

```

1 local help= []
2 NB:
3 (c)2022 Tim Menzies, timm@ieee.org
4
5 OPTIONS:
6 -b --Bins      max number of bins      = 16
7 -k --k         handle rare classes     = 1
8 -m --m         handle rare attributes  = 2
9 -p --p         distance coefficient     = 2
10 -S --small     small leaf size         = .5
11 -w --wait      wait before classifying = 5
12
13 OPTIONS (other):
14 -f --file      file                    = .././data/auto93.csv
15 -g --go        start-up goal           = nothing
16 -h --help      show help               = false
17 -s --seed      seed                    = 10019]
18
19 == 1 1 1 1 1 1 1 1 ==
20
21 local _ = require "lib"
22 local argmax,big      = _:argmax, _:big
23 local cli, csv, demos, is, normpdf = _:cli, _:csv, _:demos, _:is, _:normpdf
24 local oo, push, read, rnd, same, str = _:oo, _:push, _:read, _:rnd, _:same, _:str
25
26 local THE={}
27 help:gsuub('[-|-|](["%s"]|["n"]%s(["%s"]+),function(key,x) THE[key] = read(x) end)
28
29 local NB, NUM, SYM, COLS, ROW, ROWS= is"NB", is"NUM", is"SYM", is"COLS", is"ROW", is"ROWS"
30 local FEW, RANGE, TREE = is"FEW", is"RANGE", is"TREE"
31
32 == 1 1 1 1 1 1 1 1 ==
33
34 function RANGE.new(i, xlo, xhi, ys) i.xlo, i.xhi, i.ys = xlo, xhi, ys end
35 function RANGE.add(i, x, y)
36 if x < i.xlo then i.xlo = x end -- works for string or num
37 if x > i.xhi then i.xhi = x end -- works for string or num
38 i.y:addd(y) end
39
40 function RANGE._:tostring(i)
41 local x, lo, hi = i.ys.txt, i.xlo, i.xhi
42 if lo == hi then return fmt("%s==%s", x, lo)
43 elseif hi == big then return fmt("%s>%s", x, lo)
44 elseif lo == -big then return fmt("%s<%s", x, hi)
45 else return fmt("%s<%s<%s", lo, x, hi) end end
46
47 == 1 1 1 1 1 1 1 1 ==
48
49 function FEW.new(i) i.n, i.t, i.ok=0, {}, true end
50 function FEW.has(i) i.t=i.ok and i.t or sort(i.t); i.ok=true; return i.t end
51 function FEW.add(i, x)
52 if x=="?" then return x end
53 i.n=i.n+1
54 if #i.t < THE.some then i.ok=false; push(i.t, x)
55 elseif rand() < THE.some/i.n then i.ok=false; i.t[rand(#i.t)]=x end end
56
57 function NUM.new(i) i.n, i.mu, i.m2, i.mu, i.lo, i.hi, i.few=0, 0, 0, 0, big, FEW(i) end
58 function NUM.mid(i, p) return rnd(i.mu, p) end
59 function NUM.like(i, x, ...) return normpdf(x, i.mu, i.sd) end
60 function NUM.bin(x)
61 b=(i.hi - i.lo)/THE.bins; return i.lo==i.hi and 1 or math.floor(x/b+.5)*b end
62
63 function NUM.add(i, NUM, v, number)
64 if v=="?" then return v end
65 i.few:addd(v)
66 i.n = i.n + 1
67 local d = v - i.mu
68 i.mu = i.mu + d/i.n
69 i.m2 = i.m2 + d*(v - i.mu)
70 i.sd = i.n<2 and 0 or (i.m2/(i.n-1))^0.5
71 i.lo = math.min(v, i.lo)
72 i.hi = math.max(v, i.hi) end
73
74 function NUM.merge(i, j, k)
75 local k = NUM(i.at, i.txt)
76 for _, n in pairs(i.few.t) do k:addd(x) end
77 for _, n in pairs(j.few.t) do k:addd(x) end
78 return k end
79
80 function NUM.mergeRanges(i, b4, min)
81 local t, j, a, b, c, A, B, C = {}, 1
82 while j < #b4 do
83 a, b = b4[j], b4[j+1]
84 if b then
85 A, B = a.ys, b.ys
86 C = A:merge(B)
87 if A.n<min or B.n<min or C:div() <= (A.n*A:div() + B.n*B:div())/C.n then
88 j = j + 1
89 a = RANGE(a.xlo, b.xhi, C) end end
90 t[#t+1] = a
91 j = j + 1 end
92 if #t < #b4 then return i:mergeRanges(t, min) end
93 for j=2, #t do t[j].xlo = t[j-1].xhi end
94 t[1].xlo, t[#t].xhi = -big, big
95 return t end
96
97 function SYM.new(i) i.n, i.syms, i.most, i.mode = 0, {}, 0, nil end
98 function SYM.mid(i, ...) return i.mode end
99 function SYM.like(i, x, prior) return ((i.syms[x] or 0) + THE.m*prior)/(i.n+THE.m) end
100 function SYM.bin(x) return x end
101 function SYM.add(i, v, inc)
102 if v=="?" then return v end
103 inc=inc or 1
104 i.n = i.n + inc
105 i.syms[v] = inc + (i.syms[v] or 0)
106 if i.syms[v] > i.most then i.most, i.mode = i.syms[v], v end end
107
108 function SYM.merge(i, j, k)
109 local k = SYM(i.at, i.txt)
110 for x, n in pairs(i.has) do k:addd(x, n) end
111 for x, n in pairs(j.has) do k:addd(x, n) end
112 return k end
113
114 function SYM.mergeRanges(i, t, ...) return t end

```

```

115 == 1 1 1 1 1 1 1 1 ==
116
117 local function usep(x) return not x:find"%$" end
118 local function nump(x) return x:find"%[A-Z]" end
119 local function goalp(x) return x:find"%[!-~$]" end
120 local function klassp(x) return x:find"%[S]" end
121 local function new(at, txt, i)
122 txt = txt or ""
123 i = (nump(txt) and NUM or SYM)()
124 i.txt, i.usep, i.at, i.w = txt, usep(txt), at or 0, txt:find"%$" and -1 or 1
125 return i end
126
127 function COLS.new(i, t, col)
128 i.all, i.xs, i.ys, i.names = {}, {}, {}, t
129 for at, x in pairs(t) do
130 col = push(i.all, new(at, x))
131 if col.usep then
132 if klassp(col.txt) then i.klass=col end
133 push(goalp(col.txt) and i.ys or i.xs, col) end end end
134
135 function COLS.add(i, t)
136 for _, cols in pairs(i.xs, i.ys) do
137 for _, col in pairs(cols) do col:addd(t[col.at]) end end
138 return t end
139
140 == 1 1 1 1 1 1 1 1 ==
141
142 function ROW.new(i, of, cells) i.of, i.cells, i.eval=of, cells, false end
143 function ROW.klass(i) return i.cells[i.of.cols.klass.at] end
144 function ROW.within(i, range, lo, hi, at, v)
145 lo, hi, at = range.xlo, range.xhi, range.ys.at
146 v = i.i:lll(i)
147 return v=="?" or (lo==hi and v==lo) or (lo<v and v<=hi) end
148
149 function ROW.b4(i, j, at, x, y)
150 x, y = i.cells[at], j.cells[at]
151 x = x=="?" and -big or x
152 y = y=="?" and -big or y
153 return x < y end
154
155 == 1 1 1 1 1 1 1 1 ==
156
157 local function load(src, fun)
158 if type(src)~="string" then for _, t in pairs(src) do fun(t) end
159 else for t in csv(src) do fun(t) end end end
160
161 function ROWS.new(i, t) i.cols=COLS(t); i.rows={} end
162 function ROWS.add(i, t)
163 t=t.cells and t or ROW(i, t)
164 i.cols:addd(t.cells)
165 return push(i.rows, t) end
166
167 function ROWS.mid(i, cols, p, t)
168 t={}; for _, col in pairs(cols or i.cols.ys) do t[col.txt]=col:mid(p) end; return t end
169
170 function NB._:known(i, k)
171 j= ROWS(i.cols.names); for _, row in pairs(t or {}) do j:addd(row) end; return j end
172
173 function ROWS.like(i, t, nklasses, nrows, prior, like, inc, x)
174 prior = (#i.rows + THE.k) / (nrows + THE.k * nklasses)
175 like = math.log(prior)
176 for _, col in pairs(i.cols.xs) do
177 x = t.cells[col.at]
178 if x and x ~= "?" then
179 inc = col:like(x, prior)
180 like = like + math.log(inc) end end
181 return like end
182
183 == 1 1 1 1 1 1 1 1 ==
184
185 -- (0) Use row1 to initial our 'overall' knowledge of all rows.
186 -- After that (1) add row to 'overall' and (2) ROWS about this row's klass.
187 -- (3) After 'wait' row, classify row BEFORE updating training knowledge
188 function NB.new(i, src, report, row)
189 report = report or print
190 i.overall.i.dict, i.list = nil, {}, {}
191 load(src, function(row, k)
192 if not i.overall then i.overall = ROWS(row) else -- (0) eat row1
193 row = i.overall:addd(row) -- add to overall
194 if #i.overall.rows > THE.wait then report(row:klass(), i:guess(row)) end
195 i:train(row) end end) add tp row's klass
196
197 function NB:train(i, row) i:known(row:klass()):addd(row) end
198 function NB._:known(i, k)
199 i.dict[k] = i.dict[k] or push(i.list, i.overall:clone()) -- klass is known
200 i.dict[k].txt = k -- each klass knows its name
201 return i.dict[k] end
202
203 function NB.guess(i, row)
204 return argmax(i.dict,
205 function(klass) return klass:like(row, #i.list, #i.overall.rows) end) end
206
207 function TREE.new(i, listOfRows, gaurd)
208 -- i.gaurd, i.kids = gaurd, {}
209 -- of = listOfRows[1][1].of
210 -- best = sort(map(of.cols.x,
211 -- function(col) i:bins(col, listOfRows) end), lt"div")[1]
212 -- i.kids = map(best.ranges, function(range)
213 -- listOfRowsl = {}
214 -- local function within(row) return row:within(best) end
215 -- local function withins(rows) return map(rows, within) end
216 -- map(listOfRanges, function(rows) return withins(rows) end) end
217 -- tmp=map(rows, within)
218 -- if #tmp > stop then
219 -- end)
220
221 function TREE.bins(i, xcol, yklass, y, rows)
222 local n, list, dict = 0, {}, {}
223 for _, row in pairs(rows) do
224 local v = row.cells[xcol.at]
225 if v ~= "?" then
226 n = n + 1
227 local pos = xcol:bin(v)
228 dict[pos] = dict[pos] or push(list, RANGE(v, y, yklass(xcol.at, xcol.txt)))
229 dict[pos]:addd(v, y(row)) end end
230 list = xcol:mergeRanges(sort(list, lt"xlo"), n^THE.min)
231 return {ranges=list,
232 div = sum(list, function(z) return z.ys:div() * z.ys.n/n end) end} end

```

```

234 == 1 1 1 1 1 1 1 1 ==
235
236 local no, go = {}, {}
237 function go.the() print(1); print(THE); return type(THE.p)=="number" and THE.p==2 end
238
239 function go.argmax(t, fun)
240 fun=function(x) return -x end
241 t={50,40,0,40,50}
242 return 3 == argmax(t, fun) end
243
244 function go.num(n) n=NUM(); for x=1,100 do n:addd(x) end; return n.mu==50.5 end
245
246 function go.sym(s)
247 s=SYM(); for
248 return s.mode=="a" end
249
250 function go.csv(n, s)
251 n,s=0,0; for row in csv(THE.file) do n=n+1; if n>1 then s=s+row[1] end end
252 return rnd(s/n, 3) == 5.441 end
253
254 function go.rows(rows)
255 load(THE.file, function(t) if rows then rows:addd(t) else rows=ROWS(t) end end)
256 return rnd(rows.cols.ys[1].sd, 0) == 847 end
257
258 function go.nb()
259 return 268 == #NB(".././data/diabetes.csv").dict["positive"].rows end
260
261 local function _:classify(file)
262 local abcd=require"abcd"
263 local abcd=Abcd()
264 NB(file, function(got, want) abcd:addd(got, want) end)
265 abcd:pretty(abcd:report())
266 return true end
267
268 function go.soybean() return _:classify(".././data/soybean.csv") end
269 function go.diabetes() return _:classify(".././data/diabetes.csv") end
270
271 == 1 1 1 1 1 1 1 1 ==
272
273 if pcall(debug.getlocal, 4, 1)
274 then return (ROW=ROW, ROWS=ROWS, NUM=NUM, SYM=SYM, THE=THE, lib=lib)
275 else THE = cli(THE, help)
276 demos(THE, go) end
277
278 == 1 1 1 1 1 1 1 1 ==
279
280 -- [ ]
281 -- [ ]
282 -- [ ]
283 -- [ ]
284 -- [ ]
285 -- [ ]
286 -- [ ]
287 -- [ ]
288 -- [ ]
289 -- [ ]
290 -- [ ]
291 -- [ ]
292 -- [ ]
293 -- [ ]
294 -- [ ]
295 -- [ ]
296 -- [ ]
297 -- [ ]
298 -- [ ]
299 -- [ ]
300 -- [ ]
301 -- [ ]
302 -- [ ]
303 -- [ ]
304 -- [ ]
305 -- [ ]
306 -- [ ]
307 -- [ ]
308 -- [ ]
309 -- [ ]
310 -- [ ]
311 -- [ ]
312 -- [ ]
313 -- [ ]
314 -- [ ]
315 -- [ ]
316 -- [ ]
317 -- [ ]
318 -- [ ]
319 -- [ ]
320 -- [ ]
321 -- [ ]
322 -- [ ]
323 -- [ ]
324 -- [ ]
325 -- [ ]
326 -- [ ]
327 -- [ ]
328 -- [ ]
329 -- [ ]
330 -- [ ]
331 -- [ ]
332 -- [ ]
333 -- [ ]
334 -- [ ]
335 -- [ ]
336 -- [ ]
337 -- [ ]
338 -- [ ]
339 -- [ ]
340 -- [ ]
341 -- [ ]
342 -- [ ]
343 -- [ ]
344 -- [ ]
345 -- [ ]
346 -- [ ]
347 -- [ ]
348 -- [ ]
349 -- [ ]
350 -- [ ]
351 -- [ ]
352 -- [ ]
353 -- [ ]
354 -- [ ]
355 -- [ ]
356 -- [ ]
357 -- [ ]
358 -- [ ]
359 -- [ ]
360 -- [ ]
361 -- [ ]
362 -- [ ]
363 -- [ ]
364 -- [ ]
365 -- [ ]
366 -- [ ]
367 -- [ ]
368 -- [ ]
369 -- [ ]
370 -- [ ]
371 -- [ ]
372 -- [ ]
373 -- [ ]
374 -- [ ]
375 -- [ ]
376 -- [ ]
377 -- [ ]
378 -- [ ]
379 -- [ ]
380 -- [ ]
381 -- [ ]
382 -- [ ]
383 -- [ ]
384 -- [ ]
385 -- [ ]
386 -- [ ]
387 -- [ ]
388 -- [ ]
389 -- [ ]
390 -- [ ]
391 -- [ ]
392 -- [ ]
393 -- [ ]
394 -- [ ]
395 -- [ ]
396 -- [ ]
397 -- [ ]
398 -- [ ]
399 -- [ ]
400 -- [ ]
401 -- [ ]
402 -- [ ]
403 -- [ ]
404 -- [ ]
405 -- [ ]
406 -- [ ]
407 -- [ ]
408 -- [ ]
409 -- [ ]
410 -- [ ]
411 -- [ ]
412 -- [ ]
413 -- [ ]
414 -- [ ]
415 -- [ ]
416 -- [ ]
417 -- [ ]
418 -- [ ]
419 -- [ ]
420 -- [ ]
421 -- [ ]
422 -- [ ]
423 -- [ ]
424 -- [ ]
425 -- [ ]
426 -- [ ]
427 -- [ ]
428 -- [ ]
429 -- [ ]
430 -- [ ]
431 -- [ ]
432 -- [ ]
433 -- [ ]
434 -- [ ]
435 -- [ ]
436 -- [ ]
437 -- [ ]
438 -- [ ]
439 -- [ ]
440 -- [ ]
441 -- [ ]
442 -- [ ]
443 -- [ ]
444 -- [ ]
445 -- [ ]
446 -- [ ]
447 -- [ ]
448 -- [ ]
449 -- [ ]
450 -- [ ]
451 -- [ ]
452 -- [ ]
453 -- [ ]
454 -- [ ]
455 -- [ ]
456 -- [ ]
457 -- [ ]
458 -- [ ]
459 -- [ ]
460 -- [ ]
461 -- [ ]
462 -- [ ]
463 -- [ ]
464 -- [ ]
465 -- [ ]
466 -- [ ]
467 -- [ ]
468 -- [ ]
469 -- [ ]
470 -- [ ]
471 -- [ ]
472 -- [ ]
473 -- [ ]
474 -- [ ]
475 -- [ ]
476 -- [ ]
477 -- [ ]
478 -- [ ]
479 -- [ ]
480 -- [ ]
481 -- [ ]
482 -- [ ]
483 -- [ ]
484 -- [ ]
485 -- [ ]
486 -- [ ]
487 -- [ ]
488 -- [ ]
489 -- [ ]
490 -- [ ]
491 -- [ ]
492 -- [ ]
493 -- [ ]
494 -- [ ]
495 -- [ ]
496 -- [ ]
497 -- [ ]
498 -- [ ]
499 -- [ ]
500 -- [ ]
501 -- [ ]
502 -- [ ]
503 -- [ ]
504 -- [ ]
505 -- [ ]
506 -- [ ]
507 -- [ ]
508 -- [ ]
509 -- [ ]
510 -- [ ]
511 -- [ ]
512 -- [ ]
513 -- [ ]
514 -- [ ]
515 -- [ ]
516 -- [ ]
517 -- [ ]
518 -- [ ]
519 -- [ ]
520 -- [ ]
521 -- [ ]
522 -- [ ]
523 -- [ ]
524 -- [ ]
525 -- [ ]
526 -- [ ]
527 -- [ ]
528 -- [ ]
529 -- [ ]
530 -- [ ]
531 -- [ ]
532 -- [ ]
533 -- [ ]
534 -- [ ]
535 -- [ ]
536 -- [ ]
537 -- [ ]
538 -- [ ]
539 -- [ ]
540 -- [ ]
541 -- [ ]
542 -- [ ]
543 -- [ ]
544 -- [ ]
545 -- [ ]
546 -- [ ]
547 -- [ ]
548 -- [ ]
549 -- [ ]
550 -- [ ]
551 -- [ ]
552 -- [ ]
553 -- [ ]
554 -- [ ]
555 -- [ ]
556 -- [ ]
557 -- [ ]
558 -- [ ]
559 -- [ ]
560 -- [ ]
561 -- [ ]
562 -- [ ]
563 -- [ ]
564 -- [ ]
565 -- [ ]
566 -- [ ]
567 -- [ ]
568 -- [ ]
569 -- [ ]
570 -- [ ]
571 -- [ ]
572 -- [ ]
573 -- [ ]
574 -- [ ]
575 -- [ ]
576 -- [ ]
577 -- [ ]
578 -- [ ]
579 -- [ ]
580 -- [ ]
581 -- [ ]
582 -- [ ]
583 -- [ ]
584 -- [ ]
585 -- [ ]
586 -- [ ]
587 -- [ ]
588 -- [ ]
589 -- [ ]
590 -- [ ]
591 -- [ ]
592 -- [ ]
593 -- [ ]
594 -- [ ]
595 -- [ ]
596 -- [ ]
597 -- [ ]
598 -- [ ]
599 -- [ ]
600 -- [ ]
601 -- [ ]
602 -- [ ]
603 -- [ ]
604 -- [ ]
605 -- [ ]
606 -- [ ]
607 -- [ ]
608 -- [ ]
609 -- [ ]
610 -- [ ]
611 -- [ ]
612 -- [ ]
613 -- [ ]
614 -- [ ]
615 -- [ ]
616 -- [ ]
617 -- [ ]
618 -- [ ]
619 -- [ ]
620 -- [ ]
621 -- [ ]
622 -- [ ]
623 -- [ ]
624 -- [ ]
625 -- [ ]
626 -- [ ]
627 -- [ ]
628 -- [ ]
629 -- [ ]
630 -- [ ]
631 -- [ ]
632 -- [ ]
633 -- [ ]
634 -- [ ]
635 -- [ ]
636 -- [ ]
637 -- [ ]
638 -- [ ]
639 -- [ ]
640 -- [ ]
641 -- [ ]
642 -- [ ]
643 -- [ ]
644 -- [ ]
645 -- [ ]
646 -- [ ]
647 -- [ ]
648 -- [ ]
649 -- [ ]
650 -- [ ]
651 -- [ ]
652 -- [ ]
653 -- [ ]
654 -- [ ]
655 -- [ ]
656 -- [ ]
657 -- [ ]
658 -- [ ]
659 -- [ ]
660 -- [ ]
661 -- [ ]
662 -- [ ]
663 -- [ ]
664 -- [ ]
665 -- [ ]
666 -- [ ]
667 -- [ ]
668 -- [ ]
669 -- [ ]
670 -- [ ]
671 -- [ ]
672 -- [ ]
673 -- [ ]
674 -- [ ]
675 -- [ ]
676 -- [ ]
677 -- [ ]
678 -- [ ]
679 -- [ ]
680 -- [ ]
681 -- [ ]
682 -- [ ]
683 -- [ ]
684 -- [ ]
685 -- [ ]
686 -- [ ]
687 -- [ ]
688 -- [ ]
689 -- [ ]
690 -- [ ]
691 -- [ ]
692 -- [ ]
693 -- [ ]
694 -- [ ]
695 -- [ ]
696 -- [ ]
697 -- [ ]
698 -- [ ]
699 -- [ ]
700 -- [ ]
701 -- [ ]
702 -- [ ]
703 -- [ ]
704 -- [ ]
705 -- [ ]
706 -- [ ]
707 -- [ ]
708 -- [ ]
709 -- [ ]
710 -- [ ]
711 -- [ ]
712 -- [ ]
713 -- [ ]
714 -- [ ]
715 -- [ ]
716 -- [ ]
717 -- [ ]
718 -- [ ]
719 -- [ ]
720 -- [ ]
721 -- [ ]
722 -- [ ]
723 -- [ ]
724 -- [ ]
725 -- [ ]
726 -- [ ]
727 -- [ ]
728 -- [ ]
729 -- [ ]
730 -- [ ]
731 -- [ ]
732 -- [ ]
733 -- [ ]
734 -- [ ]
735 -- [ ]
736 -- [ ]
737 -- [ ]
738 -- [ ]
739 -- [ ]
740 -- [ ]
741 -- [ ]
742 -- [ ]
743 -- [ ]
744 -- [ ]
745 -- [ ]
746 -- [ ]
747 -- [ ]
748 -- [ ]
749 -- [ ]
750 -- [ ]
751 -- [ ]
752 -- [ ]
753 -- [ ]
754 -- [ ]
755 -- [ ]
756 -- [ ]
757 -- [ ]
758 -- [ ]
759 -- [ ]
760 -- [ ]
761 -- [ ]
762 -- [ ]
763 -- [ ]
764 -- [ ]
765 -- [ ]
766 -- [ ]
767 -- [ ]
768 -- [ ]
769 -- [ ]
770 -- [ ]
771 -- [ ]
772 -- [ ]
773 -- [ ]
774 -- [ ]
775 -- [ ]
776 -- [ ]
777 -- [ ]
778 -- [ ]
779 -- [ ]
780 -- [ ]
781 -- [ ]
782 -- [ ]
783 -- [ ]
784 -- [ ]
785 -- [ ]
786 -- [ ]
787 -- [ ]
788 -- [ ]
789 -- [ ]
790 -- [ ]
791 -- [ ]
792 -- [ ]
793 -- [ ]
794 -- [ ]
795 -- [ ]
796 -- [ ]
797 -- [ ]
798 -- [ ]
799 -- [ ]
800 -- [ ]
801 -- [ ]
802 -- [ ]
803 -- [ ]
804 -- [ ]
805 -- [ ]
806 -- [ ]
807 -- [ ]
808 -- [ ]
809 -- [ ]
810 -- [ ]
811 -- [ ]
812 -- [ ]
813 -- [ ]
814 -- [ ]
815 -- [ ]
816 -- [ ]
817 -- [ ]
818 -- [ ]
819 -- [ ]
820 -- [ ]
821 -- [ ]
822 -- [ ]
823 -- [ ]
824 -- [ ]
825 -- [ ]
826 -- [ ]
827 -- [ ]
828 -- [ ]
829 -- [ ]
830 -- [ ]
831 -- [ ]
832 -- [ ]
833 -- [ ]
834 -- [ ]
835 -- [ ]
836 -- [ ]
837 -- [ ]
838 -- [ ]
839 -- [ ]
840 -- [ ]
841 -- [ ]
842 -- [ ]
843 -- [ ]
844 -- [ ]
845 -- [ ]
846 -- [ ]
847 -- [ ]
848 -- [ ]
849 -- [ ]
850 -- [ ]
851 -- [ ]
852 -- [ ]
853 -- [ ]
854 -- [ ]
855 -- [ ]
856 -- [ ]
857 -- [ ]
858 -- [ ]
859 -- [ ]
860 -- [ ]
861 -- [ ]
862 -- [ ]
863 -- [ ]
864 -- [ ]
865 -- [ ]
866 -- [ ]
867 -- [ ]
868 -- [ ]
869 -- [ ]
870 -- [ ]
871 -- [ ]
872 -- [ ]
873 -- [ ]
874 -- [ ]
875 -- [ ]
876 -- [ ]
877 -- [ ]
878 -- [ ]
879 -- [ ]
880 -- [ ]
881 -- [ ]
882 -- [ ]
883 -- [ ]
884 -- [ ]
885 -- [ ]
886 -- [ ]
887 -- [ ]
888 -- [ ]
889 -- [ ]
890 -- [ ]
891 -- [ ]
892 -- [ ]
893 -- [ ]
894 -- [ ]
895 -- [ ]
896 -- [ ]
897 -- [ ]
898 -- [ ]
899 -- [ ]
900 -- [ ]
901 -- [ ]
902 -- [ ]
903 -- [ ]
904 -- [ ]
905 -- [ ]
906 -- [ ]
907 -- [ ]
908 -- [ ]
909 -- [ ]
910 -- [ ]
911 -- [ ]
912 -- [ ]
913 -- [ ]
914 -- [ ]
915 -- [ ]
916 -- [ ]
917 -- [ ]
918 -- [ ]
919 -- [ ]
920 -- [ ]
921 -- [ ]
922 -- [ ]
923 -- [ ]
924 -- [ ]
925 -- [ ]
926 -- [ ]
927 -- [ ]
928 -- [ ]
929 -- [ ]
930 -- [ ]
931 -- [ ]
932 -- [ ]
933 -- [ ]
934 -- [ ]
935 -- [ ]
936 -- [ ]
937 -- [ ]
938 -- [ ]
939 -- [ ]
940 -- [ ]
941 -- [ ]
942 -- [ ]
943 -- [ ]
944 -- [ ]
945 -- [ ]
946 -- [ ]
947 -- [ ]
948 -- [ ]
949 -- [ ]
950 -- [ ]
951 -- [ ]
952 -- [ ]
953 -- [ ]
954 -- [ ]
955 -- [ ]
956 -- [ ]
957 -- [ ]
958 -- [ ]
959 -- [ ]
960 -- [ ]
961 -- [ ]
962 -- [ ]
963 -- [ ]
964 -- [ ]
965 -- [ ]
966 -- [ ]
967 -- [ ]
968 -- [ ]
969 -- [ ]
970 -- [ ]
971 -- [ ]
972 -- [ ]
973 -- [ ]
974 -- [ ]
975 -- [ ]
976 -- [ ]
977 -- [ ]
978 -- [ ]
979 -- [ ]
980 -- [ ]
981 -- [ ]
982 -- [ ]
983 -- [ ]
984 -- [ ]
985 -- [ ]
986 -- [ ]
987 -- [ ]
988 -- [ ]
989 -- [ ]
990 -- [ ]
991 -- [ ]
992 -- [ ]
993 -- [ ]
994 -- [ ]
995 -- [ ]
996 -- [ ]
997 -- [ ]
998 -- [ ]
999 -- [ ]
1000 -- [ ]

```