Arduino e le Tabelle Hash

Gennai Gian Maria

2022

1 Introduzione

Quale sono le difficoltà o limitazioni che troviamo nell'andare ad utilizzare Arduino per l'implementazione di una tabella Hash?
Leggendo troverai prima la spiegazione di come ho fatto ad implementare su Arduino una tabella Hash e poi le varie considerazioni che ne ho tratto avendo fra le mani dei dati misurati sperimentalmente.

2 Arduino

Arduino è una scheda programmabile con vari ingressi e uscite che permettono di comunicare con il mondo esterno, come parte software ha un suo IDE che permette all'utente di interfacciarsi con questo.
La scheda utilizzata per questi esperimenti non è un vero e proprio Arduino ma un **Elegoo**; è stato parlato solo di Arduino, anche se non è effettivamente quello, perchè è un dispositivo open-source e quindi molti lo hanno riprodotto e messo in commercio con le stesse caratteristiche ma sotto diverso nome. Detto questo le due schede sono identiche e hanno solo il nome differente. Entrambi funzionano grazie ad un **ATmega328** come processore con 32KB di SRAM e 1KB di EEPROM.
Questi dispositivi permettono di fare operazioni anche interagendo con altri strumenti collegabili tramite pin(come LED, motori, display, pulsanti,...)come vedremo in seguito.
La memoria e le capacità di calcolo su questi dispositivi sono molto ridotte, non sono stati creati per contenere grandi quantità di dati ne tanto meno per eseguire calcoli complessi e in modo rapido.

2.1 Codice per Arduino

I vari algoritmi implementati con Arduino sono stati tutti scritti seguendo la sintassi del C++, ovvero quella di default; non è possibile scrivere del codice in altri linguaggi e dopo andare a caricarlo al suo interno se non con detemrminate schede.
All'interno IDE Arduino vengono forniti di default due metodi che sono:

- **setup()**: metodo che mi permette di inizializzare la scheda, andando a fare operazioni preliminari.
- **loop()**: metodo che si ripete all'infinito (è possibile fermarlo solo tramite specifici comandi).

Nel codice fatto per questi test ci sono anche variabili o vettori al di fuori di tutti i metodi e quelle sono *variabili globali* ovvero delle variabili che sono visibili ed utilizzabili su tutti i metodi che ci sono.

3 Tabelle Hash

Queste sono particolari tabelle formate da un certo numero di spazi su cui poter scrivere dei valori. Il tipo dei valori non è particolarmente importante anche se noi per semplicità abbiamo utilizzato numeri interi.
In queste tabelle ci sono vari metodi che mi permettono di inserire valori in determinati punti per poi andare a ritrovarli nel minor tempo possibile e risolvere il **problema delle collisioni** spiegato più avanti.
Ci sono vari tipi di esplorazione della tabella per l'inserimento dei valori all'interno di questa e io ne ho implementati tre:

- Metodo lineare
- Metodo quadratico
- Doppio hash

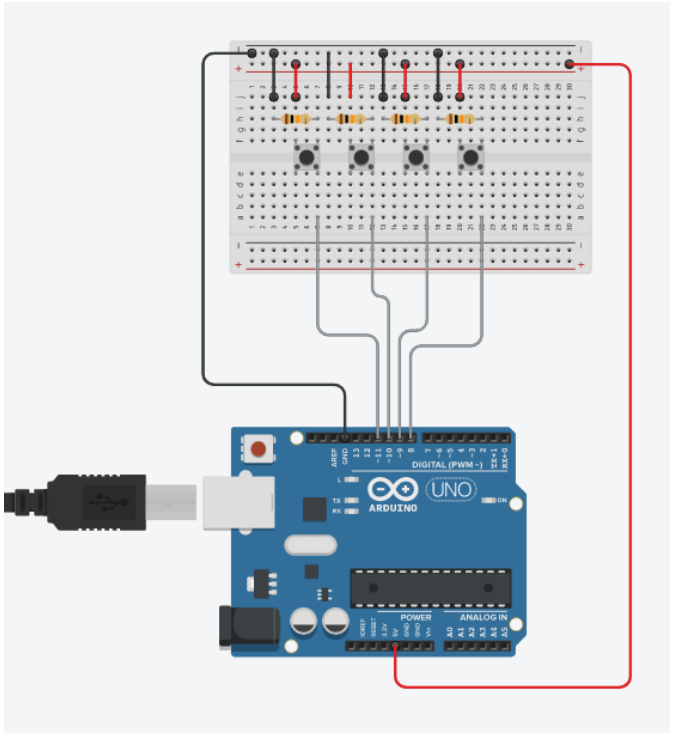
Per l'implementazione dei vari metodi che abbiamo elencato sopra, è importante spiegare cos'è una **funzione hash**, ovvero una vera e propria funzione matematica che prende il valore da inserire nella tabella e lo "trasforma" in qualcos'altro. La trasformazione del valore da inserire avviene tramite alcuni calcoli e per i nostri test abbiamo utilizzato il *metodo delle divisioni*, un calcolo che dato il valore da inserire e la dimensione della tabella ne calcola il modulo. L'unica accortezza da prendere per creare una tabella di questo tipo è quella di prenderla di una grandezza pari ad un numero primo non troppo vicino ad una potenza di due.

3.1 Codice per Tabella Hash

Possiamo vedere i tre metodi principali per l'implementazione di una tabella Hash ovvero *inserimento, cancellazione* e *ricerca*.
Come annunciato prima, nelle tabelle di questo tipo ci possono essere delle collisioni e questo perchè magari si cerca di inserire un nuovo elemento nella tabella ma in una posizione già presa da un valore precedente. Ovviamente quando arriviamo ad avere la tabella piena non sarà più possibile inserire valori. In verità ci sarebbe un metodo che mi permette di continuare ad inserire elementi nella tabella anche se questa è piena ovvero far partire da ogni casella un lista con puntatori e continuare ad inserire elementi sulla lista; questa idea è stata abbandonata sul nascere in quanto la limitatissima memoria di Arduino ci avrebbe permesso di inserire una tabella davvero piccola contando che doveva essere creata una lista per ogni casella della tabella.
La prima parte è sempre uguale per tutte le implementazioni che abbiamo. Per l'inserimento si va a prendere un valore randomico in un range compreso fra 0 e 999 e si va a calcolare la funzione hash come spiegato sopra; ottenuta la funzione hash andiamo a controllare se la casella della tabella corrispondente al valore della funzione è libera, se lo è mettiamo il valore randomico li ma se questo non succede dobbiamo spostarci in avanti e trovare una nuova posizione. Per la ricerca il funzionamento è lo stesso, andiamo a prendere un valore random tra 0 e 999 e andiamo e controllare se questo elemento è presente nella posizione calcolata tramite la funzione hash sennò ci spostiamo, come accennato sopra ci fermiamo in 3 casi: se il valore viene trovato, se la tabella è stata percorsa tutta senza trovare il valore oppure se durante la ricerca trovo un valore che corrisponde al valore di tabella vuota(1000).
La cancellazione è una ripetizione del codice di ricerca con la sola modifica di eliminare il valore trovato al suo interno con un nuovo valore speciale differente da quello utilizzato per inizializzare la tabella sennò, dato che il metodo per la ricerca dei valori si blocca anche nel caso in cui trova il valore *1000*(valore assegnato per indicare la casella vuota *NIL*) per evitare che la macchina controlli tutte le caselle inutilmente. Il valore per indicare che il valore all'interno della tabella è stato eliminato è *1111*.
Ogni qual volta una casella è occupata sia nel momento in cui devo inserire un nuovo valore oppure sto facendo la ricerca e trovo il valore sbagliato devo spostarmi e per fare questo ci sono diversi metodi, ne sono stati implementati tre.
Nel **metodo lineare** abbiamo un indice che come dice il nome si incrementa in modo lineare, quindi se la posizione calcolata dalla funzione hash mi crea un collisione, questo metodo mi va ad incrementare un indice appartenente ai numeri interi **j** il quale va a sommarsi al valore della funzione hash. Dopo questa somma per trovare quindi la posizione nella tabella viene calcolato di nuovo il modulo, sennò ovviamente ci potremmo trovare nella situazione di andare fuori dal range.
Nei due metodi successivi il principio è molto simile solamente che nel **metodo quadratico** avremo che sempre viene controllato il posto calcolato dalla funzione hash, se questo riscontra una collisione allora entra in campo di nuovo l'indice intero **j** il quale però non andra semplicemente a sommarsi al valore della funzione hash ma ci sarà una costante moltiplicativa associata e un'altra somma composta dal quadrato dell'indice j e sempre moltiplicato per una costante (posizione = h(key)+j*A+(j*j)*B , con A e B costanti). Notiamo bene che se la costante moltiplicata all'indice quadro mi va a 0 io ho di nuovo un metodo di esplorazione lineare.
Nell'ultimo metodo di esplorazione, ovvero il doppio hash, andiamo a fare la stessa cosa che abbiamo nel metodo lineare ma in più moltiplichiamo all'indice j una nuova funzione hash che viene creata andando a sommare 1 al valore ottenuto dal modulo tra la chiave e un valore *m'*, ovvero il valore un valore più piccolo della grnadezza di tutta la tabella (es. m'=sizeTable-1).

4 Hardware e schemi

Per lo svolgimento di questo esperimento ho voluto, per ogni metodo di esplorazione creare un nuovo file. Avendo però dovuto fare quattro operazioni differenti ho voluto sfruttare uno dei vantaggi che ha arduino ovvero quello di poter essere connesso con dei pin a vari elementi, ho preso quattro bottoni e li ho collegati ai pin 8, 9, 10 e 11 e per ognuno ho fatto in modo che quando l'uscita risulta alta (grazie alla possibilità di leggere il valore impostando il pin come pin di input con il metodo pinMode(n°pin,INPUT/OUTPUT) e dopo andando ad utilizzare digitalRead(n°pin) che è il metodo che controlla lo stato del pin se in ingresso riceve un valore alto, 5V oppure basso 0V) mi viene svolta un operazione differente.
La quarta operazione che ho citato sopra serve semplicimente per stampare la tabella sul moitor del pc.
Grazie all'aiuto di *Tinkercad* un software AUTODESK gratuito ho potuto creare un immagine della struttura generale.



5 Spazio in memoria

Utilizzando Arduino infatti abbiamo potuto creare una tabella di dimensioni molto ridotte, ovvero 631 slot, e questo perchè doveva essere un numero primo e non doveva superare una certa grandezza a causa della memoria limitata del dispositivo. Forse potevamo osare con un altro po' di slot in più ma avremmo sovraccaricato inutilmente il dispositivo rischiando problemi nell'esecuzione del codice.
Questa grandezza della tabella è stata adottata per tutti e tre gli scketch dato che comunque le dimensioni non cambiano molto e l'IDE di Arduino ci consigliava di non aumentare ancora.
Sempre grazie all'IDE di Arduino posso già capire le dimensioni in byte che occupa il nostro codice compreso di memoria utilizzata per la tabella. Possiamo vedere i dati riassunti nella seguente tabella.

	Lineare	Quadratico	Doppio Hash
Sketch	4810	4194	4842
v. locali	516	516	514
v. globali	1532	1532	1534

Sapendo che lo spazio totale per lo sketch è di *32256 bytes* e che per le variabili è *2048 byte* possiamo capire bene che anche con un semplice codice come quello applicato qui abbiamo occupato una grande quantità di memoria e ovviamente è sempre consigliato non arrivare mai al limite per non incidere sulle capacià del dispositivo.

5.1 Tempo di esecuzione

Per quanto riguarda il tempo, le misure sono state fatte grazie alla funzione *micros()* che ci ha permesso di andare a prendere il tempo in un punto di partenza e in uno di arrivo per poi calcolare il tempo effettivo tra i due punti tramite una semplice sottrazione. Ho fatto una decina di prove per ogni metodo e ho fatto una media dei valori. Qui sotto possiamo vedere una tabella con i vari tempi di esecuzione.

	Lineare	Quadratico	Doppio Hash
Insert	304625.6	125871.6	128650
True Search	155.2	922	131.2
False Search	5097.6	6982.5	7088.4
Delete	468	44364.8	40465.2

Da questi tempi possiamo capire che la differenza tra i tre metodi non è enorme e delle volte mi va a vantaggio uno e delle volte un altro. Su una decina di prove per ogni test però possiamo notare come la cancellazione nel metodo lineare risulti molto più veloce al contrario dell'inserimento che risulta abbastanza lento. Poi per quanto riguarda la ricerca dell'elemento inesistente nella tabella i valori si avvicinano molto anche se vanno ad aumentare partendo da lineare fino ad quadratico e invece per la ricerca dell'elemento corretto il metodo quadratico passa in vantaggio. Ovviamente molto dipende anche dalla posizione dell'elemento che vado ad ispezionare e quindi il tempo varia anche a causa dell'utilizzo di valori randomici.

6 Conclusioni

Tirando le somme possiamo notare come le capacità di questi dispositivi siano davvero limitate, sia in termini di spazio che in termini di tempo di esecuzione. Ovviamente stiamo parlando di un dispositivo con un prezzo inferiore alle centinaia di euro e molto piccolo non possiamo metterlo a confronto con un normale calcolatore che ha molta più potenza di calcolo e spazio in memoria ma ha un costo molto più elevato ed è molto più grande.