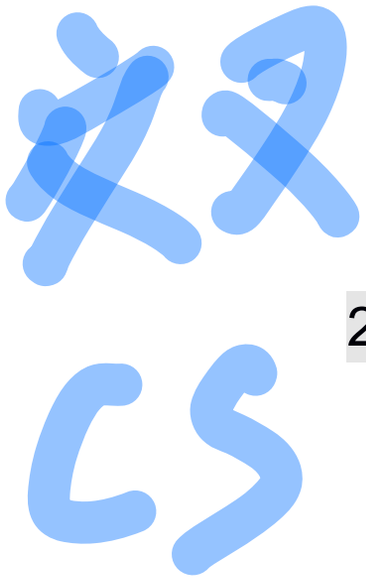


北京航空航天大学

2019－2020 学年 第一学期期末

《编译技术》

B 卷

班 级_____学 号 _____

姓 名_____成 绩 _____

2020 年 5 月 25 日

班号_____ 学号_____ 姓名_____ 成绩_____

《 编译技术 》期末考试卷

注意事项：

1. 试卷共 6 页（不含封面和目录），请仔细检查。

2. 在监考老师统计完试卷后，再离开考场；

题目：

一、简答题…… (20 分)

二、正则文法与自动机…… (20 分)

三、算符优先分析法……(15 分)

四、SLR 分析法…… (15 分)

五、符号表构造与运行时存储分析…… (15 分)

六、代码优化……(15 分)

题号	得分	教师签字
1		
2		
3		
4		
5		
6		
总分		

一. 简答题：(20 分)

1. 简述什么是错误的局部化处理，主要作用是什么?

错误局部化处理:指当编译程序发现错误后，尽可 能把错误的影响限制在一个局部的范围，避免错误扩散和影响 程序其他部分的分析。

2. 分别简述语法分析的任务和语义分析的任务。

语法分析功能:根据文法规则，从源程序单词符号串中识别出语法 成分，并进行语法检查。
基本任务:识别符号串 S 是否为某语法成分。

语义分析是审查源程序有无语义错误，为代码生成阶段收集类型信息。

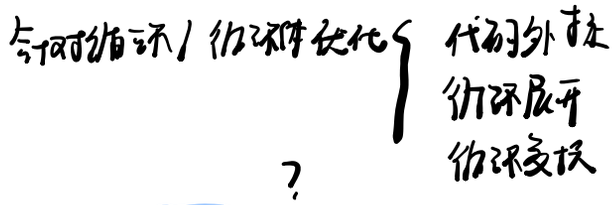
3. 分别简述静态存储分配和动态存储，二者的使用场景有什么不同。

静态存储分配
在编译阶段由编译程序实现对存储空间的 管理和为源程序中的变量分配存储的方法。
条件
如果在编译时能够确定源程序中变量在运 行时的数据空间大小，且运行时不改变，那么 就可以采用静态存储分配方法。

动态存储分配
在目标程序运行阶段由目标程序实现对存 储空间的组织与管理，和为源程序中的变量 分配存储的方法。
特点
• 在目标程序运行时进行变量的存储分配。 • 编译时要生成进行动态分配的目标指令。

4. 分别说明什么是局部优化、全局优化和循环优化。

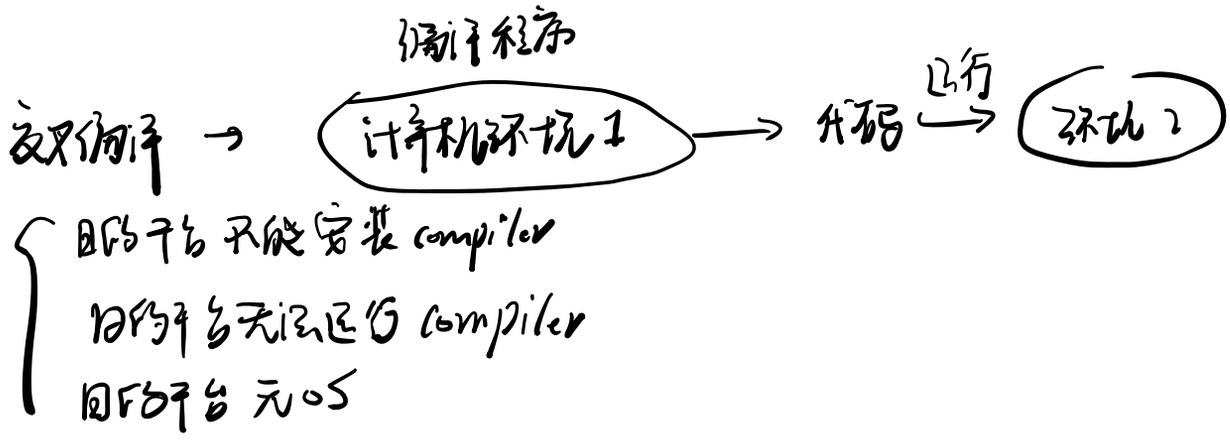
- 局部优化技术
– 指在基本块内进行的优化
– 例如，局部公共子表达式删除
- 全局优化技术
– 函数/过程内进行的优化 – 跨越基本块
– 例如，全局数据流分析
- 跨函数优化技术 – 整个程序
– 例如，跨函数别名分析 ，逃逸分析 等



5. 说明什么是交叉编译，什么时候需要使用交叉编译?

在一种计算机环境中运行的编译程序，能编译出在另外一种环境下运行的代码，我们就称这种编译器支持交叉编译。这个编译过程就叫交叉编译。

有时是因为目的平台上不允许或不能够安装我们所需要的编译器，而我们又需要这个编译器的某些特征；有时是因为目的平台上的资源贫乏，无法运行我们所需要编译器；有时又是因为目的平台还没有建立，连操作系统都没有，根本谈不上运行什么编译器。



二. 在输入字母表Σ={0, 1} 上定义语言 L = {x|x∈ Σ*，且 x 以 0 开头，以 1 结尾}。(20 分)

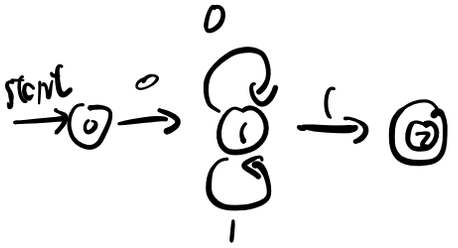
1. 给出识别该语言 L 的正则表达式。

0(0|1)*1

$0(0|1)^*1$

2. 根据正则表达式构造 NFA。

X



3. 设计识别该语言的极小化 DFA。(给出求解过程：①对应的)

确定化

I-closure[0] = 0

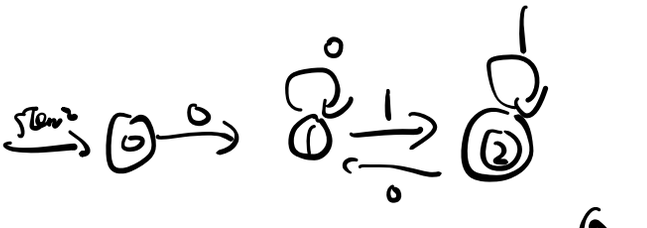
Z	Z_0	Z_1
{0}	{1}	ϕ
{1}	{1}	{1,2}
{1,2}	{1}	{1,2}

0→1　0→ϕ

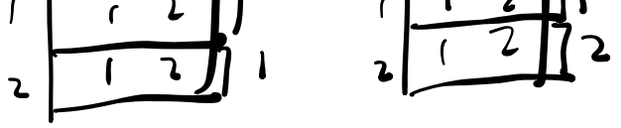
1→1　1→1,2

1→1　1→1,2

状态 \ 符	0	1
0	1	-
1	1	2
2	1	2



最小化



三. 有如下文法 G[S]：（15 分）

S → V

V → T | V i T

T → F | T + F

F → b V* | a

文法中无 U::= …VW…

V, W, ∈ Vn

1. 判断该文法是否是算符文法，并说明理由。

是，因为文法中无形如 U:: = …VW…的规则

2. 求每个非终结符的 FIRSTVT 和 LASTVT 集合。

F(S) = {i, +, b, a}

L(S) = {i, +, *, a}

F(V) = {i, +, b, a}

L(V) = {i, +, *, a}

F(T) = { +, b, a}

L(T) = {+, *, a}

F(F) = {b, a}

L(F) = {*, a}

FIRSTVT(S)=FIRSTVT(V) ={a,b,i,+}

FIRSTVT(V)=FIRSTVT(T) ∪ {i} ={a,b,i,+}

FIRSTVT(T)=FIRSTVT(T+V)={a,b,+}

FIRSTVT(T+F)={a,b}

LASTVT(S) =LASTVT(V) = {*, a, +, i}

LASTVT(V) = LASTVT(T) ∪ {i} = {*, a, +, i}

LASTVT(V) = LASTVT(T+F) ∪ {+} = {*, a, +}

FIRSTVT(F)={*, a}

#S#

#< FIRST(S)

LASTVT(S) > #

V i T

LASTVT(V) > i

i < FIRSTVT(T)

T+F

LASTVT(T) > +

+ < FIRSTVT(T+F)

3. 构造算法优先关系矩阵。

	i	+	b	*	a	#
i	>	<	<	>	<	>
+	>	>	<	>	<	>
b	<	<	<	=	<	
*	>	>		>		>

b ∨ *

b ⋙ * b ⋖ FIRSTVT(V) LASTVT(V) ⋗ *

a	⋗	⋗		⋗		⋗
#	⋖	⋖	⋖		⋖	

4. 判断该文法是否为算符优先文法。

是，因为算符的优先关系唯一 ✓

四. 有如下文法 G[S]: (15 分)

S→ SaBc| aBc
B→ bAc
A→ a | bS

1. SLR 分析表（Action 表和 GOTO 表）。

(1) S→ SaBc
(2) S→ aBc
(3) B→ bAc
(4) A→ a
(5) A→ bS

I0:
S = .SaBc
S = .aBc

I1: I0 + S
S = S.aBc

I2: I0 + a
S = a.Bc
B = .bAc

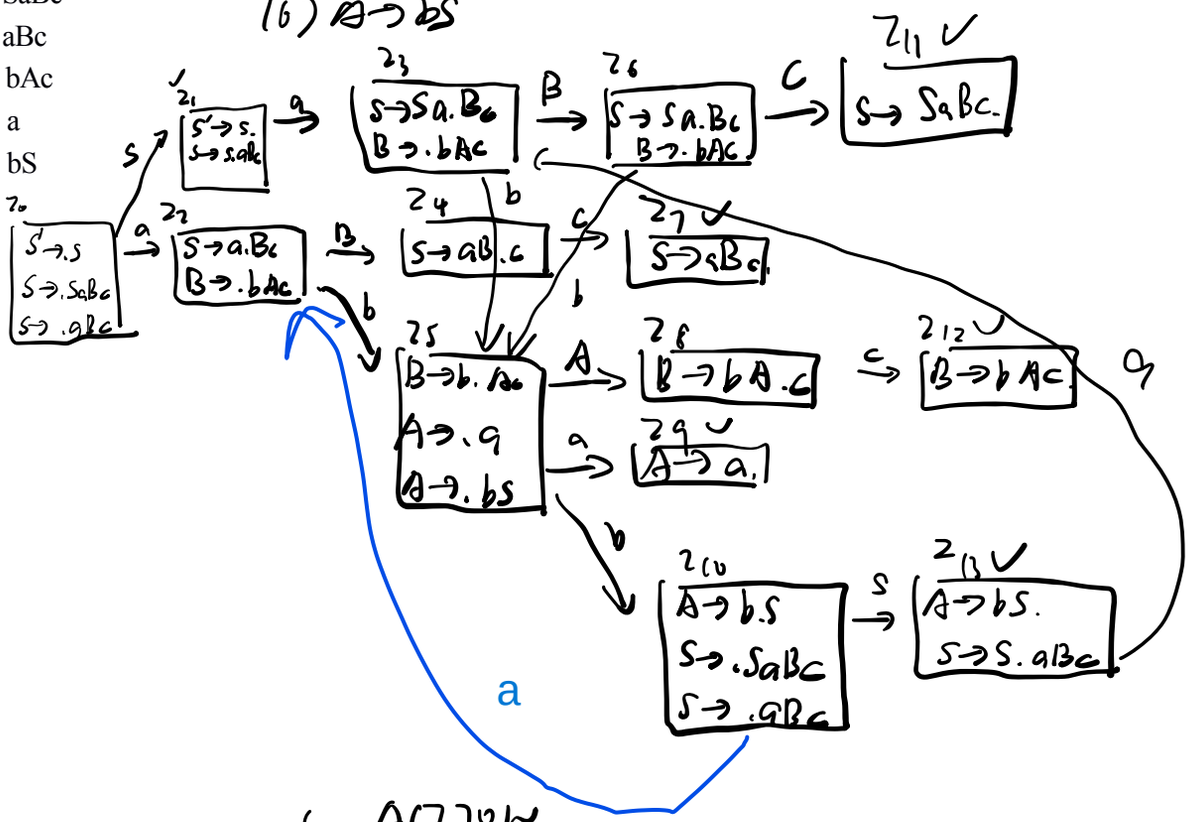
I3: I1 + a
S = Sa.Bc
B = .bAc

I4: I2 + B
S = aB.c

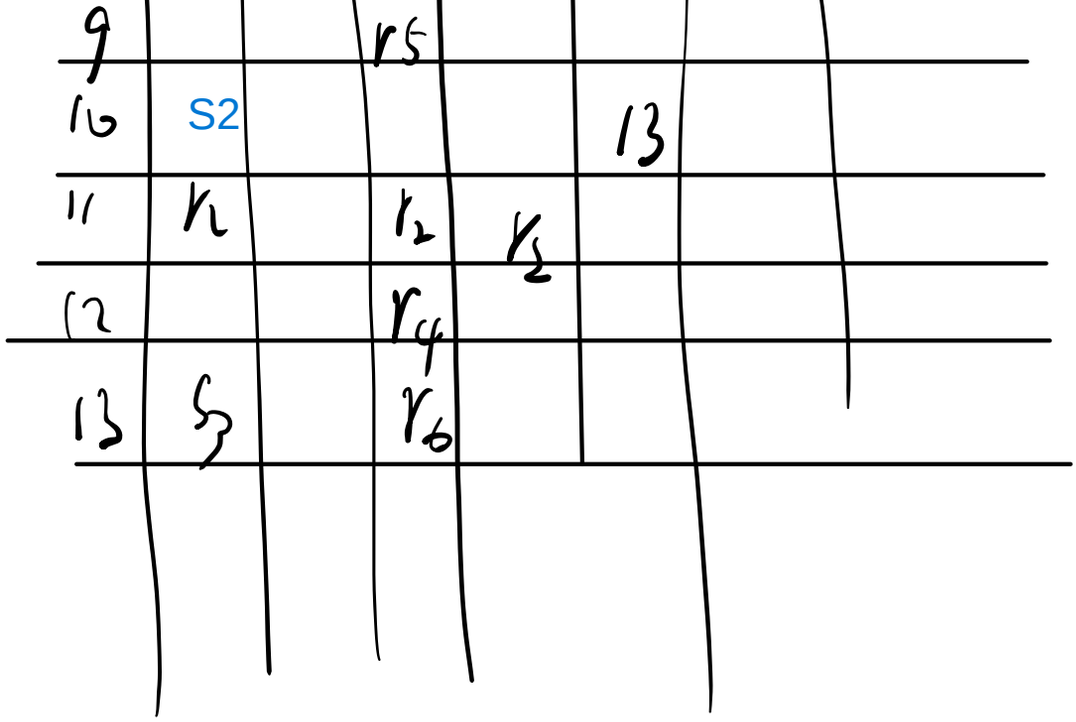
I5: I2 + b
B = b.Ac
A = .a
A = .bS

I6: I3 + B
S = SaB.c

*I7: I3 + b
goto I5



	a	b	c	#	S	A	B
0	S2				1		
1	S3			acc-pt			
2		S5					4
3		S5					6
4			S7				
5	S9	S10				8	
6			S11				
7	R3		R3	R3			
8		S12					



I7: I4 + c
S = aBc.

I8: I5 + A
B = bA.c

I9 : I5 + a
A = a.

I10 : I5+b
A = b.S
S = .SaBc
S = .aBc

I11: I6 +c
S = SaBc.

I12: I8 +c
B = bAc.

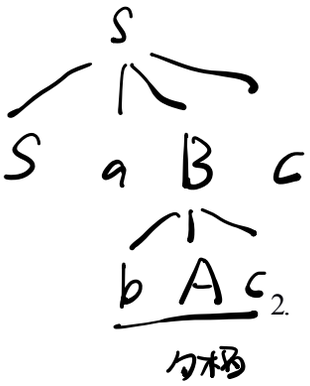
I13: I10 + S
A = bS.
S = S.aBc

*I14: I10 + a
goto I2

*I14: I13 + a
goto I3

	a	b	c	#	S	A	B
0	S2				1		
1	S3						
2		S5					4
3		S5					6
4			S7				
5	S9	S10				8	
6			S11				
7				r2			
8			S12				
9							
10	S2				13		
11				r1			
12				r3			
13	S3						

S'



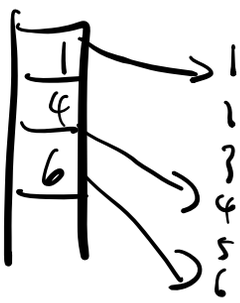
2. 识别句型 SabAcc 活前缀 SabAc 的有效项目集 。

Z_{12} [S → bA c.]

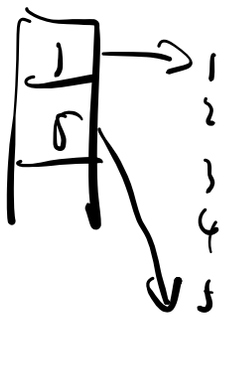
五. 有如下程序段：（15 分）

```
program paser;
    var filename:string;
        curchar:char;
procedure getsym;
    var symsize:integer;
    procedure getchar;
        var errorinfo:string;
        begin
          …          ---------- (1)
        end;
    begin
        …
        call getchar;
        …
    end;
procedure lexer;
    var linelength:integer;
    begin
        …
        call getsym;---------- (2)
        …
    end;
begin
    …
                                call lexer;
    …
end.
```

1. 请分别画出当编译到位置（1）和（2）时的符号表。



	变量名	类型	维数 lev
1	filename	string	0 1
2	curchar	char	0 1
3	getsym	proc	1
4	symsize	integer	0 2
5	getchar	proc	2
6	errorinfo	string	0 3

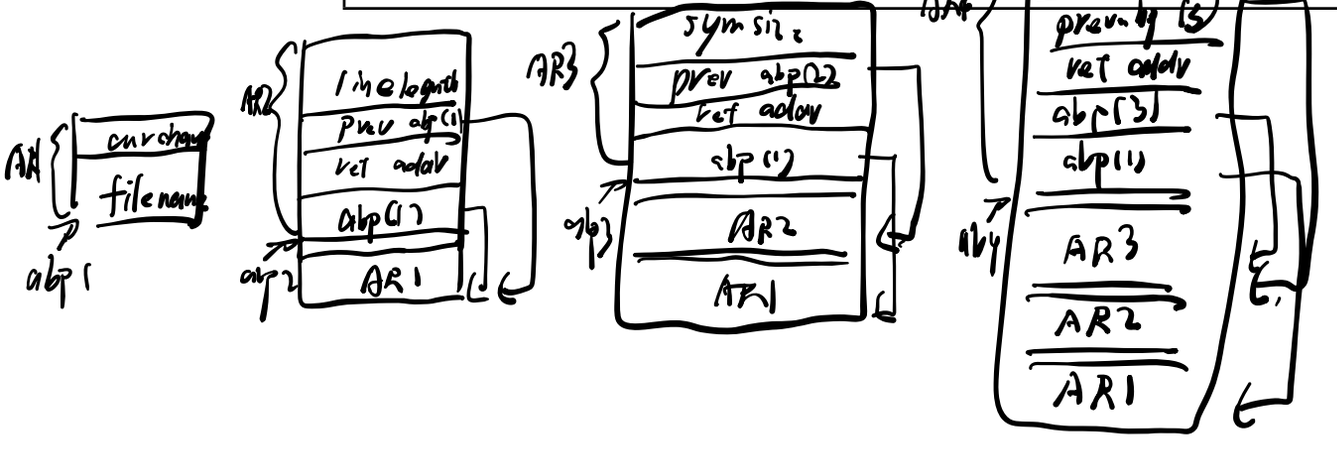


	变量名	类型	维数
1	filename	string	0 1
2	curchar	char	0 1
3	getsym	proc	1
4	lexer	proc	1
5	linelength	integer	2

symsize	integer	0
getchar	proc	
errorinfo	string	0
lexer	proc	
linelength	integer	0

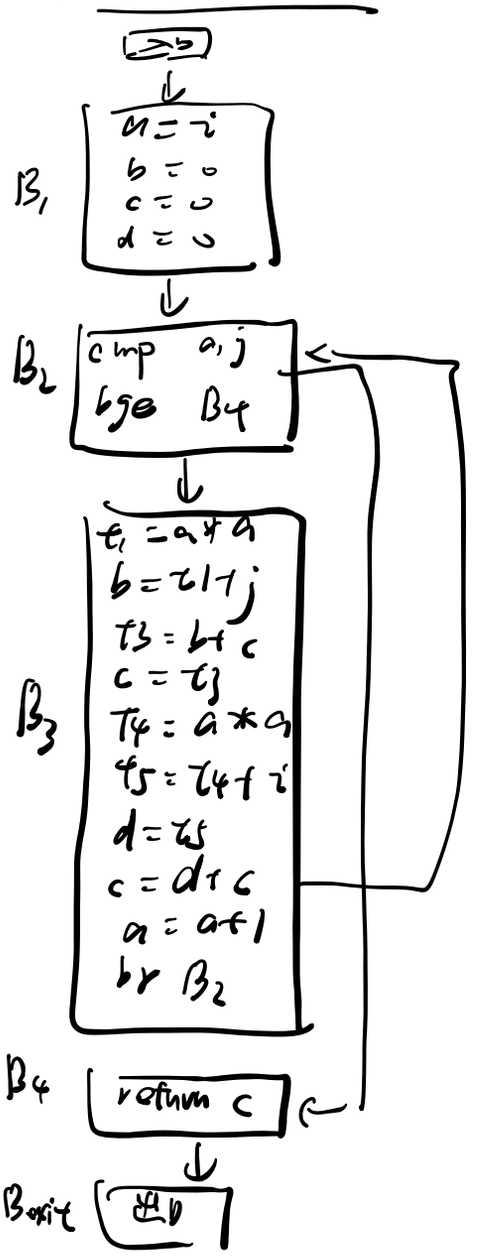
2. 运行到位置（1）时的运行栈。

errorinfo
prev abp(3)
ret addr(4)
abp(4)
abp(3)
abp(2)
abp(1)
symsize
prev abp(2)
ret addr(3)
abp(3)
abp(2)
abp(1)
linelength
prev abp(1)
ret addr(2)
abp(2)
abp(1)
AR1



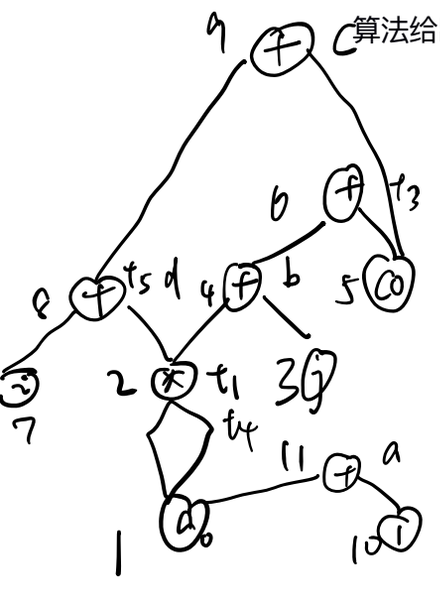
六. 有如下中间代码序列，其中{a,b,c,d}为局部变量，{i,j}为形参，{t1,t2,t3,t4,t5,t6}为临时变量（15 分）：

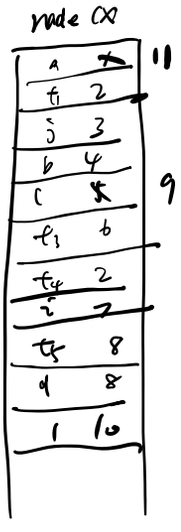
```
        a = i
        b = 0
        c = 0
        d = 0
_loop_begin:
        cmp a, j
        bge _loop_end //a>=j
        t1 = a * a
        b = t1 + j
        t3 = b + c
        c = t3
        t4 = a * a
        t5 = t4 + i
        d = t5
        c = d + c
        a = a + 1
        br _loop_begin // jmp _loop_begin
_loop_end:
        return c
```



1. 对图中代码序列划分基本块，构建流图。

2. 对循环体所在基本块利用 DAG 图做“局部公共子表达式删除”优化，并根据启发式算法给出优化后的中间代码序列。





11 a
1 c
8 t_5
2 此有4条
6 t_3
4 b
2 t_1

a c t_5 t_3 b t_1

优化序

$t_1 = a_0 * a_0$
$b = t_1 + j$
$t_3 = b + c_0$
$t_5 = t_1 + i$
$c = t_5 + t_3$
$a = a_0 + 1$