CS322 Fall 1997
Assignment 5 — Solution

Due: 1:30pm, Friday 10 October 1997.

Question 1

In this question you will look at backtracking versus arc consistency for solving CSP problems.

Consider a scheduling problem, where there are 5 activities to be scheduled in 4 time slots. Lets represent the activities by the variables A, B, C, D, and E, and suppose the domain of each variable is $\{1, 2, 3, 4\}$ and the constraints are: $A > D$, $D > E$, $C \neq A$, $C > E$, $C \neq D$, $B \geq A$, $B \neq C$, $C \neq D + 1$.

1. [6 marks] Show how backtracking can be used to solve this problem. To do this you should draw the search tree generated to find all answers. Indicate clearly the valid schedule(s).

   Here is a solution tree based on the variable ordering C, D, A, B, E.

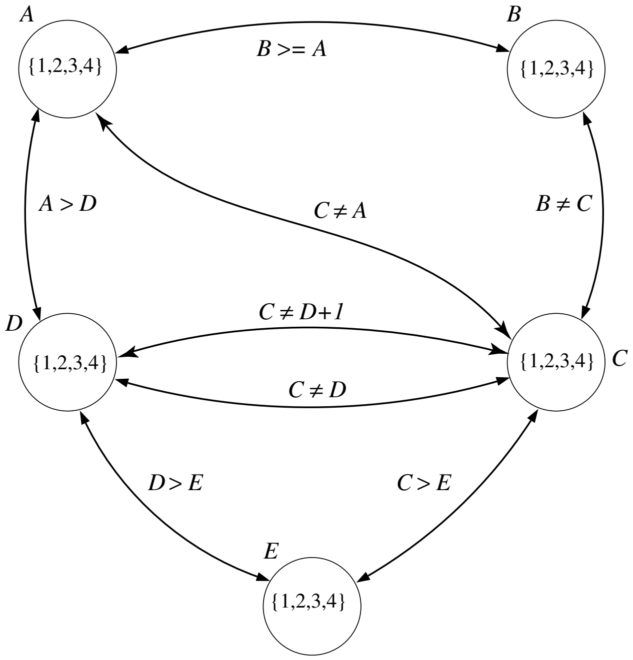
```
C=1 D=1 X
    D=2 A=1 X
        A=2 X
        A=3 B=1 X
            B=2 X
            B=3 E=1 X
                E=2 X
                E=3 X
                E=4 X
            B=4 E=1 X
                E=2 X
                E=3 X
                E=4 X
        A=4 B=1 X
            B=2 X
            B=3 X
            B=4 E=1 X
                E=2 X
                E=3 X
                E=4 X
    D=3 A=1 X
        A=2 X
```

```
          A=3  X
          A=4  B=1  X
               B=2  X
               B=3  X
               B=4  E=1  X
                    E=2  X
                    E=3  X
                    E=4  X
     D=4  A=1  X
          A=2  X
          A=3  X
          A=4  X
C=2  D=1  X
     D=2  X
     D=3  A=1  X
          A=2  X
          A=3  X
          A=4  B=1  X
               B=2  X
               B=3  X
               B=4  E=1  SOLUTION
                    E=2  X
                    E=3  X
                    E=4  X
     D=4  A=1  X
          A=2  X
          A=3  X
          A=4  X
C=3  D=1  A=1  X
          A=2  B=1  X
               B=2  X
               B=3  X
               B=4  E=1  X
                    E=2  X
                    E=3  X
                    E=4  X
     D=2  X
     D=3  X
     D=4  A=1  X
          A=2  X
          A=3  X
          A=4  X
C=4  D=1  A=1  X
          A=2  B=1  X
```

```
            B=2 E=1 X
                E=2 X
                E=3 X
                E=4 X
            B=3 E=1 X
                E=2 X
                E=3 X
                E=4 X
            B=4 X
        A=3 B=1 X
            B=2 X
            B=3 E=1 X
                E=2 X
                E=3 X
                E=4 X
            B=4 X
        A=4 X
    D=2 A=1 X
        A=2 X
        A=3 B=1 X
            B=2 X
            B=3 E=1 SOLUTION
                E=2 X
                E=3 X
                E=4 X
            B=4 X
        A=4 X
    D=3 A=1 X
        A=2 X
        A=3 X
        A=4 X
    D=4
```

2. [6 marks] Show how arc consistency can be used to solve this problem.

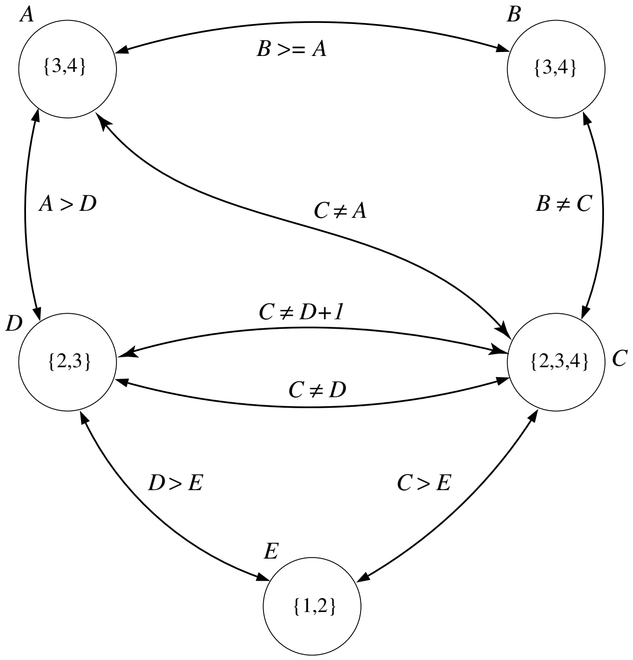
   - Here is the initial constraint graph:



- The following table shows which elements of a domain are deleted at each step, and which arc is responsible for removing the element:

Arc	Relation	Value(s) Removed
$\langle D, E\rangle$	$D > E$	$D = 1$
$\langle E, D\rangle$	$D > E$	$E = 4$
$\langle C, E\rangle$	$C > E$	$C = 1$
$\langle D, A\rangle$	$A > D$	$D = 4$
$\langle A, D\rangle$	$A > D$	$A = 1 \& A = 2$
$\langle B, A\rangle$	$B \geq A$	$B = 1 \& B = 2$
$\langle E, D\rangle$	$D > E$	$E = 3$

At this stage arc consistency has stopped. Here is the arc-constraint graph:



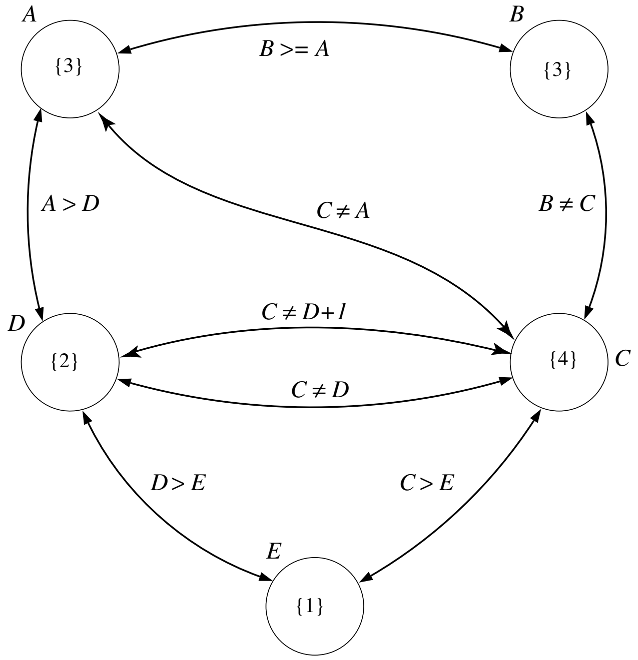
Note that, if you had one arc between C and D labelled with $C \neq D+1 \wedge C \neq D$, then $C = 3$ can be removed by considering the arc $\langle C, D\rangle$. If you have two arcs, $C = 3$ cannot be removed by arc consistency.

- Show how splitting a domain can be used to solve this problem. Lets split the domain of D. There are two cases, $D = 2$ and $D = 3$.

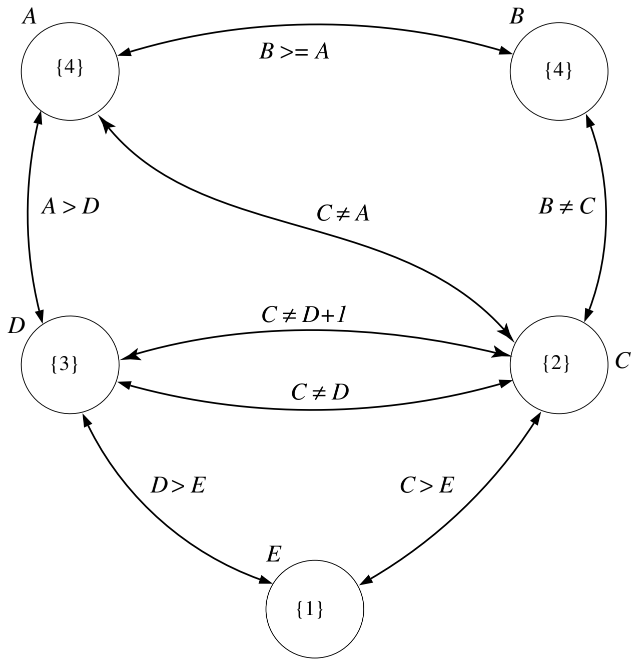
  Case 1. Lets put $D = 2$ into the constraint net (i.e., prune $D = 3$ from the constraint net) and run are consistency again.

Arc	Relation	Value(s) Removed
$\langle E, D\rangle$	$D > E$	$E = 2$
$\langle C, D\rangle$	$C \neq D$	$C = 2$
$\langle C, D\rangle$	$C \neq D + 1$	$C = 3$
$\langle A, C\rangle$	$C \neq A$	$A = 4$
$\langle B, C\rangle$	$B \neq C$	$B = 4$

  Here is the resulting constraint net.



Case 2. Lets put $D = 3$ into the constraint net (i.e., prune $D = 2$ from the arc-consistent constraint net) and run are consistency again. Here is the resulting constraint net.



These two constraint nets correspond to the only two solutions.

Question 2

In this question you will implement the graph searcher for finding the shortest video presentations for the previous two assigments.

For example, given the segment database (from Assignment 3),

```
% segment(SegId,Time,Topics) is true if segment SegId takes time Time
% and covers the topics in Topics.
segment(seg0,10,[welcome]).
segment(seg1,30,[skiing,views]).
segment(seg2,50,[welcome,computational_intelligence,robots]).
segment(seg3,40,[graphics,dragons]).
segment(seg4,50,[skiing,robots]).
```

you also need to assume that you have a list of all segments:

```
% all_segments(L) is true if L is the list of all segments.
all_segments([seg0,seg1,seg2,seg3,seg4]).
```

Suppose that a node is represented as $vp(To\_Cover, Segs)$ where *Segs* is a list of segments that must be in the presentation, and *To_Cover* is a list of topics that also must be covered, such that none of the segments in *Segs* covers any of the topics in *To_Cover*.

Write the *neighbors* relation, so that

```
ask neighbours(vp([welcome,robots],[]),Neighs).
```

has one answer (perhaps in the other order):

```
Answer: neighbours(vp([welcome,robots],[]),
                   [vp([],[seg2]),vp([robots],[seg0])]).
```

You also need to define the *cost* predicate, where the cost of an arc is the time of the segment added, and an *is_goal* predicate.

Show how this can be used with the *hsearch* program.

Solution:

```
% ask hsearch(depth,[node(vp([welcome,robots],[]),[],0,0)],P).
% ask hsearch(breadth,[node(vp([welcome,skiing,robots],[]),[],0,0)],P).

% segment(SegId, Duration, Covers)

segment(seg0,10,[welcome]).
segment(seg1,30,[skiing,views]).
segment(seg2,50,[welcome,computational_intelligence,robots]).
segment(seg3,40,[graphics,dragons]).
segment(seg4,50,[skiing,robots]).
```

```
all_segments([seg0,seg1,seg2,seg3,seg4]).

% A node is of the form
%   vp(ToCover,Segs) where ToCover is a list of topics to cover and
% Segs is a list of segments.

is_goal(vp([],_)).

% ask neighbours(vp([welcome,robots],[]),Neighs).

neighbours(vp([],_),[]).

neighbours(Node,Neighs) <-
   all_segments(AllSegs) &
   neighbour_segments(AllSegs,Node,Neighs).

% neighbour_segments(AllSegs,Node,Neighs). Is true if Neighs is the
% list of all neighbours that use segments in AllSegs.

neighbour_segments([],_,[]).
neighbour_segments([Seg|RSegs],vp([Topic|OtherTopics],Segs),
                        [vp(TopicsToCover,[Seg|Segs])|Neighs]) <-
   segment(Seg,_,SegTopics) &
   member(Topic,SegTopics) &
   listdiff(OtherTopics,SegTopics,TopicsToCover) &
   neighbour_segments(RSegs,vp([Topic|OtherTopics],Segs),Neighs).

neighbour_segments([Seg|RSegs],vp([Topic|OtherTopics],Segs),Neighs) <-
   segment(Seg,_,SegTopics) &
   notin(Topic,SegTopics) &
   neighbour_segments(RSegs,vp([Topic|OtherTopics],Segs),Neighs).

member(A,[A|_]).
member(A,[_|L]) <-
   member(A,L).

notin(_,[]).
notin(A,[B|L]) <-
   A \= B &
   notin(A,L).

cost(vp(_,Segs),vp(_,[Seg|Segs]),SegTime) <-
   segment(Seg,SegTime,_).
```

```
listdiff([],_,[]).
listdiff([H|T],S,RT) <-
   member(H,S) &
   listdiff(T,S,RT).
listdiff([H|T],S,[H|RT]) <-
   notin(H,S) &
   listdiff(T,S,RT).
```

The following query should give the shortest presentation:

```
ask hsearch(shortest,[node(vp([welcome,skiing,robots],[]),[],0,0)],P).
```

Show that it works. (Note that asking "how" will show the steps of the search).