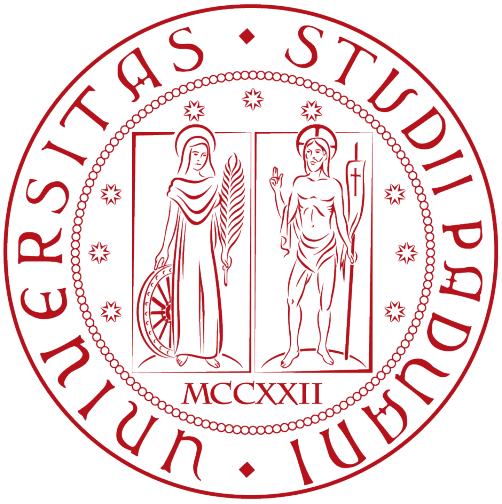


Algoritmi avanzati

Laboratorio 3

Analisi algoritmo di Karger

Studenti:
Marco Pozza

Matricole:
1219870

Introduzione

Questo documento è una relazione riguardante il terzo esperimento di laboratorio del corso di Algoritmi Avanzati della laurea Magistrale in Informatica dell'Università degli studi di Padova. Lo scopo di questa esercitazione è quello di andare ad implementare l'algoritmo randomizzato Montecarlo di Karger e di andare a valutarne le prestazioni rispetto a quattro parametri:

1. Il tempo totale di esecuzione impiegato dall'algoritmo per ripetere un numero sufficientemente alto di volte le contrazioni in modo da raggiungere una probabilità di errore dell'ordine di $\frac{1}{n^d}$;
2. Il tempo di esecuzione di un iterazione di FullContraction;
3. Il *discoveryTime* ovvero il tempo alla quale l'algoritmo trova per la prima volta la soluzione ottima;
4. L'errore della soluzione trovata rispetto alla soluzione ottima.

Descrizione del dataset

Il dataset utilizzato contiene un totale di 40 grafi non orientati del dataset stanford-algs di dimensione compresa tra i 6 e i 200 nodi.
Ogni grafo viene descritto da due file:

- *input_random_id_x.txt* che descrive, utilizzando liste di adiacenza, il grafo identificato da **id** che contiene un totale di **x** numero di vertici;
- *output_random_id_x.txt* che contiene il numero che identifica il minimum cut ottimo per quel particolare grafo.

Un grafo avrà quindi il seguente formato:

`input_random_id_6.txt:`

```
1 2 3 4
2 1 3 6
3 2 4 1 5
4 3 1 5
5 4 3
6 2
```

In cui la prima colonna indica l'etichetta del vertice, mentre gli elementi successivi formano la lista di adiacenza per quel nodo. Quello che ne risulterebbe sarebbe quindi la seguente lista di adiacenza:

```
1 -> 2 - 3 - 4
2 -> 1 - 3 - 6
3 -> 2 - 4 - 1 - 5
4 -> 3 - 1 - 5
5 -> 4 - 3
6 -> 2
```

Descrizione del problema

Quello che si vuole fare con questo esperimento è andare a valutare le prestazioni dell'algoritmo di Karger secondo i 4 parametri specificati nell'introduzione.

Algoritmo di Karger

L'algoritmo di Karger è un algoritmo randomizzato di tipo Montecarlo per andare a risolvere il problema del *minimum cut* di un grafo o, più precisamente, di un *multigrafo*.
Prima di descrivere l'algoritmo è necessario capire cosa si sta cercando di risolvere e per fare questo è necessario capire cosa sia un *minimum cut* e il contesto applicativo in cui ci muoviamo.
Diamo quindi le seguenti definizioni:

- Un **multigrafo** è un grafo in cui, tra due nodi, ci può essere più di un lato che li connette. Un grafo, di fatto, è un caso speciale di un multigrafo in cui tra due nodi c'e solamente un lato e non più di uno;
- Dato un multigrafo connesso un **taglio** è un sottoinsieme di lati tale per cui la sua rimozione fa si che il grafo non sia più connesso;
- Un **minimum cut** abbreviato a **minCut** è un taglio di cardinalità minima.

Passiamo ora a descrivere l'algoritmo di Karger e i principi sulla quale si basa.
Dato un multigrafo connesso il minCut può essere trovato tramite i seguenti passaggi:

1. scelta casuale di un lato;
2. *contrazione* dei due nodi adiacenti al lato scelto;
3. ripetizione della procedura dal punto 1. fino a rimanere con due nodi distinti;
4. il minCut è il numero di lati che connette questi due ultimi nodi.

Come si può capire la parte principale su cui si basa l'algoritmo di Karger è la **contrazione** dei nodi che è un operazione in cui viene creato un nuovo multigrafo tramite i seguenti passi:

1. dati due nodi (x e y), ogni lato che li connette viene rimosso;
2. viene creato un nuovo nodo (Z_{xy}) che rappresenta la contrazione dei due nodi originali;
3. ogni lato che ha x o y come nodo adiacente viene eliminato e i nodi adiacenti restanti vengono connessi al nuovo nodo Z_{xy};
4. i due nodi x e y vengono rimossi.

Questa operazione di contrazione è ripetuta finché il multigrafo non rimane con due nodi. Questa nuova operazione è chiamata **FullContraction**.
La teoria ci insegna che quando si contrae rispetto ad un lato, tutti i tagli che contemplavano quel lato, scompaiono nel nuovo multigrafo.
Quest'ultima considerazione è un problema ai fini della risoluzione del problema del minCut perché, se nella scelta randomica del lato, si seleziona un lato del taglio minimo automaticamente questo scompare è non è più possibile trovare il minCut ottimo. Ecco perché, per aumentare la probabilità che questo algoritmo trovi il minCut ottimo, è necessario ripetere le operazioni di FullContraction un ben preciso numero di volte e, ad ogni iterazione, mantenere il minCut minore trovato fino a quel momento.
Per avere una probabilità di $1 - \frac{1}{n^d}$ di trovare il minCut ottimo (alta probabilità) è necessario ripetere l'operazione di FullContraction un numero di volte pari a

$$d\frac{n^2}{2}\ln n$$

dove abbiamo :

- **n**: numero di vertici nel multigrafo;
- **d**: costante che esprime il grado di "accuratezza" che, al suo aumentare, aumenta la probabilità di trovare la soluzione ottima in funzione del numero di vertici del multigrafo.

L'algoritmo di Karger ha una complessità che dipende da due fattori principali: l'operazione di FullContraction e il numero di volte che questa viene ripetuta. Possiamo quindi concludere l'analisi dell'algoritmo mostrando che:

- FullContraction ha una complessità di $\mathcal{O}(n^2)$
- L'algoritmo completo di Karger ripete un certo numero di volte l'operazione di FullContraction raggiungendo una complessità di $\mathcal{O}(n^4 \log n)$

Rappresentazione dei grafi

In questo esperimento la rappresentazione adottata per i grafi è indotta dai dati forniti ovvero una rappresentazione secondo liste di adiacenza.

Presentazione dei dati e analisi

Andiamo ora a presentare i dati raccolti dopo aver eseguito l'algoritmo su tutti i grafi del dataset fornito.
Siccome, per alcune istanze gradi dei grafi, il tempo di calcolo necessario a completare tutte le iterazioni è risultato eccessivo è stato messo un timeout di 10 minuti dopo il quale è stata ritornata la soluzione migliore trovata fino a quel momento.

Linguaggio e ambiente d'esecuzione

L'esperimento è stato condotto su una macchina con le seguenti specifiche:

- Processore Intel Core i5-7200U 3.1 Ghz;
- Memoria RAM 8GB.

Il linguaggio di programmazione scelto per la scrittura dell'algoritmo è stato Typescript.

Tabella dei risultati totali

					Analisi minCut con Karger				
Grafo	Nodi	d	Ripetizioni	MinCut ottimo	MinCut calcolato	Tempo di esecuzione	Tempo medio FullCont.	Discovery Time	Errore
input_random_01_6	6	3	97	2	2	0,008210 s	0,000081 s	0,000567 s	0,00%
input_random_02_6	6	3	97	1	1	0,006080 s	0,000060 s	0,000203 s	0,00%
input_random_03_6	6	3	97	3	3	0,004656 s	0,000046 s	0,000326 s	0,00%
input_random_04_6	6	3	97	4	4	0,066561 s	0,000684 s	0,000091 s	0,00%
input_random_05_10	10	3	346	4	4	0,031253 s	0,000088 s	0,000584 s	0,00%
input_random_06_10	10	3	346	3	3	0,026877 s	0,000076 s	0,000065 s	0,00%
input_random_07_10	10	3	346	2	2	0,027161 s	0,000077 s	0,000925 s	0,00%
input_random_08_10	10	3	346	1	1	0,027947 s	0,000080 s	0,000171 s	0,00%
input_random_09_25	25	3	3018	7	7	1,522903 s	0,000500 s	0,006149 s	0,00%
input_random_10_25	25	3	3018	6	6	1,541968 s	0,000506 s	0,000962 s	0,00%
input_random_11_25	25	3	3018	8	8	1,504505 s	0,000498 s	0,001039 s	0,00%
input_random_12_25	25	3	3018	9	9	1,956101 s	0,000647 s	0,003081 s	0,00%
input_random_13_50	50	3	14671	15	15	55,862204 s	0,003806 s	0,105783 s	0,00%
input_random_14_50	50	3	14671	16	16	56,113932 s	0,003824 s	0,026554 s	0,00%
input_random_15_50	50	3	14671	14	14	47,500851 s	0,003237 s	0,010064 s	0,00%
input_random_16_50	50	3	14671	10	10	50,125808 s	0,003416 s	0,007297 s	0,00%
input_random_17_75	75	3	36429	19	19	367,948603 s	0,010099 s	0,060515 s	0,00%
input_random_18_75	75	3	36429	15	15	411,393045 s	0,011292 s	0,023603 s	0,00%
input_random_19_75	75	3	36429	18	18	383,565742 s	0,010528 s	0,012333 s	0,00%
input_random_20_75	75	3	36429	16	16	358,569438 s	0,009842 s	0,139206 s	0,00%
input_random_21_100	100	3	69078	22	22	600,023862 s	0,008685 s	0,358717 s	0,00%
input_random_22_100	100	3	69078	23	23	600,020344 s	0,008685 s	0,145916 s	0,00%
input_random_23_100	100	3	69078	19	19	600,030749 s	0,008685 s	0,423114 s	0,00%
input_random_24_100	100	3	69078	24	24	600,034153 s	0,008685 s	0,076251 s	0,00%
input_random_25_125	125	3	113164	34	34	600,023827 s	0,005302 s	0,123753 s	0,00%
input_random_26_125	125	3	113164	29	29	600,037490 s	0,005302 s	0,414368 s	0,00%
input_random_27_125	125	3	113164	36	36	600,080006 s	0,005302 s	1,167751 s	0,00%
input_random_28_125	125	3	113164	31	31	600,046571 s	0,005302 s	0,909240 s	0,00%
input_random_29_150	150	3	169109	37	37	601,476198 s	0,003557 s	18,794088 s	0,00%
input_random_30_150	150	3	169109	35	35	600,148030 s	0,003549 s	1,420800 s	0,00%
input_random_31_150	150	3	169109	41	41	600,116392 s	0,003548 s	0,370980 s	0,00%
input_random_32_150	150	3	169109	39	39	600,504565 s	0,003551 s	45,015743 s	0,00%
input_random_33_175	175	3	237258	42	42	601,034209 s	0,002533 s	64,327599 s	0,00%
input_random_34_175	175	3	237258	45	45	601,037591 s	0,002533 s	10,653631 s	0,00%
input_random_35_175	175	3	237258	53	53	600,472239 s	0,002531 s	34,409900 s	0,00%
input_random_36_175	175	3	237258	43	43	600,824849 s	0,002532 s	85,599731 s	0,00%
input_random_37_200	200	3	317900	54	54	600,606470 s	0,001889 s	58,314939 s	0,00%
input_random_38_200	200	3	317900	52	52	601,603280 s	0,001892 s	53,295302 s	0,00%
input_random_39_200	200	3	317900	51	51	600,377294 s	0,001888 s	50,060799 s	0,00%
input_random_40_200	200	3	317900	61	61	601,147199 s	0,001891 s	80,373348 s	0,00%

Tabella dei risultati medi

N. Nodi	Tempo esecuzione medio	Tempo Full Cont. Medio
6	0,021377 s	0,000064 s
10	0,028310 s	0,000080 s
25	1,631369 s	0,000538 s
50	52,400699 s	0,003571 s
75	380,369207 s	0,010440 s
100	600,027277 s	0,008685 s
125	600,046973 s	0,005302 s
150	600,561296 s	0,003551 s
175	600,842222 s	0,002532 s
200	600,933561 s	0,001890 s

Analisi dei risultati

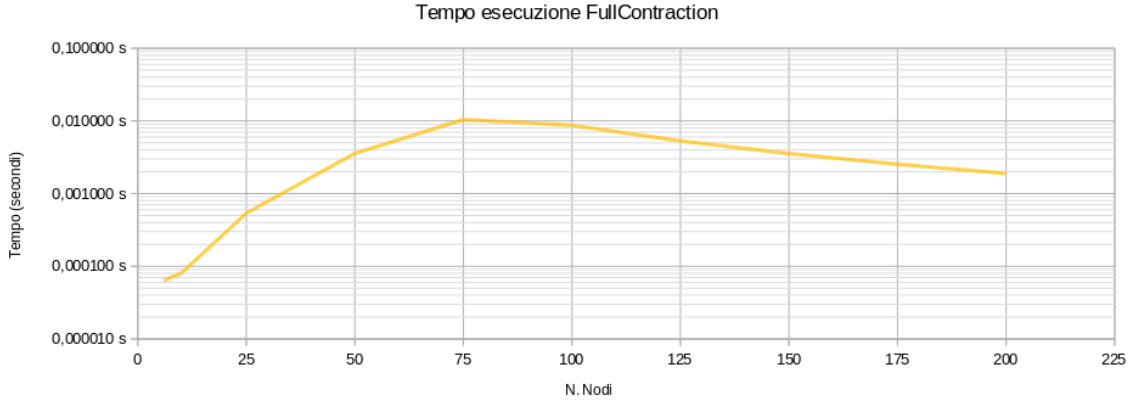
Domanda 1. Analisi FullContraction

La complessità asintotica di un operazione di FullContraction, come abbiamo già visto, risulta essere $\mathcal{O}(n^2)$.
Andando a vedere i dati presenti nella tabella dei valori medi e prendendo i dati a coppie di due osserviamo che si ha sempre un incremento dei tempi di esecuzione in funzione dell'incremento del numero di nodi. Questo trend risulta essere vero per tutti i grafi fino ad arrivare a quelli

composti da 100 nodi. Dai dati forniti si può notare come il tempo di esecuzione inizi a diminuire. Questo è dovuto al fatto che è stato impostato un timeout di 10 minuti che, quando scade, arresta l'esecuzione dell'algoritmo. I grafi con più di 100 nodi non riescono quindi a terminare l'esecuzione di FullContraction e quindi dell'algoritmo di Karger.
Osservando i dati abbiamo quindi:

- da 6 a 10 nodi (incremento di circa 67%) il tempo di esecuzione aumenta da 0.000064 secondi a 0.000080 secondi (incremento di circa 25%);
- da 10 a 25 nodi (incremento di circa 150%) il tempo di esecuzione aumenta da 0.000080 secondi a 0.000538 secondi (incremento di circa 570%);
- da 25 a 50 (incremento del 100%) il tempo di esecuzione aumenta da 0.000538 secondi a 0.00357 secondi (incremento di circa 560%);
- da 50 a 75 (incremento del 50%) il tempo di esecuzione aumenta da 0.00357 a 0.010440 secondi (incremento del 190%).

Confrontando la complessità asintotica e i risultati ottenuti possiamo dire che è rispettata in quanto, all'aumentare del numero di nodi, si ha sempre un incremento del tempo di esecuzione della procedura di FullContraction. Per un numero di nodi superiore, come già osservato, l'esecuzione non raggiunge il suo termine risultando in un decremento delle prestazioni in termini di tempo.
Questi risultati sono visibili anche nella seguente tabella



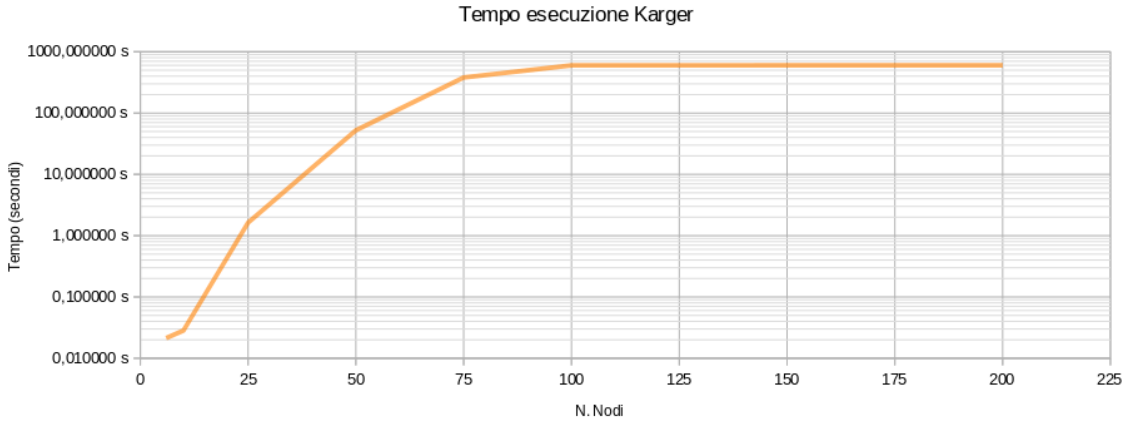
Domanda 2. Analisi algoritmo di Karger

Considerazioni simili possono essere fatte per un analisi del tempo totale di esecuzione dell'algoritmo di Karger che ha una complessità totale di $\mathcal{O}(n^4 \log n)$
Analizzando la tabella dei risultati medi possiamo osservare come:

- da 6 a 10 nodi (incremento di circa 67%) il tempo di esecuzione aumenta da 0.021377 secondi a 0.028310 secondi (incremento di circa 32%);
- da 10 a 25 nodi (incremento di circa 150%) il tempo di esecuzione aumenta da 0.028310 secondi a 1.631369 secondi (incremento di circa 5600%);
- da 25 a 50 (incremento del 100%) il tempo di esecuzione aumenta da 1.631369 secondi a 52.4 secondi (incremento di circa 3000%);

- da 50 a 75 (incremento del 50%) il tempo di esecuzione aumenta da 52.4 a 380.37 secondi (incremento del 720%);
- dopo i 100 nodi il tempo di esecuzione risulta essere di circa 600 (10 minuti) che corrisponde al timeout impostato. Questo ci fa capire che, con un numero di nodi uguale o superiore a 100, l'algoritmo di Karger impieghi più di 10 minuti per terminare la sua esecuzione.

Da queste considerazioni si può notare come gli incrementi più sostanziali nel tempo di esecuzione si hanno quando il numero di nodi almeno raddoppia rispetto all'iterazione precedente (da 10 a 25 e da 25 a 50). Questi risultati sono visibili nella seguente tabella:



Domanda 3. Analisi Discovery Time

Per quanto riguarda il Discovery Time ovvero il tempo in cui, per la prima volta, l'algoritmo trova la soluzione ottima, si può notare come, in ogni istanza, questo risulti essere di molto inferiore rispetto al tempo di esecuzione dell'intero algoritmo.
Presentiamo questi dati in una tabella riassuntiva:

N. Nodi	Tempo esecuzione medio	Tempo Disc.Time medio
6	0,021377 s	0,000297 s
10	0,028310 s	0,000436 s
25	1,631369 s	0,002808 s
50	52,400699 s	0,037424 s
75	380,369207 s	0,058915 s
100	600,027277 s	0,251000 s
125	600,046973 s	0,653778 s
150	600,561296 s	16,400403 s
175	600,842222 s	48,747715 s
200	600,933561 s	60,511097 s

Da questa tabella possiamo notare come:

- con 6 nodi il discovery time è di 0.000297 secondi a fronte di 0.021377 secondi di esecuzione totale (circa il 1.5%);
- con 10 nodi il discovery time è di 0.000436 secondi a fronte di 0.02831 secondi di esecuzione totale (circa il 1.4%);

- con 25 nodi il discovery time è di 0.0028 secondi a fronte di 1.63 secondi di esecuzione totale (circa il 1.36%);
- con 50 nodi il discovery time è di 0.037 secondi a fronte di 52.4 secondi di esecuzione totale (circa il 0.07%);
- con 75 nodi il discovery time è di 0.059 secondi a fronte di 380.4 secondi di esecuzione totale (circa il 0.016%);
- con 100 nodi il discovery time è di 0.251 secondi a fronte di circa 600 secondi di esecuzione totale (circa il 0.04%);
- con 125 nodi il discovery time è di 0.654 secondi a fronte di circa 600 secondi di esecuzione totale (circa il 0.11%);
- con 150 nodi il discovery time è di 16.4 secondi a fronte di circa 600 secondi di esecuzione totale (circa il 2.7%);
- con 175 nodi il discovery time è di 48.7 secondi a fronte di circa 600 secondi di esecuzione totale (circa il 8.11%);
- con 200 nodi il discovery time è di 60.5 secondi a fronte di circa 600 secondi di esecuzione totale (circa il 10.1%);

Quello che si può notare è che il discovery time aumenta all'aumentare del numero di nodi del grafo ma risulta essere sempre una parte esigua (al massimo il 10%) del tempo totale di esecuzione. Questo ci fa capire come la soluzione ottima venga trovata in tempi brevi rispetto al tempo totale di esecuzione ma che, per avere una probabilità di correttezza di $1 - \frac{1}{n^d}$ (quindi in alta probabilità) bisogna far eseguire Karger per un tempo di molto superiore rispetto al tempo richiesto per trovare la soluzione ottima.

Altre osservazioni

Altre osservazioni che si possono fare sono:

1. l'algoritmo di Karger implementato trova sempre la soluzione ottima ottenendo quindi un errore dello 0.00% indipendentemente dal grafo.
2. il numero di ripetizioni richieste, in funzione del numero di nodi e della costante di "accuratezza" d, aumenta di:
   - da 6 a 10 nodi (incremento di circa 67%) il numero di ripetizioni va da 97 a 346 (incremento del 256%);
   - da 10 a 25 (incremento del 150%)il numero di ripetizioni va da 346 a 3018 (incremento del 772%);
   - da 25 a 50 (incremento del 100%) il numero di ripetizioni va da 3018 a 14671 (incremento del 386%);
   - da 50 a 75 (incremento del 50%) il numero di ripetizioni va da 14671 a 36429 (incremento del 148%);
   - da 75 a 100 (incremento del 33%) il numero di ripetizioni va da 36429 a 69078 (incremento del 90%);

- da 100 a 125 (incremento del 25%) il numero di ripetizioni va da 69078 a 113164 (incremento del 63%);
- da 125 a 150 (incremento del 20%) il numero di ripetizioni va da 113164 a 169109 (incremento del 49%);
- da 150 a 175 nodi (incremento di circa 17%) il numero di ripetizioni va da 169109 a 237258 (incremento del 40%);
- da 175 a 200 (incremento del 14%) il numero di ripetizioni va da 237258 a 317900 (incremento del 34%).

Questo ci fa capire come non solo il discovery time sia molto basso rispetto all'esecuzione totale di Karger ma che la soluzione trovata è quella ottima. Il numero di ripetizioni che ci serve per aumentare il grado di affidabilità dell'algoritmo per trovare questa soluzione, diminuisce, in percentuale, all'aumentare del numero di nodi.

Conclusioni

Questa esperienza mi ha permesso di studiare, progettare e mettere mano su una categoria di algoritmi che, personalmente, non conoscevo se non per degli accenni visti durante la laurea triennale. L'algoritmo di Karger è un algoritmo di tipo Montecarlo quindi un algoritmo randomizzato che non assicura la correttezza della soluzione ritornata. E' stato davvero interessante vedere e capire, prima in via teorica e poi in pratica, come sia possibile mettere insieme l'algoritmica e lo studio delle probabilità per arrivare ad avere degli algoritmi che, nonostante compiano scelte casuali, garantiscano una soluzione corretta.
Rispetto agli obiettivi iniziali dell'esperimento possiamo osservare e concludere che l'algoritmo richiede un gran numero di ripetizioni per garantire una probabilità del $1 - \frac{1}{n^d}$ di trovare la soluzione corretta ma che questa, di fatto, viene trovata in tempi relativamente brevi rispetto al tempo di esecuzione totale richiesto per completare tutte le iterazioni. Nonostante l'operazione di FullContraction richieda un tempo di esecuzione breve per un numero di nodi oltre il 100 il tempo di esecuzione va oltre i 10 minuti rendendo quindi l'algoritmo inutilizzabile per usi al di fuori dell'ambito accademico.
In conclusione ritengo che l'esperienza sia stata molto utile per capire come funzionano questi algoritmi che, personalmente, non avevo mai utilizzato e che ho trovato molto interessanti da conoscere e capire.