```
%% Genetic Algorithm Example:  Binary Strings (BinStrings.m)
%
%  Problem:  Define the population as binary strings of length 20,
%  with the fitness function:  Sum of 1's.  Use a population of
%  100 with certain probabilities defined below to see how long
%  it takes to reach a string of all 1's.

%%  Setup the GA parameters

ff=inline('sum(x,2)'); % objective function

maxit=200;        % max number of iterations (for stopping)
maxcost=9999999; %Maximum allowable cost (for stopping)
popsize=100; % set population size (it is constant)
mutrate=0.001; % set mutation rate (a small probability)

lenx=20; % Length of the chromosome (20 here)

%% Create the initial population
pop=round(rand(popsize,lenx)); % random population of 1s
                               % and 0s (using default, 100x20)

% Initialize cost and other items to set up the main loop
cost=feval(ff,pop); % calculates population cost using ff

[cost,ind]=sort(cost,'descend'); % max element in first entry
pop=pop(ind,:); % sorts population with max cost first

probs=cost/sum(cost);  %Simple normalization for probabilities.

% We'll be tracking the following quantities for our plot:
maxc(1)=max(cost); % minc contains min of population
meanc(1)=mean(cost); % meanc contains mean of population


%% MAIN LOOP

iga=0;  % Initalize the variable used in the loop below.
        %  This will keep track of how many iterations we've
        %  used and will get us out of the loop at the max

while iga<maxit

    iga=iga+1; % increments generation counter

% Choose mates

    M=ceil(popsize/2); % number of matings
    ma=RandChooseN(probs,M); % mate #1
    pa=RandChooseN(probs,M); % mate #2
% ma and pa contain the *indices* of the chromosomes that will mate

% Select crossover values for each set of parents
    xp=randi([1,lenx],1,M);  % M integers from 1 to lenx
                             % In this code, crossover is different
                             %  for each set of parents.

    Temp=pop;  %Just temporary storage as we perform crossover

%Crossover: One offspring will be stored in the odd indices,
%           the other in the even indices.
    for k=1:M
      pop(2*k-1,:)=[Temp(ma(k),1:xp(k)) Temp(pa(k),xp(k)+1:20)];
      pop(2*k,:)=[Temp(pa(k),1:xp(k)) Temp(ma(k),xp(k)+1:20)];
    end

% Mutate the population
```

```
68      nmut=ceil((popsize-1)*lenx*mutrate); % total number of mutations
69      mrow=ceil(rand(1,nmut)*(popsize-1))+1; % row to mutate
70      mcol=ceil(rand(1,nmut)*lenx); % column to mutate
71      for ii=1:nmut
72          pop(mrow(ii),mcol(ii))=abs(pop(mrow(ii),mcol(ii))-1); % toggles bits
73      end % ii
74
75  %% The population is re-evaluated for cost
76      cost=feval(ff,pop); % calculates population cost using ff
77      [cost,ind]=sort(cost,'descend'); % max element in first entry
78      pop=pop(ind,:); % sorts population with max cost first
79
80      probs=cost/sum(cost);  %Re-set probabilities
81
82  % We keep track of some values for graphical output:
83      maxc(iga+1)=max(cost);
84      meanc(iga+1)=mean(cost);
85
86  %% Stopping criteria.  The double bar: || is an "or"
87      if iga>maxit || cost(1)>maxcost
88          break
89      end
90
91  end %iga
92
93
94  %% Displays the output
95  day=clock;
96  disp(datestr(datenum(day(1),day(2),day(3),day(4),day(5),day(6)),0))
97  %disp(['optimized function is ' ff])
98  format short g
99  disp(['popsize = ' num2str(popsize) ' mutrate = ' num2str(mutrate)]);
100 disp(['#generations=' num2str(iga) ' best cost=' num2str(cost(1))]);
101 fprintf('best solution\n%s\n',mat2str(int8(pop(1,:))));
102 figure(1)
103 iters=0:length(maxc)-1;
104 plot(1:(iga+1),maxc,1:(iga+1),meanc);
105 xlabel('generation');ylabel('cost');
106 legend('Max cost', 'Mean Cost')
```