Alessio Barletta 5686474

Compito 2.1. Implementazione di LVQuickSort

Calcolare valore medio e varianza del numero di confronti effettuati per ogni esecuzione di LVQuickSort su 10^5 run. |S| = 10^4 e numeri casuali da 0 a 10^4. Successivamente limitare dall'alto la probabilità che LVQuickSort effettui il doppio e il triplo dei confronti rispetto al valor medio e commentare i risultati.

Svolgimento

Ho implementato la funzione LVQuickSort come da pseudocodice nelle dispense, aumentando il numero di confronti ogni volta che un elemento dell'array viene confrontato con il pivot. Il pivot viene scelto randomicamente usando la funzione rand() dopo aver opportunamente inizializzato il generatore di numeri casuali.
Tutti i risultati di ogni iterazione ho deciso di salvarli in un file per poter successivamente produrre l'istogramma utilizzando Python, mentre questa parte dell'esercitazione l'ho svolta in c++. Successivamente ho calcolato valor medio, deviazione standard e la probabilità che la variabile casuale X (cioè, il numero di confronti ad ogni iterazione di LVQuickSort) sia il doppio o il triplo del valore medio. Questi calcoli gli ho effettuati utilizzando le formule indicate e ho salvato tutti i risultati in altri due file.

Risultati ottenuti

Variabile casuale X = numero di confronti effettuati per ogni run. X_R indica il numero di confronti effettuati al passo r-esimo.

- Valore medio (μ) = 156337
- Deviazione standard (σ) = 6506

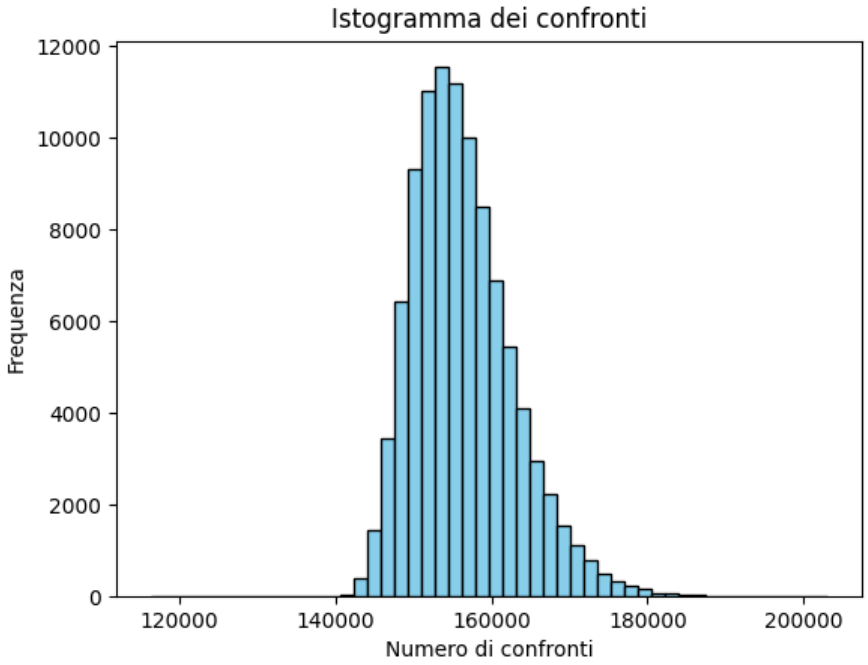
Formula utilizzata per il valor medio, R=10^5:

$$\frac{1}{R}\sum_{r=1}^{R} X_R$$

Formula utilizzata per la varianza, R=10^5:

$$\sigma^2 = \frac{1}{R-1}\sum_{r=1}^{R}(X_R - \mu)^2$$

Utilizzando i dati ricavati da ogni run, è stato possibile generare il seguente istogramma con 50 bin:



L'istogramma ci mostra come la grande maggioranza di valori si concentri attorno a circa 155000 confronti per run e che sono distribuiti per la gran parte tra circa 14500 e 16200, questo conferma i valori calcolati di valor medio e deviazione standard (di quanti confronti, in media, ogni valore si discosta dal valor medio).
Inoltre, sappiamo che il valore atteso di LVQuickSort è $E[X] \approx 2n\ln(n) \approx 184206$. Valore simile a quello ottenuto empiricamente.

I limiti dall'alto delle probabilità che il numero di confronti siano il doppio o il triplo del valore medio sono i seguenti:

- Doppio, con disuguaglianza (6) = 0.5
- Doppio con disuguaglianza (7) = 0.00173183
- Triplo, con disuguaglianza (6) = 0.33
- Triplo, con disuguaglianza (7) = 0.000432957

Le disuguaglianze (6) e (7) sono derivate dalla disuguaglianza di Markov e di Chebishev e sono le seguenti:

- (6): $pr\{X \geq v\mu\} \leq \frac{\mu}{v\mu} = \frac{1}{v}$
- (7): $pr\{X \geq v\mu\} \leq \frac{\sigma^2}{(v-1)^2\mu^2}$

Mettendo $v = 2\ e\ v = 3$ possiamo quindi vedere qual è il limite superiore che il numero di confronti sia il doppio e il triplo del valore medio, cioè la probabilità che i confronti siano il doppio e il triplo del valore medio è inferiore a ciò che è a destra della disuguaglianza. Per esempio, seguendo la formula (7), la probabilità che i confronti effettuati siano il doppio del valore medio è inferiore o uguale a 0.00173183.
Possiamo inoltre notare come la misura diventi molto più precisa se siamo a conoscenza della varianza oltre che il valore medio.
Empiricamente, non ho mai ottenuto un valore superiore al doppio, né tantomeno al triplo, del valor medio, un risultato aspettato in quando la probabilità che accada è estremamente bassa.
Possiamo concludere che la complessità di LVQuickSort è effettivamente O(nlogn), comunque siano distribuiti gli input, questo coincide con quanto calcolato empiricamente.

Codice C++

```cpp
#include <iostream>
#include <cmath>
#include <time.h>
#include <fstream>

#define N 10000

#define RUNS 100000

using namespace std;

int lvQuickSort(int s[], int left, int right);
long long int valoreMedio(int comparisons[], int n);
long long int deviazioneStandard(int comparisons[], int n);
float chebyshev1(int val);
float chebyshev2(long long int med, long long int dev, int val);


int main() {

    int s[N];
    int comparisons[RUNS];
```

```
    srand(time(NULL));

    for (int i = 0; i < N; i++) {
        s[i] = rand() % N;
    }

    ofstream comparisonsFile;
    ofstream resultsFile;
    ofstream chebyshevFile;

    comparisonsFile.open("comparisons.txt");
    resultsFile.open("results.txt");
    chebyshevFile.open("chebyshev.txt");

    for(int i = 0; i < RUNS; i++) {
        comparisons[i] = lvQuickSort(s, 0, N - 1);
        comparisonsFile << comparisons[i] << endl;
    }

    long long int med = valoreMedio(comparisons, RUNS);
    long long int dev = deviazioneStandard(comparisons, RUNS);
    resultsFile << med << endl;
    resultsFile << dev << endl;

    float cheb1Double = chebyshev1(2);
    float cheb2Double = chebyshev2(med, dev, 2);

    float cheb1Triple = chebyshev1(3);
    float cheb2Triple = chebyshev2(med, dev, 3);

    cout << "Chebyshev 1 double: " << cheb1Double << endl;
    chebyshevFile << cheb1Double << endl;
    cout << "Chebyshev 2 double: " << cheb2Double << endl;
    chebyshevFile << cheb2Double << endl;

    cout << "Chebyshev 1 triple: " << cheb1Triple << endl;
    chebyshevFile << cheb1Triple << endl;
    cout << "Chebyshev 2 triple: " << cheb2Triple << endl;
    chebyshevFile << cheb2Triple << endl;

    chebyshevFile.close();
    resultsFile.close();
    comparisonsFile.close();

    return 0;
}

int lvQuickSort(int s[], int left, int right) {

    int confronti = 0;
```

```
    if (left >= right) return confronti;

    int i = left, j = right;
    int pivot = s[rand() % (right - left + 1) + left];

    while (i <= j) {
        while (s[i] < pivot){ confronti++; i++; };
        while (s[j] > pivot){ confronti++; j--; };
        if (i <= j) {
            std::swap(s[i], s[j]);
            i++;
            j--;
        }
    }

    confronti += lvQuickSort(s, left, j);
    confronti += lvQuickSort(s, i, right);

    return confronti;
}

long long int valoreMedio(int comparisons[], int n) {
    long long int med = 0;
    for (int i = 0; i < n; i++) {
        med += comparisons[i];
    }
    return med / n; // 1 / n * med;
}

long long int deviazioneStandard(int comparisons[], int n) {
    long long int var = 0;
    long long int med = valoreMedio(comparisons, n);
    for (int i = 0; i < n; i++) {
        var += pow(comparisons[i] - med, 2);
    }
    var = var / (n - 1);
    return sqrt(var);
}


float chebyshev1(int val) {
    return (1.0 / val);
}

float chebyshev2(long long int med, long long int dev, int val) {
    return pow(dev, 2) / ( pow(val - 1, 2) * pow(med, 2) );
}
```

Codice Python

```python
import matplotlib.pyplot as plt

data = []

with open('comparisons.txt', 'r') as file:
    for line in file:
        data.append(int(line.strip()))

plt.hist(data, bins=50, color='skyblue', edgecolor='black')

plt.xlabel('Numero di confronti')
plt.ylabel('Frequenza')
plt.title('Istogramma dei confronti')

plt.show()
```