati altri due parametri, left e right, che indicano i confini della partizione su cui RQSS sta lavorando.

Una volta calcolato il pivot viene invocata sorting.partitionDet (**PD**), che effettua il lavoro di partizionamento e restituisce l’indice del pivot. Dunque, RQSS lavora ricorsivamente sulle partizioni ottenute, ordinando infine la lista di input.

Implementazione di SMS e SM

Diversamente da S (descritto nel modulo selection.Selection), SMS estrae l’elemento partizionando attorno a un pivot calcolato mediante l’invocazione di SM. A seconda se SMS lavora ricorsivamente su una delle due partizioni oppure restituisce l[indice(p)].

SM costruisce un sottoinsieme *V* di l partizionando la lista di input in tuple di lunghezza , estraendo un elemento a caso da ciascuna tupla e inserendolo in V. Se la condizione (con ) non si verifica ripete ricorsivamente le suddette operazioni, altrimenti si calcola il mediano di *V* invocando quickSelectRand (**QSR**) e lo restituisce.

Grazie al lavoro di partizionamento effettuato da SM, i valori estratti casualmente e inseriti in sono distribuiti in modo omogeneo su tutta la lista. Ciò aumenta la probabilità che il mediano estratto da corrisponda al mediano effettivo della lista, migliorando di conseguenza i tempi di esecuzione di SMS, poiché partiziona più velocemente.

È interessante notare come il parametro non sia un valore fissato, bensì in funzione di : gli effetti al variare di questi parametri sono discussi nella sezione di **Raccolta ed Elaborazione Dati**.

Strategie Scartate

Durante l’implementazione di SM, riguardo la scelta dell’algoritmo di selezione da utilizzare per estrarre il mediano da , è stato proposto trivialSelect (**TV**). Questo perché inizialmente la lunghezza delle tuple era stata fissata pari a 5, sulla falsariga di S. Studiando le variazioni dei tempi di esecuzione di SM al variare di , è stato scoperto che a un valore elevato di quest’ultimo è associato un miglioramento generale dei tempi di esecuzione di SM e SMS, oscurato da un ben più sensibile peggioramento dei tempi di TV, che presenta un tempo quadratico nel caso peggiore. Dunque, è stato ritenuto opportuno sostituire TV con una chiamata a QSR, che presenta un tempo nel caso peggiore.

Inoltre, mentre attualmente il parametro , ovvero la dimensione del sottoinsieme , può assumere valori tali che , inizialmente si era optato di definire , e dunque di costruire estraendo 3 elementi a caso dalla lista. L’idea è stata sostituita dal metodo delle partizioni, poiché quest’ultimo presenta una probabilità maggiore di restituire un valore il più vicino possibile al mediano della lista.