1.

min x6

st

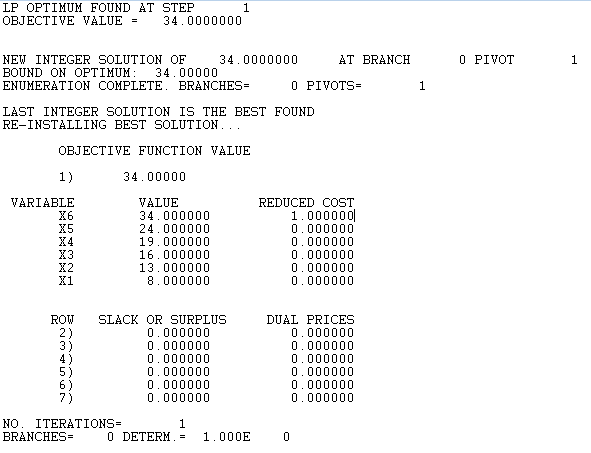
x6 - x5 = 10

x5 - x4 = 5

x4 - x3 = 3

x3 - x2 = 3

x2 - x1= 5

x1 = 8

end

gin 6

2.

min 3x01 + 5x02 + 2x12 + 1x32 + 6x13 + 4x23 + 3x40 + 2x34 + 7x43 + 6x24

st

x01 + x21 - x13 - x12 = 0

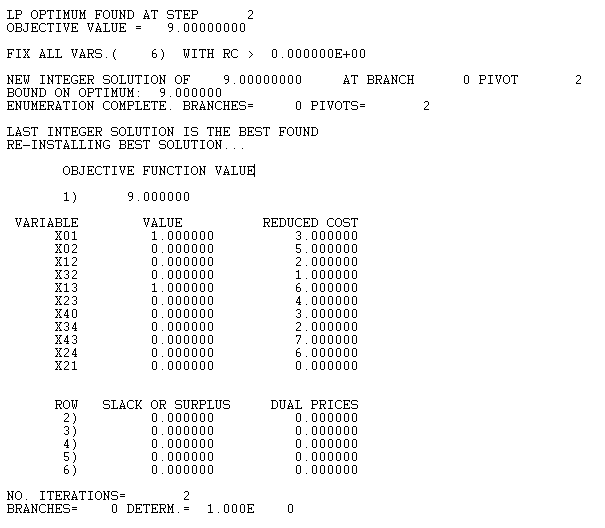
x02 + x12 -x23 - x24 - x21 = 0

x13 + x23 + x43 = 1

x24 -x43 = 0

x01 + x02 = 1

end

inte 103.

min y1 + y2 + y3 + y4

st

y1 = 1

y2 = 1

y3 = 1

y4 = 1

x10 + x11 + x12 + x13 + x14 + x15 + x16 + x17 + x18 + x19 + x110 = 1

x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29 + x210 = 1

x30 + x31 + x32 + x33 + x34 + x35 + x36 + x37 + x38 + x39 + x310 = 1

x40 + x41 + x42 + x43 + x44 + x45 + x46 + x47 + x48 + x49 + x410 = 1

y1 - x01 >= 0

y1 - x11 >= 0

y1 - x21 >= 0

y1 - x31 >= 0

y1 - x41 >= 0

y1 - x51 >= 0

y1 - x61 >= 0

y1 - x71 >= 0

y1 - x81 >= 0

y1 - x91 >= 0

y1 - x101 >= 0

y2 - x02 >= 0

y2 - x12 >= 0

y2 - x22 >= 0

y2 - x32 >= 0

y2 - x42 >= 0

y2 - x52 >= 0

y2 - x62 >= 0

y2 - x72 >= 0

y2 - x82 >= 0

y2 - x92 >= 0

y2 - x102 >= 0

y3 - x03 >= 0

y3 - x13 >= 0

y3 - x23 >= 0

y3 - x33 >= 0

y3 - x43 >= 0

y3 - x53 >= 0

y3 - x63 >= 0

y3 - x73 >= 0

y3 - x83 >= 0

y3 - x93 >= 0

y3 - x103 >= 0

y4 - x04 >= 0

y4 - x14 >= 0

y4 - x24 >= 0

y4 - x34 >= 0

y4 - x44 >= 0

y4 - x54 >= 0

y4 - x64 >= 0

y4 - x74 >= 0

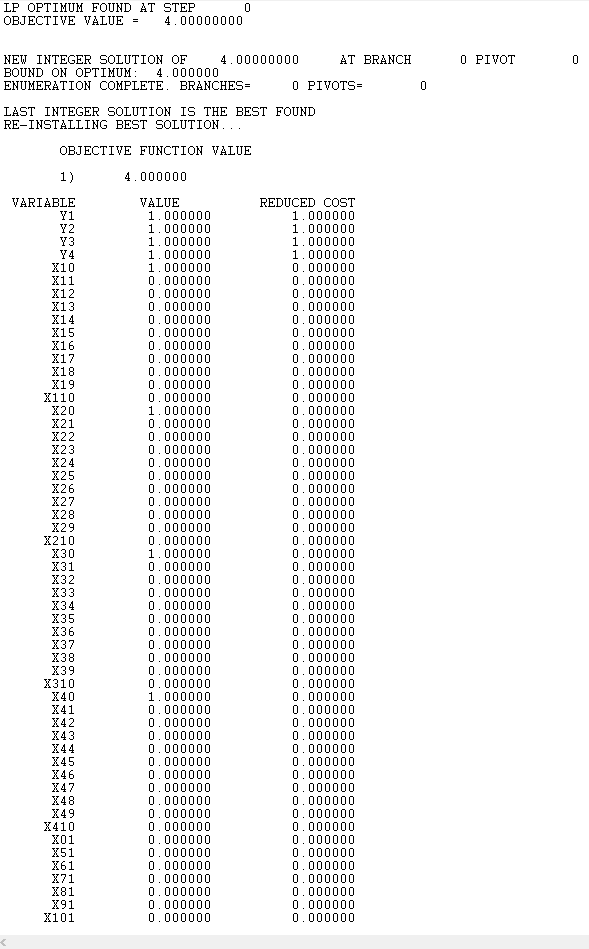
y4 - x84 >= 0

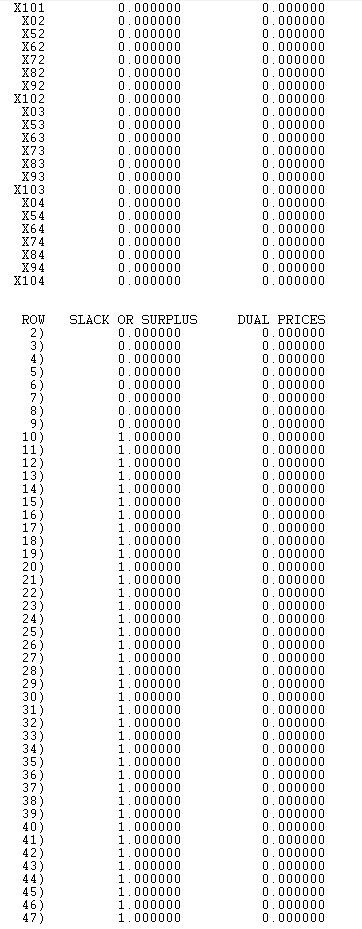
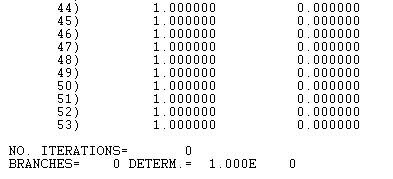
y4 - x94 >= 0

y4 - x104 >= 0

end

inte 48





4.

Slide 3 - Type 2 Context-free grammar – To be a type 3 grammar slide 3 would have to have only one terminating symbol.

Slide 6 – Type 3 Regular grammar – Has only one symbol on the left, one terminating symbol at the far right, and right-hand side has one or more symbols.

5.

LL parsers begin from the starting symbol and repeatedly expand the leftmost non-terminal symbol progressing to the right until it reaches a terminal symbol; a top to bottom left-to-right non-terminal symbol to terminating symbol

LR parsers scan left to right as well but constructs the right-most terminal symbol back into the beginning non-terminal symbol; a bottom to top reconstruction of a terminal symbol into a non-terminal beginning symbol.

An example of a real-world LR parser would be the POSIX yacc compiler.

An example of a real-world LL parser would be the older versions of the gcc compiler before they added parser generators.

Sources:

<http://stackoverflow.com/questions/5975741/what-is-the-difference-between-ll-and-lr-parsing>

<http://www.unix.com/man-page/POSIX/1posix/yacc/>

<https://www.quora.com/What-are-the-parsing-techniques-used-by-modern-compilers>

<https://en.wikipedia.org/wiki/Comparison_of_parser_generators>

6. p864 29.2-4

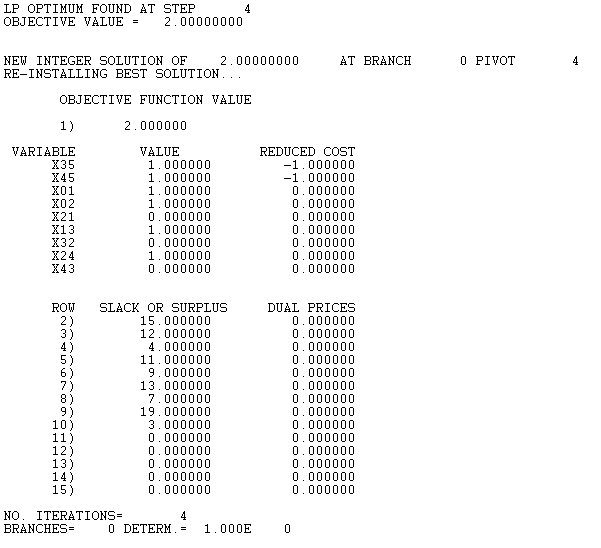
max x35 + x45

st

x01 <= 16

x02 <= 13

x21 <= 4

x13 <= 12

x32 <= 9

x24 <= 14

x43 <= 7

x35 <= 20

x45 <= 4

x35 + x45 - x01 - x02 = 0

x01 + x21 - x13 = 0

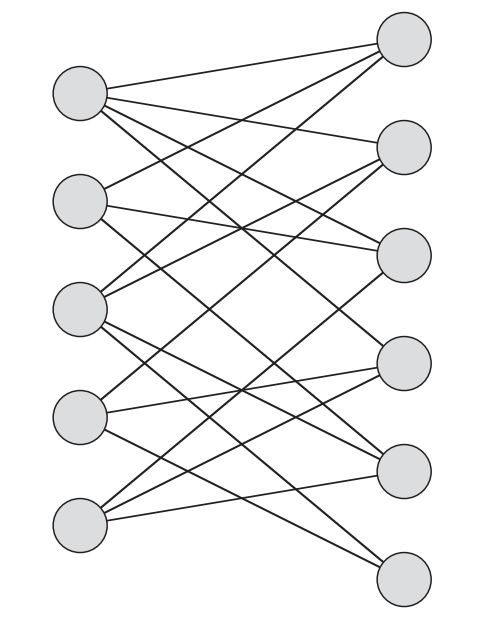
x02 + x32 - x24 - x21 = 0

x24 - x43 - x45 = 0

x13 + x43 - x32 - x35 = 0

end

inte 9

7. 34.2-2

A graph is hamiltonian if it visits each node exactly once and produces a cycle.

An undirected bipartite graph with an odd number of verticies is nonhamiltonian. Let k represent an odd number of nodes. For k+2 nodes, every odd-node graph there will be k/2 nodes in one column and k/2 -1 nodes in the other column. A cycle cannot be completed with unequal nodes in both columns of the undirected bipartite graph. Therefore all k and k+2 graphs are nonhamiltonian.

9. 34.4-7

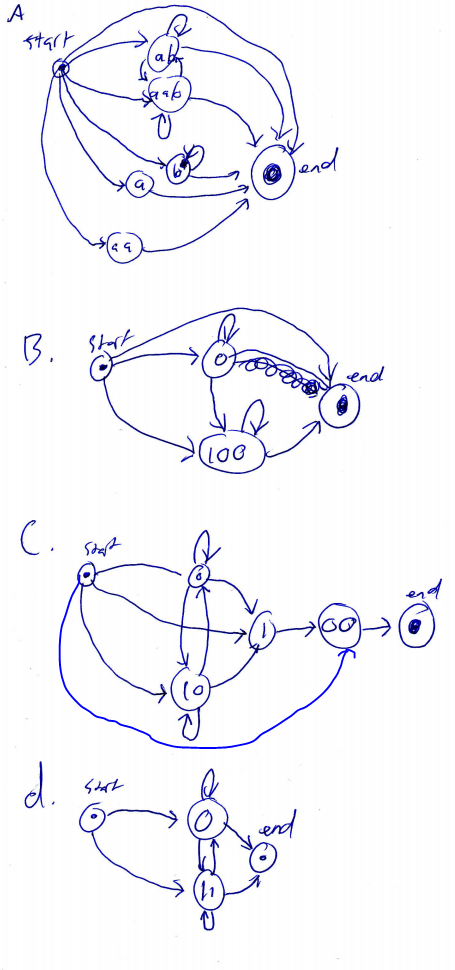
2-CNF possibilities (x1 v x2) (-x1 v –x2) (-x1 v x2) (x1 v –x2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X1 | X2 | X1 v x2 | (-x1 v –x2) | (-x1 v x2) | (x1 v –x2) |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |

2-CNF is in P since 2-CNF is always satisfiable one varaible will always imply the other variable is the same after negation. If x1 is one then x2 will also be one, if x1 is zero then –x2 will be 1.

The book was convoluted so I used the link below as a reference/source.

<http://stackoverflow.com/questions/8467676/how-is-2-cnf-sat-is-in-p-while-3-cnf-sat-is-in-npc>

10.

a.

((ab v aab)\* v a v aa v b\*)

b.

0\*(100)\*

c.

( (0\* v 10 )\* v 1 )00

d.

(0 v 11)\*