Relazione sulle prime lezioni di informatica

Emanuele

In queste prime lezioni abbiamo affrontato l'informatica procedendo a soluzioni di problemi, quindi prima cercando di risolvere i problemi presentati e successivamente generalizzando ed analizzando la soluzione.

Sono stati quindi proposti dei problemi che il nostro cervello risolve automaticamente ogni giorno ma con meccanismi che non sono completamente consci e quindi non espliciti: cercare di far risolvere al computer gli stessi basilari problemi ci consente quindi di capire come il nostro cervello svolge molte funzioni; i problemi proposti sono tutti impostati sul confronto di oggetti differenti: si presupponeva di avere otto oggetti differenti e di avere a disposizione soltanto una bilancia a due piatti per confrontare gli oggetti.

Questi sono i problemi che sono stati affrontati:

a) Trovare il maggiore numero fra un insieme di otto numeri arbitrari.
b) Trovare il più grande e il più piccolo fra un insieme di otto numeri arbitrari.
c) Trovare i due numeri più grandi fra un insieme di otto numeri arbitrari.

Si è dunque proceduto nel seguente modo:

a) Presentazione del problema;
b) Analisi e sviluppo di un algoritmo risolvente;
c) Ottimizzazione dell'algoritmo;
d) Traduzione dell'algoritmo da linguaggio naturale a linguaggio macchina (Pascal);
e) Ottimizzazione del codice Pascal;
f) Considerazioni sul numero di confronti in funzione del numero di dati analizzati.

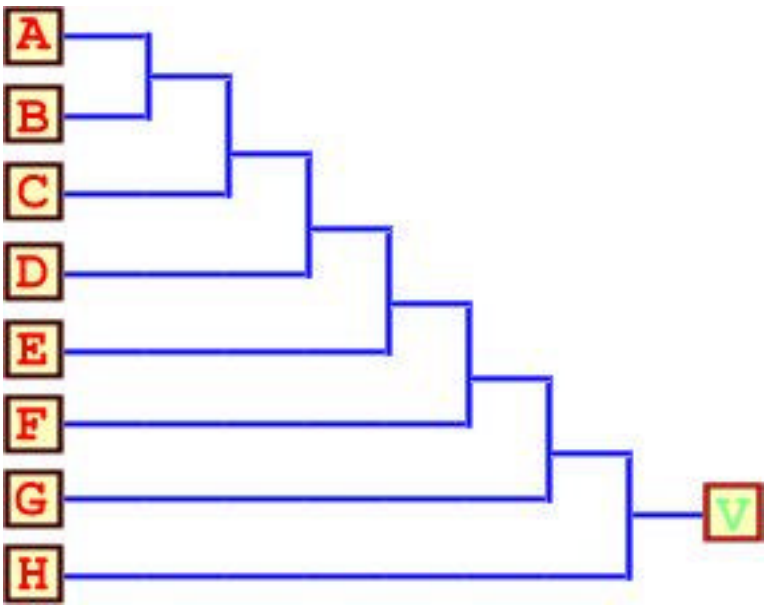
Il primo problema

Presentazione del problema.

Trovare il più grande numero, potendoli soltanto confrontare a due a due, in un insieme di otto numeri arbitrari.

Analisi e sviluppo dell'algoritmo risolvente ottimizzazione.

Si può procedere confrontando tutti i numeri fra di loro ma così avremmo una sovrabbondanza di informazioni inutilizzate.
Questi sono i due alberi che schematizzano i due dei migliori (con minor numero di confronti) algoritmi risolventi:



Ogni incontro di tre linee è un confronto; come si può notare il numero di confronti è sette (numero di oggetti – 1); A,B,C,D,E,F,G,H sono gli oggetti e V è il maggiore alla fine dell'elaborazione.

Traduzione dell'algoritmo da linguaggio naturale a linguaggio macchina (Pascal) e ottimizzazione.

Il codice Pascal che applica il secondo algoritmo (utilizzato da Scarpa e Segatto) deve conservare (in un array) tutti gli elementi e confrontare i numeri a coppie. Si ha dunque che per raggiungere lo scopo si devono allocare una variabile array di otto elementi integer più un contatore(per i cicli for).

Si nota subito invece che il primo algoritmo è quello che conserva meno informazioni (che in questo caso sono inutili) e che analizza contemporaneamente il minor numero di elementi: è dunque questo l'algoritmo che permette di creare un codice Pascal ottimizzato al massimo.

Si ha infatti che bisogna allocare soltanto tre variabili integer: i due oggetti da confrontare e un contatore (per i cicli for). Inoltre non ci interessa conoscere alla fine dell'elaborazione l'insieme iniziale dei numeri ma soltanto il maggiore (questo spiega l'assenza di un array contenente gli otto numeri) e dobbiamo confrontare due numeri alla volta tenendo in memoria soltanto il più grande (fino a quel momento) e quello con cui confrontarlo.

Il codice (minimo) Pascal risolvente è il seguente:

```
uses crt;
```

```
var a,b,c:integer;

begin
clrscr;
readln(a);
for c:=0 to 6 do
begin
readln(b);
if b>a then a:=b;                    (* Totale di 7 confronti *)
end;
writeln('Il maggiore è: ',a);
readln;
end.
(* tutto in 13 righe di codice Pascal! *)
```

Considerazioni sul numero di confronti in funzione del numero di dati analizzati.

Effettuando varie prove si può empiricamente osservare che il numero minimo di confronti è *n-1* dove *n* è il numero di oggetti. Ma perché?

La risposta sta nella natura stessa del confronto: esso è binario, cioè prende in esame soltanto due elementi alla volta, quindi poiché si prende la coppia di elementi e il maggiore lo si confronta con tutti gli altri si ha che dalla prima coppia (1° confronto) esce un numero solo che poi si confronta (2° confronto) con il terzo (*n-1* sono gli elementi rimasti) fino a rimanere con *n-n+1* elementi quindi il numero di confronti è *n-1* (cioè *n-k* dove *k* è il numero di oggetti maggiori [sempre 1]).

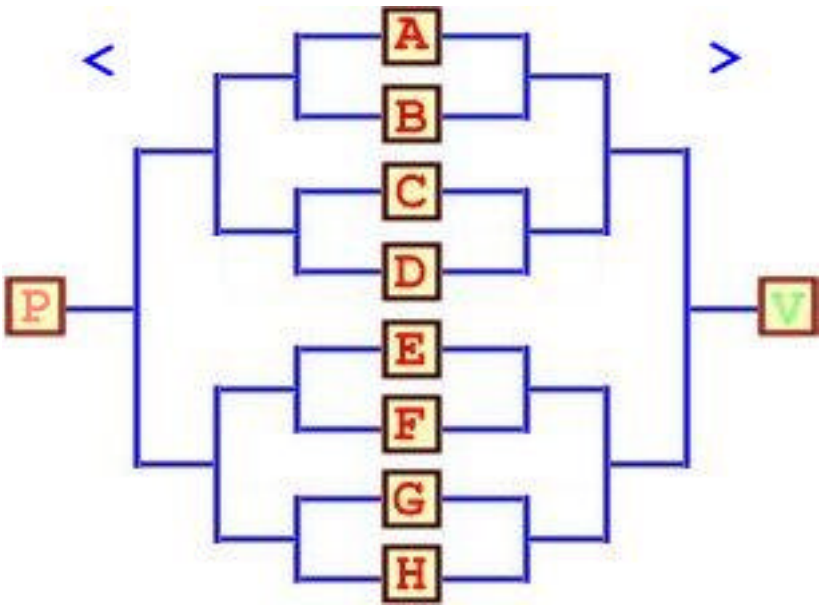
Il secondo problema

Presentazione del problema.

Trovare il più grande e il più piccolo numero, potendoli soltanto confrontare a due a due, in un insieme di otto numeri arbitrari.

Analisi e sviluppo dell'algoritmo risolvente ottimizzazione.

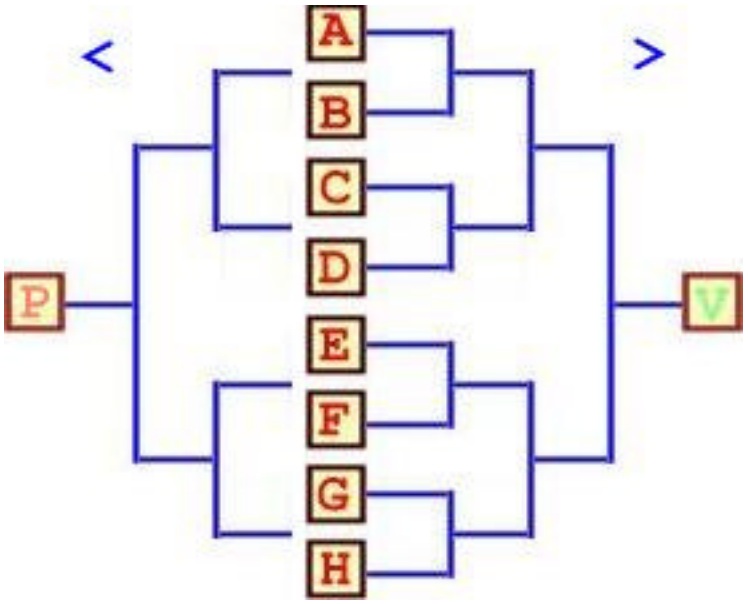
Si può procedere applicando prima l'algoritmo sopra descritto (quello del primo problema) per trovare il maggiore e poi applicarlo al contrario trovando il minore



Da questo lato procedono solo i numeri minori

Da questo lato procedono solo i numeri maggiori

ma così avremmo una sovrabbondanza di informazioni inutilizzate: dai primi quattro confronti infatti sappiamo già qual'è il maggiore e il minore di ogni coppia iniziale; questa informazione ci consente quindi di eliminare i primi quattro confronti a sinistra che sono inutili.
Questo è l' albero che schematizza il migliore (con minor numero di confronti) algoritmo risolvente:



Ogni incontro di tre linee è un confronto; come si può notare il numero di confronti è dieci;
A,B,C,D,E,F,G,H sono gli oggetti e V è il maggiore e P è il minore alla fine dell'elaborazione.

Traduzione dell'algoritmo da linguaggio naturale a linguaggio macchina (Pascal) e ottimizzazione.

Il codice Pascal che applica il secondo algoritmo deve conservare (in un array) tutti gli elementi e confrontare i numeri a coppie. Si ha dunque che per raggiungere lo scopo si devono allocare una variabile array di otto elementi integer più un contatore(per i cicli for) e una variabile temporanea di cui vedremo in seguito l'utilità.
Abbiamo visto quindi che si ha la necessità di mantenere in memoria tutte le variabili iniziali perché i numeri minori devono essere anch'essi confrontati per l'individuazione del minore.
Possiamo a questo punto agire in questo modo: si operano i primi quattro confronti e si mette al primo posto di ogni coppia il maggiore e nel secondo il minore;
ad esempio avendo inizialmente la tabella

A	B	C	D	E	F	G	H
3	2	4	5	8	6	1	7

Dopo i primi quattro confronti si avrà:

A	B	D	C	E	F	H	G
3	2	5	4	8	6	7	1

quindi nelle caselle a indice pari (partendo da zero!) avremo tutti i numeri che nei primi quattro confronti si sono rivelati maggiori: possiamo dunque procedere con l'algoritmo del primo problema per selezionare il maggiore fra gli elementi a indice pari e il minore fra gli elementi a indice dispari.

Questo è il codice Pascal:

```
program minmax;
uses crt;
var c,tmp:integer; o:array[0..7]of integer;

begin
clrscr;
c:=0;
repeat
 readln(o[c]);
 readln(o[c+1]);
 if o[c]<o[c+1] then
 begin
  tmp:=o[c+1];
  o[c+1]:=o[c];
  o[c]:=tmp;
 end;
 c:=c+2;
until c>7;
for c:=1 to 3 do
begin
 if o[0]<o[c*2] then o[0]:=o[c*2];
 if o[1]>o[c*2+1] then o[1]:=o[c*2+1]
end;
writeln('Il maggiore è: ',o[0]);
writeln('il minore è:   ',o[1]);
readln;
end.
```

(* tutto in 25 righe di codice Pascal! *)

La variabile temporanea (nel codice *tmp*) è utilizzata per l'inversione degli elementi dell'array: per invertire i valori di due variabili *a* e *b* bisogna infatti procedere in questo modo:

```
tmp:=a;
a:=b;
b:=tmp;
```

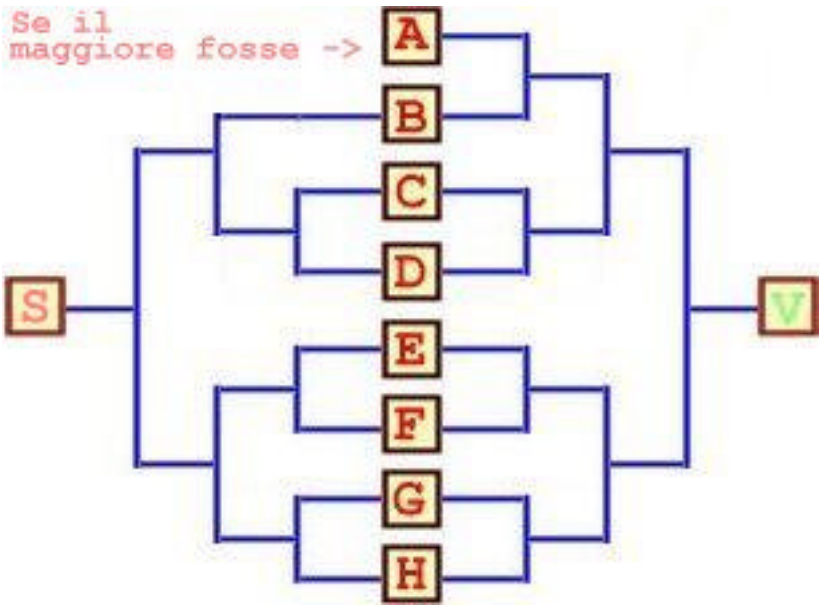
Il terzo problema

Presentazione del problema.

Trovare i due numeri più grandi, potendoli soltanto confrontare a due a due, in un insieme di otto numeri arbitrari.

Analisi e sviluppo dell'algoritmo risolvente ottimizzazione.

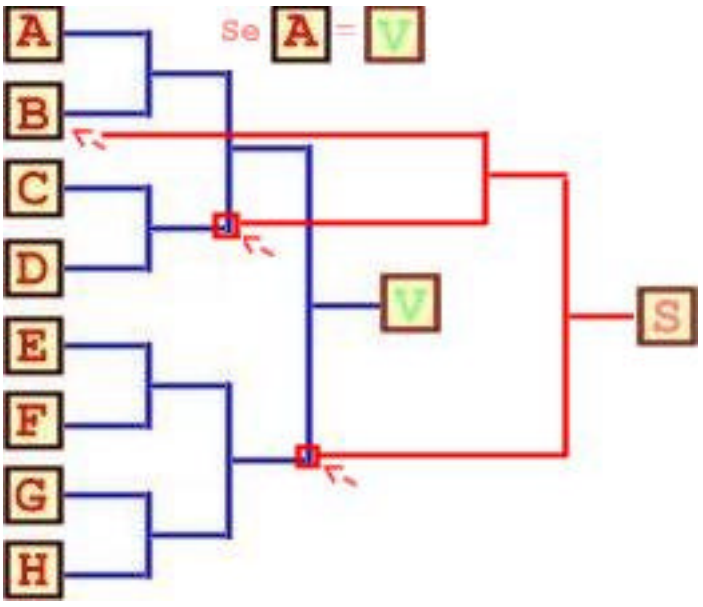
Si può procedere applicando prima l'algoritmo del primo problema per trovare il maggiore e poi applicarlo con tutti gli elementi rimanenti



ma così avremmo una sovrabbondanza di informazioni inutilizzate: dai primi quattro confronti infatti sappiamo già qual'è il maggiore e il minore di ogni coppia iniziale; questa informazione ci consente quindi di eliminare i confronti fra E-F e G-H (nell'esempio) che sono inutili in quanto sappiamo già quale fra E-F e G-H è maggiore e quindi può concorrere ad essere il secondo numero maggiore come schematizzato in questo secondo albero:

Si noti come si sia passati da 13 confronti a 10.

Ma anche questa soluzione è ulteriormente ottimizzabile e mantiene alcune informazioni inutili. Sfruttando tutte le informazioni ricavate dai primi sette confronti (quelli per ricavare il primo massimo numero) siamo riusciti a creare un algoritmo che risolve il problema con nove confronti. Come? Analizziamo le informazioni che possediamo:
sappiamo che gli unici tre elementi che possono ambire al "secondo posto" sono quelli che hanno perso il confronto diretto con il massimo; infatti sappiamo che quelli si sono rivelati maggiori agli altri perché si sono "guadagnati" il confronto con il maggiore: trovando il maggiore fra questi tre elementi (con l'algoritmo del primo problema) troviamo il secondo numero più grande.
Questo è l' albero che schematizza il migliore algoritmo risolvente:



Ogni incontro di tre linee è un confronto; come si può notare il numero di confronti è nove; A,B,C,D,E,F,G,H sono gli oggetti e V è il maggiore e S è il secondo numero maggiore alla fine dell'elaborazione. Supponendo che A sia il maggiore il secondo può solo che essere solo o B oppure il maggiore fra C e D oppure il maggiore fra E-F-G-H.

Traduzione dell'algoritmo da linguaggio naturale a linguaggio macchina (Pascal) e ottimizzazione.

Il codice Pascal che applica il terzo algoritmo deve conservare (in un array) tutti gli elementi e confrontare i numeri a coppie. Si ha dunque che per raggiungere lo scopo si devono allocare una variabile array di otto elementi integer più un contatore(per i cicli for) e una variabile temporanea di cui abbiamo parlato per il secondo problema.
Abbiamo visto quindi che si ha la necessità di mantenere in memoria tutte le variabili iniziali perché i numeri minori devono essere anch'essi confrontati per l'individuazione del secondo.
Possiamo a questo punto agire in questo modo: si divide (mentalmente) in due l'array e si operano i primi quattro confronti e si mette al primo posto di ogni coppia il maggiore e nel secondo il minore; ad esempio avendo inizialmente la tabella

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
3	2	4	5	8	6	1	7

Essa verrà raffigurata nel seguente modo:

A	3
C	4
E	8
G	1

B	2
D	5
F	6
H	7

Dopo i primi quattro confronti si avrà:

A	3
D	5
E	8
H	7

B	2
C	4
F	6
G	1

[4 confronti]

quindi nelle caselle a sinistra (a indice pari, partendo da zero) avremo tutti i numeri che nei primi quattro confronti si sono rivelati maggiori.
Passiamo ora al confronto dei maggiori per stabilire il maggiore.
C'è però un fatto importante: le coppie originali non devono essere sciolte!
Se nel confronto fra A e D vince D, A e D si scambiano le posizioni e lo fanno anche i loro corrispettivi compagni C e B a prescindere dal loro valore.
Dopo la seconda serie di confronti:

D	5
A	3
E	8
H	7

C	4
B	2
F	6
G	1

[2 confronti]

Dopo l'ultimo confronto per decretare il massimo:

E	8
A	3
D	5
H	7

F	6
B	2
C	4
G	1

[1 confronto]

Quindi ora sappiamo il maggiore (E) che è nella prima casella dell'array.
Sappiamo inoltre che quelli che si sono confrontati direttamente con E sono nelle caselle (partendo da zero!) 1 5 e 6 (F,D,H).
Da qui basta applicare l'algoritmo del primo problema per stabilire il massimo fra questi tre trovando il secondo.

F > D
F < H

[2 confronti]

Il totale di confronti è dunque nove: si è giunti alla massima ottimizzazione.

Questo è il codice Pascal:

```
program stnd;
uses crt;
var c,tmp:integer;o:array[0..7]of integer;
begin
clrscr;
for c:=0 to 3 do
begin
 readln(o[c]);
 readln(o[c+4]);
 if o[c]<o[c+4] then                    (*   4 confronti   *)
 begin
  tmp:=o[c+4];
  o[c+4]:=o[c];
  o[c]:=tmp;
 end;
end;
for c:=0 to 3 do
begin
  if o[c]<o[c+1] then                   (*   2 confronti   *)
  begin
   tmp:=o[c];o[c]:=o[c+1];o[c+1]:=tmp;
   tmp:=o[c+4];o[c+4]:=o[c+5];o[c+5]:=tmp;
  end;
  c:=c+1;
end;
if o[0]<o[2] then                       (*   1 confronto   *)
begin
 tmp:=o[0];o[0]:=o[2];o[2]:=tmp;
 tmp:=o[4];o[4]:=o[6];o[6]:=tmp;
end;
if o[1]<o[2] then o[1]:=o[2];
if o[1]<o[4] then o[1]:=o[4];           (*   2 confronti   *)
writeln('Il primo è:    ',o[0]);
writeln('il secondo è:  ',o[1]);
readln;
end.                                    (* TOT: 9 confronti *)(*36 righe...*)
```

Conclusioni

Abbiamo dunque risolto e applicato all'informatica i tre problemi.
Non è stato trattato per tutti i problemi l'aspetto del numero di confronti in funzione del numero di oggetti ma sarà presto affrontato anche questo argomento.
Per ora è tutto.

Emanuele