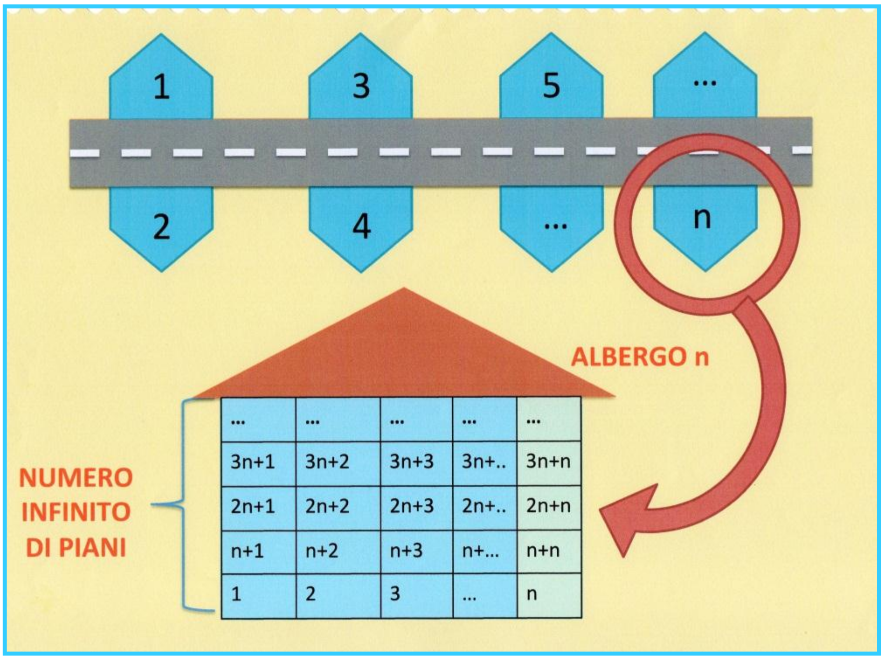
Il problema dell’albergatore

Liceo Scientifico “Marie Curie” di Tradate (VA)
Classe 2A
Insegnante di riferimento: Marina Wagner
Ricercatrice: Fiammetta Carniani
Partecipanti: Riccardo Bernacchi, Giacomo Bottinelli, Giorgia Bortolan, Sara Brun, Chiara Cieri, Alba Desogus, Jacopo Figuriello, Daniele Galante, Chiara Gallivanone, Marco Lomazzi, Edoardo Marelli, Armando Marra, Nicolas Mattara, Ilaria Meazza, Jacopo Peron, Sara Pizzo, Silvia Randi, Silvano Rezzonico, Federico Saitta, Federico Vignati, Federica Zuccoli.

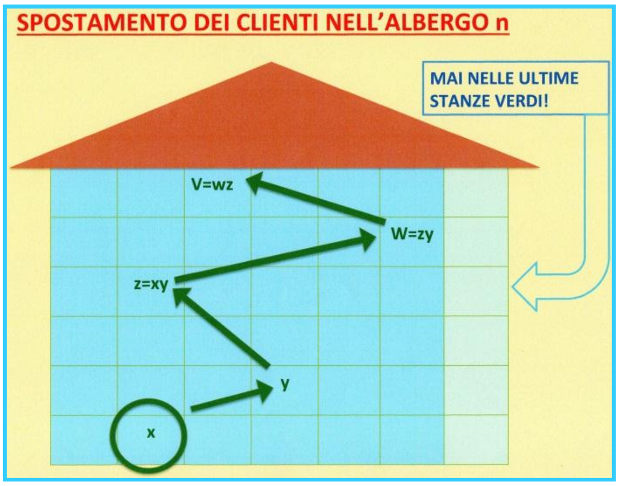
In una via vi sono infiniti alberghi. Ogni albergo ha un numero civico. Gli alberghi hanno tante stanze per piano quanto è il numero civico.
Se si considera l’albergo di numero civico n, le stanze sono così numerate

- *da 1 a n per quelle del primo piano;*
- *da n+1 a n+n quelle del secondo piano;*
- *da 2n+1 a 2n+n quelle del terzo piano;*

… e così via.
Il numero di piani di ciascun albergo è infinito.
L’ultima stanza di ogni piano (cioè la stanza kn con $k \in N - \{0\}$) è l‘uscita.



L’albergo di numero civico n è pieno. Arriva un cliente, l’albergatore non vuole perderlo e quindi lo manda in una qualunque stanza, per esempio x. L’inquilino della stanza x va nella stanza y e manda l’inquilino della stanza y nella stanza xy e così via. Gli inquilini non devono però finire nell’ultima stanza per piano, cioè all’uscita, perché in questo caso l’albergatore perderebbe il cliente.



Si chiede:

1) *Quali sono i numeri civici degli alberghi che non perdono clienti all'arrivo di un ospite nuovo?*
2) *Se n è il numero civico di un albergo che ha la caratteristica del punto 1), allora quanti alberghi esistono fatti in questo modo e aventi numero civico minore di n? (con n≥2)*

Lavoro svolto

Il problema ci ha interessato fin da subito.
Definendo le seguenti due regole abbiamo dato una risposta al primo quesito.

1. La persona A che entra nell'albergo non andrà mai direttamente all'uscita;
2. Se la persona A va nella stanza x della persona B, la persona A non può mandare la persona B alla stanza n, che è l'uscita.

Con queste due regole, possiamo dire che **gli alberghi da cui non usciranno mai clienti sono quelli che hanno come numero civico n un numero primo**.
Infatti se n è primo ha come fattori della scomposizione solo 1 e se stesso.
Per quanto detto prima, A potrà mandare B, che si trova nella stanza x, solo in una stanza y diversa da n o da un multiplo di n.
Abbiamo tre casi:

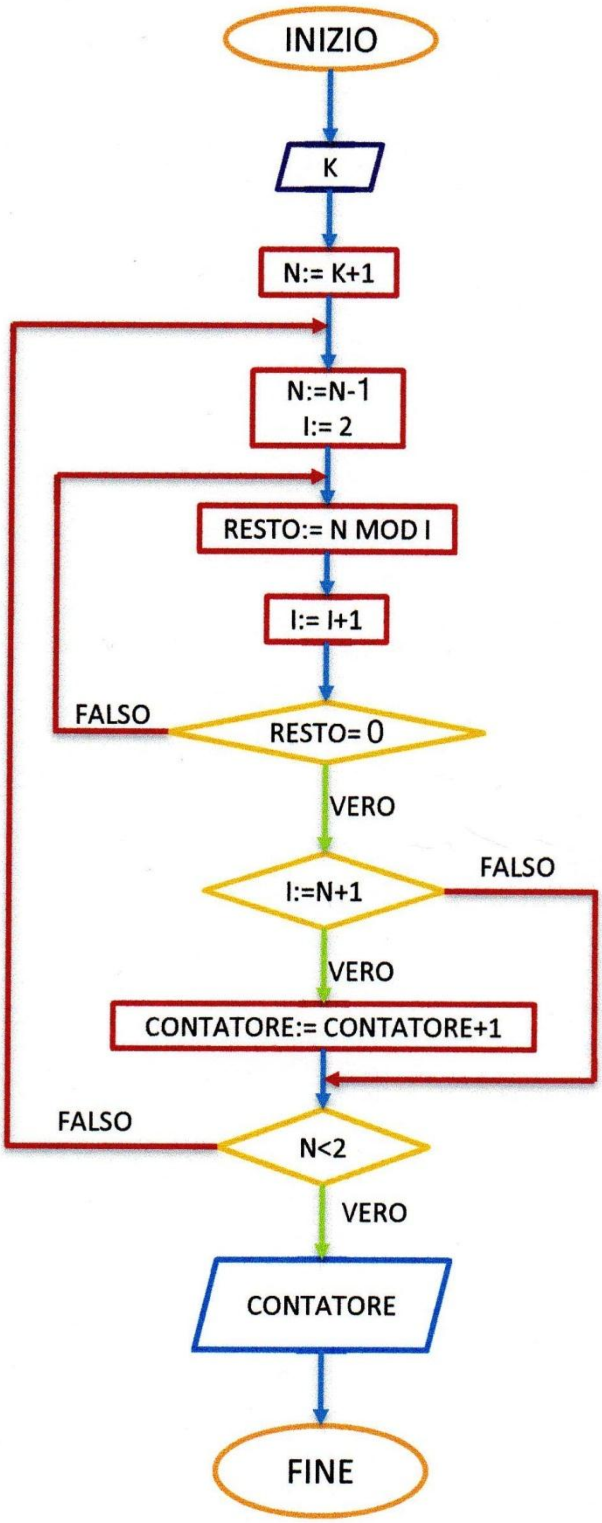
a) A manda B nella stanza y diversa da 1 quando B si trova in una stanza generica x (con x diverso da n perché nessuno alloggia all'uscita). In questo caso nessuno uscirà dall'albergo perché se n è primo nessun numero y, diverso da 1, moltiplicato per un altro, potrà mai dare come risultato n.
b) A manda B nella stanza y=1 quando B si trova nella stanza x=1. In questo caso i due clienti andranno avanti all'infinito a scambiarsi di posto nella stanza 1, e nessuno uscirà dall'albergo.
c) A manda B nella stanza y=1 quando B si trova in una stanza x diversa da 1 e da n (nessuno alloggia nell'uscita). Allora B manderà C nella stanza z=x*1, ovvero la stessa stanza x. x è diverso da 1 e x non divide n, perché n è primo, quindi nessuno uscirà mai dall'albergo.
d) Quindi in nessun caso i clienti usciranno dall'albergo.

Il problema dell'albergatore tratta quindi **i numeri primi.**
"Quanti sono i numeri primi?" La domanda è scaturita immediatamente, ma la risposta riuscivamo solo ad intuirla: sono infiniti. Non siamo riusciti a dimostrarlo, ma facendo delle ricerche abbiamo trovato che già Euclide aveva fornito una dimostrazione per assurdo nel libro "Gli elementi", libro IX, proposizione 20. La comprensione non è stata affatto semplice perché ci siamo scontrati con un linguaggio diverso da quello moderno.

Per rispondere al secondo quesito proposto dalla ricercatrice, ci siamo resi conto che dovevamo conoscere la successione dei numeri primi. Si potevano rispolverare le tavole utilizzate alla scuola media, ma non avevamo intenzione di andare a contare uno a uno quanti sono i numeri primi minori di un dato numero. L'informatica poteva risolvere i nostri problemi: abbiamo quindi implementato dei programmi in Free Pascal Lazarus, lavorando con variabili di tipo integer e quindi con numeri interi minori di 32000.
Di seguito viene riportato il listato del programma con il relativo diagramma di flusso che determina quanti sono i numeri primi minori di un dato numero k. La variabile contatore al termine dell'esecuzione del programma assume il valore cercato.

<u>Listato programma CONTAPRIMI</u>

```
program contaprimi;
(*questo programma determina i numeri primi
minori di k*)
uses crt;
var n, resto, i, k, b, contatore: integer;
risposta:char;
begin
risposta:='y';
while risposta='y' do begin contatore:=0;
clrscr;
writeln('il programma determina i numeri primi
minori di k');
writeln('inserisci k');
readln(k);
writeln('i numeri primi minori di k sono');
n:=k+1;
repeat
n:=n-1;
i:=2;
    repeat
    resto:=n mod i;
    i:=i+1;
    until resto=0;
if i=n+1 then begin
contatore:=contatore+1;
write(' ',n,' ');
end;
until n<2;
writeln('e sono ',contatore);
writeln('inserisci y se vuoi continuare altrimenti
premi un''altra lettera');
readln(risposta);
end;
end.
```



Questo programma risponde al quesito 2 del problema, ma non ci siamo accontentati di lasciar calcolare questo valore ad una macchina.
Abbiamo iniziato a studiare come sono disposti i numeri primi all’interno della successione dei numeri naturali. Abbiamo inserito i primi 100 numeri naturali all’interno di tabelle che avevano diversi numeri di colonne e abbiamo osservato che i numeri primi venivano a solo in alcune colonne speciali, ad eccezione della prima riga di ogni tabella.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36
37	38	39	40	41	42
43	44	45	46	47	48
49	50	51	52	53	54
55	56	57	58	59	60
61	62	63	64	65	66
67	68	69	70	71	72
73	74	75	76	77	78
79	80	81	82	83	84
85	86	87	88	89	90
91	92	93	94	95	96
97	98	99	100		

1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96
97	98	99	100								

Nelle tabelle i numeri primi sono quelli evidenziati in giallo.
Abbiamo osservato che i numeri primi si trovano soltanto all’interno delle colonne che hanno un indice coprimo con il numero di colonne totali della tabella. Infatti, se consideriamo la tabella che ha 10 colonne, i numeri primi si trovano (sempre ad eccezione della prima riga) solo nelle colonne dei numeri coprimi con 10, cioè nella colonna di indice 1, 3, 7, 9. Per la tabella con 6 colonne, i numeri primi si trovano (escludendo ancora una volta la prima riga) solo nelle colonne dei numeri coprimi con 6, cioè in quelle di indice 1 e 5.

Abbiamo lavorato quindi sulla dimostrazione della seguente affermazione:
Si scrivano i numeri naturali in una tabella di **k** *colonne.* ***I numeri primi, tranne quelli della prima riga, si trovano solo nelle colonne il cui indice x è coprimo con k.***
Per dimostrare l’affermazione dimostreremo che in una cella qualsiasi della colonna **i,** dove **i** non è coprimo con **k**, si trova un numero non primo.

Dimostrazione:
Nella tabella, consideriamo un numero **n** generico che si trovi nella casella di colonna **i** e di riga **j+1**.

	1	2	...	i-1	**i**	i+1	...	k
1								
2								
...								
j								
j+1					**n**			
j+2								
...								

Si ha **n=j*k+i,** dove **k** è il numero di colonne della tabella.
Allora consideriamo il **M.C.D(k;i)** e chiamiamo questo numero **m**. Se **k** e **i** non sono coprimi **m≠1**.
Dunque possiamo scrivere:

k= m*a con **a€N***
i=m*b con **b€N*** **(m≠1)**

Così l'elemento generico di posto **(j+1,i)** si può esprimere come

n=j*m*a+m*b

Raccogliendo **m** troviamo

n=m*(j*a+b)

Quindi **n** risulta multiplo di due numeri naturali **diversi da 1**, e quindi non è primo.

Abbiamo notato che se si scrivono i numeri naturali minori di n in una tabella di k colonne, i numeri primi non si trovano mai nelle colonne di indice non coprimo con k. Ma non tutte le celle delle colonne dei numeri coprimi con k contengono numeri primi.
Siano f(n) la funzione che indica i numeri primi minori di n e g(n) la funzione che conta il numero massimo di celle in cui si possono trovare i numeri primi, allora f(n) ≤ g(n). Inoltre

g(n) = numero di colonne di indice coprimo con k * (numero di righe della tabella -1)
+ numero di numeri primi della prima riga

Indicheremo con coprimi(k) il numero di colonne di indice coprimo con k e con p il numero di numeri primi che si trovano nella prima riga della tabella.
Poiché n/k è il numero di righe della tabella (arrotondato per eccesso se n non è multiplo di k)

g(n)= coprimi(k) * n/k + p

e quindi

f(n) ≤ coprimi(k)* n/k + p

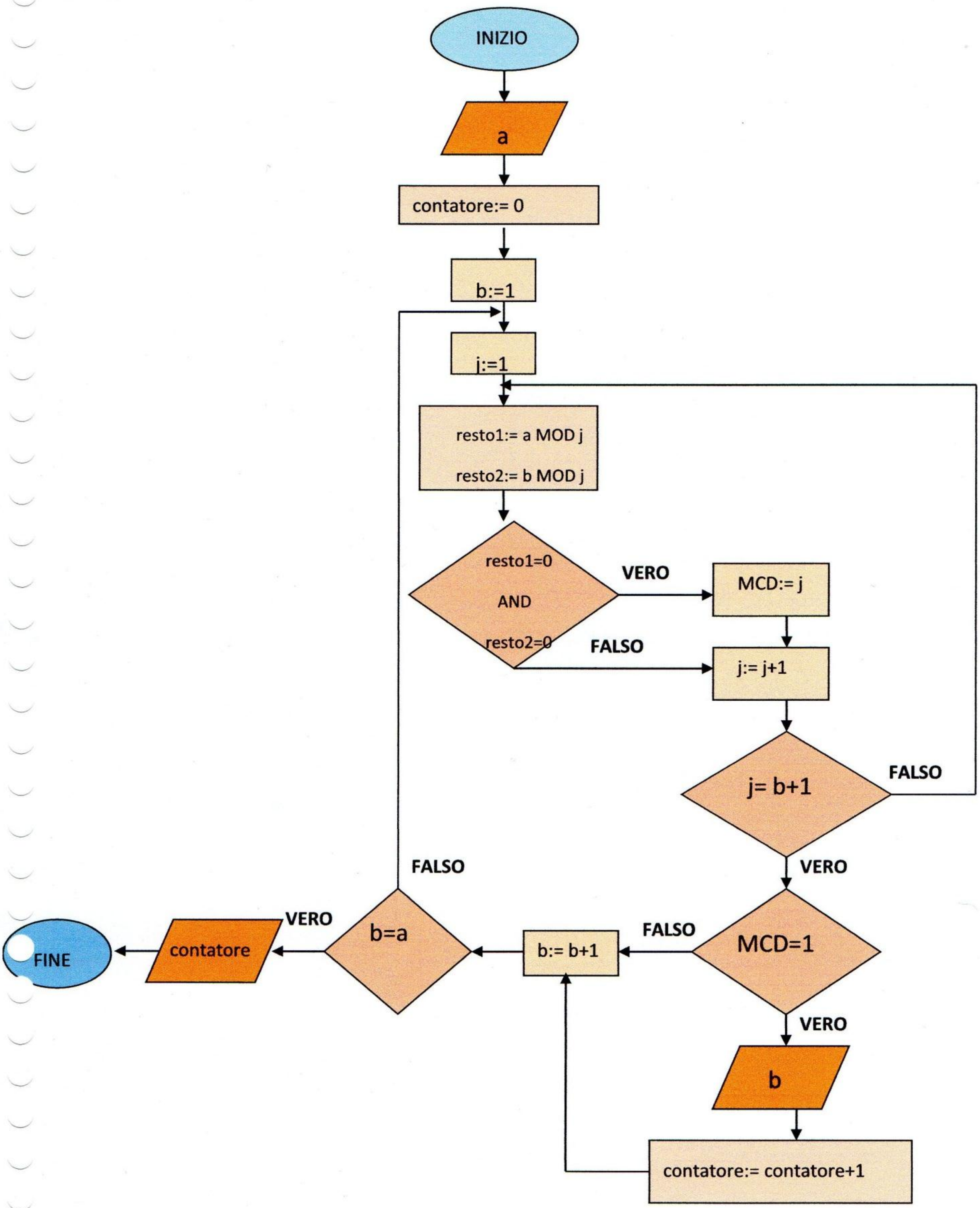
Abbiamo lavorato ancora con Free Pascal Lazarus per contare quanti sono i numeri coprimi con un numero qualunque a e per quale a (per a da 2 a 40) è minimo il rapporto tra il numero dei numeri coprimi con a minori di a e a stesso.

Di seguito sono riportati i listati dei programmi e i relativi diagrammi di flusso.

Listato programma COPRIMI

```
program coprimi3;
var a, b, j, MCD, resto1, resto2, contatore: integer;
begin
contatore:=0;
writeln(‘ il programma determina e conta i numeri minori di a che sono coprimi con a ‘);
writeln(‘ inserisci a ‘);
readln( a);
b:=1;
writeln(‘i numeri coprimi con ‘,a, ‘ sono ‘);
repeat
j:=1;
repeat
resto1:=a mod j;
resto2:=b mod j;
if (resto1=0) AND (resto2=0) then MCD:=j;
j:=j+1;
until j=b+1;
if MCD=1 then begin
 write ( b:5,’  ‘);
 contatore:=contatore+1; end;
b:=b+1;
until b=a;
writeln;
writeln (‘Quindi i numeri primi con ‘,a,’ sono ‘,contatore);
readln;
end.
```

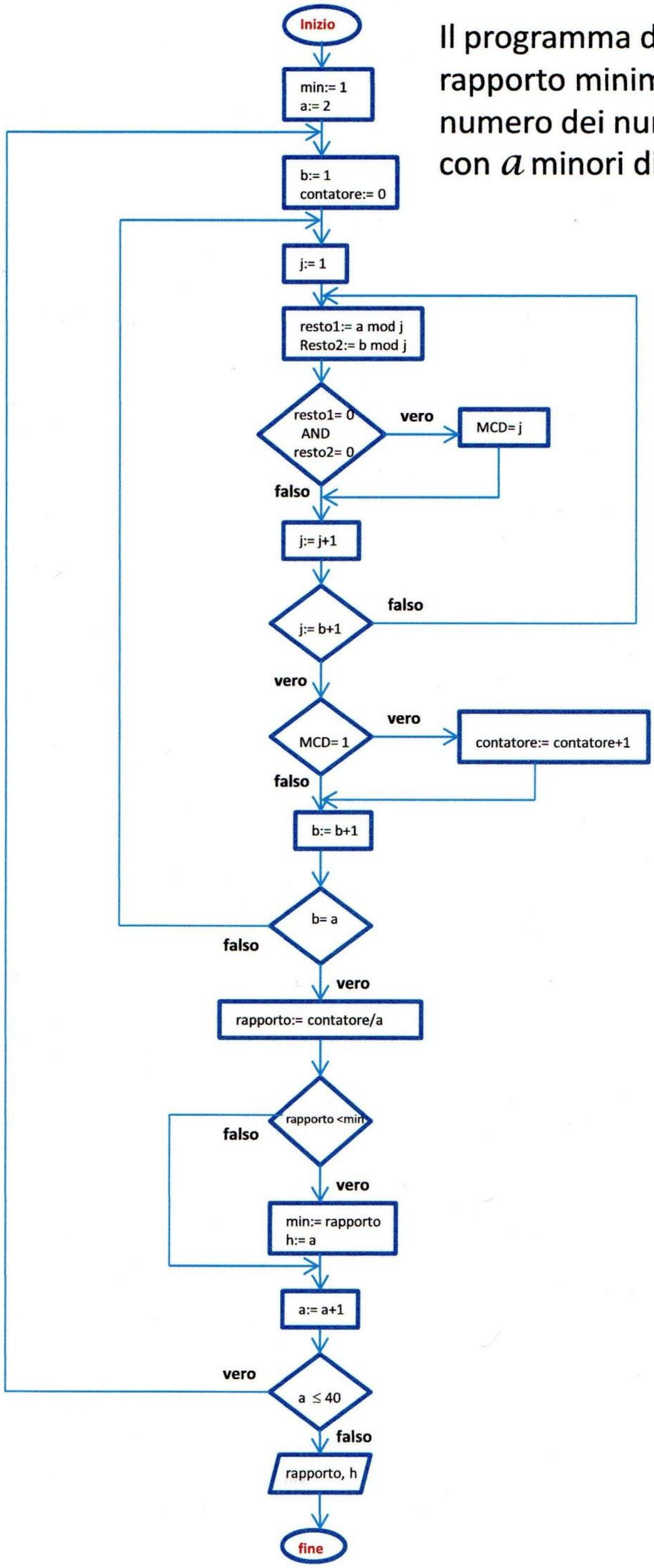
Il programma determina e conta i numeri minori di a che sono coprimi con a



Listato programma MIGLIORRAPPORTO

```
program migliorrapporto;
var a, b, j, MCD, resto1, resto2, contatore, h: integer;
rapporto, min:real;
begin
writeln(‘ il programma determina il minimo tra i rapporto tra i numeri coprimi con a e a’);
writeln(‘a e’’ un numero da 2 a 40’);
min:=1;
for a:=2 to 40 do begin
b:=1;
contatore:=0;
repeat
j:=1;
repeat
resto1:=a mod j;
resto2:=b mod j;
if (resto1=0) AND (resto2=0) then MCD:=j;
j:=j+1;
until j=b+1;
if MCD=1 then begin
 contatore:=contatore+1; end;
b:=b+1;
until b=a;
rapporto:=contatore/a;
writeln (‘il rapporto tra i numeri coprimi con ‘,a,’ e ‘,a, ‘ e’’ ‘,rapporto:20:2);
if rapporto<min then begin min:=rapporto; h:=a; end;
end;
writeln (‘il rapporto minore e’’ ‘,min:5:2, ‘ ottenuto con  ‘ ,h);
readln;
end.
```

Il programma determina il rapporto minimo tra il numero dei numeri coprimi con a minori di a e a stesso.



Si trova che il rapporto minimo è dato da a =30.

Abbiamo quindi rappresentato i grafici delle funzioni g(x) con k=30 e k=6 utilizzando un foglio di calcolo: k=30 è stato scelto perché il valore a=30 è stato determinato come miglior rapporto eseguendo il programma pascal; k=6 è stato scelto perché, benché il rapporto sia maggiore del precedente, il numero di numeri coprimi con 6 minori di 6 è minore del numero di numeri coprimi con 30 e minori di 30 (cioè quelli che si trovano nella prima riga delle tabelle).
Abbiamo quindi confrontato questi grafici con quello di f(x), ottenuto grazie ai valori dati come output dal seguente programma.

<u>Listato FUNZIONECONTAPRIMI</u>

```
 program funzionecontaprimi;
uses crt;
var n, resto, i, k,m, contatore: integer;
  var Myfile:TEXT;
begin
clrscr;
Assign(Myfile,'c:\tmp\nomefile.csv');
Rewrite(Myfile);
writeln ('il programma calcola ')  ;
writeln('quanti sono i numeri primi per ogni numero minore di m');
writeln ('inserisci m');
readln (m);
for k:=m downto 2 do begin
contatore:=0;
n:=k;
repeat
n:=n-1;
i:=2;
     repeat
     resto:=n mod i;
     i:=i+1;
     until resto=0;
if i=n+1 then contatore:=contatore+1;
   until n<2;
writeln ('i numeri primi minori di ',k,' sono ',contatore);
Writeln(Myfile, k, ';',contatore);
end;
readln;
Close(Myfile);
end.
```

Ecco i grafici delle funzioni.

	k=30	k=6
coprimi	8	2
p	10	3

$f(n)$= numero di numeri primi minori di n

$g_k(n)$= coprimi(k)*n/k+p

n	f(n)	$g_6(n)$	$g_{30}(n)$
2	0	3	10
3	1	4	10
4	2	4	11
5	2	4	11
6	3	5	11
7	3	5	11
8	4	5	12
9	4	6	12
10	4	6	12
11	4	6	12
12	5	7	13
13	5	7	13
14	6	7	13
15	6	8	14
16	6	8	14
17	6	8	14
18	7	9	14
19	7	9	15
20	8	9	15
21	8	10	15
22	8	10	15
23	8	10	16
24	9	11	16
25	9	11	16
26	9	11	16
27	9	12	17
28	9	12	17
29	9	12	17
30	10	13	18
31	10	13	18
32	11	13	18
33	11	14	18
34	11	14	19
35	11	14	19
36	11	15	19
37	11	15	19
38	12	15	20
39	12	16	20
40	12	16	20
41	12	16	20
42	13	17	21
43	13	17	21
44	14	17	21
45	14	18	22
46	14	18	22
47	14	18	22
48	15	19	22
49	15	19	23
50	15	19	23
51	15	20	23
52	15	20	23
53	15	20	24
54	16	21	24
55	16	21	24
56	16	21	24
57	16	22	25
58	16	22	25
59	16	22	25
60	17	23	26
61	17	23	26
62	18	23	26
63	18	24	26
64	18	24	27
65	18	24	27
66	18	25	27
67	18	25	27
68	19	25	28
69	19	26	28
70	19	26	28
71	19	26	28
72	20	27	29

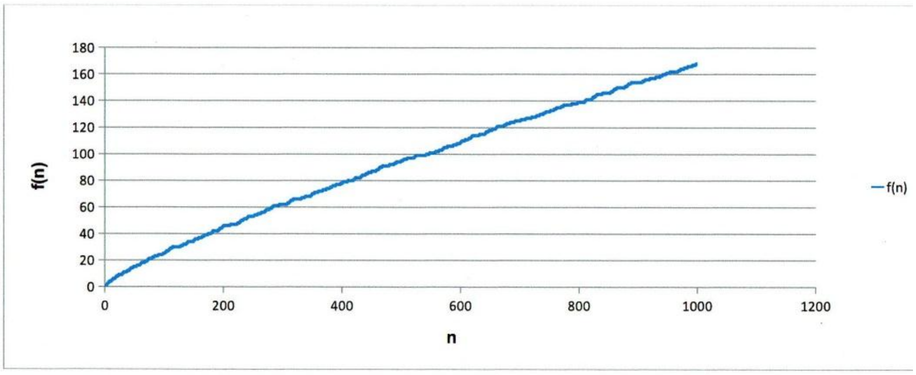
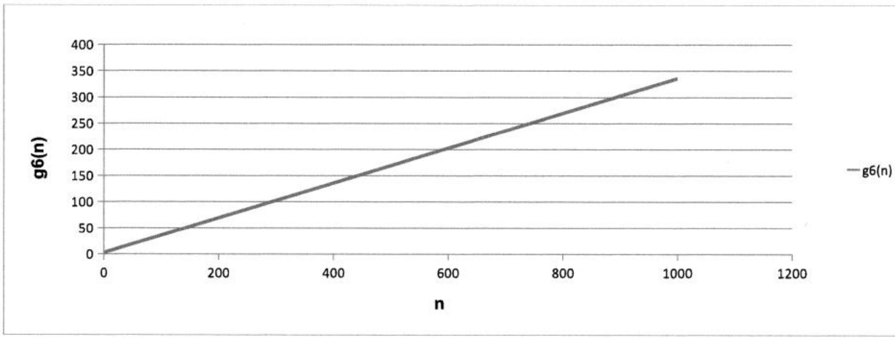
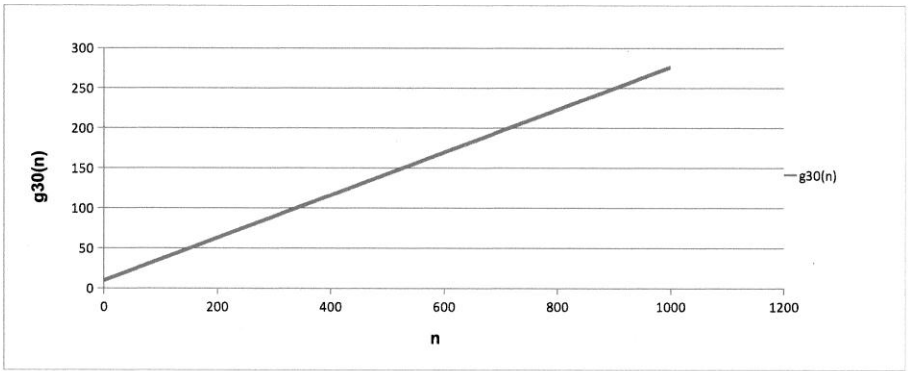


Grafico della funzione f(n). La funzione non è una retta.
Si hanno infatti intervalli in cui è costante e intervalli in cui varia.

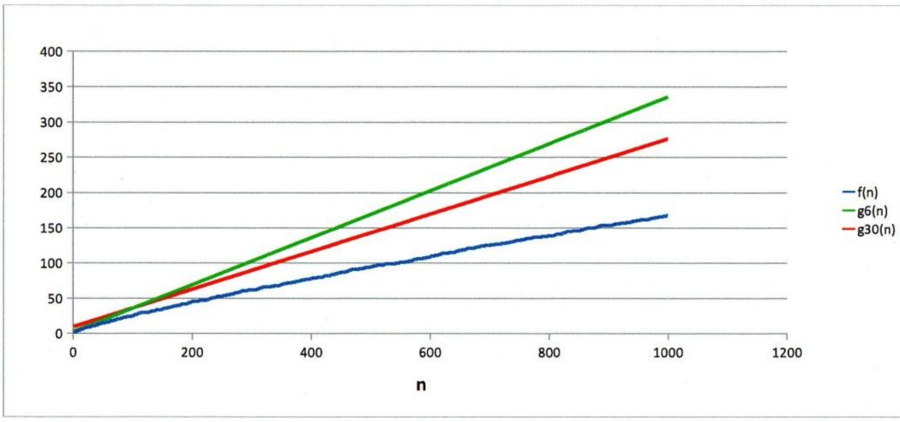


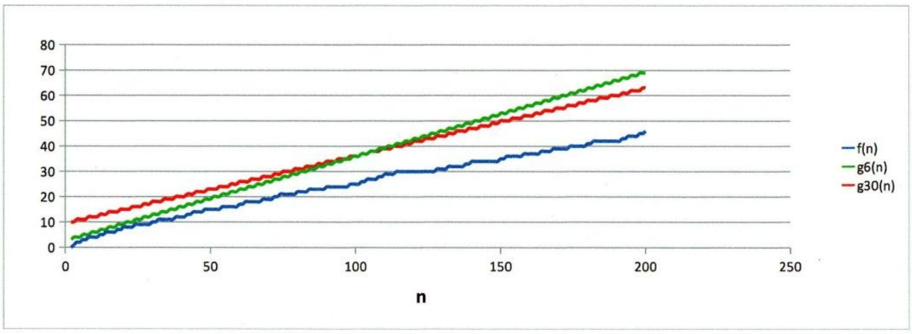
Il grafico rappresenta la funzione $g_6(n)$.



Il grafico rappresenta la funzione $g_{30}(n)$.

73	20	27	29
74	21	27	29
75	21	28	30
76	21	28	30
77	21	28	30
78	21	29	30
79	21	29	31
80	22	29	31
81	22	30	31
82	22	30	31
83	22	30	32
84	23	31	32
85	23	31	32
86	23	31	32
87	23	32	33
88	23	32	33
89	23	32	33
90	24	33	34
91	24	33	34
92	24	33	34
93	24	34	34
94	24	34	35
95	24	34	35
96	24	35	35
97	24	35	35
98	25	35	36
99	25	36	36
100	25	36	36
101	25	36	36
102	26	37	37
103	26	37	37
104	27	37	37
105	27	38	38
106	27	38	38
107	27	38	38
108	28	39	38
109	28	39	39
110	29	39	39
111	29	40	39
112	29	40	39





Per 2 < n < 105 la funzione che meglio approssima f(n) è g_6(n).
Per n > 105 la funzione che meglio approssima f(n) è g_{30}(n)

Dagli ultimi due grafici si deduce che

- per $2<n<105$ la funzione che meglio approssima f(n) è $g_{k=6}(n)$;
- per $n>105$ la funzione che meglio approssima f(n) è $g_{k=30}(n)$;
- per n=105 , $g_{K=6}(105)=g_{K=30}(105)$.

Conclusione

Siamo partiti dal problema dell'albergatore, abbiamo scoperto che **gli alberghi che rispettano le caratteristiche richieste sono quelli con il numero civico primo**, rispondendo così al primo quesito.

Per rispondere al secondo quesito, cioè quanti sono gli alberghi che hanno la caratteristica richiesta fino al civico n, abbiamo tradotto la domanda in "**quanti sono i numeri primi minori di un dato numero n?**". Quello che possiamo dire è che non abbiamo ancora trovato l'espressione matematica della funzione f(n)!

Noi abbiamo trovato che

$$f(n) < \frac{2}{6}n + 3 \qquad 2 < n < 105$$

$$f(n) < \frac{8}{30}n + 10 \qquad n \geq 105$$

Ringraziamenti:

Ringraziamo la dott. Carniani che ci ha proposto questo interessante problema, che ci ha seguito costantemente e che ci ha sempre elogiato anche per i nostri seppur piccoli successi.