COSC202 Mid-Year Test

Department of Computer Science
University of Canterbury

3 July 1998

Problem 1 [8 marks]

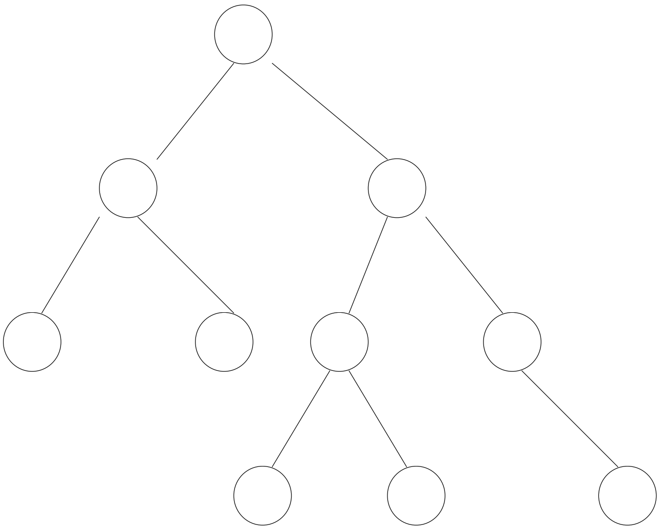
We try to convert the quicksort program from a recursive one to an iterative one from the following observation. If we partition with the pivot as the leftmost element, we have

(41, 24, 76, 11, 45, 64, 21, 69, 19, 36) partitioned into

(36, 24, 19, 11, 21, 41, 64, 69, 45, 76)

Here partition(1, 10, m) will return m=6. Then we push pairs (1, 5) and (7, 10) into a stack, resulting in ((1,5), (7,10)). Take the first pair from the stack and keep doing the same until the stack becomes empty. Note that the stack is initialized to ((1, 10)), and the general appearance of the stack looks like ((a,b),(c,d), ..., (x,y))

Problem 2. [8 marks] Insert 77 into the following AVL tree and rotate it if necessary. Show the subtrees A, B, etc., and the labels L, R, and E at the nodes, indicating left, right, and equal.

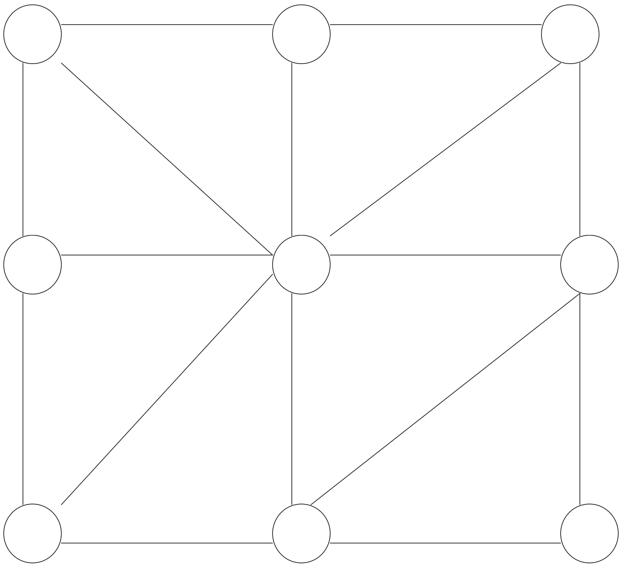


Problem 3 [8 marks]
(1) Make a graph with 6 vertices and not less than 10 edges with edge costs of your own choice. [2 marks]

(2) Trace Floyd's algorithm with your graph to compute the all-pairs shortest distances.

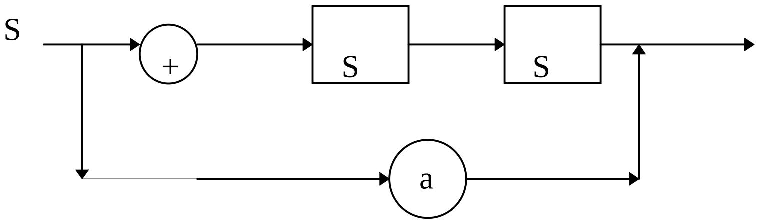
Problem 4 [8 marks] Trace Tarjan’s algorithm in page 36 for strongly connected components with the following graph.

Problem 5 [8 marks] Trace Kruskal’s algorithm for minimum cost spanning tree with the following graph



Problem 6 [12 marks]

The following is a syntax chart for prefix Polish notations for operator + and operand a. The infix form a+a is expressed by its equivalent Polish notation +aa. This is a natural expression of English sentence “add a to a”.



The Polish notations specifies the order of operations by operators, so we do not need parentheses. For example, we have the following correspondence between infix forms and Polish notations.

(a+a)+a ===> ++aaa interpreted as +(+aa)a

a+(a+a) ===> +a+aa interpreted as +a(+aa)

The recursive descent parser for the above syntax chart is given below.

```
procedure S;
begin
  if current = “+” then begin getsym; S; S end
  else if current = “a” then getsym
end;
begin {main program}
  set up the input string;
  getsym;
  S;
  if current = “!” then write(‘no errors’) else write(‘errors’)
end.
```

Procedure getsym is to give the next symbol to current. If we trace this parser with +a+aa!, we have the following.

History	Input	Comments
S	+a+aa!	consume +
S	a+aa!	enter S
SS	a+aa!	consume a
SS	+aa!	exit from S
S	+aa!	enter S
SS	+aa!	consume +
SS	aa!	enter S
SSS	aa!	consume a
SSS	a!	exit from S
SS	a!	enter S
SSS	a!	consume a
SSS	!	exit from S
SS	!	exit from S
S	!	exit from S
empty	!	signal ‘no errors’

(1) Draw a syntax chart for prefix Polish notations including operators + and *, and operands a and b. [3 marks]

(2) Write a recursive descent parser for the above syntax chart. [3 marks]

(3) Show the equivalent infix form for the prefix Polish notation *+ab+ba. [3 marks]

(4) Trace your parser with the string *+ab+ba! [3 marks]

Problem 7. The following is a PL0 program for computing a factorial, its object code, trace at each store operation, and the stack movie at each return.

```
VAR F,N;
PROCEDURE FACT;
VAR M;
BEGIN
  M:=N;
  N:=N-1;
  IF N=0 THEN F:=1;
  IF N>0 THEN CALL FACT;
  F:=F*M;
END;
BEGIN
  N:=3;
  CALL FACT;
END.
```

Execution trace at store operations

```
 0  JMP    0   25
 1  JMP    0    2
 2  INT    0    4
 3  LOD    1    4
 4  STO    0    3
 5  LOD    1    4
 6  LIT    0    1
 7  OPR    0    3
 8  STO    1    4
 9  LOD    1    4
10  LIT    0    0
11  OPR    0    8
12  JPC    0   15
13  LIT    0    1
14  STO    1    3
15  LOD    1    4
16  LIT    0    0
17  OPR    0   12
18  JPC    0   20
19  CAL    1    2
20  LOD    1    3
21  LOD    0    3
22  OPR    0    4
23  STO    1    3
24  OPR    0    0
25  INT    0    5
26  LIT    0    3
27  STO    0    4
28  CAL    0    2
29  OPR    0    0
```

Stack movie of 4 shots (Read this vertically. The leftmost column shows locations)

1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	1	2	6	6
5	0	0	1	0
6	1	1	1	
7	1	1	1	
8	29	29	29	
9	3	3	3	
10	1	1		
11	6	6		
12	20	20		
13	2	2		
14	1			
15	10			
16	20			
17	1			

(1) Following the example in the attached sheet, give detailed comments to the object code, execution trace, and stack movie.

(2) Write an object code for the following PL0 program for summing up 1, ... , N. [12 marks]

```
CONST N=10;
VAR A, S;
PROCEDURE SUM;
VAR M;
BEGIN
  M:=A;
  A:=A-1;
  IF A=0 THEN S:=0;
  IF A>0 THEN CALL SUM;
  S:=S+M;
END;
BEGIN
  A:=N;
  CALL SUM;
END.
```