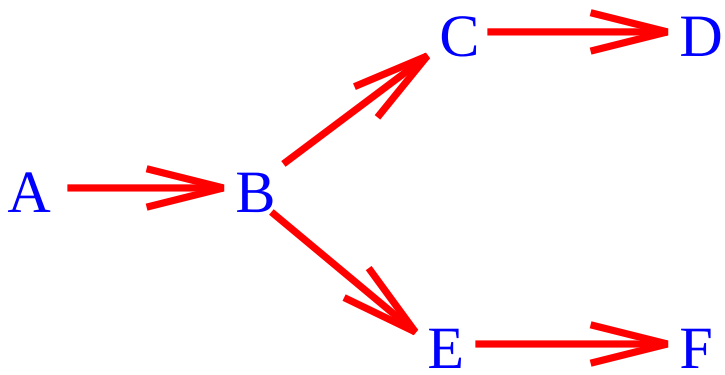
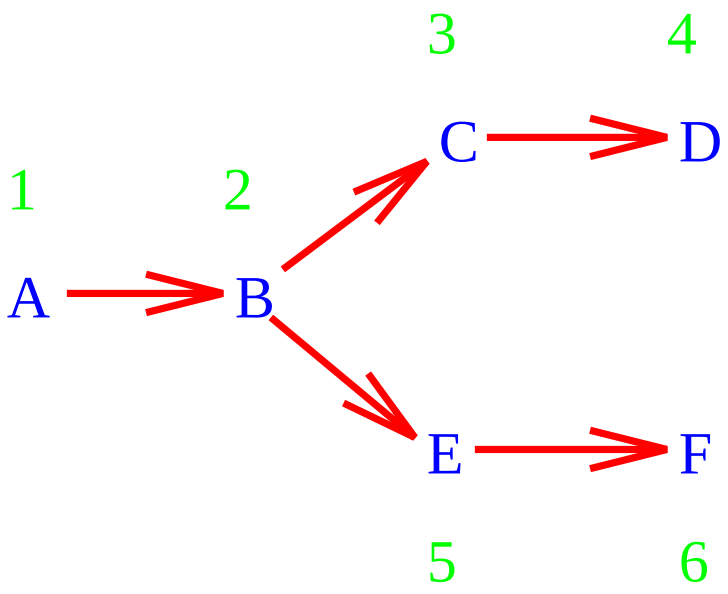
Copyright ©1996, 1997
by David A. Plaisted

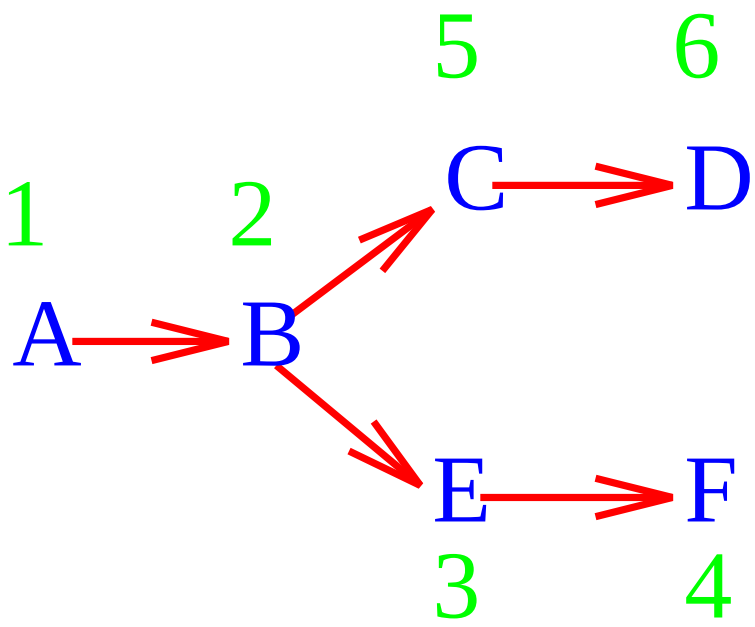
Chapter 23

Topological Sort

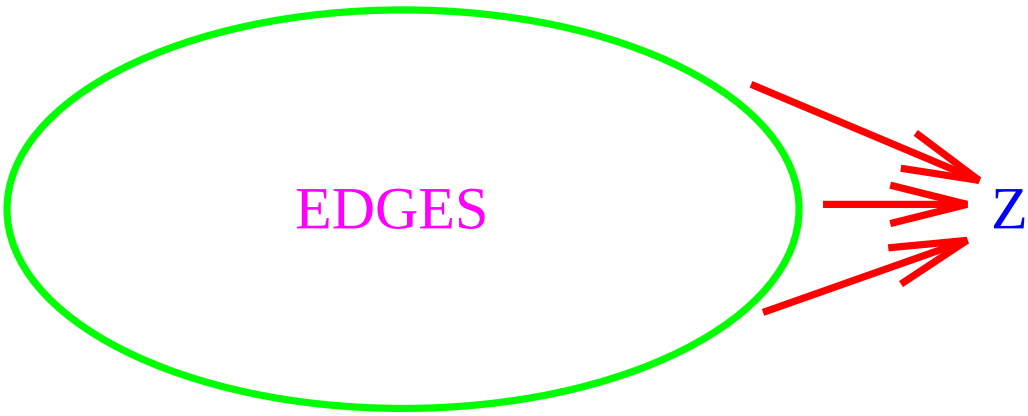

A
B
C
D
E
F



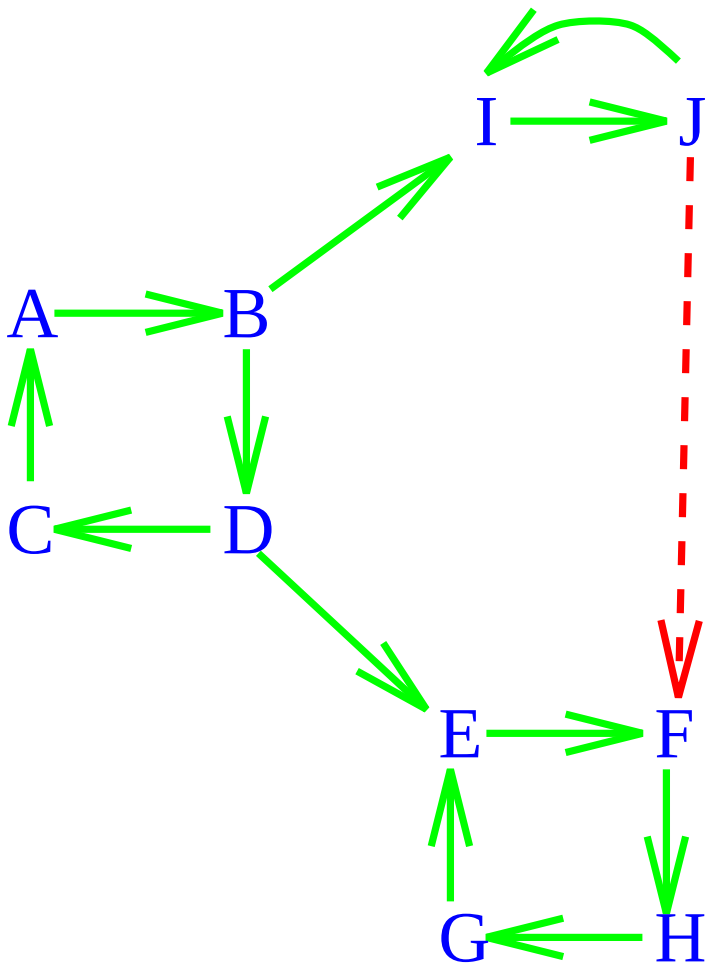
A B E F C D

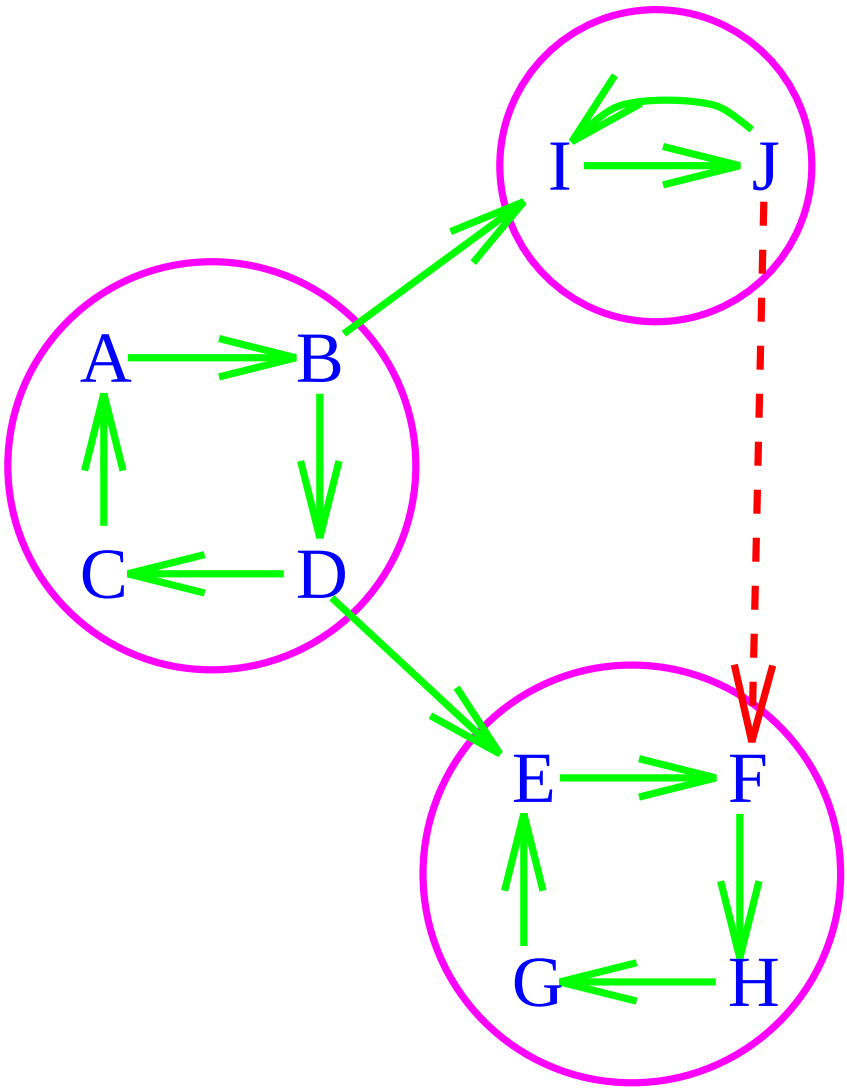


A B C D E F

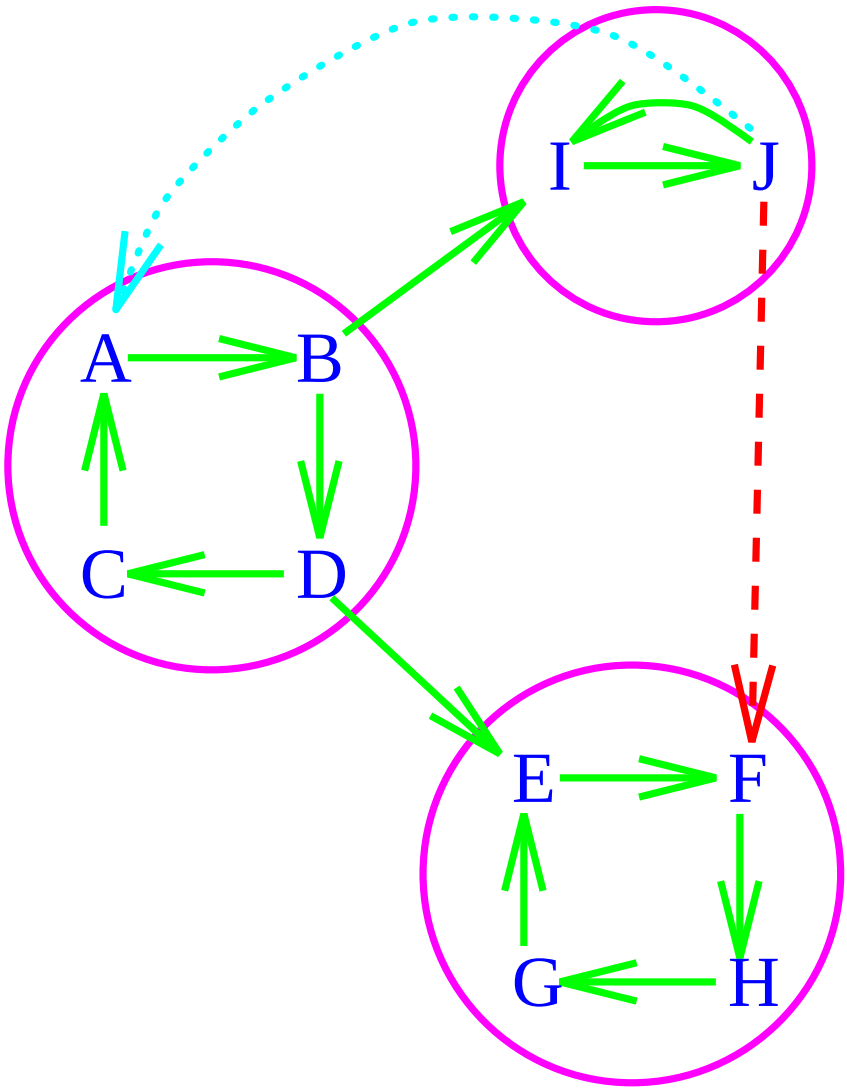


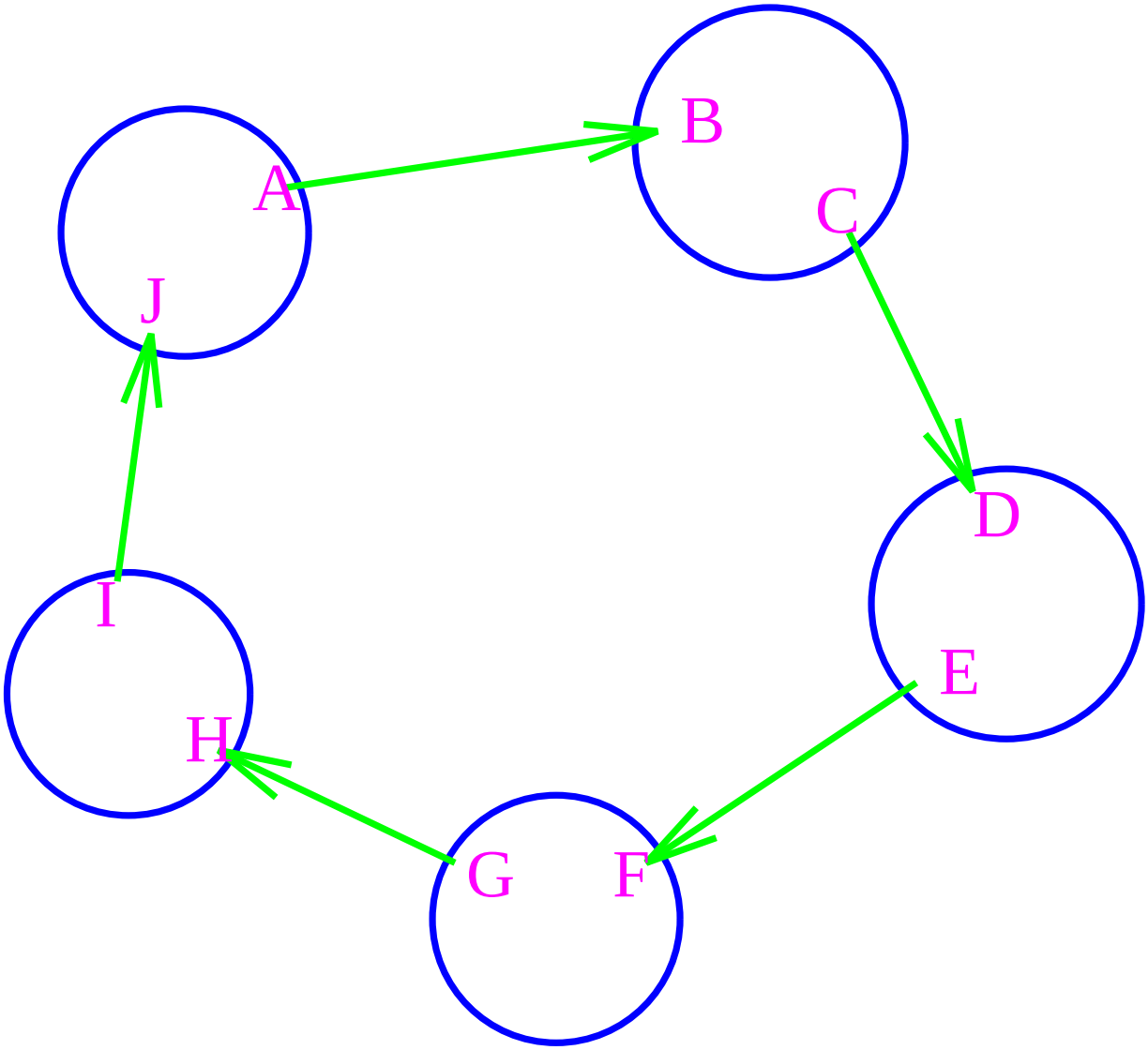
IF NO SUCH Z EXISTS, THE GRAPH HAS A CYCLE


I
J
A
B
C
D
E
F
G
H

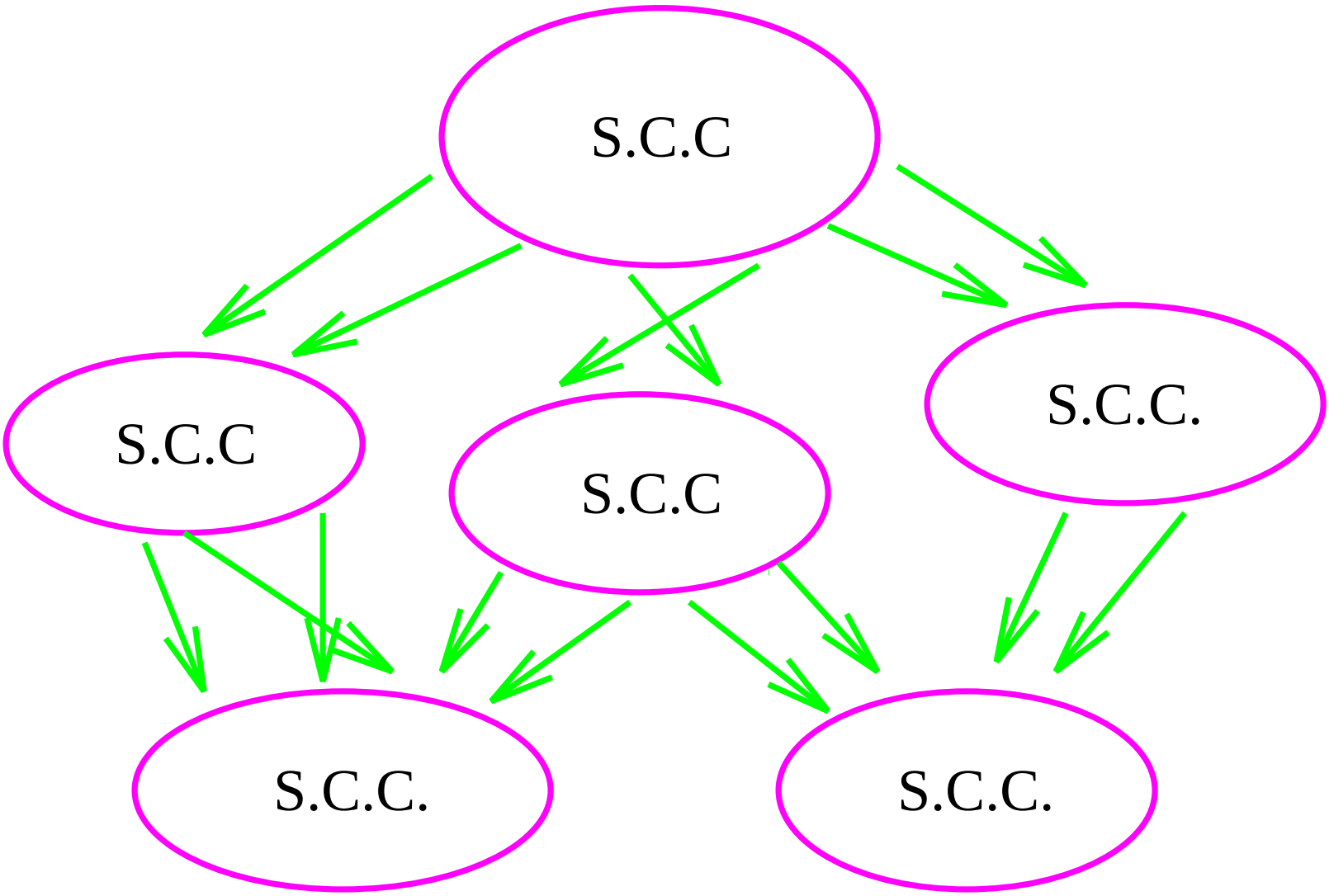


STRONGLY CONNECTED COMPONENTS

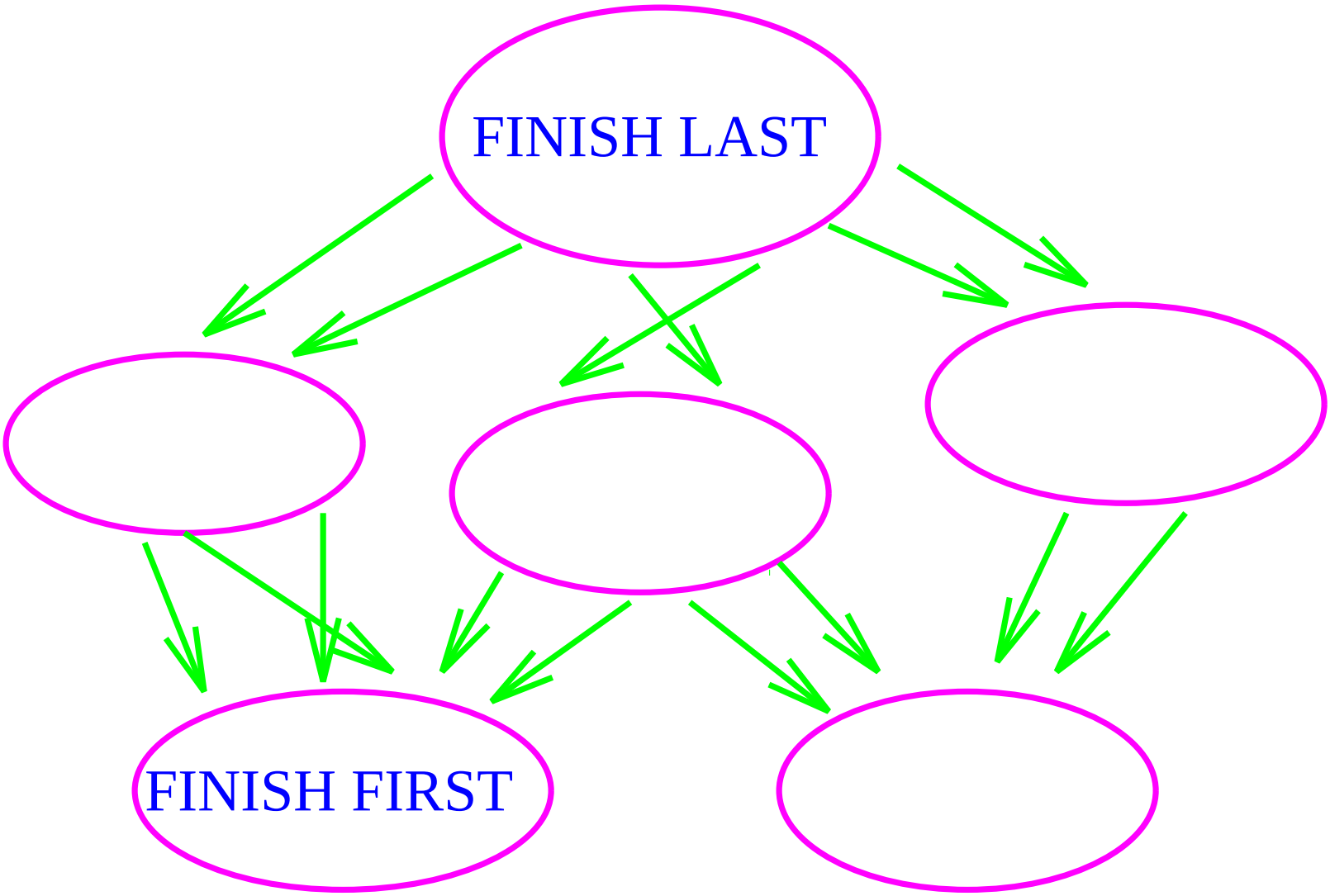

I
J
A
B
C
D
E
F
G
H

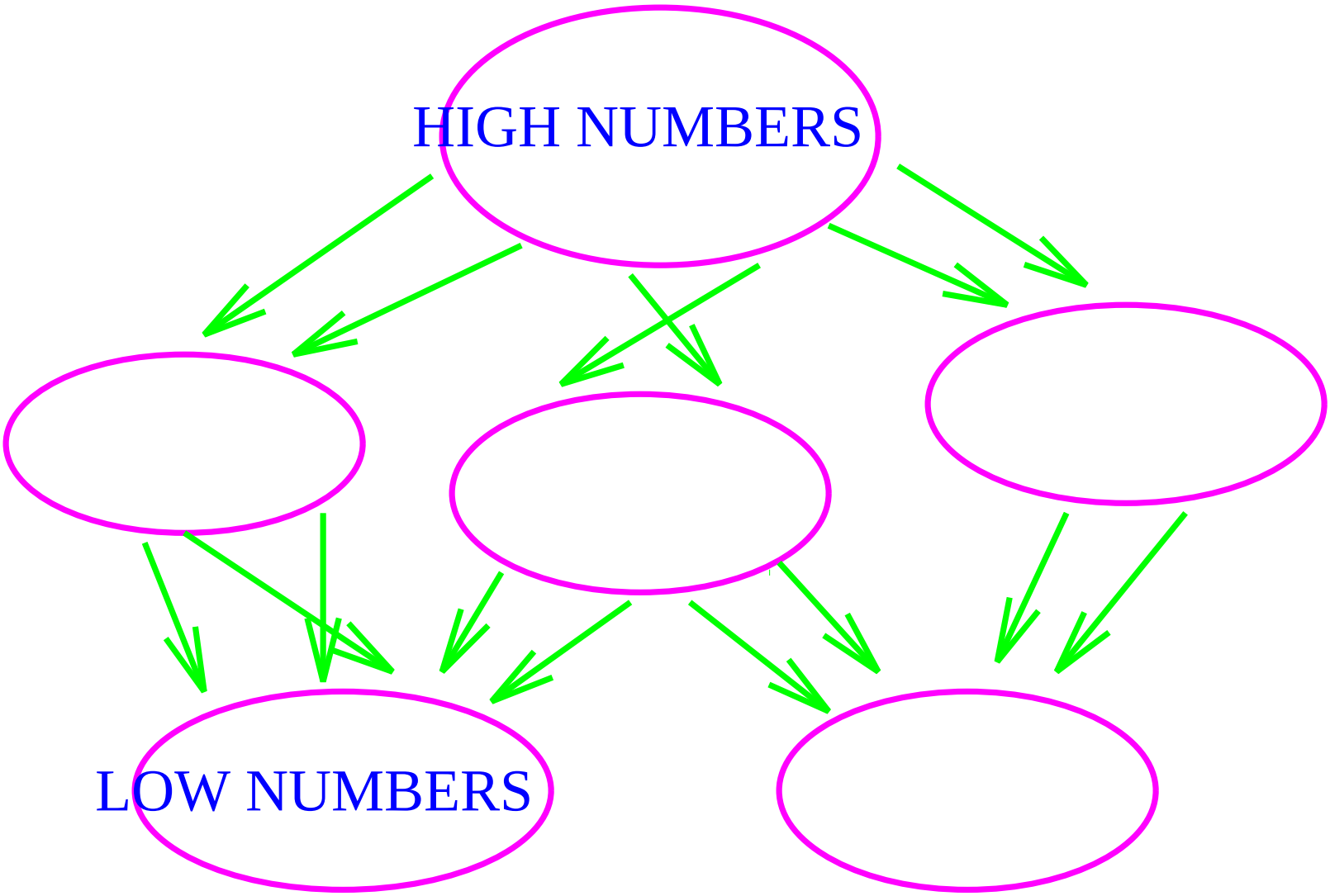


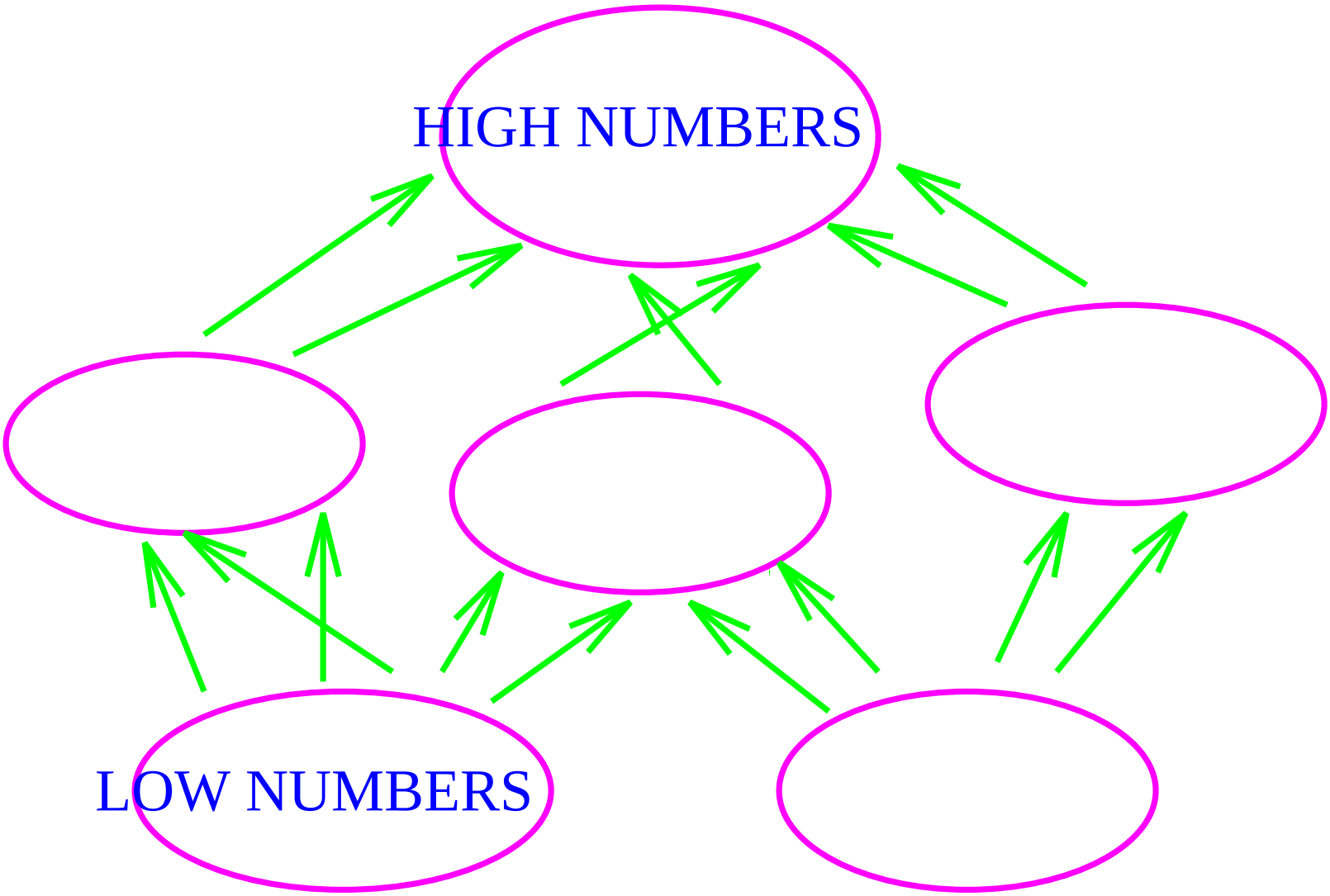
NO CYCLE POSSIBLE BETWEEN S.C.C.

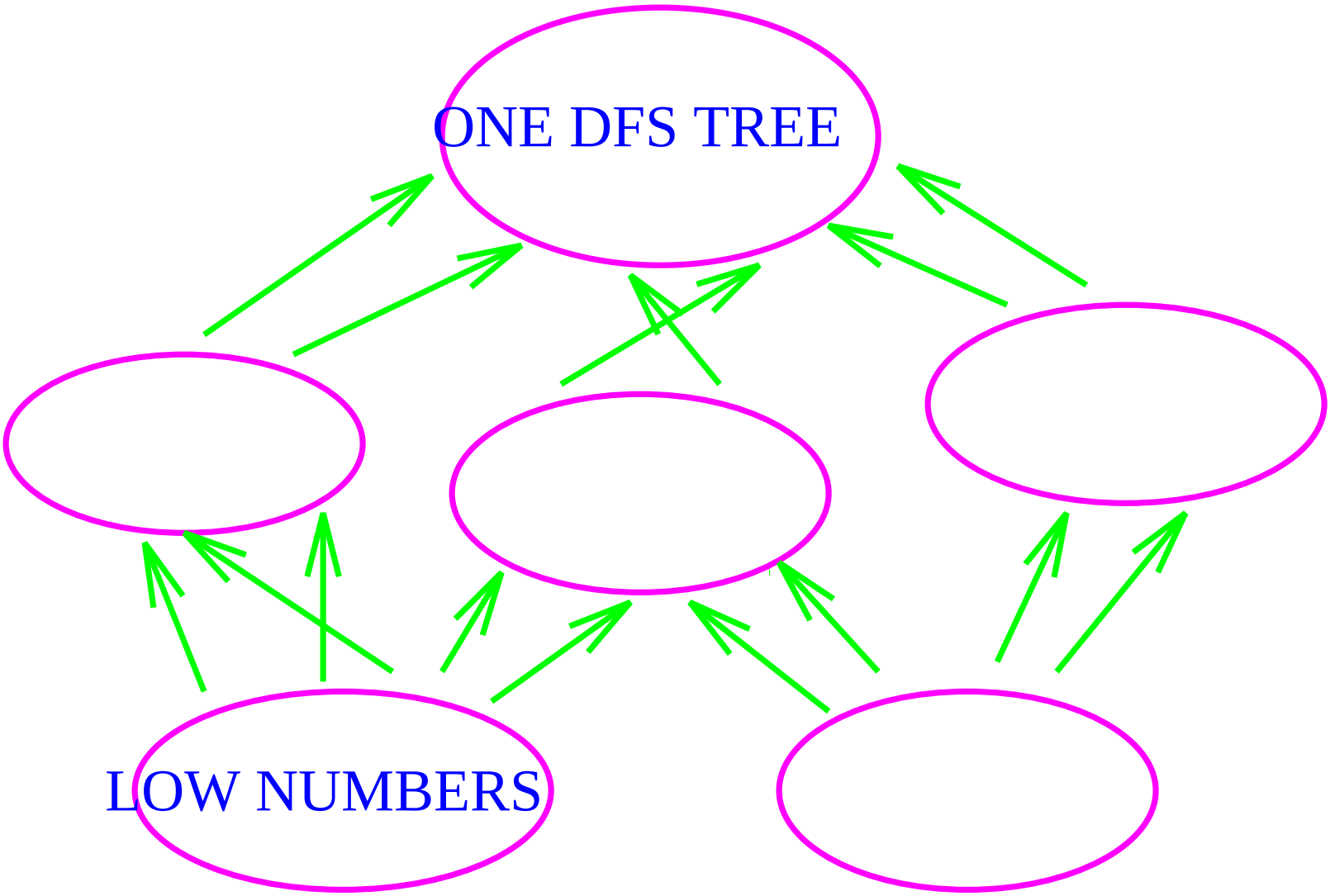


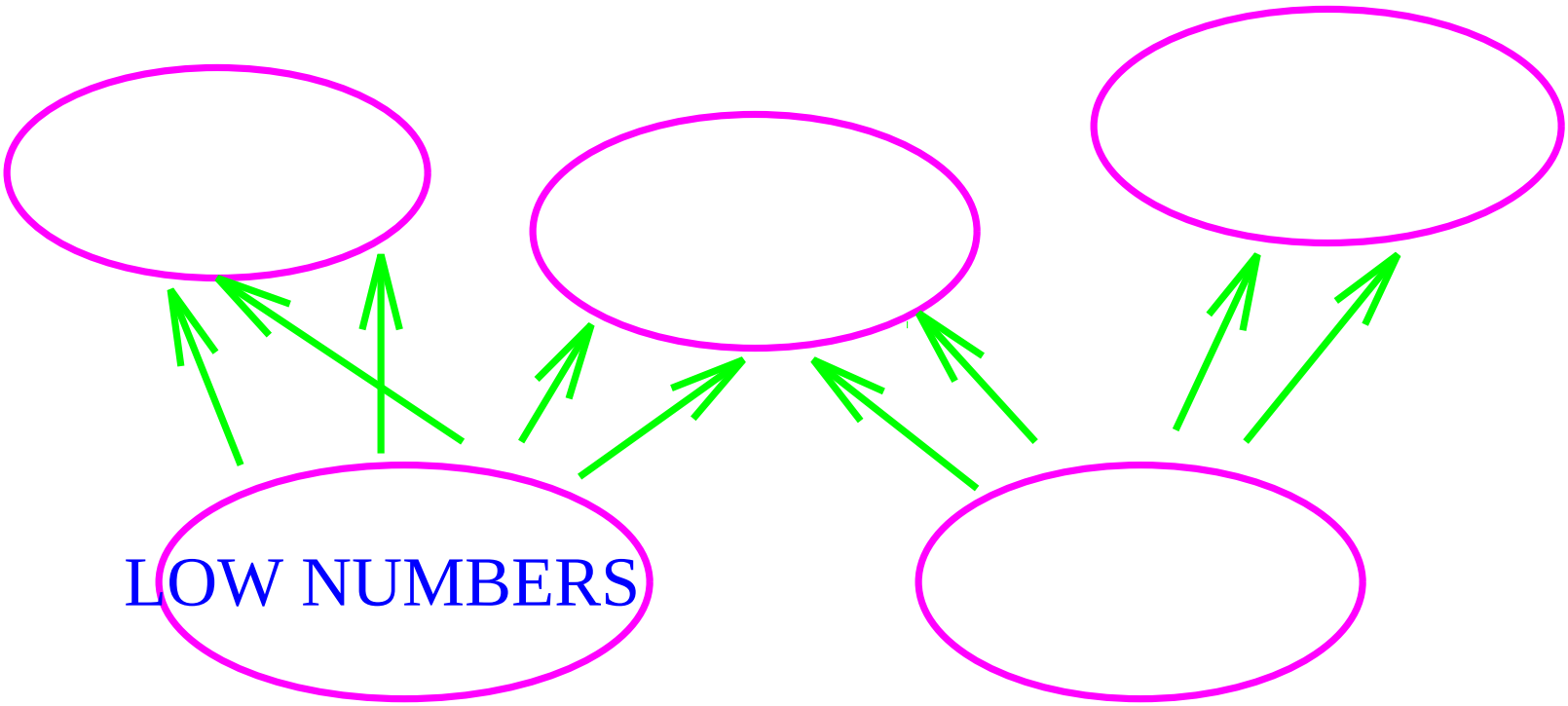
STRONGLY CONNECTED COMPONENT ALGORITHM

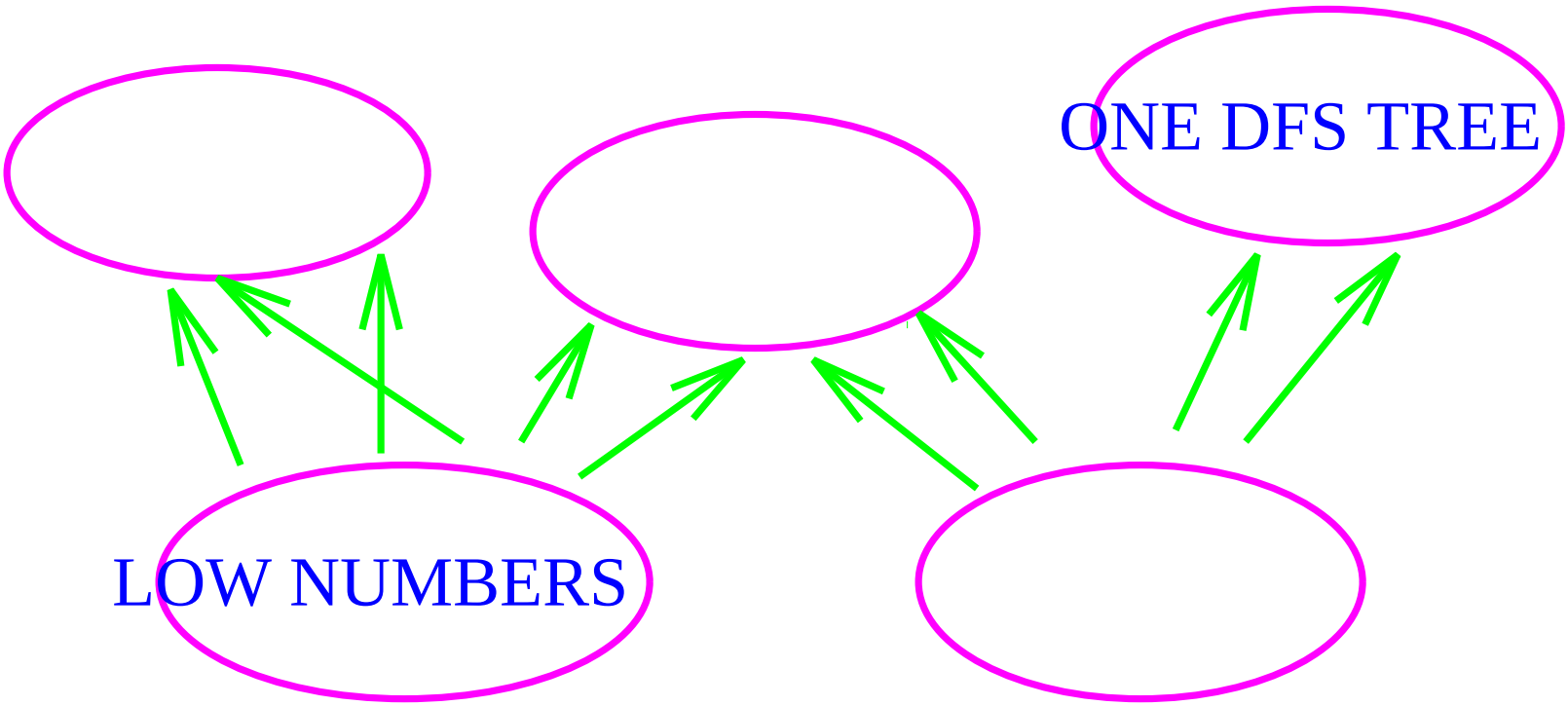

FINISH LAST
FINISH FIRST


HIGH NUMBERS
LOW NUMBERS


HIGH NUMBERS
LOW NUMBERS

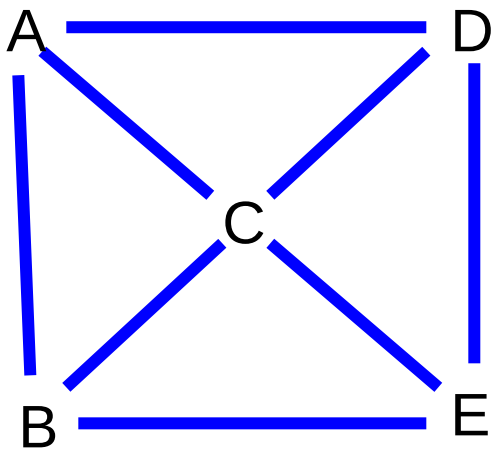

ONE DFS TREE
LOW NUMBERS

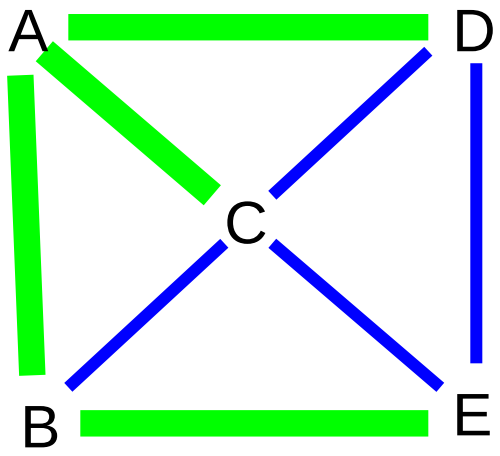

LOW NUMBERS

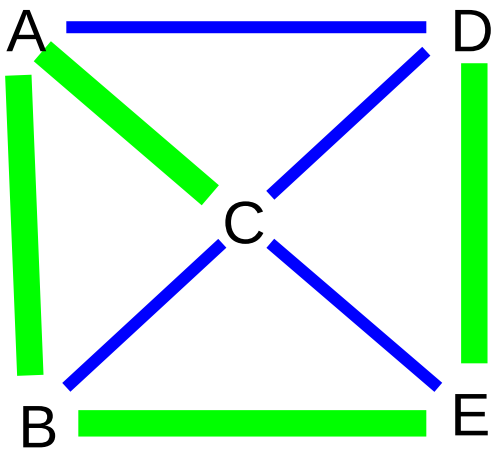

ONE DFS TREE
LOW NUMBERS

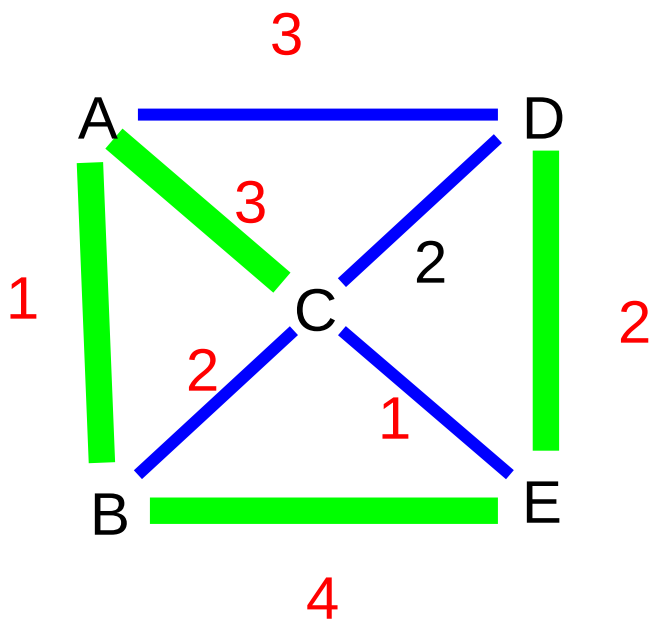
Chapter 24

Minimum Spanning Trees

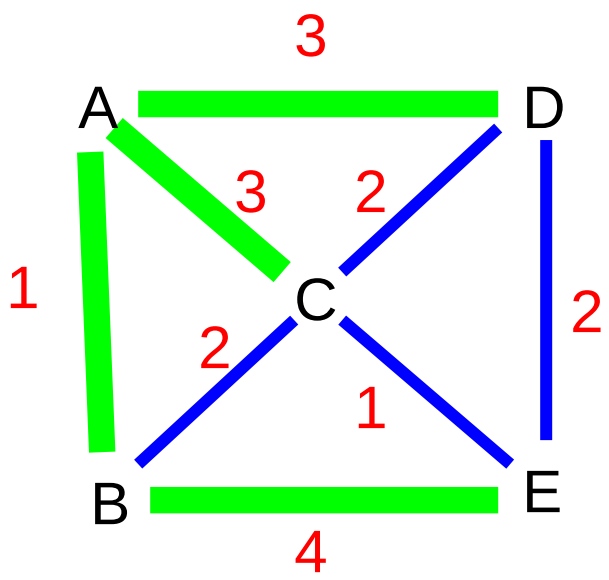

A
D
C
B
E


A
D
C
B
E

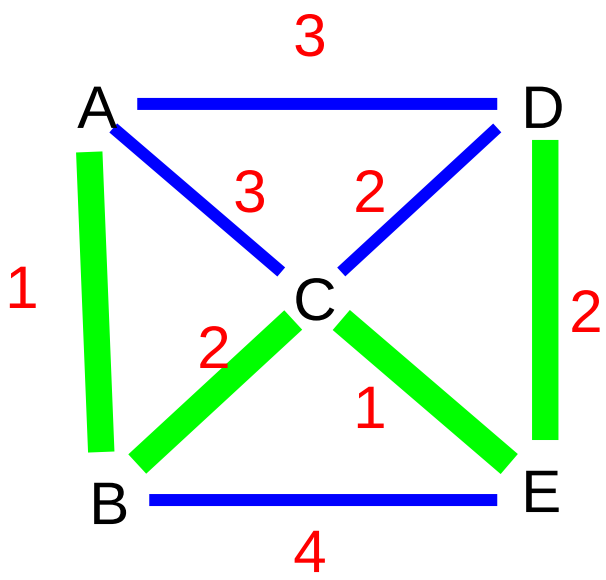

A
D
C
B
E



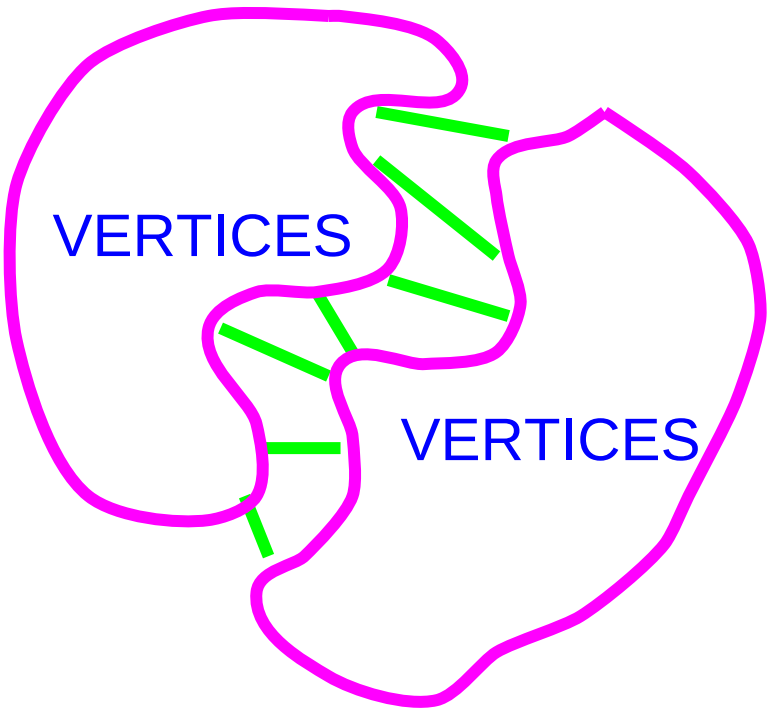
TOTAL COST = 1 + 3 + 4 + 2 = 10 .

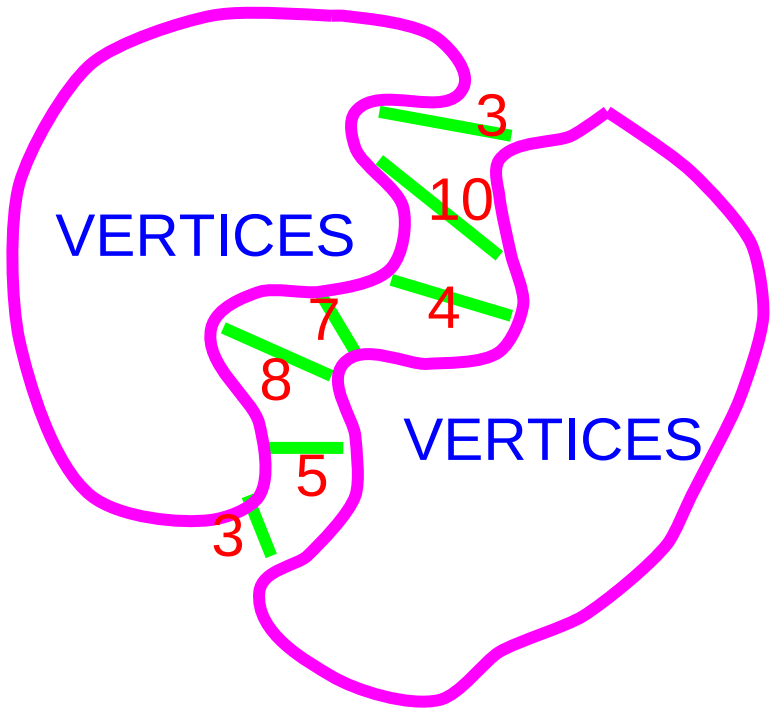


COST = 1 + 3 + 3 + 4 = 11 .

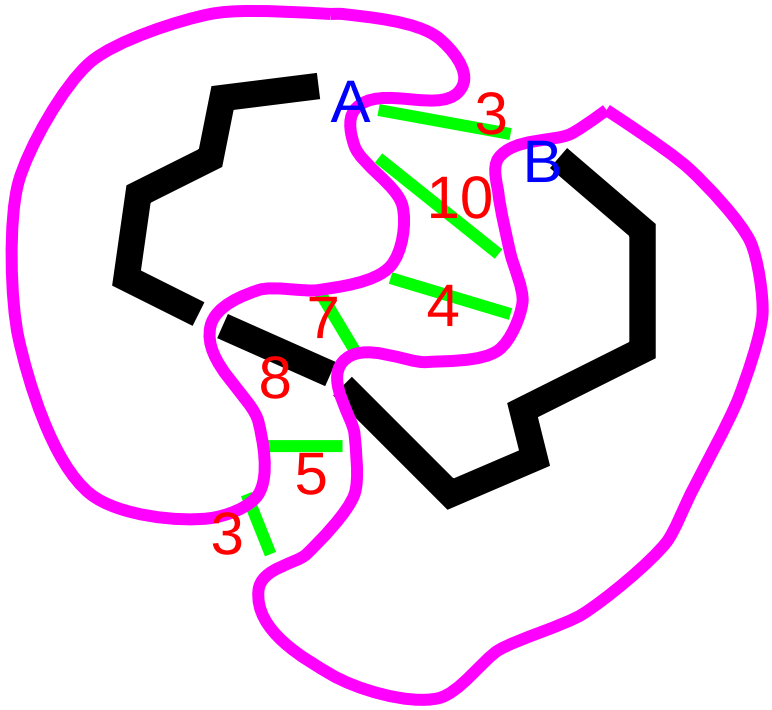


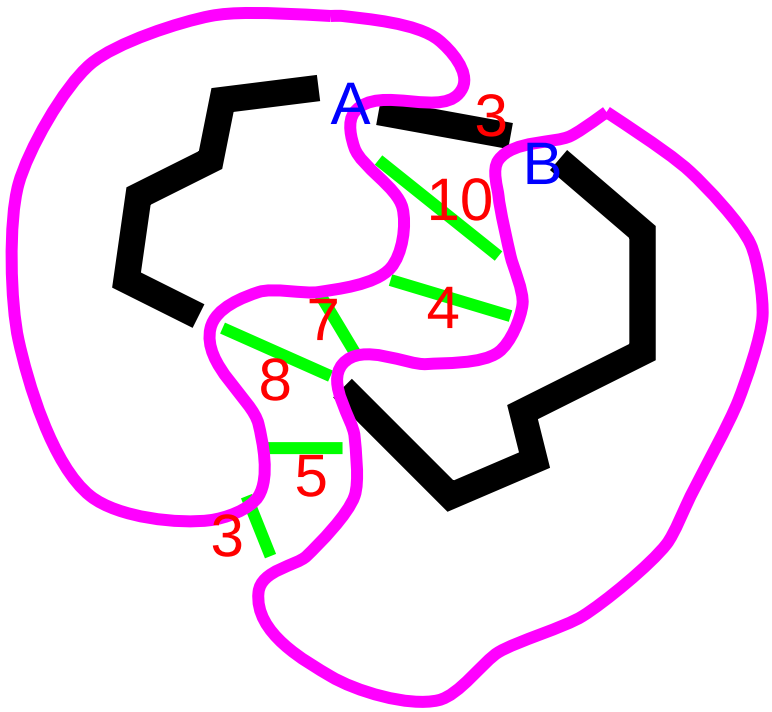
COST = 1 + 2 + 1 + 2 = 6 .


VERTICES
VERTICES

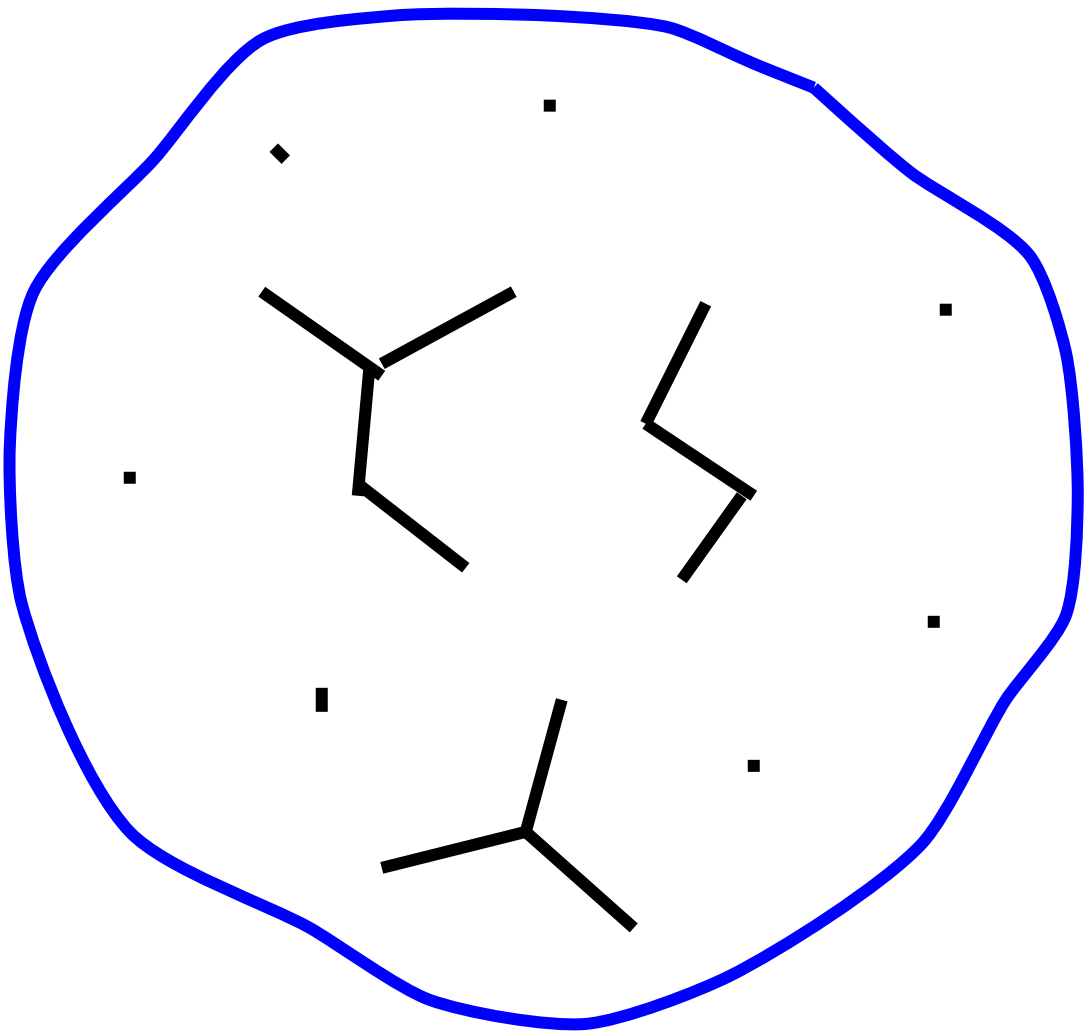


THE EDGES OF COST 3 ARE LIGHT EDGES . FOR THIS CUT.

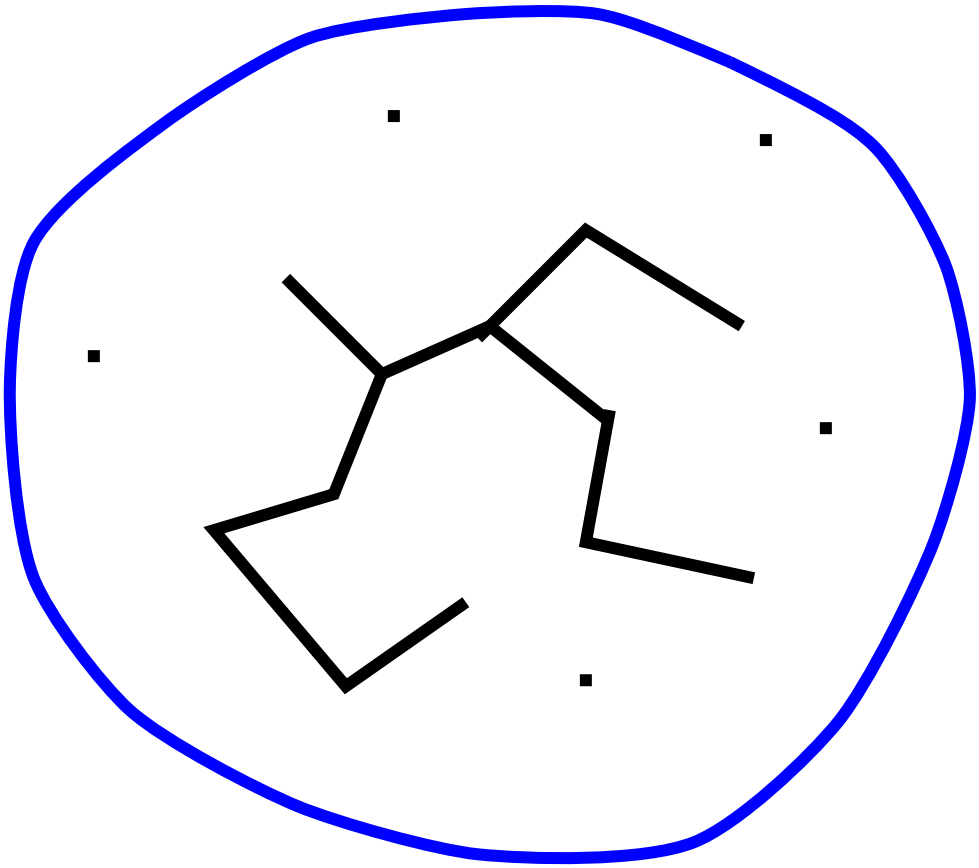

A
3
B
10
4
7
8
5
3


A
3
B
10
4
7
8
5
3

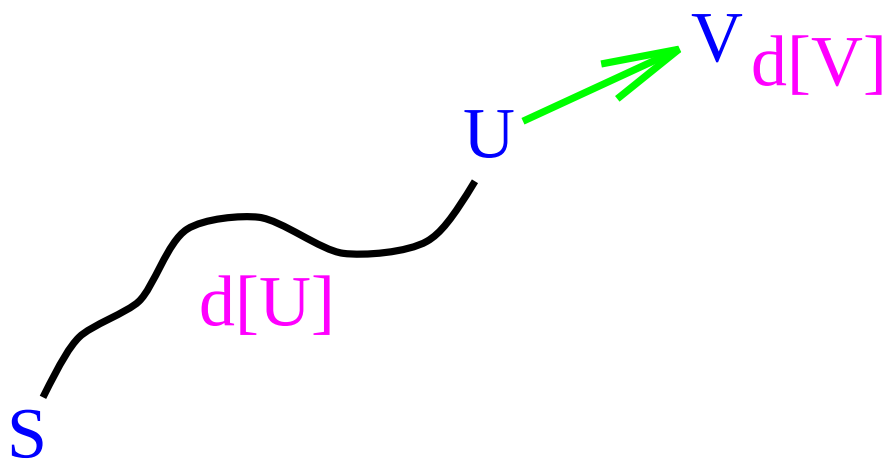
KRUSKAL'S ALGORITHM



PRIM’S ALGORITHM

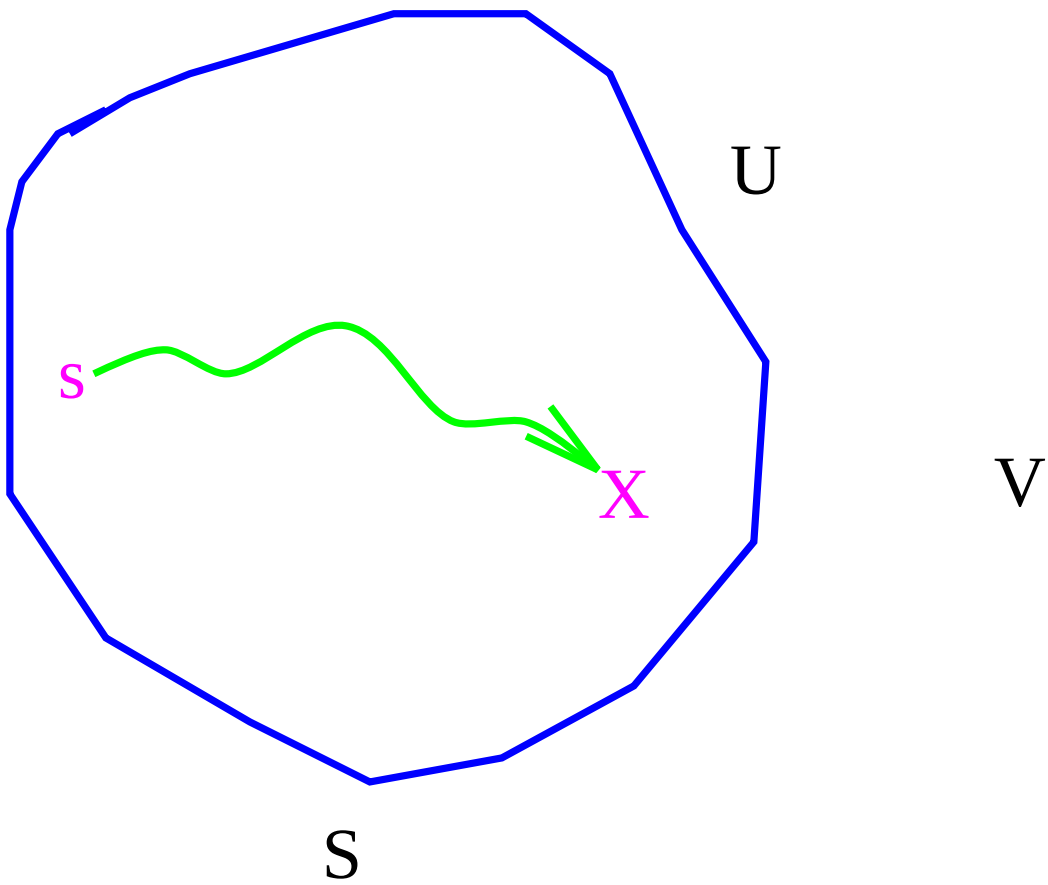


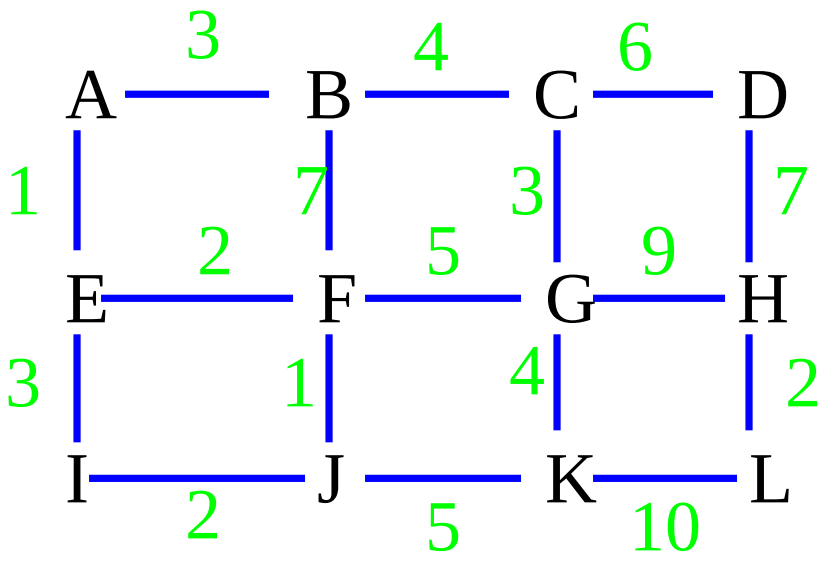
Chapter 25




```
if d[V] > d[U] + w(U,V) then
   d[V] <- d[U] + w(U,V)
   pi[V] <- U
```

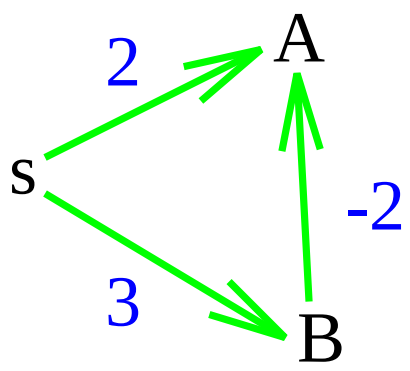
DIJKSTRA'S ALGORITHM



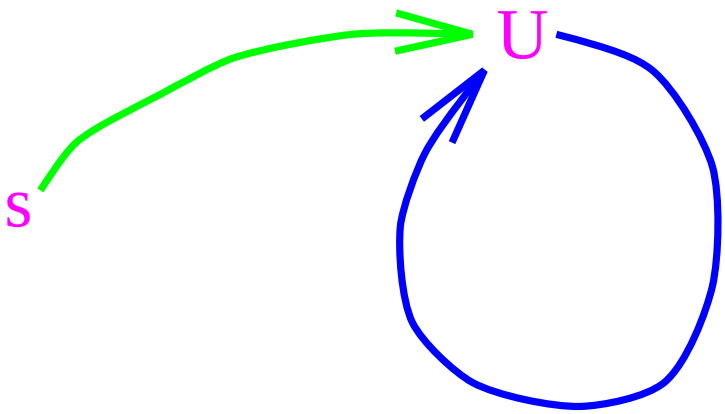


UNDIRECTED EDGES

NEGATIVE EDGES CAUSE DIJKSTRA'S ALGORITHM TO FAIL .



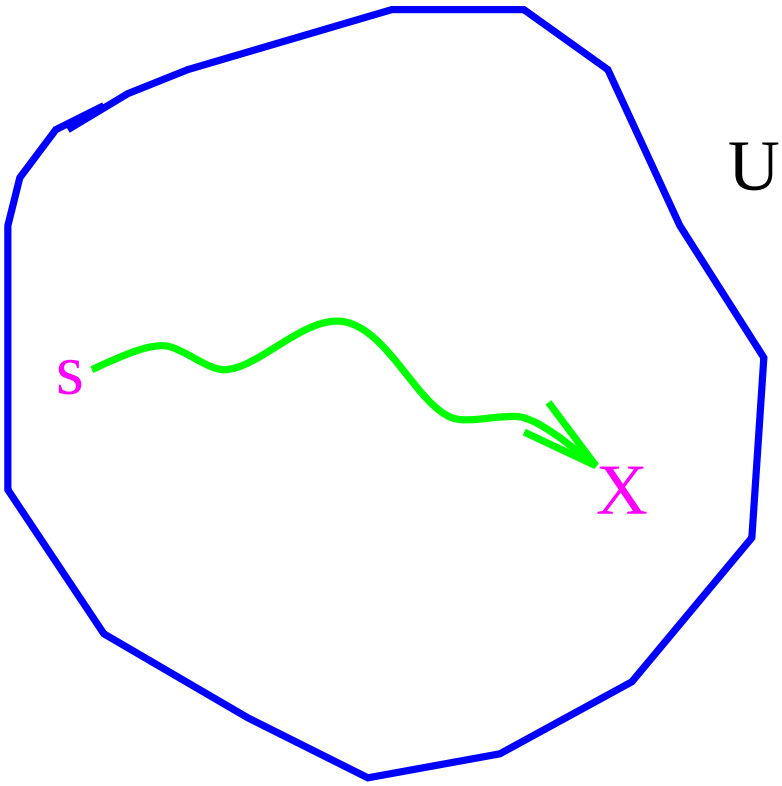
NEGATIVE CYCLES



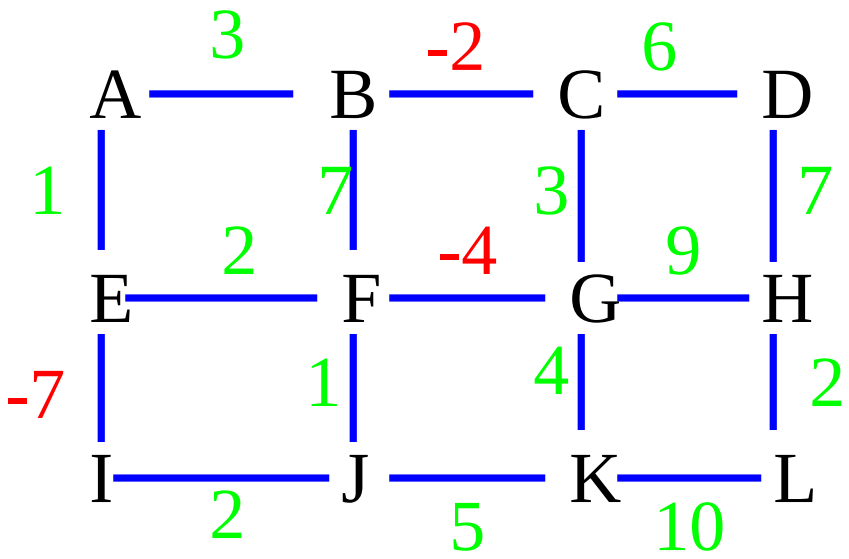
Negative Cycle

Without negative cycles, no shortest path need contain a cycle. Thus shortest paths exist, since there are only finitely many paths without cycles.

BELLMAN-FORD



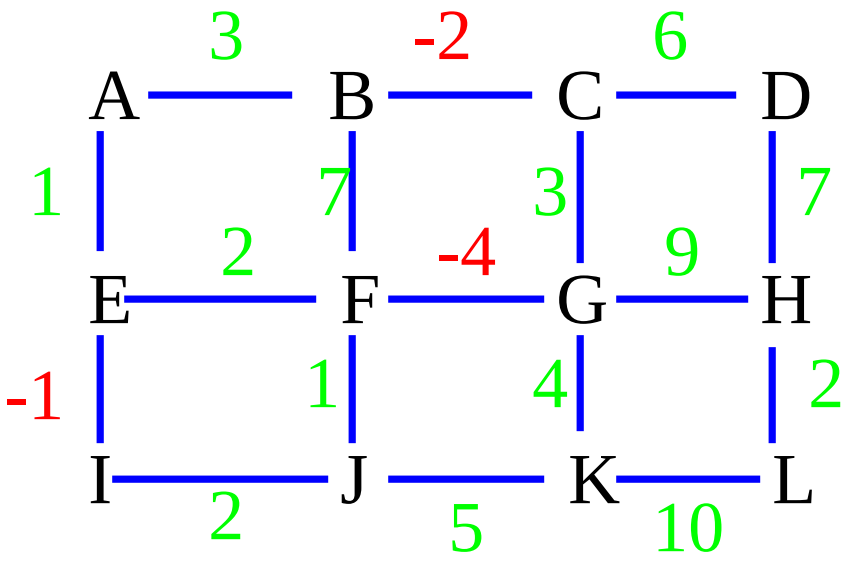
SHORTEST PATHS HAVE
FEWER THAN i EDGES



POSITIVE EDGES ARE UNDIRECTED .

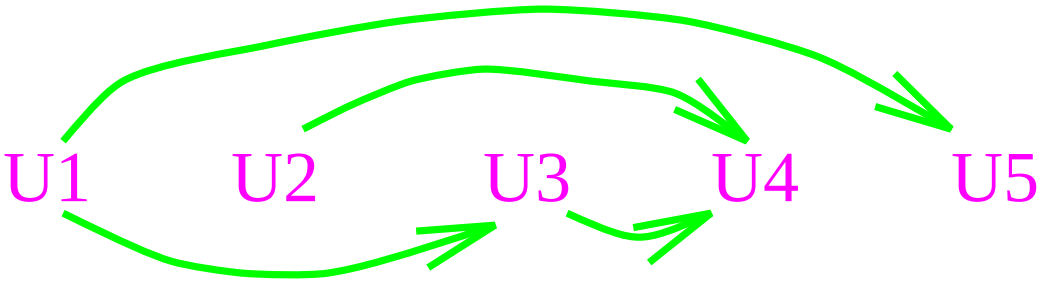
THIS HAS A NEGATIVE WEIGHT CYCLE

MODIFIED GRAPH



NO NEGATIVE WEIGHT CYCLE .

DIRECTED ACYCLIC GRAPHS

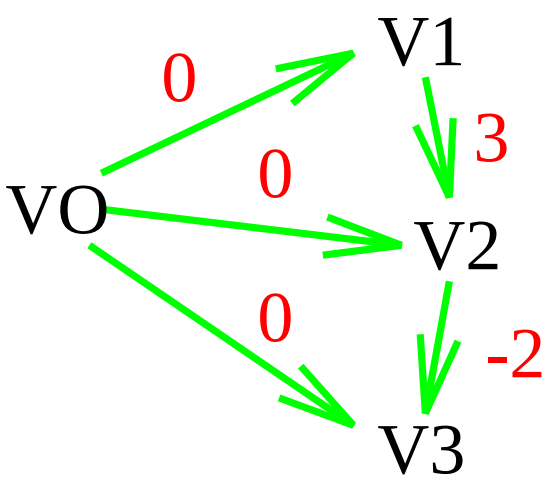


Shortest path to V goes through U which has been processed.

DIFFERENCE CONSTRAINTS

$x2 - x1 \leqslant 3$

$x3 - x2 \leqslant -2$

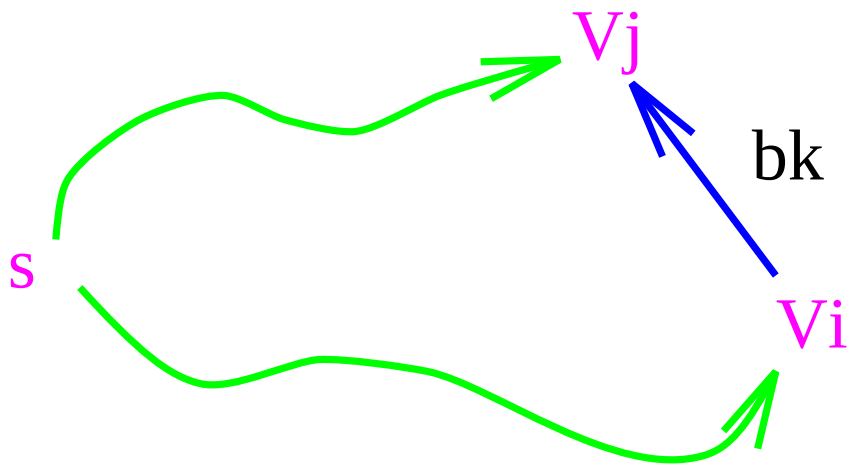


x1 = 0

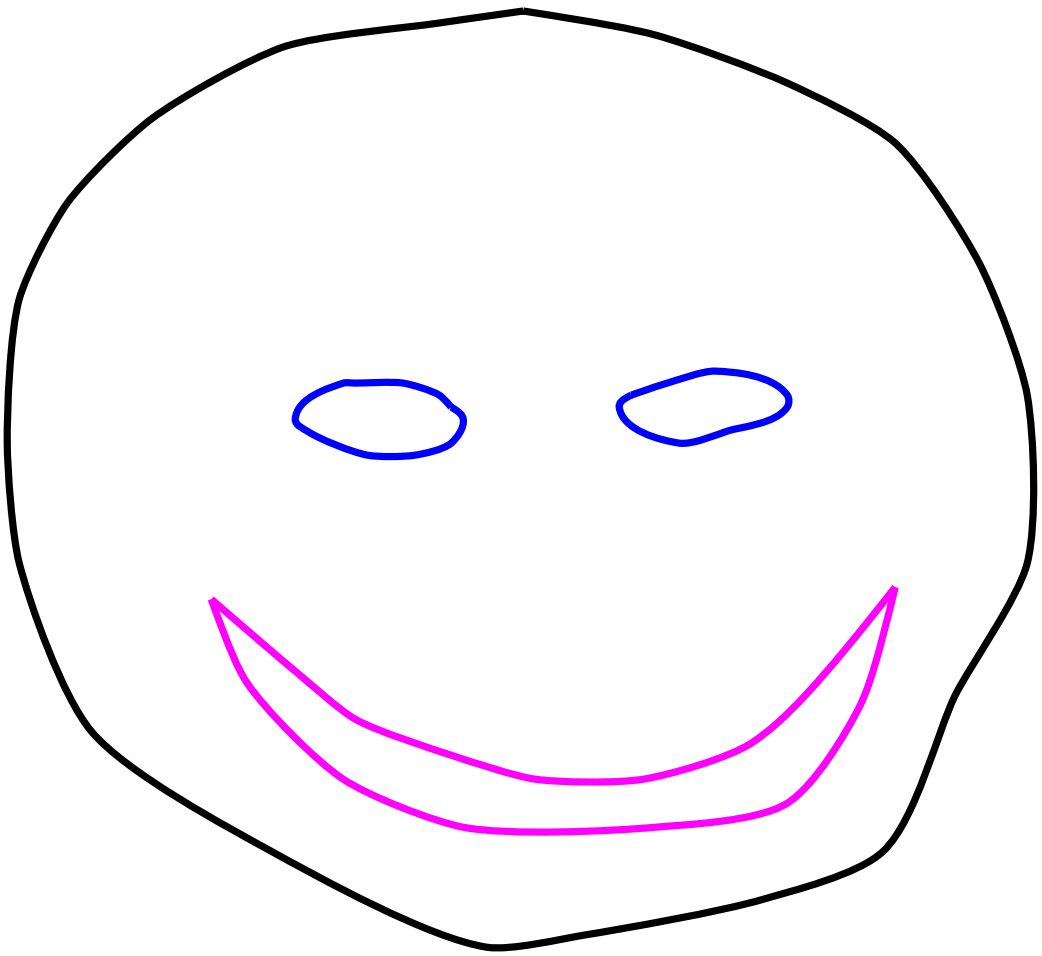
x2 = 0

x3 = -2

DIFFERENCE CONSTRAINTS .



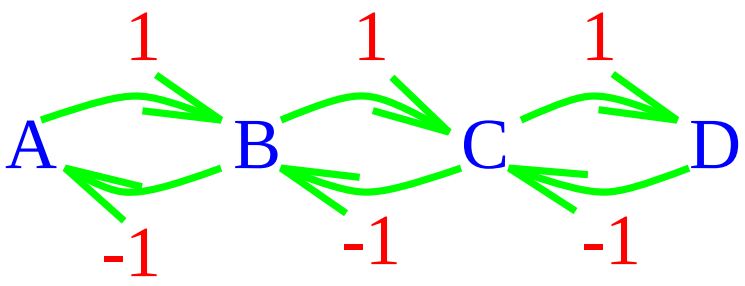
$$xj\text{-}xi \leqslant bk$$



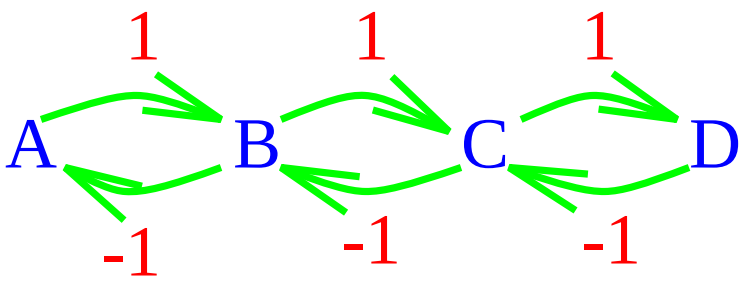
Chapter 26

All pairs Shortest Paths

DYNAMIC PROGRAMMING



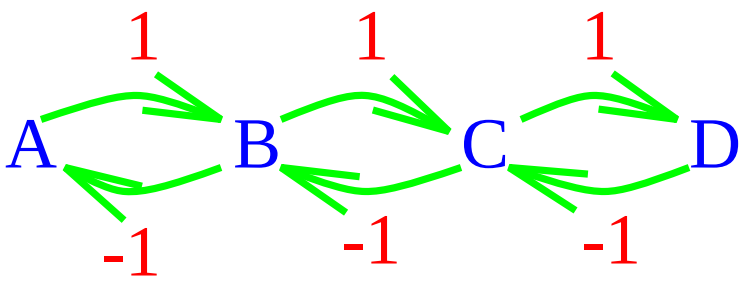
		j			
d_ij^(0)		1	2	3	4
i	1	0	oo	oo	oo
	2	oo	0	oo	oo
	3	oo	oo	0	oo
	4	oo	oo	oo	0



j

d_ij^(1)	1	2	3	4
1	0	1	oo	oo
2	-1	0	1	oo
3	oo	-1	0	1
4	oo	oo	-1	0

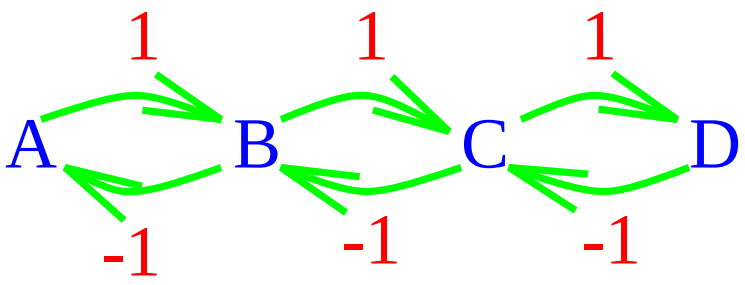
i



j

d_ij^(2)	1	2	3	4
1	0	1	2	oo
2	-1	0	1	2
3	-2	-1	0	1
4	oo	-2	-1	0

i



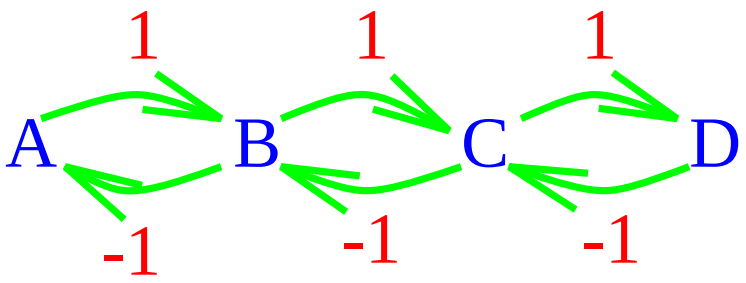
j

d_ij^(4)	1	2	3	4
1	0	1	2	3
2	-1	0	1	2
3	-2	-1	0	1
4	-3	-2	-1	0

i

FLOYD-WARSHALL ALGORITHM

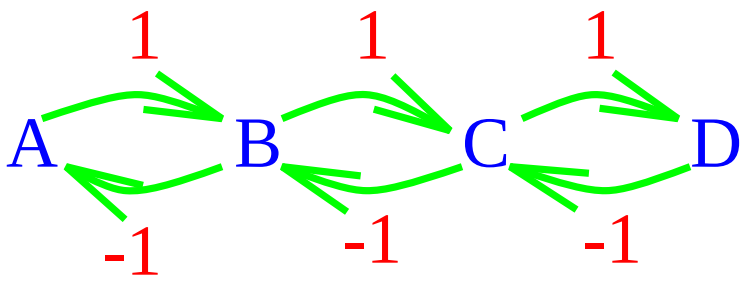
NOTE d_ij HAS A DIFFERENT MEANING .



		j			
	d_ij^(0)	1	2	3	4
	1	0	1	oo	oo
i	2	-1	0	1	oo
	3	oo	-1	0	1
	4	oo	oo	-1	0

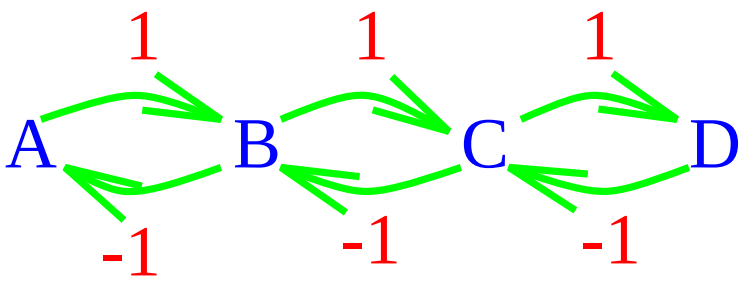
FLOYD-WARSHALL ALGORITHM

NOTE d_ij HAS A DIFFERENT MEANING .



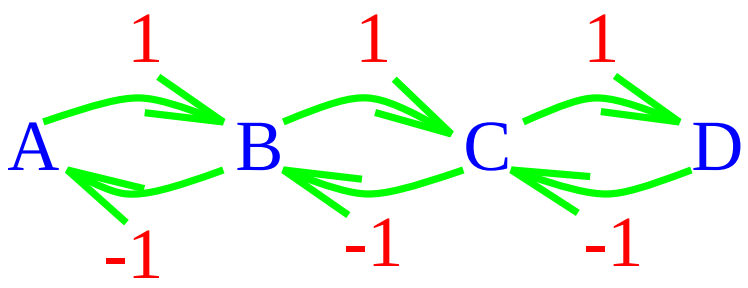
		j			
d_ij^(1)		1	2	3	4
	1	0	1	oo	oo
i	2	-1	0	1	oo
	3	oo	-1	0	1
	4	oo	oo	-1	0

FLOYD-WARSHALL ALGORITHM



		j			
	d_ij^(2)	1	2	3	4
	1	0	1	2	oo
i	2	-1	0	1	oo
	3	-2	-1	0	1
	4	oo	oo	-1	0

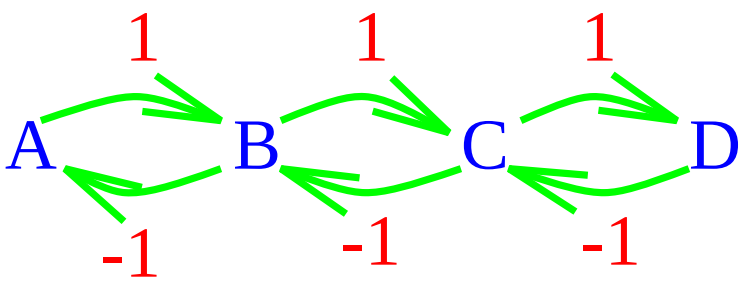
FLOYD-WARSHALL ALGORITHM



d_ij^(3)	j = 1	2	3	4
1	0	1	2	3
2	-1	0	1	2
3	-2	-1	0	1
4	-3	-2	-1	0

i

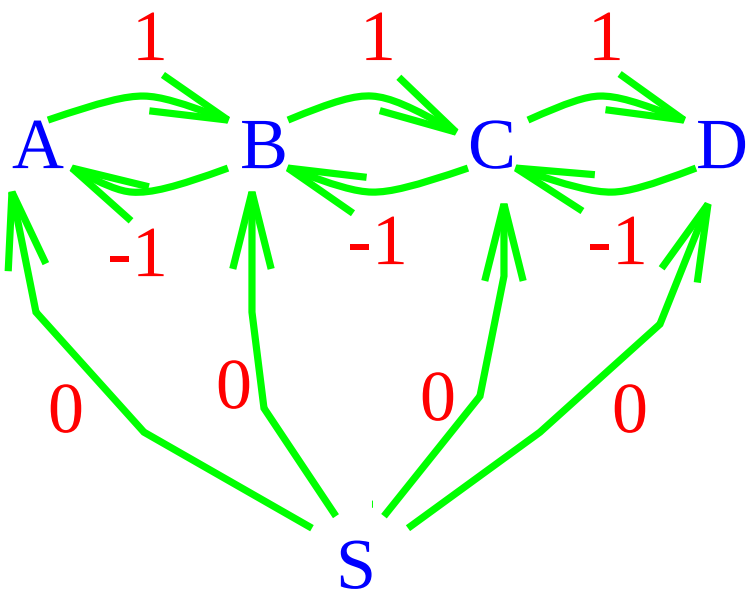
FLOYD-WARSHALL ALGORITHM



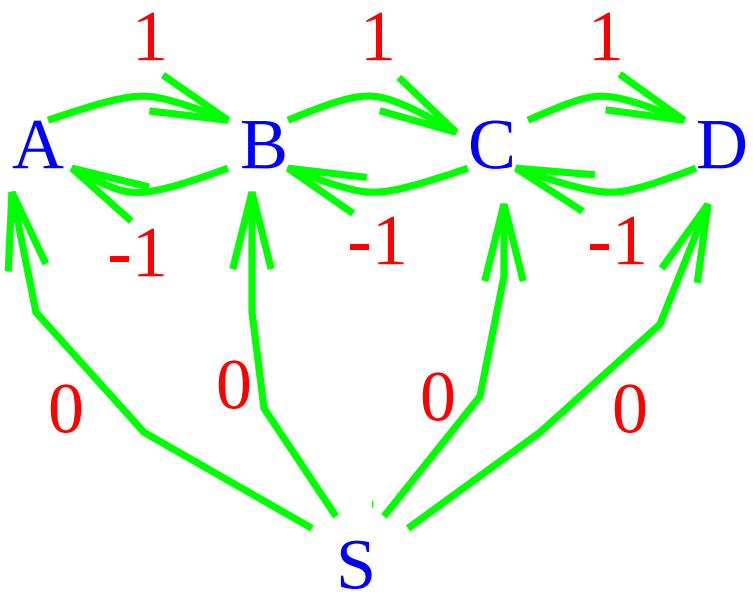
d_ij^(4)	j = 1	2	3	4
1	0	1	2	3
2	-1	0	1	2
3	-2	-1	0	1
4	-3	-2	-1	0

(i)

JOHNSON'S ALGORITHM

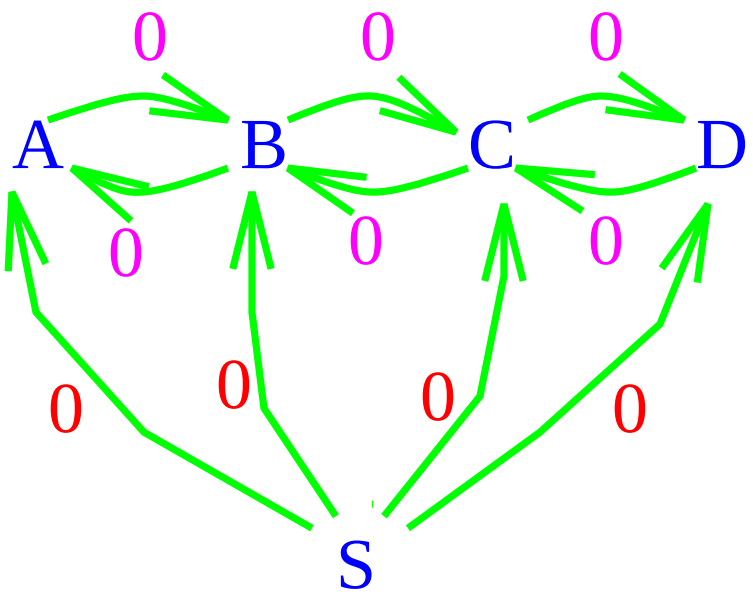


JOHNSON'S ALGORITHM



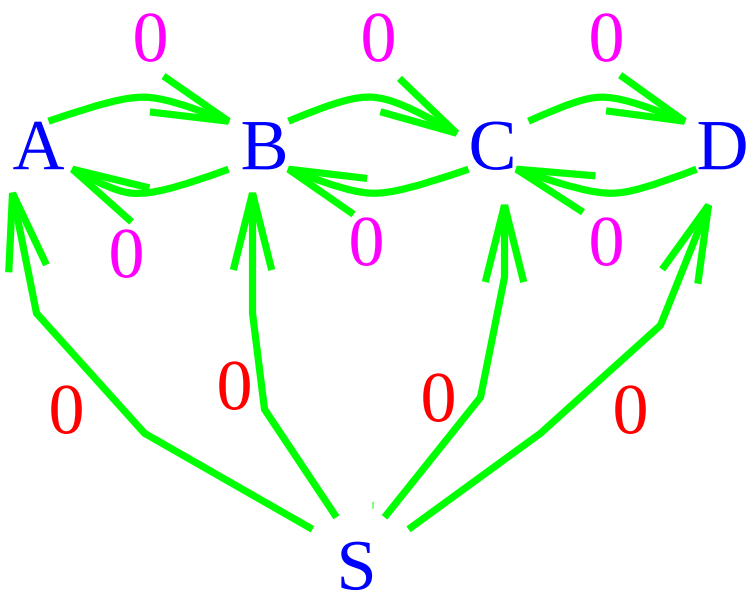
h(A) = -3
h(B) = -2
h(C) = -1
h(D) = 0

JOHNSON'S ALGORITHM



h(A) = -3
h(B) = -2
h(C) = -1
h(D) = 0

JOHNSON'S ALGORITHM



h(A) = -3
h(B) = -2
h(C) = -1
h(D) = 0

delta'(i,j) = 0 all i,j

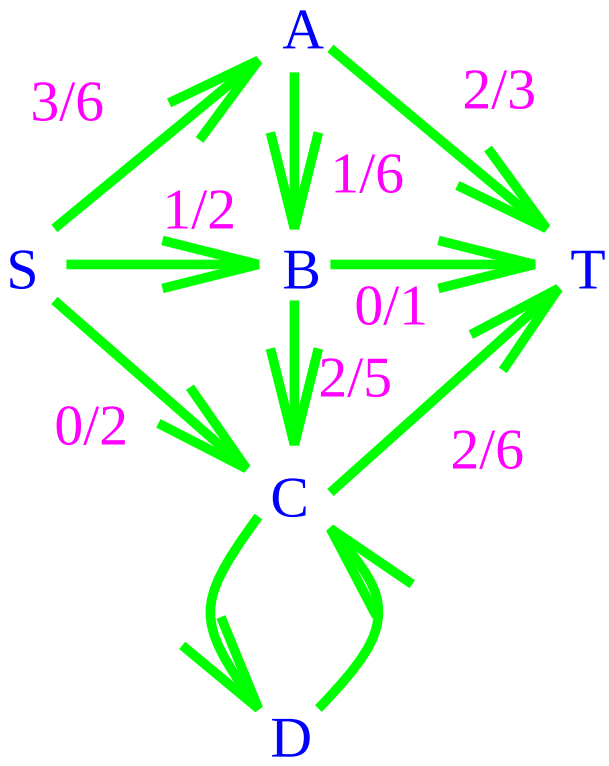
delta(C,B) = 0 + h(B) - h(C)
= -1

COMPUTE BY DIJKSTRA'S ALGORITHM

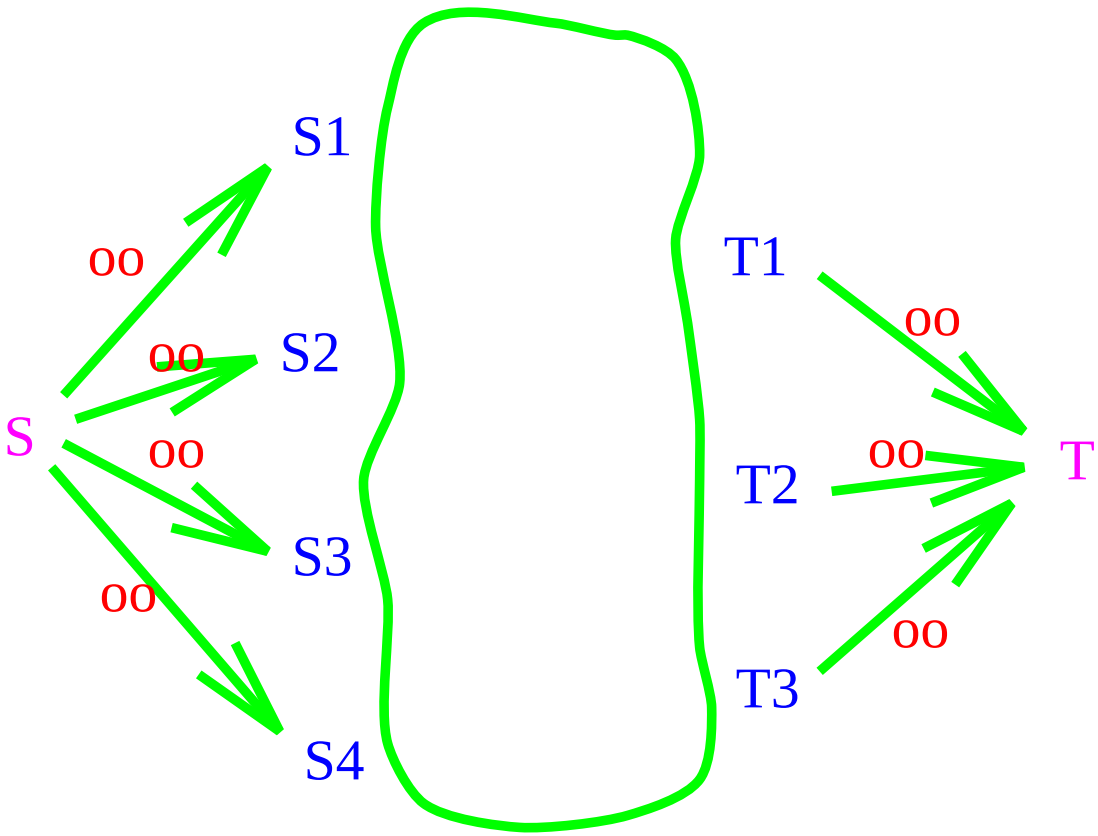
Chapter 27

Maximum Flow

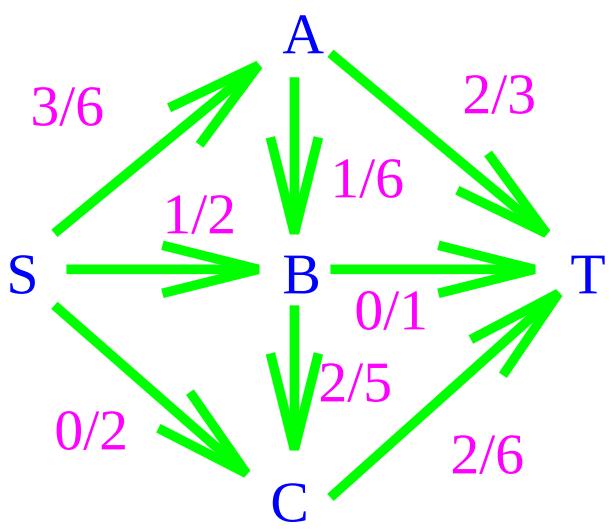
EXAMPLE FLOW

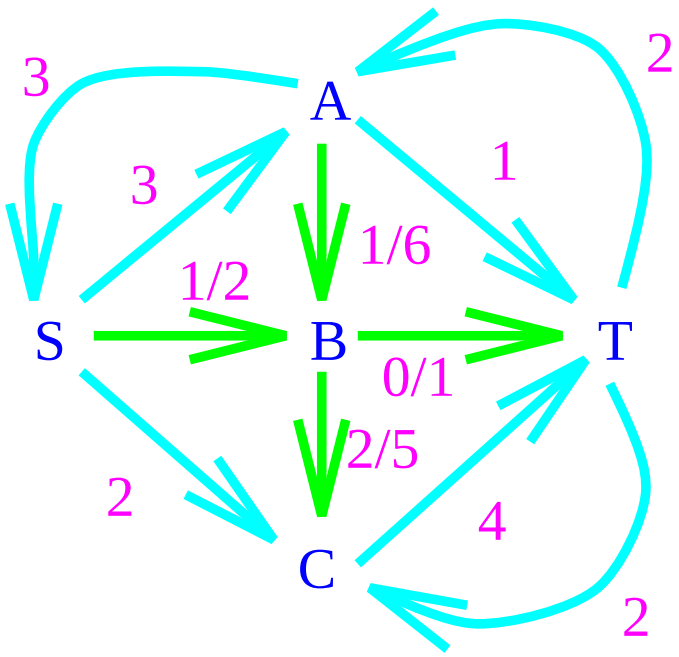


MULTIPLE SOURCES AND SINKS



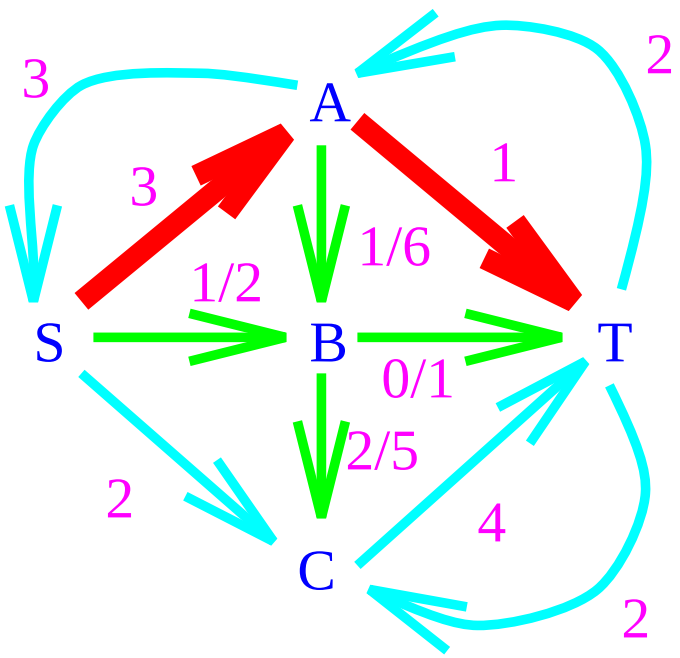
EXAMPLE FLOW



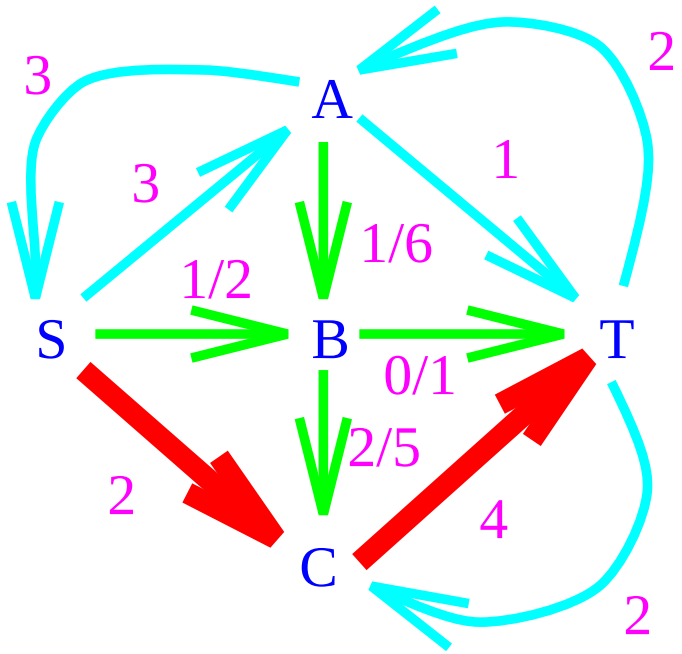


SOME CAPACITIES IN BOTH DIRECTIONS

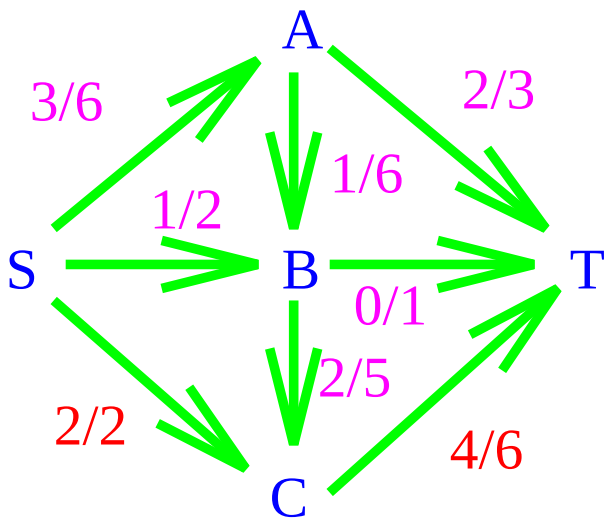
AUGMENTING PATH OF CAPACITY 1



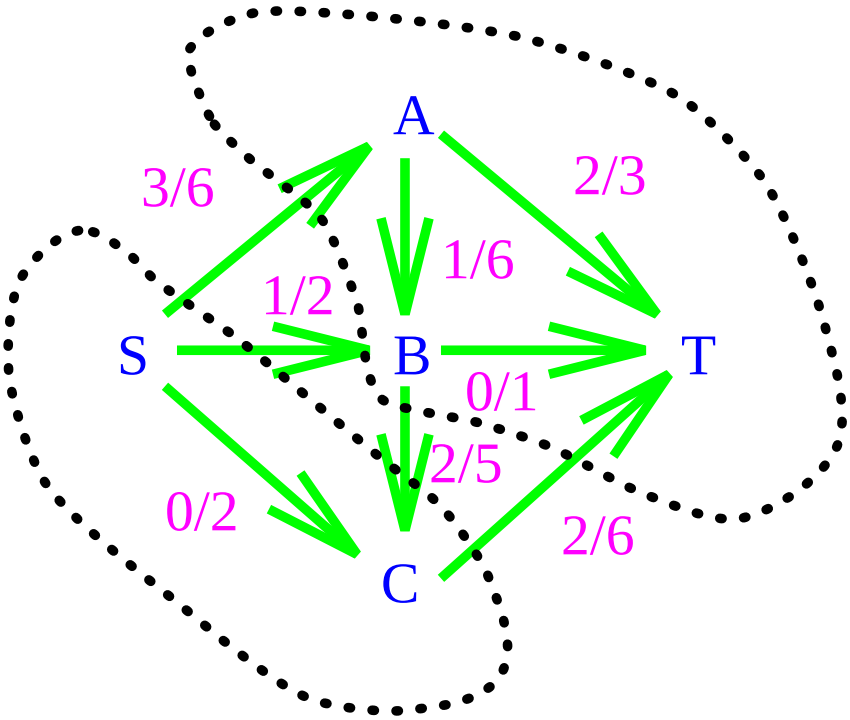
AUGMENTING PATH OF CAPACITY 2



INCREASED FLOW OBTAINED FROM AUGMENTING PATH



SAMPLE CUT

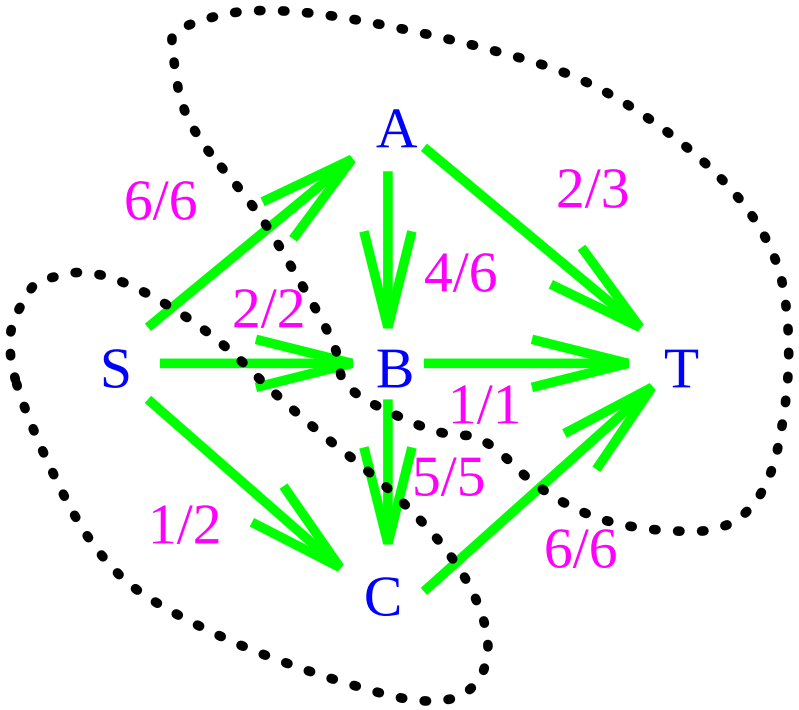


CAPACITY = 14

FLOW = 4

LEMMA 27.5 HOLDS

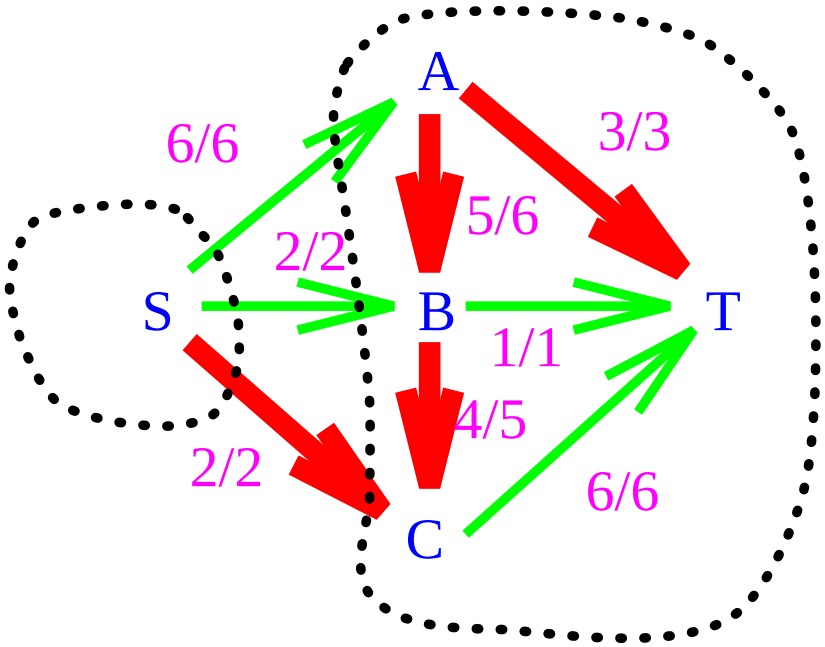
A NON MAXIMUM FLOW AND ITS NON CUT



CAPACITY OF CUT IS 14

VALUE OF FLOW IS 9

A MAXIMUM FLOW AND ITS CUT

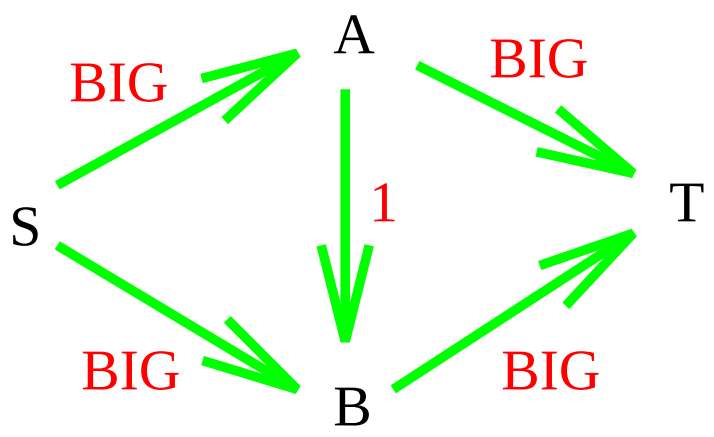


CAPACITY OF CUT IS 10

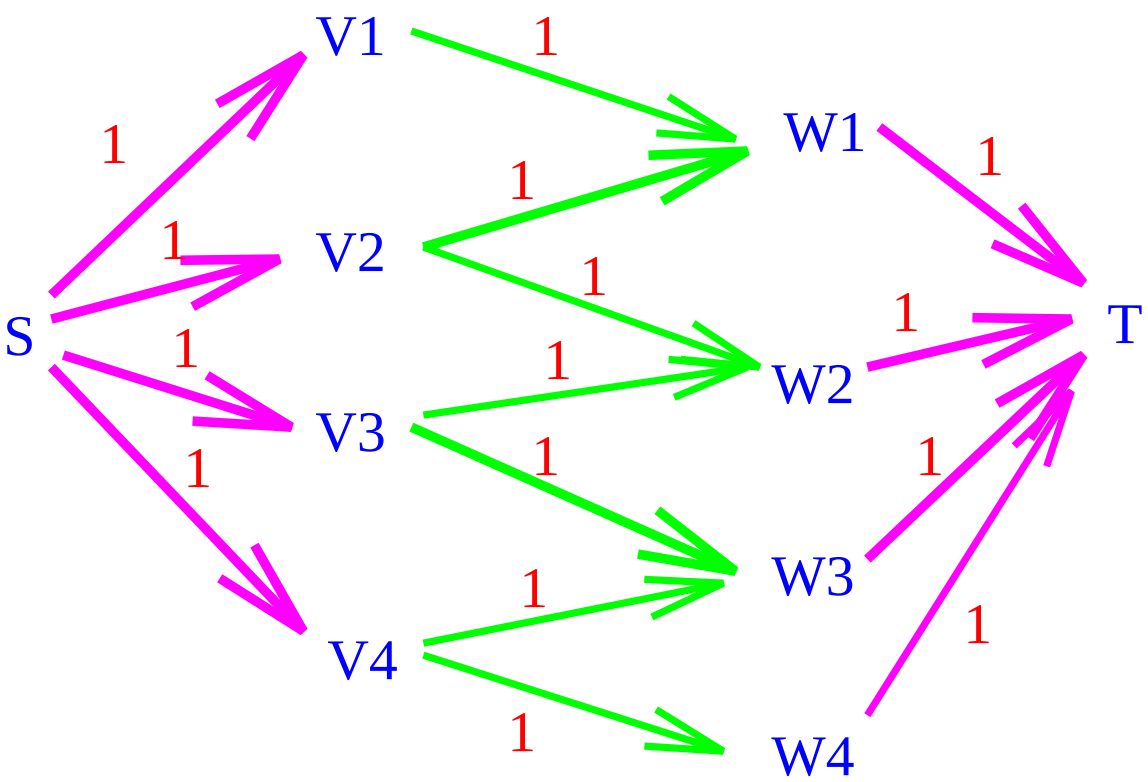
VALUE OF FLOW IS 10

NO MORE AUGMENTING PATHS EXIST

BAD EXAMPLE FOR FORD-FULKERSON

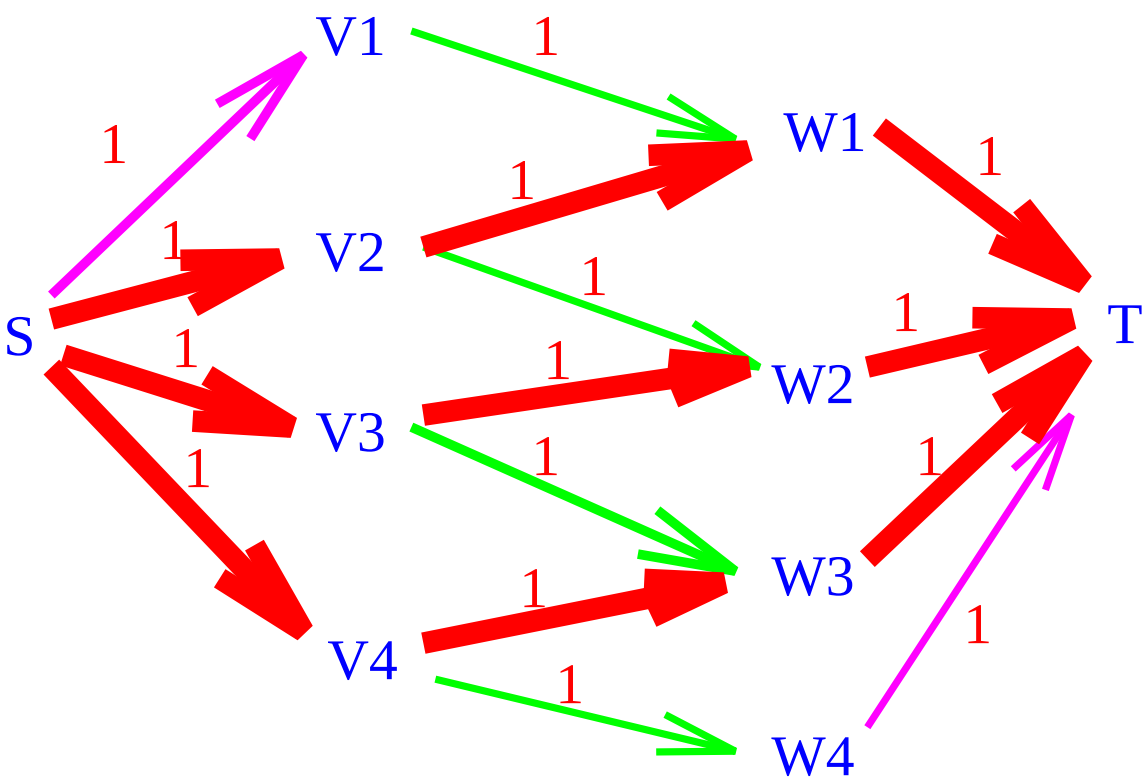


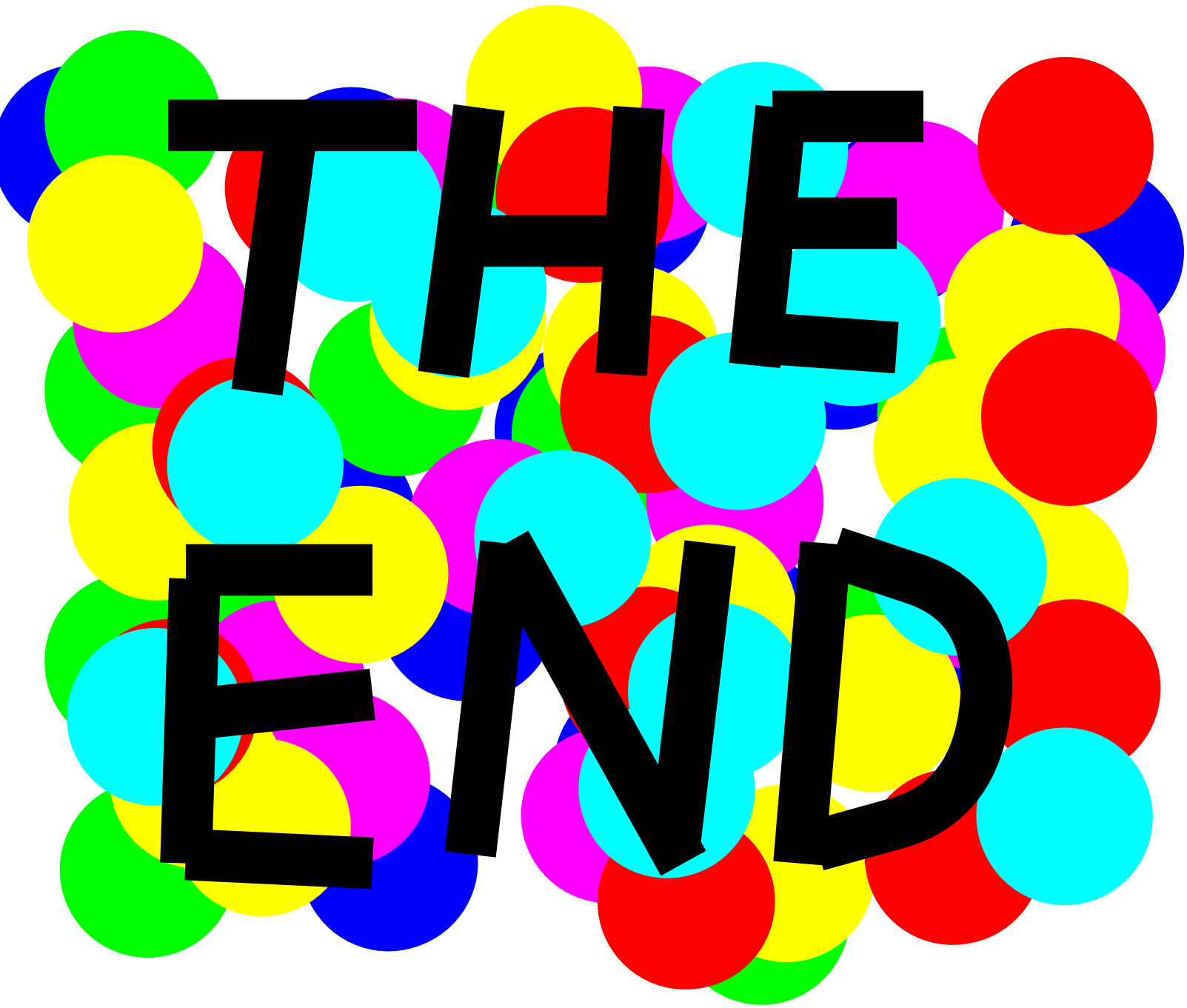
APPLICATION OF FLOW TO MAXIMUM MATCHING



APPLICATION OF FLOW TO MAXIMUM MATCHING

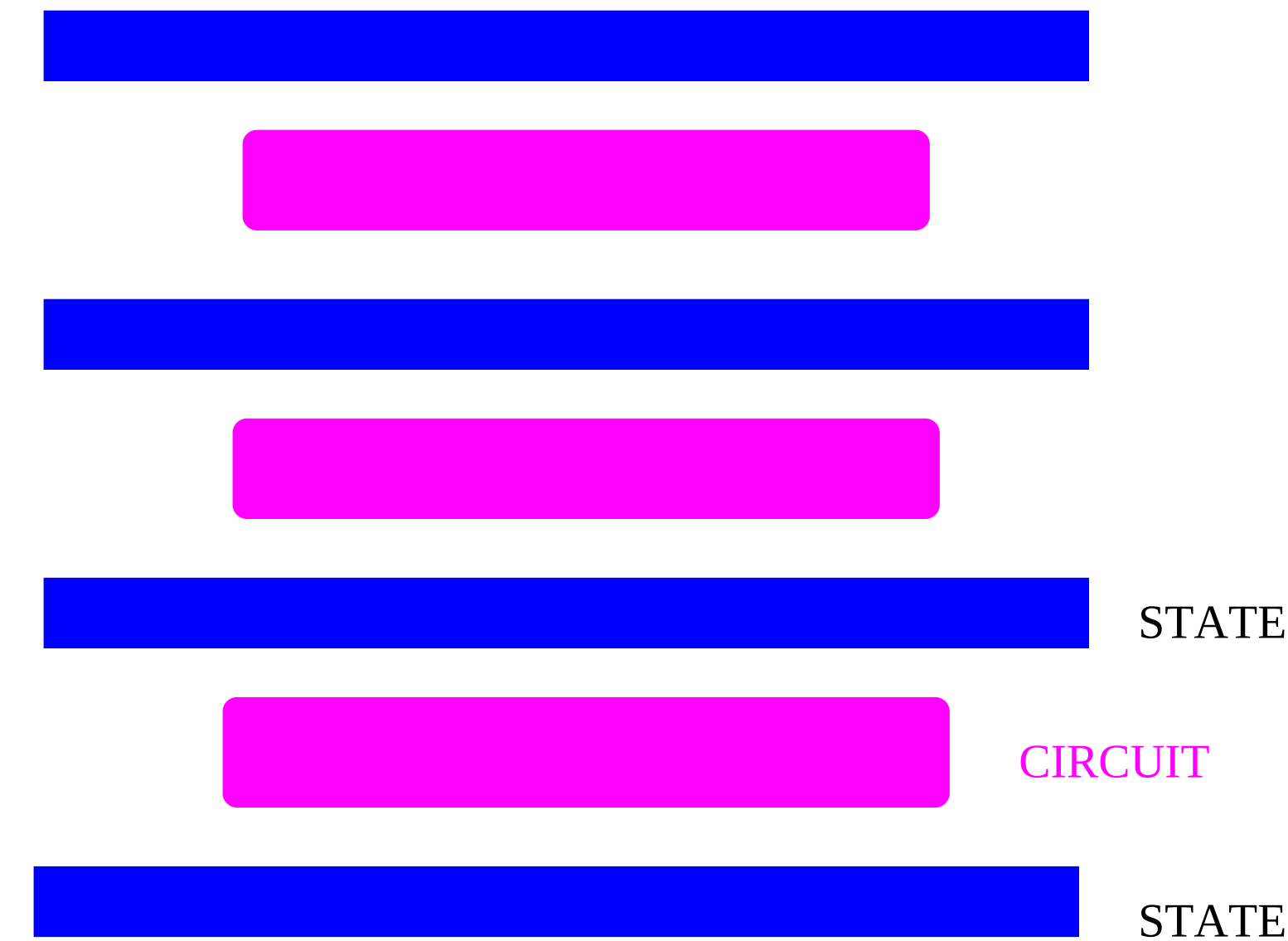
AN AUGMENTING PATH EXISTS




THE
END

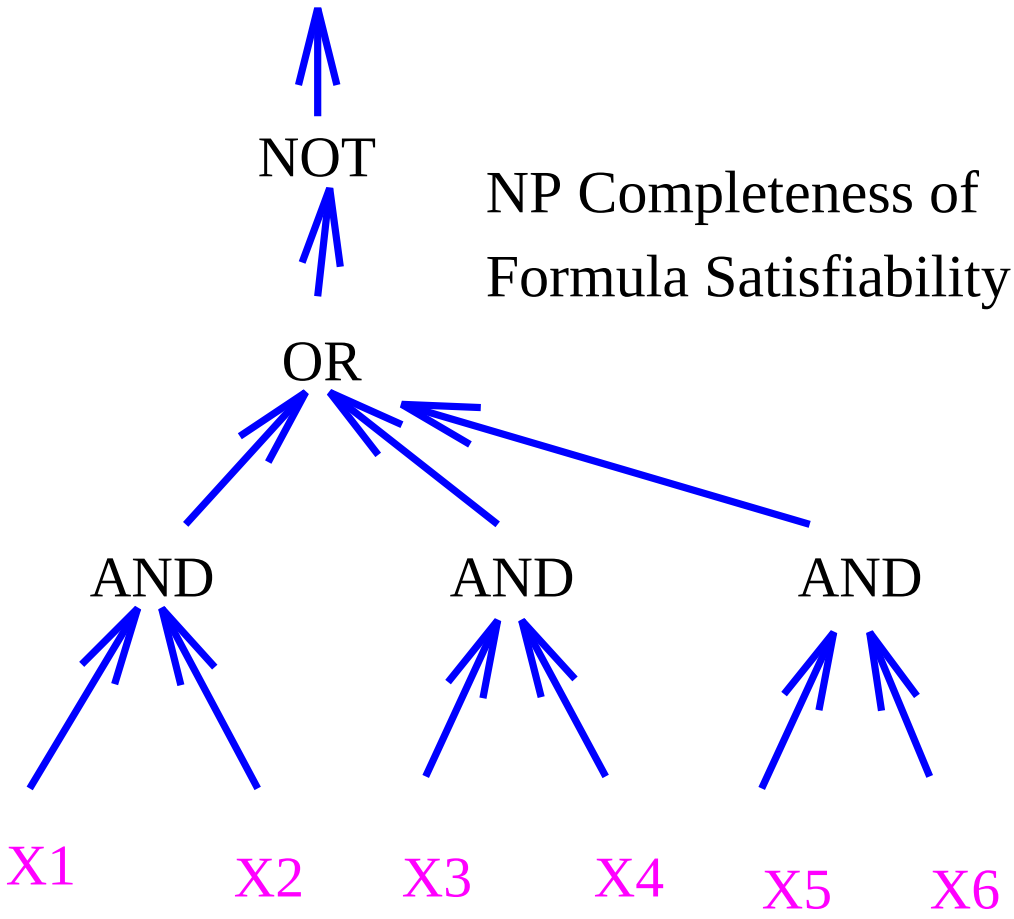
Chapter 36

NP
Completeness



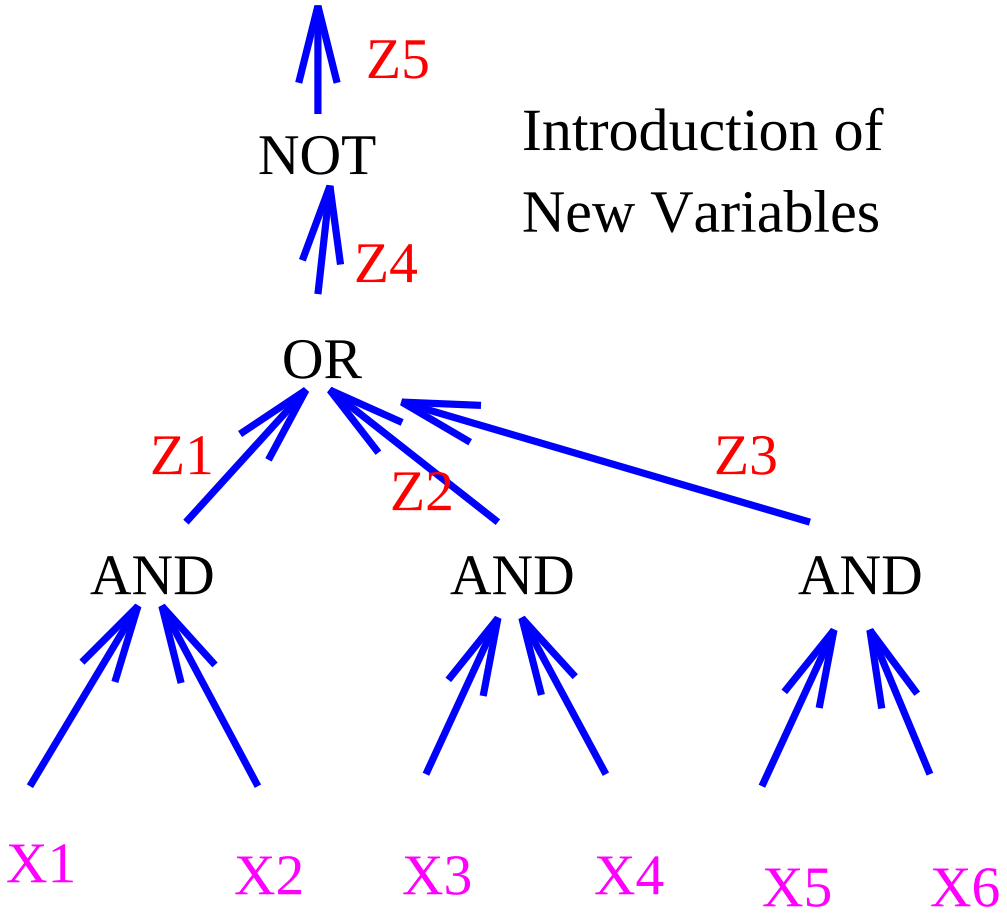
A(X,Y)

NP Completeness of Circuit Satisfiability



NOT(OR(AND(X1,X2),AND(X3,X4)))

Common subformula problem, new variables



Z1 == AND(X1,X2)

Z2 == AND(X3,X4)

Z4 == OR(Z1,Z2,Z3)

Z5

...

Z1 => AND(X1,X2)

AND(X1,X2) => Z1

Conversion of Equivalences to Clauses

not(Z1) \/ X1

not(Z1) \/ X2

not(X1) \/ not(X2) \/ Z1

Z1 == AND(X1,X2)

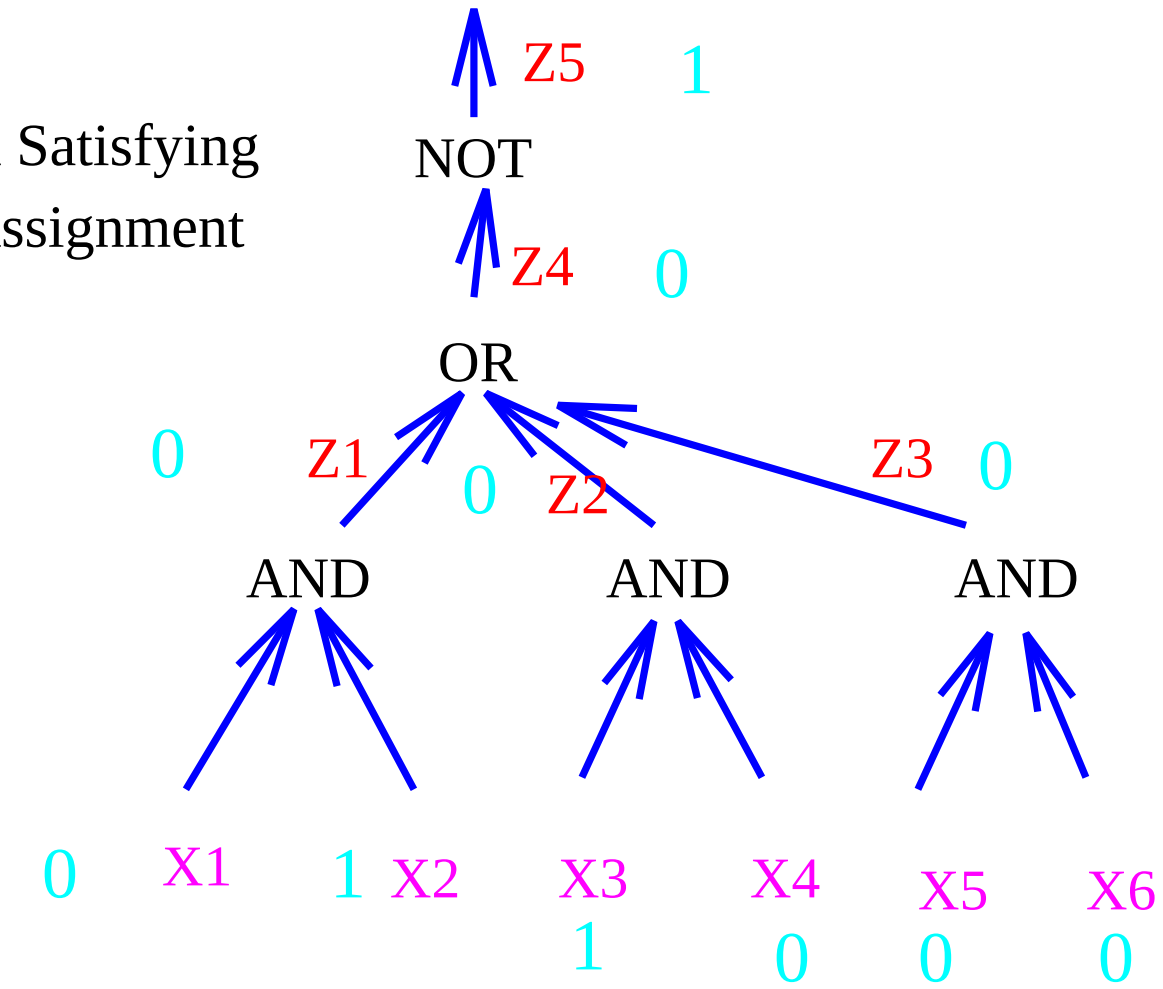
Z2 == AND(X3,X4)

Z4 == OR(Z1,Z2,Z3)

Z5

...

A Satisfying Assignment



Z1 == AND(X1,X2)

Z2 == AND(X3,X4)

Z4 == OR(Z1,Z2,Z3)

Z5 ...

Z1 => AND(X1,X2)

AND(X1,X2) => Z1

A Satisfying Assignment on the Formula

not(Z1) \/ X1

not(Z1) \/ X2

not(X1) \/ not(X2) \/ Z1

0 0

0 1 Z3 = AND(X5,X6) 1 0

Z1 == AND(X1,X2)

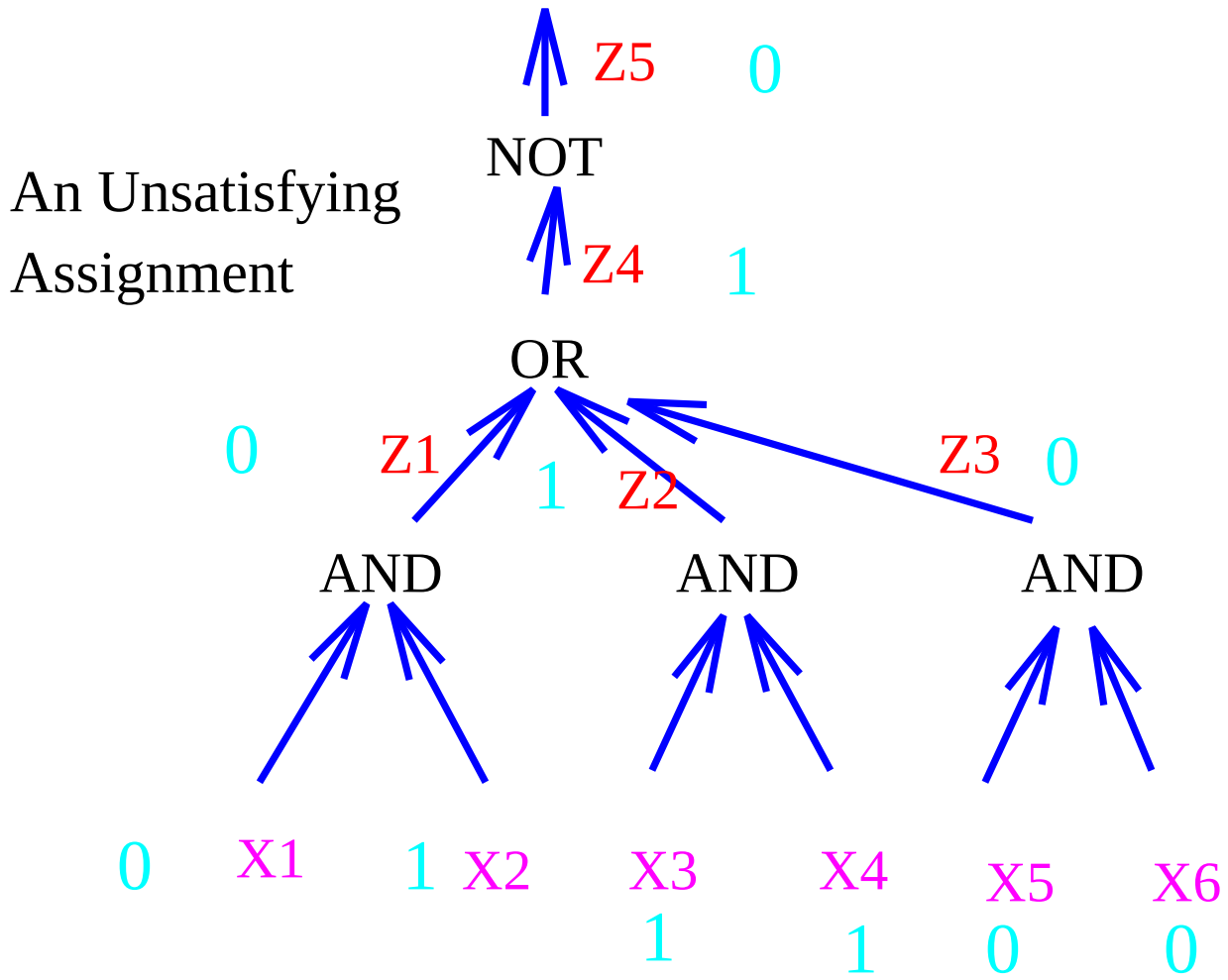
Z2 == AND(X3,X4)

Z4 == OR(Z1,Z2,Z3) Z5 1

0 0 0 Z5 = NOT(Z4)

0

An Unsatisfying Assignment



Z1 == AND(X1,X2)

Z2 == AND(X3,X4)

Z4 == OR(Z1,Z2,Z3)

Z5

...

Z1 => AND(X1,X2)

AND(X1,X2) => Z1

An Unsatisfying Assignment on the Formula

not(Z1) \/ X1

not(Z1) \/ X2

not(X1) \/ not(X2) \/ Z1

0 0

0 1 Z3 = AND(X5,X6) 1 1

Z1 == AND(X1,X2)

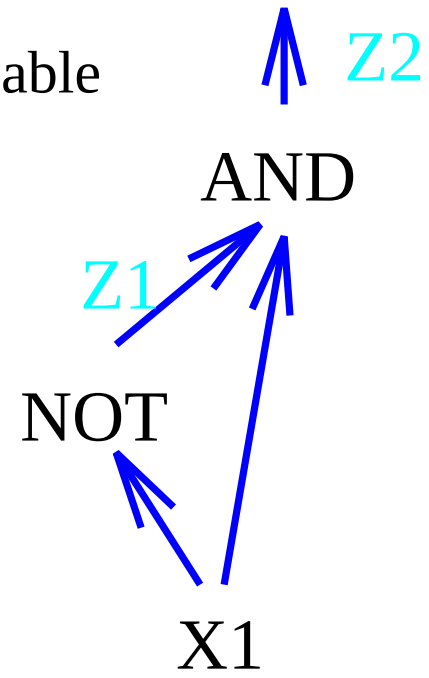
Z2 == AND(X3,X4)

Z4 == OR(Z1,Z2,Z3) Z5 0

0 1 0 Z5 = NOT(Z4)...

1

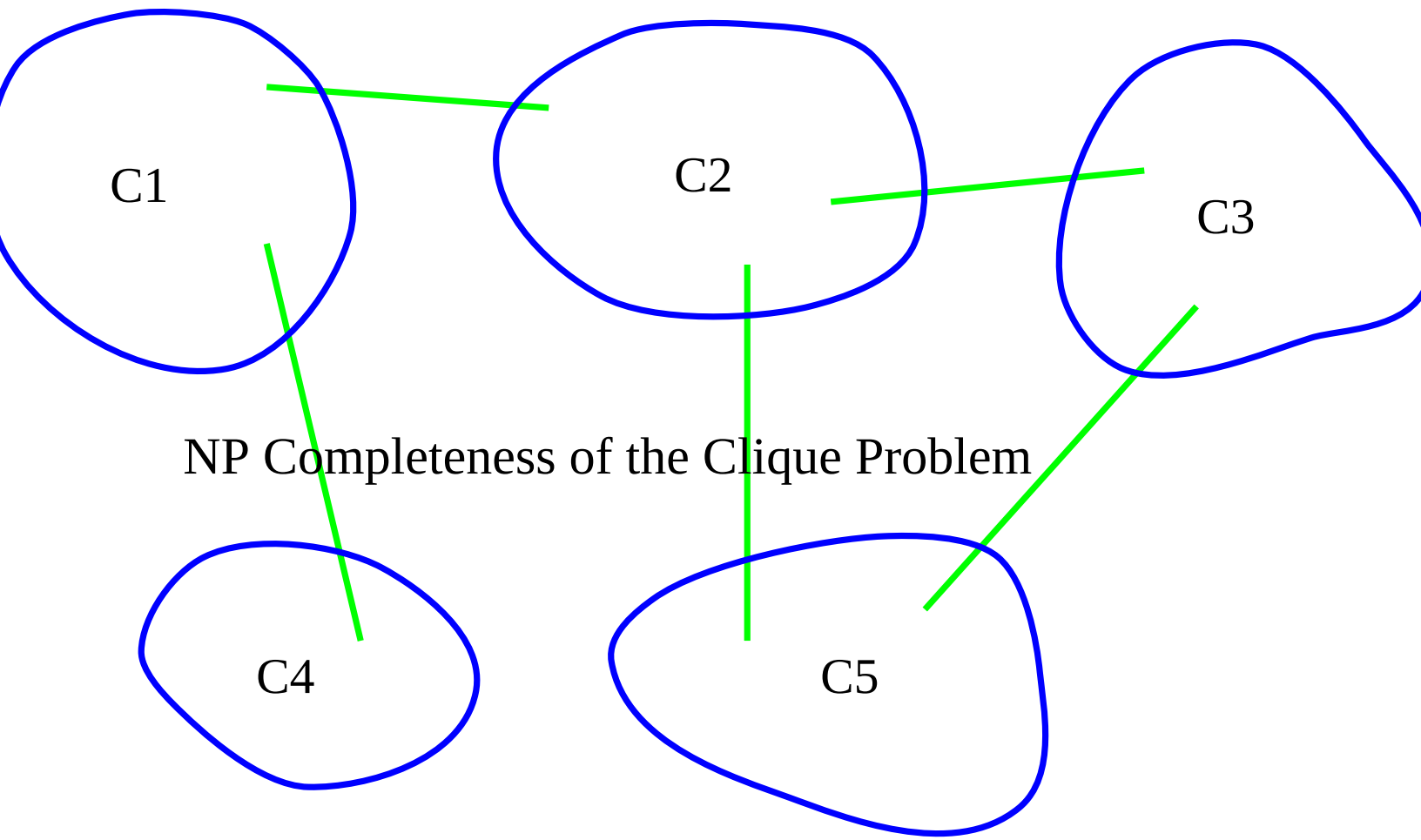
An Unsatisfiable Circuit



Z1 == NOT(X1) $\wedge$

Z2 == AND(Z1,X1) $\wedge$

Z2



NP Completeness of the Clique Problem

P or Q or R

P or Q or not(R)

P or not(Q) or R

P or not(Q) or not(R)

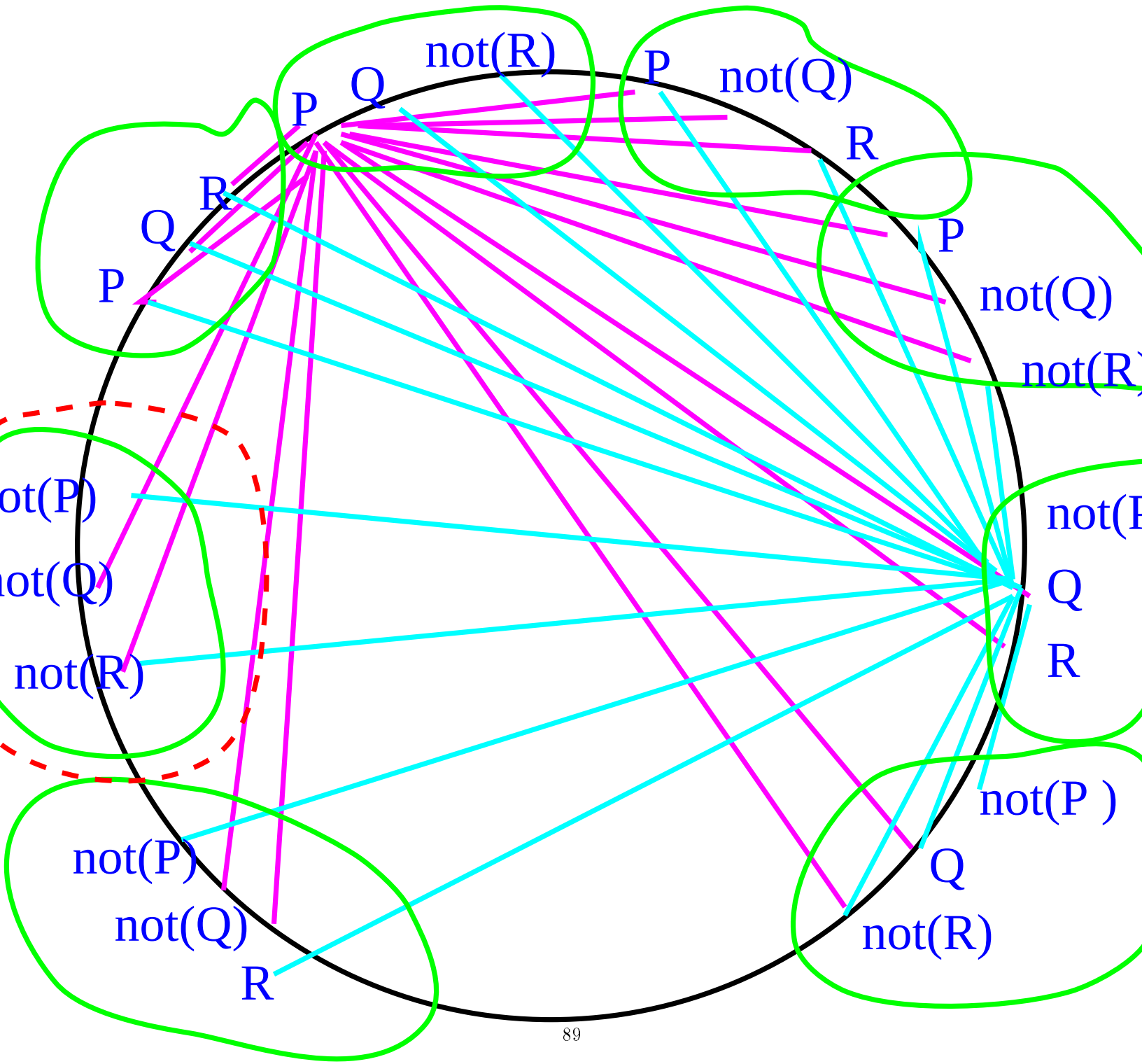
not(P) or Q or R

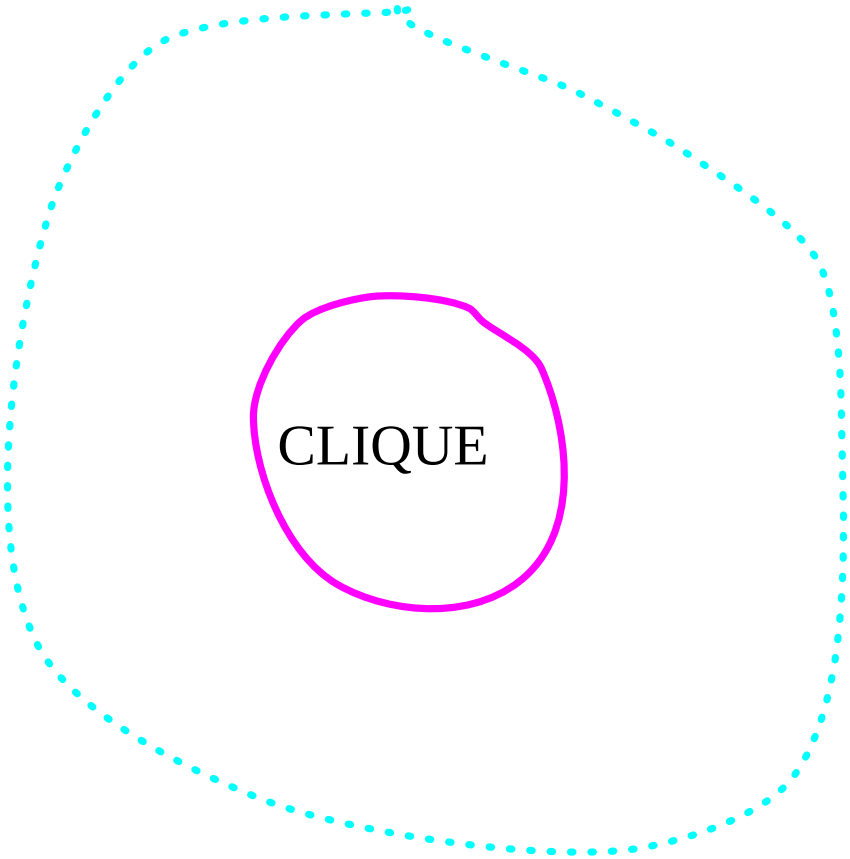
not(P) or Q or not(R)

not(P) or not(Q) or R

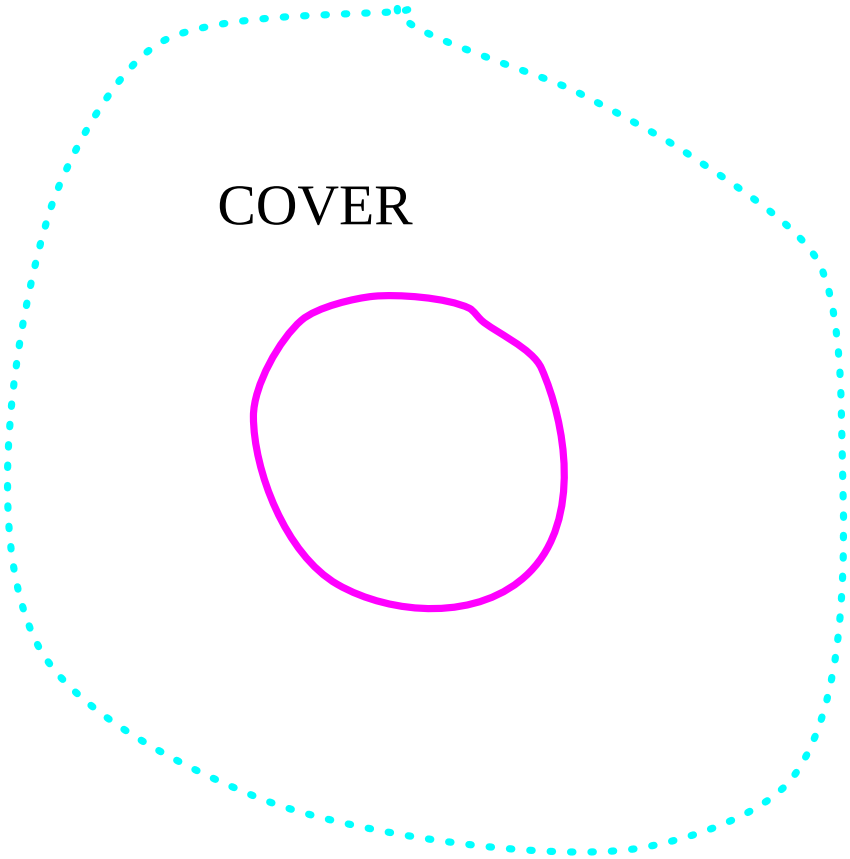
not(P) or not(Q) or not(R)

Example of Clique Reduction

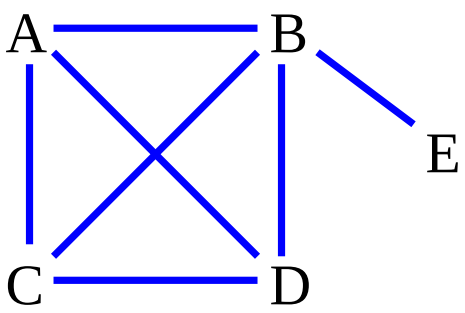

not(R)
Q
P
P
not(Q)
R
R
Q
P
P
not(Q)
not(R)
ot(P)
not(Q)
not(R)
not(P)
not(Q)
R
Q
R
not(P)
Q
not(R)



Vertex Cover Reduction

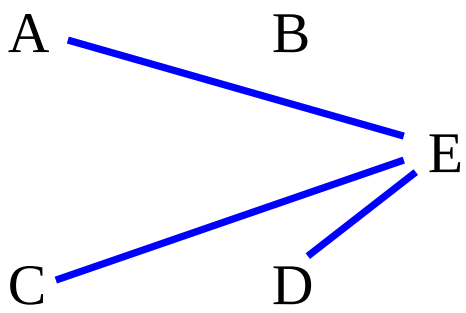


Vertex Cover Reduction



An Example Clique

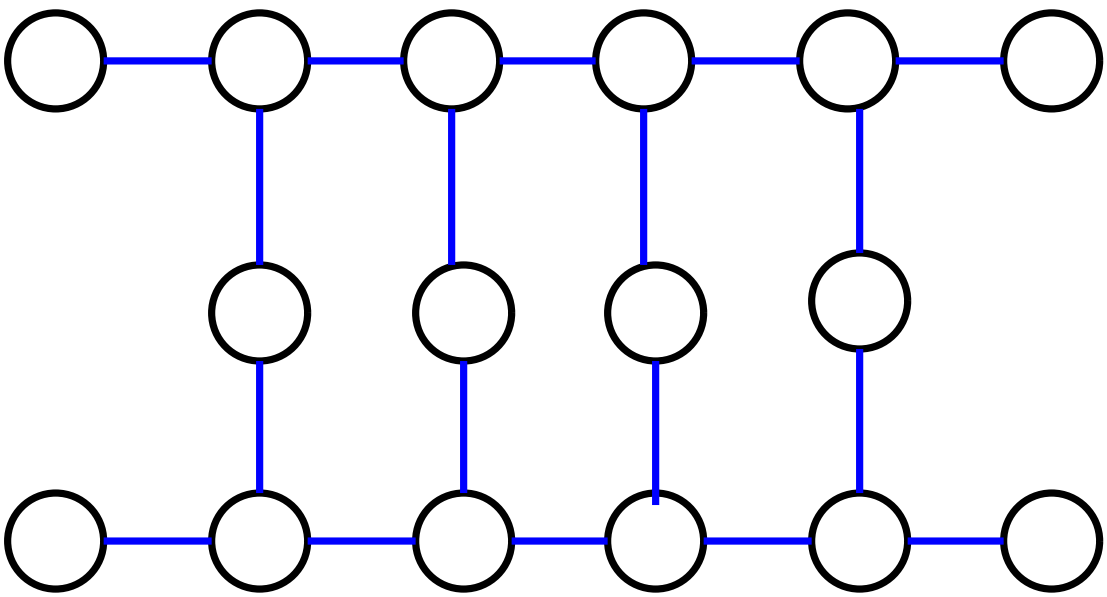
The Corresponding



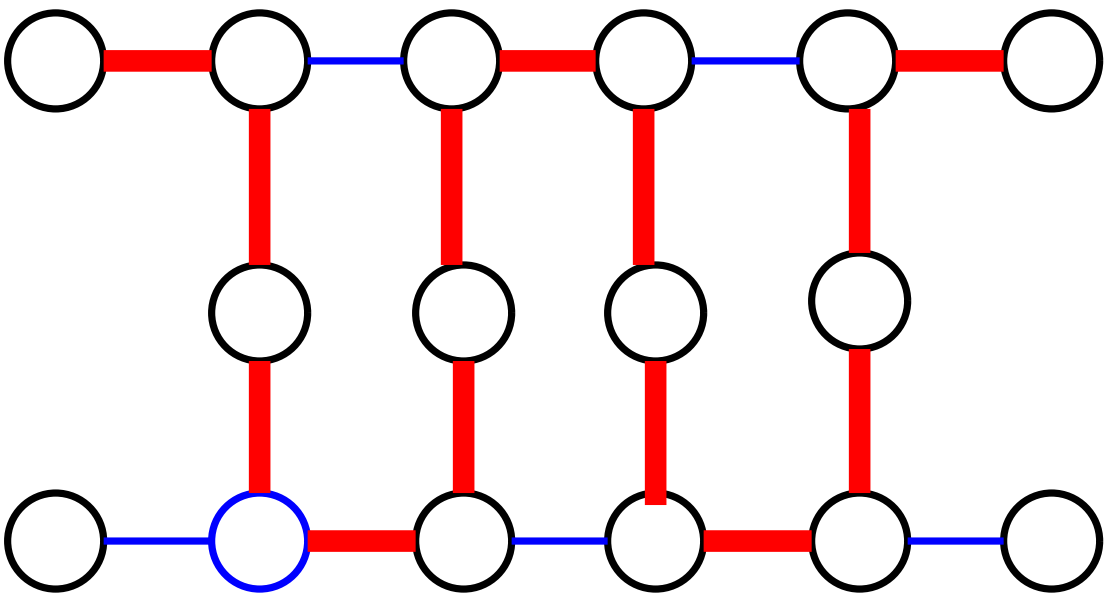
Cover

SUBSET SUM

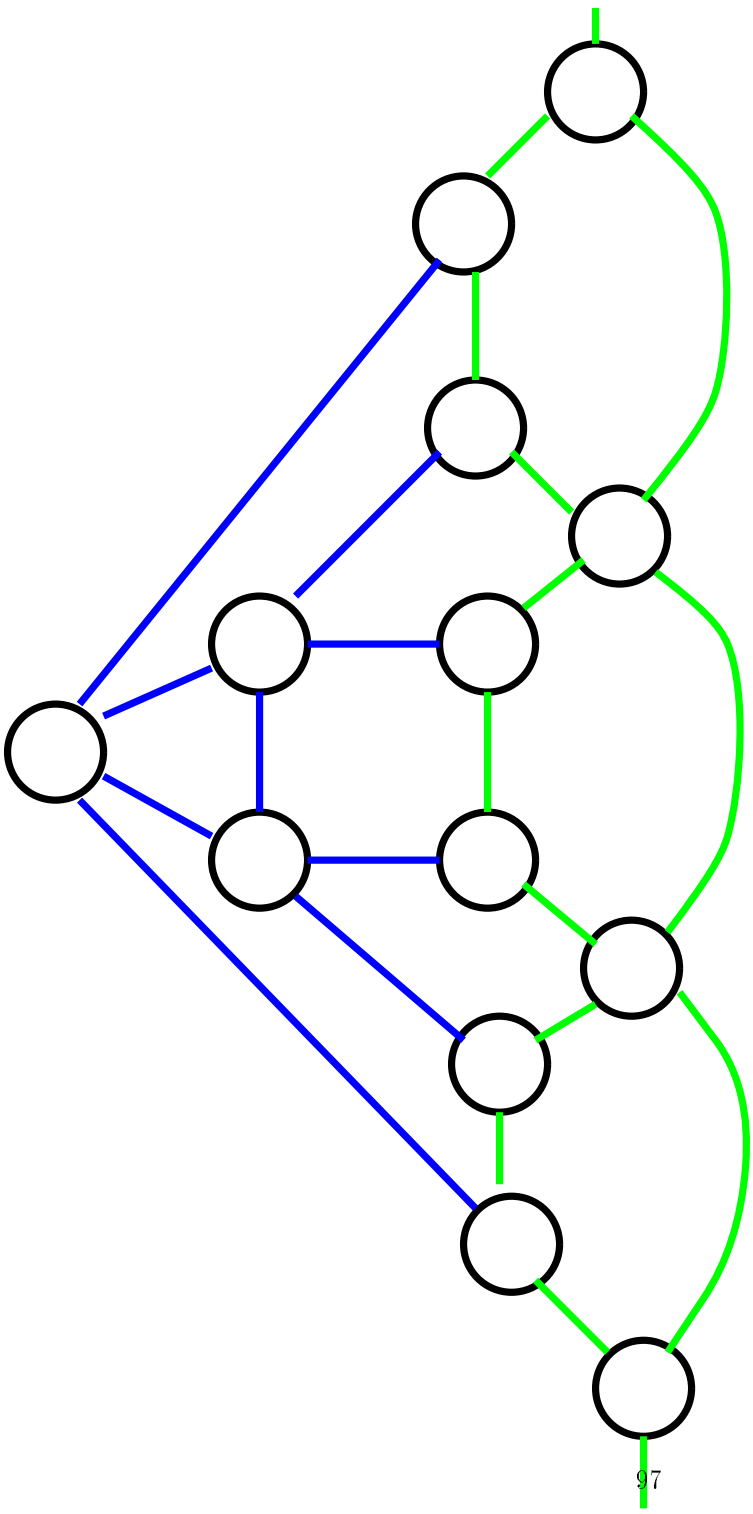
	c	em ...	ei	e1
V1	1			
V2	1		1	
...				
Vn	1	1		
e1				1
...				
em		1		
sum	k	2	2	2



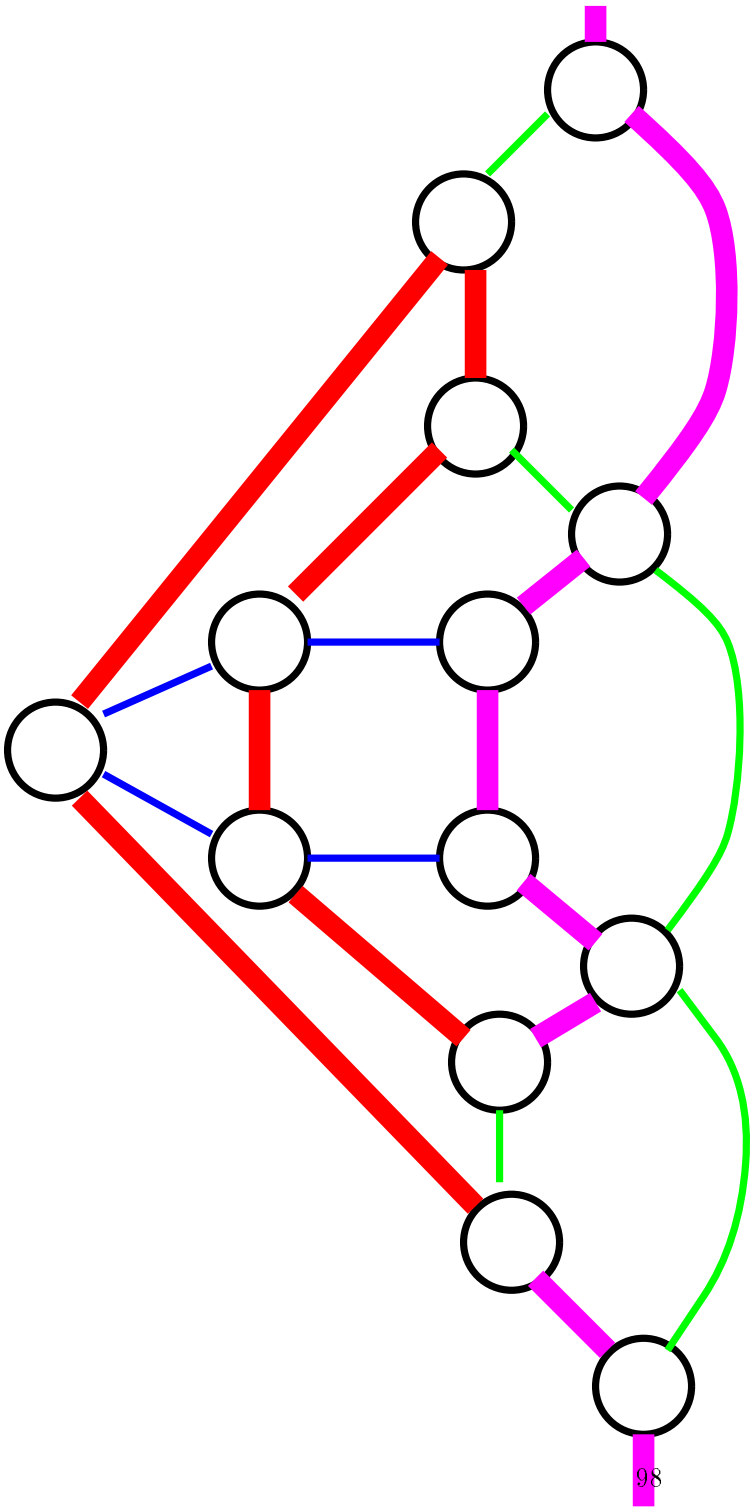
A WIDGET



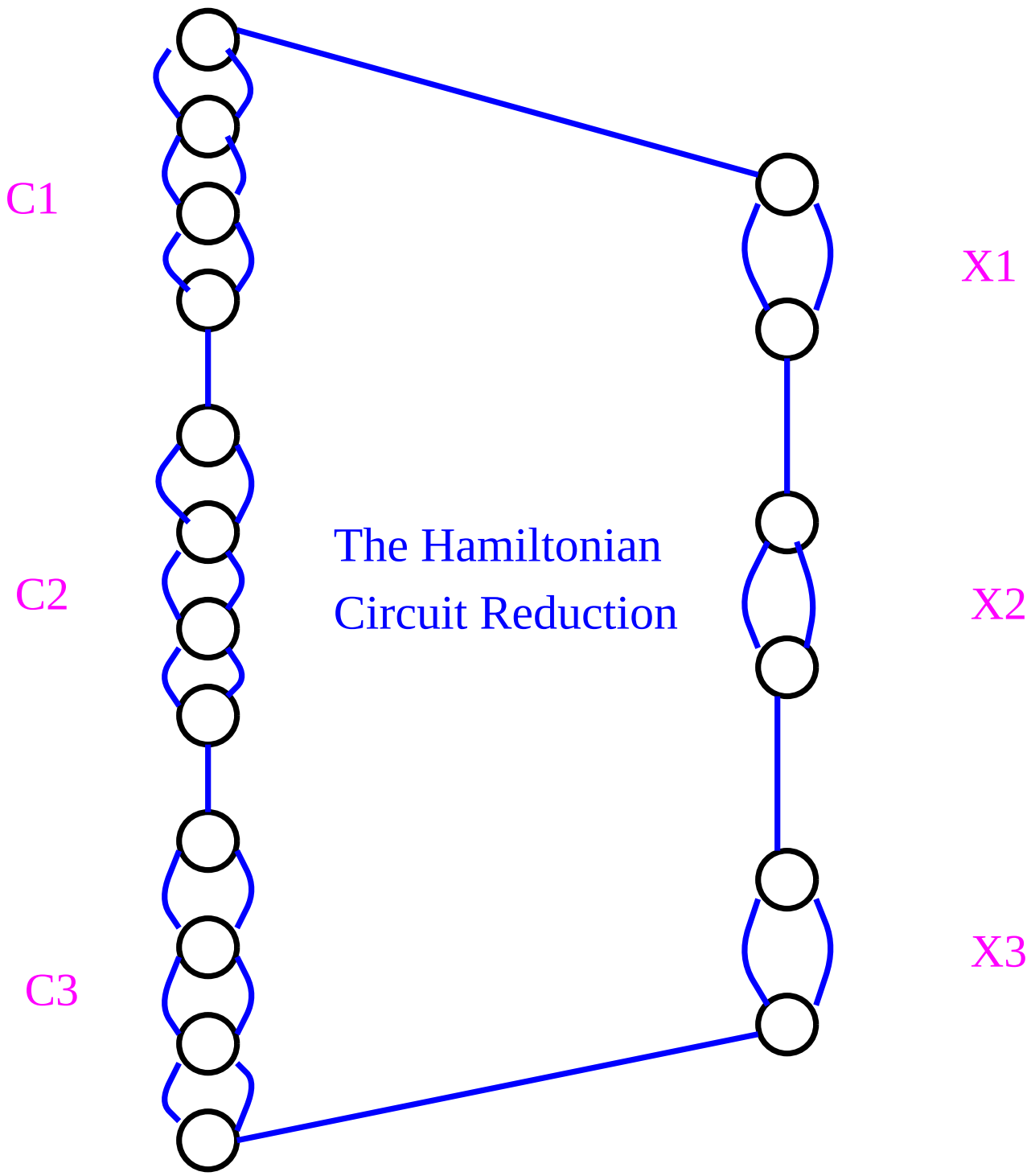
A WIDGET



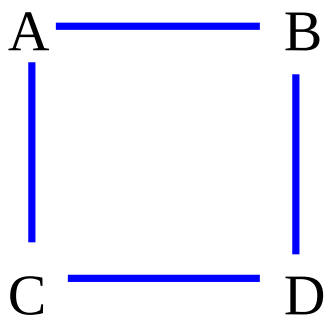
B WIDGET



B WIDGET

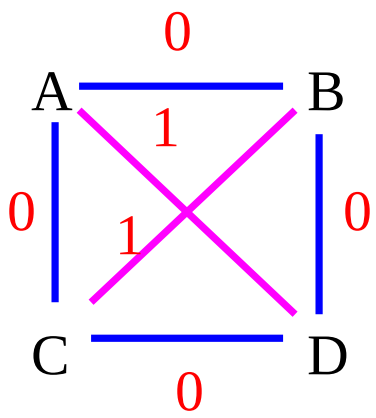

C1
C2
C3
X1
X2
X3
The Hamiltonian
Circuit Reduction

A Hamiltonian



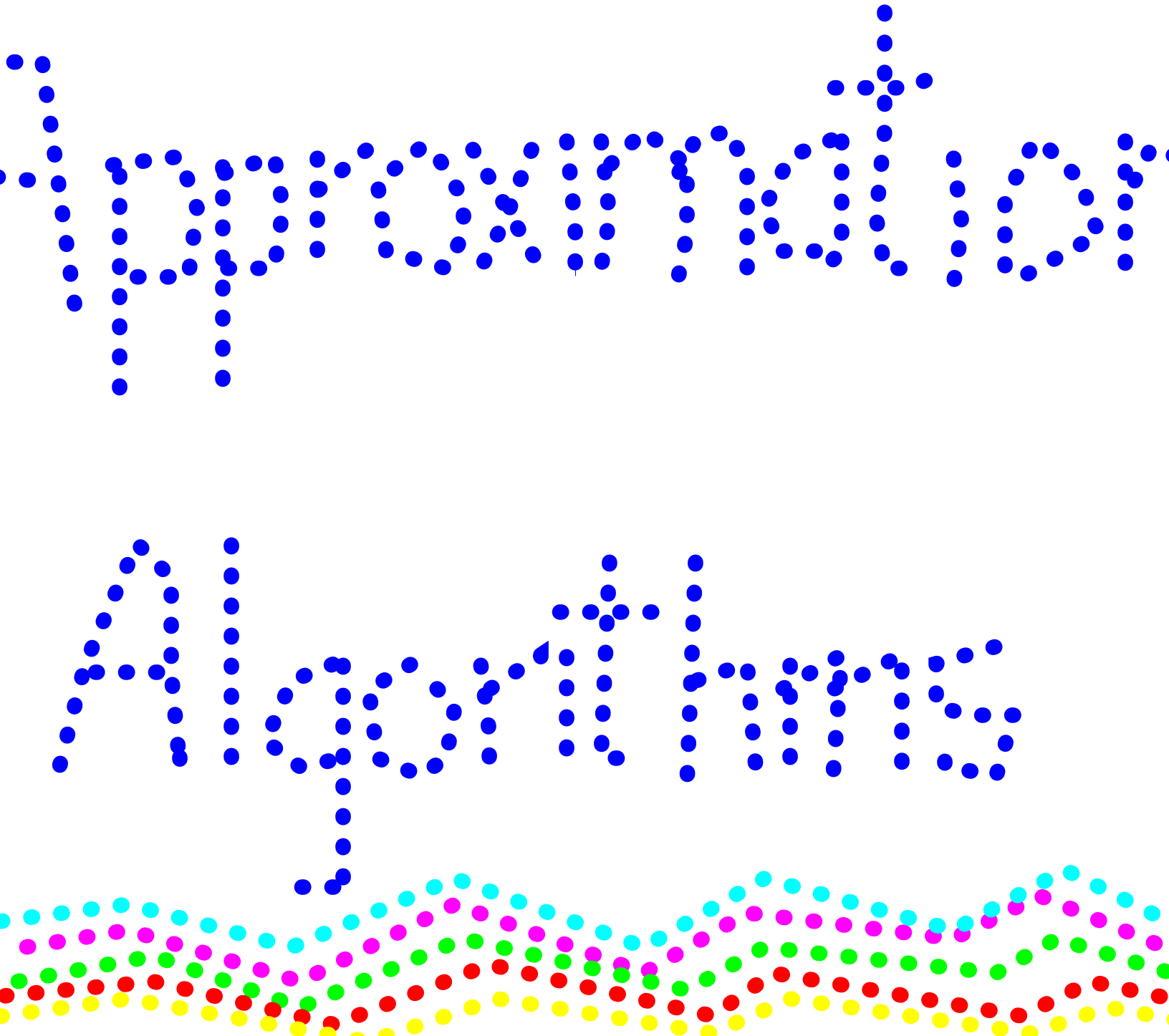
Circuit

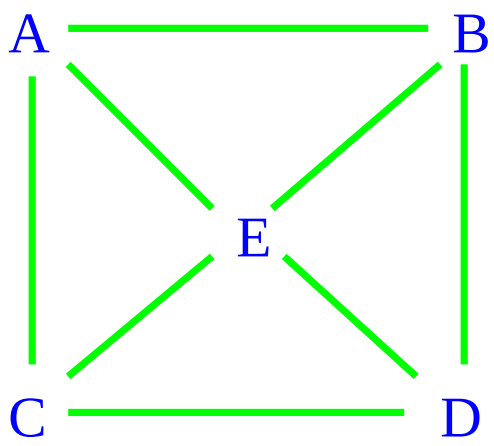
The Corresponding



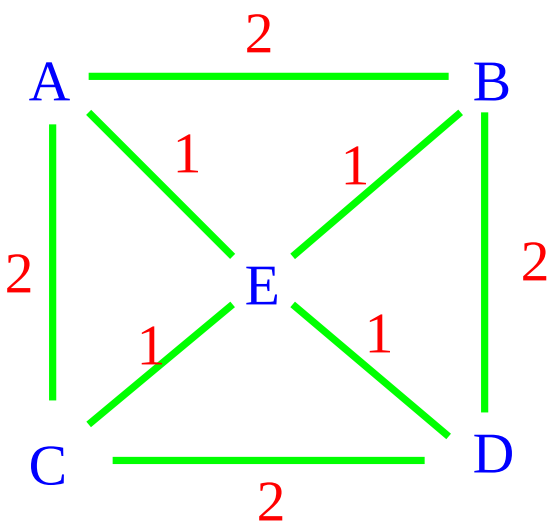
TSP Problem

Chapter 37


Approximation
Algorithms

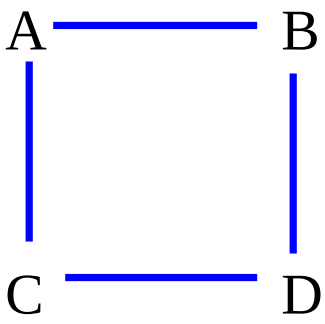


Vertex Cover Problem

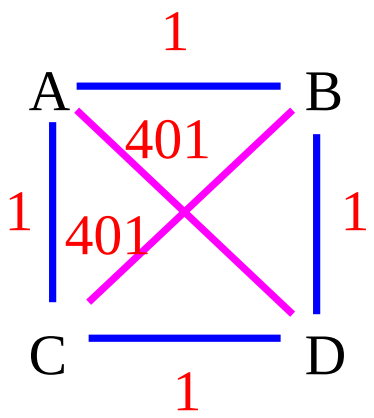


Graph Satisfying the
Triangle Inequality

TSP Can Be Approximated



A Hamiltonian Circuit



Difficulty of
Approximating a
TSP Tour
No Triangle Inequality

{ 1 2 }

{ 3 4 }

{ 5 6 }

{ 2 3 }

{ 4 5 }

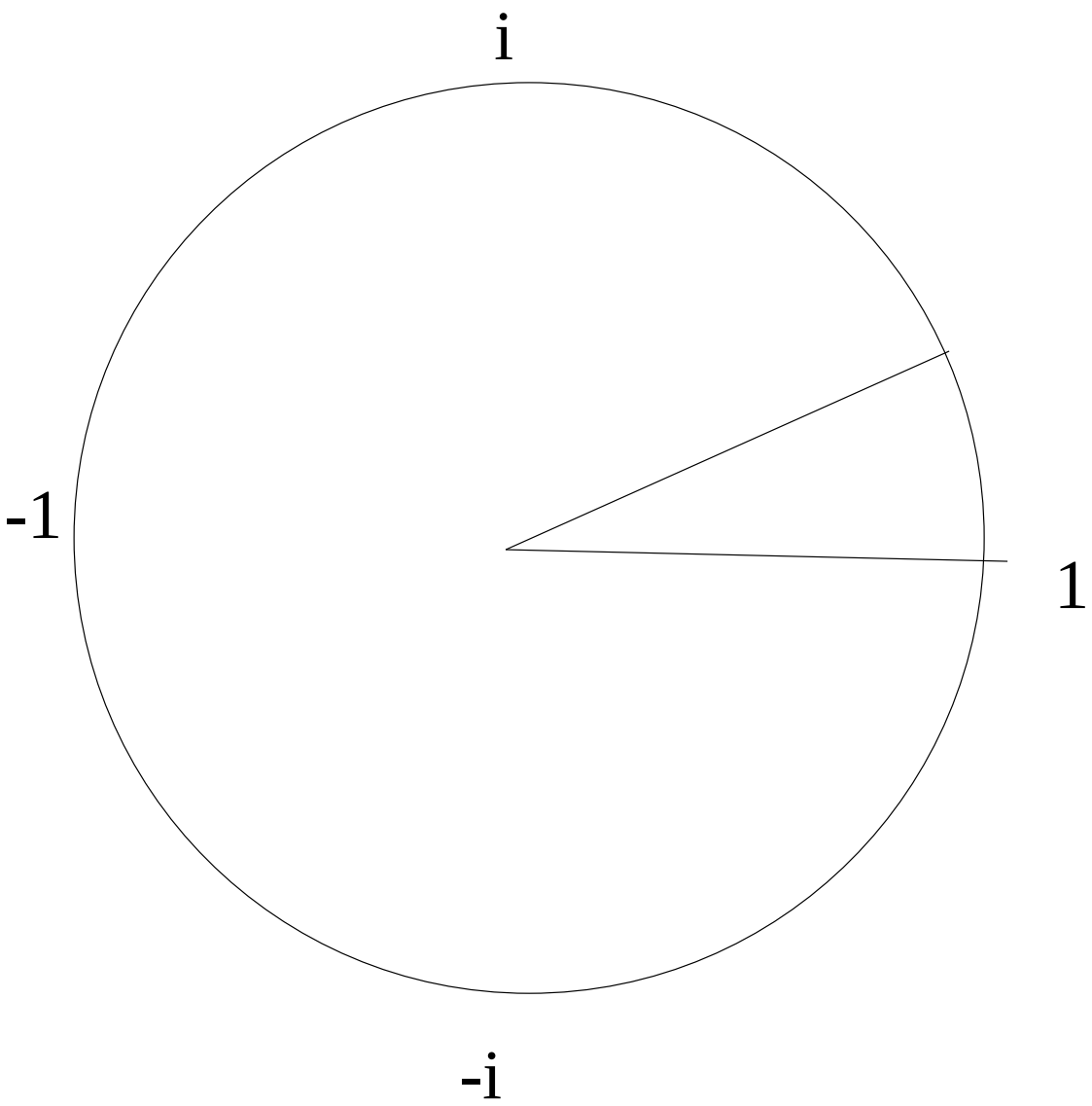
Set Cover Example

{ 2 5 8 14 22 50 }

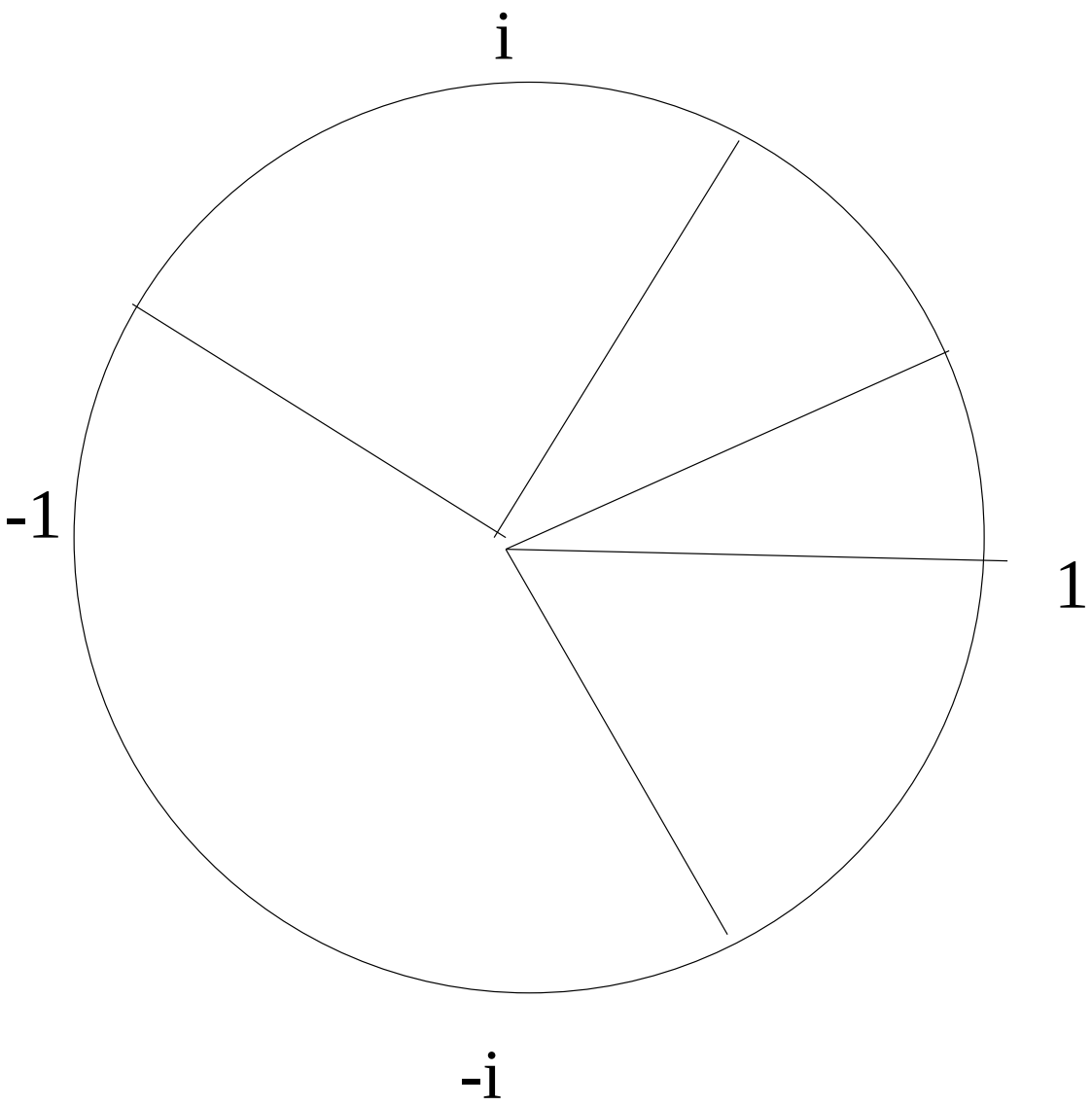
75

Example Subset Sum Problem

Chapter 32



TWELFTH ROOT OF UNITY



POWERS OF TWELFTH ROOT

INVERTING THE FOURIER TRANSFORM

1 + x-x^2

(1,1,-1,0)

(p(i),p(i^2),p(i^3),p(i^4))

(p(1),p(i),p(-1),p(-i))

(1,2+i,-1,2-i)

Consider this as a polynomial in x:

1 + (2+i)x - x^2 + (2-i)x^3

Evaluate it at x = 1:

1 +(2+i)1 - 1 + (2-i)1

1 + 2 + i - 1 + 2 - i = 4 Divide by 4 to get a_0 = 1 .

Computation of a_1

1 + x-x^2

(1,1,-1,0)

(p(i),p(i^2),p(i^3),p(i^4))

WRONG:	CORRECT:
(p(i),p(-1),p(-i),p(1))	(p(1),p(i),p(-1),p(-i))
(2+i,-1,2-i,1)	(1,2+i,-1,2-i)

evaluate this as a poly. at -i

(2+i)-x+(2-i)x^2+x^3	1 + (2+i)x - x^2 + (2-i)x^3
(2+i) +i +(2-i)(-1)+i	1 +(2+i)(-i) - (-1) + (2-i)i
2+i+i-2+i+i = 4i	1 -2i + 1 + 1 + 2i + 1 = 4 !!

1 + x-x^2

(1,1,-1,0)

(p(i),p(i^2),p(i^3),p(i^4))

Evaluate at i

(p(1),p(i),p(-1),p(-i))

(1,2+i,-1,2-i)

Consider this as a polynomial in x

1 + (2+i)x - x^2 + (2-i)x^3 Evaluate it at x=(-i)^2=-1

1 + (2+i)(-1) - (1) + (2-i)(-1)

1 -2-i -1 -2 + i = -4 Divide by 4 to get a_2=-1

$1 + x-x^2$

$(1,1,-1,0)$

$(p(i),p(i^2),p(i^3),p(i^4))$

Evaluate at i

$(p(1),p(i),p(-1),p(-i))$

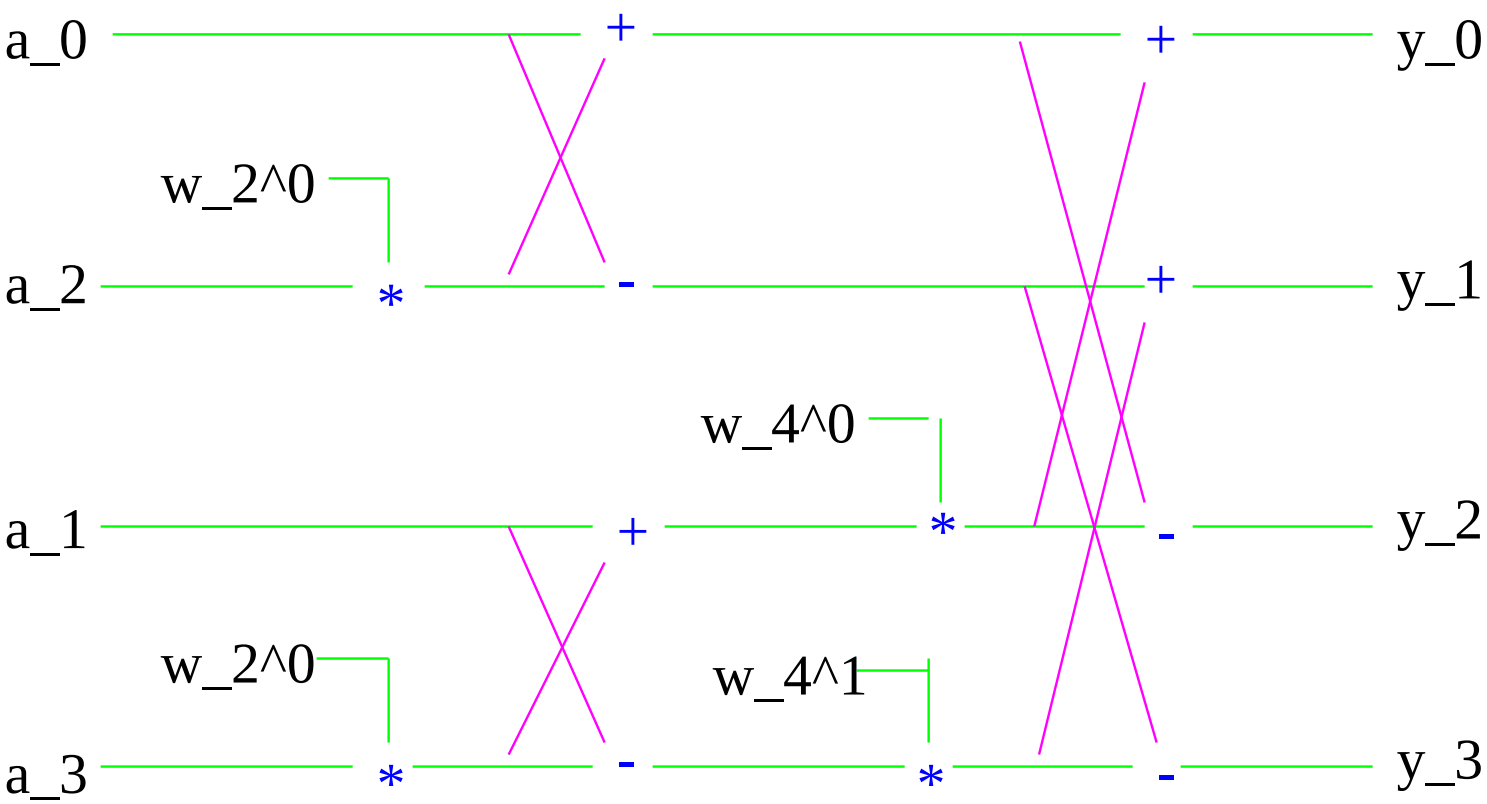
$(1,2+i,-1,2-i)$

Consider this as a polynomial in x

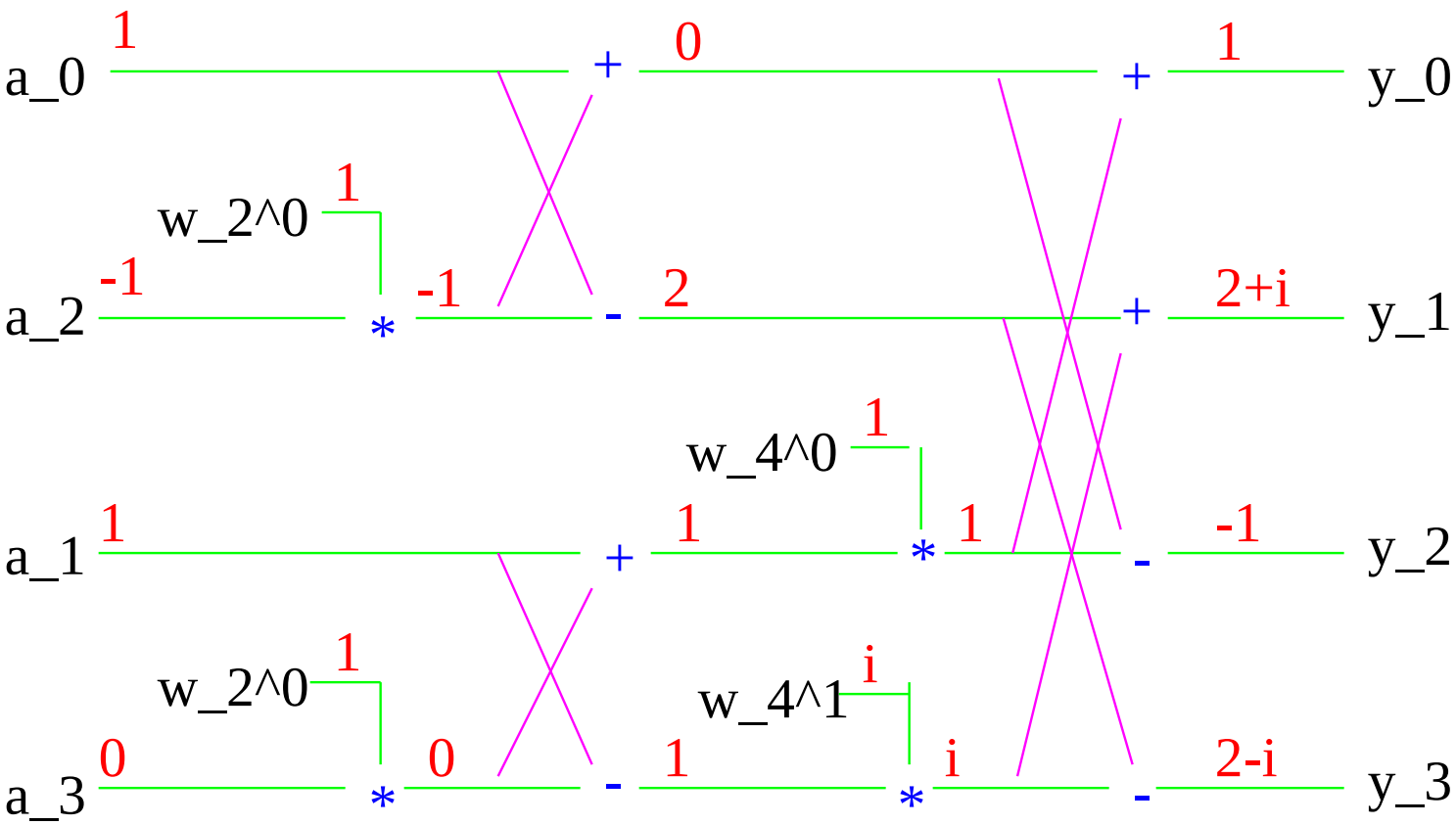
$1 + (2+i)x - x^2 + (2-i)x^3$ Evaluate it at $x=(-i)^3=i$

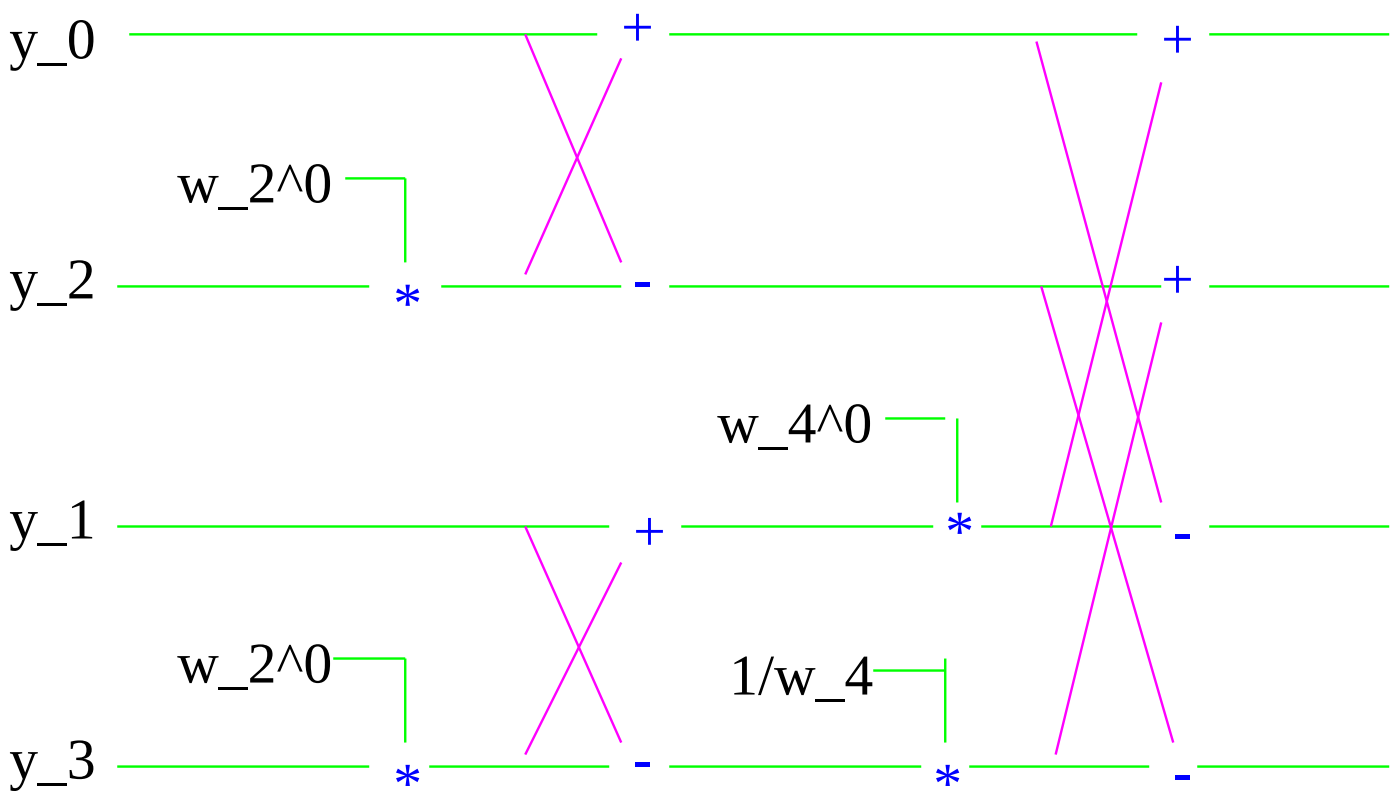
$1 + (2+i)(i) - (-1) + (2-i)(-i)$

$1 + 2i - 1 + 1 - 2i - 1 = 0$ Divide by 4 to get $a_3=0$

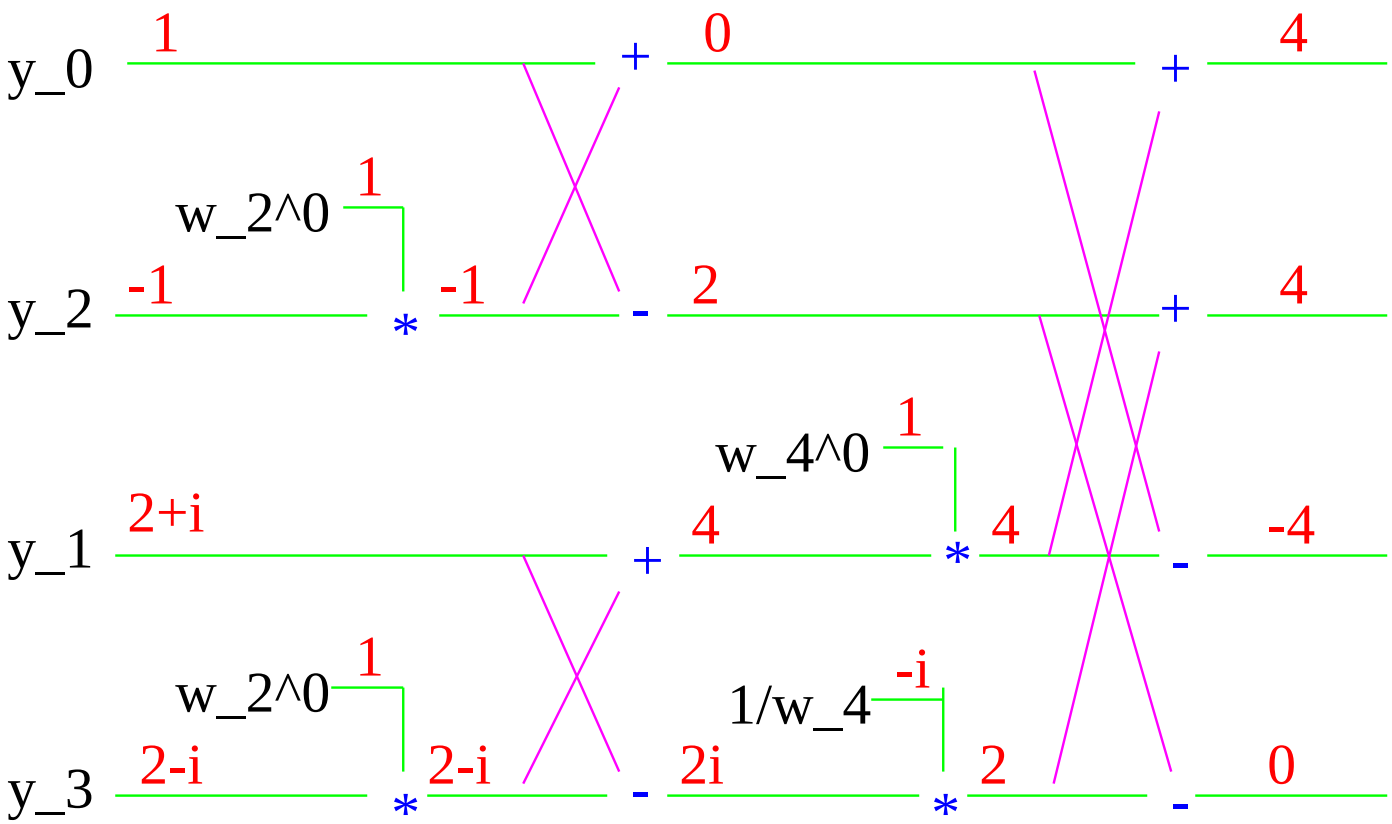

a_0
a_2
a_1
a_3
w_2^0
w_2^0
w_4^0
w_4^1
+
-
*
+
-
*
+
-
*
+
-
*
y_0
y_1
y_2
y_3

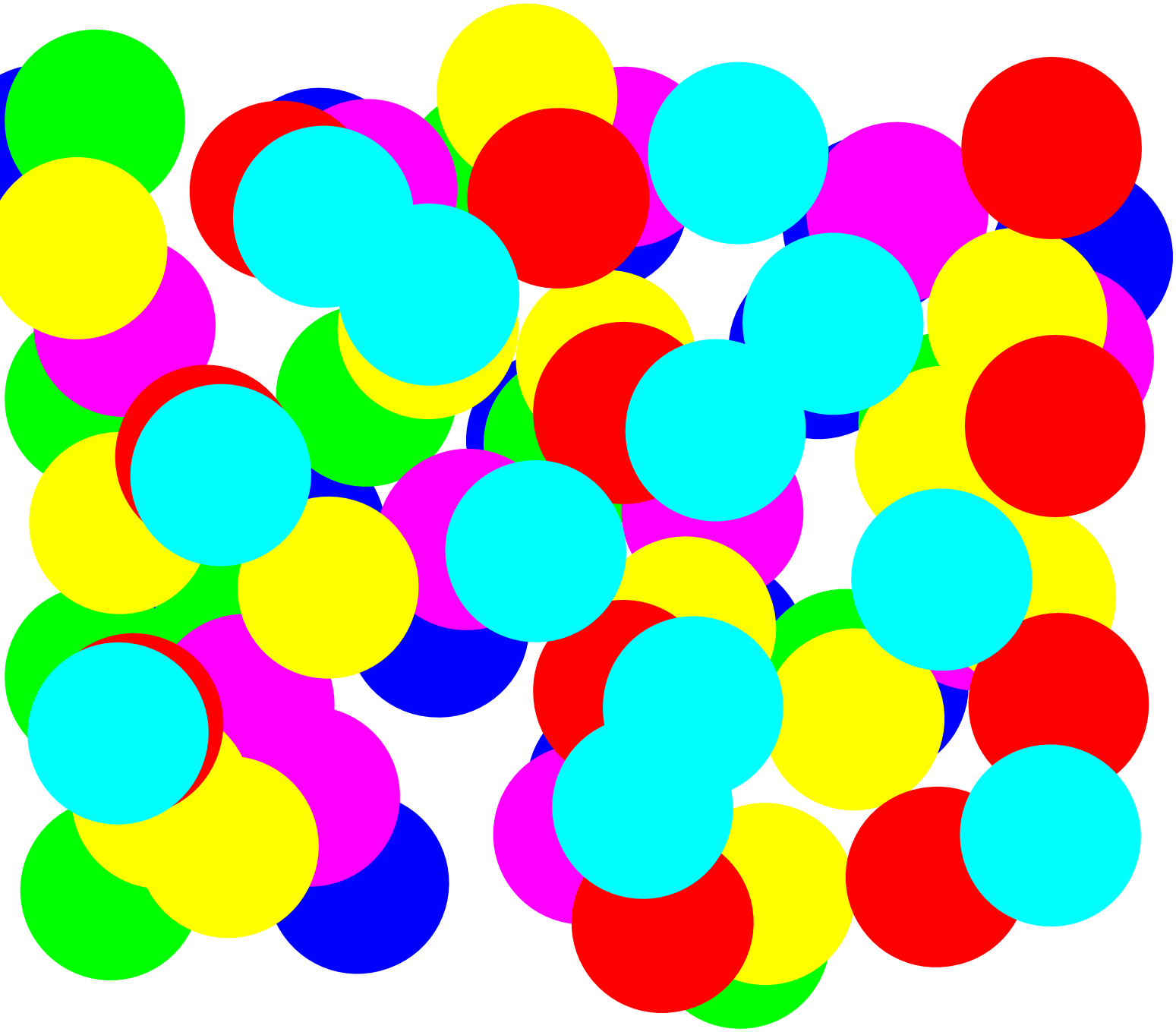
COMPUTING THE FOURIER TRANSFORM

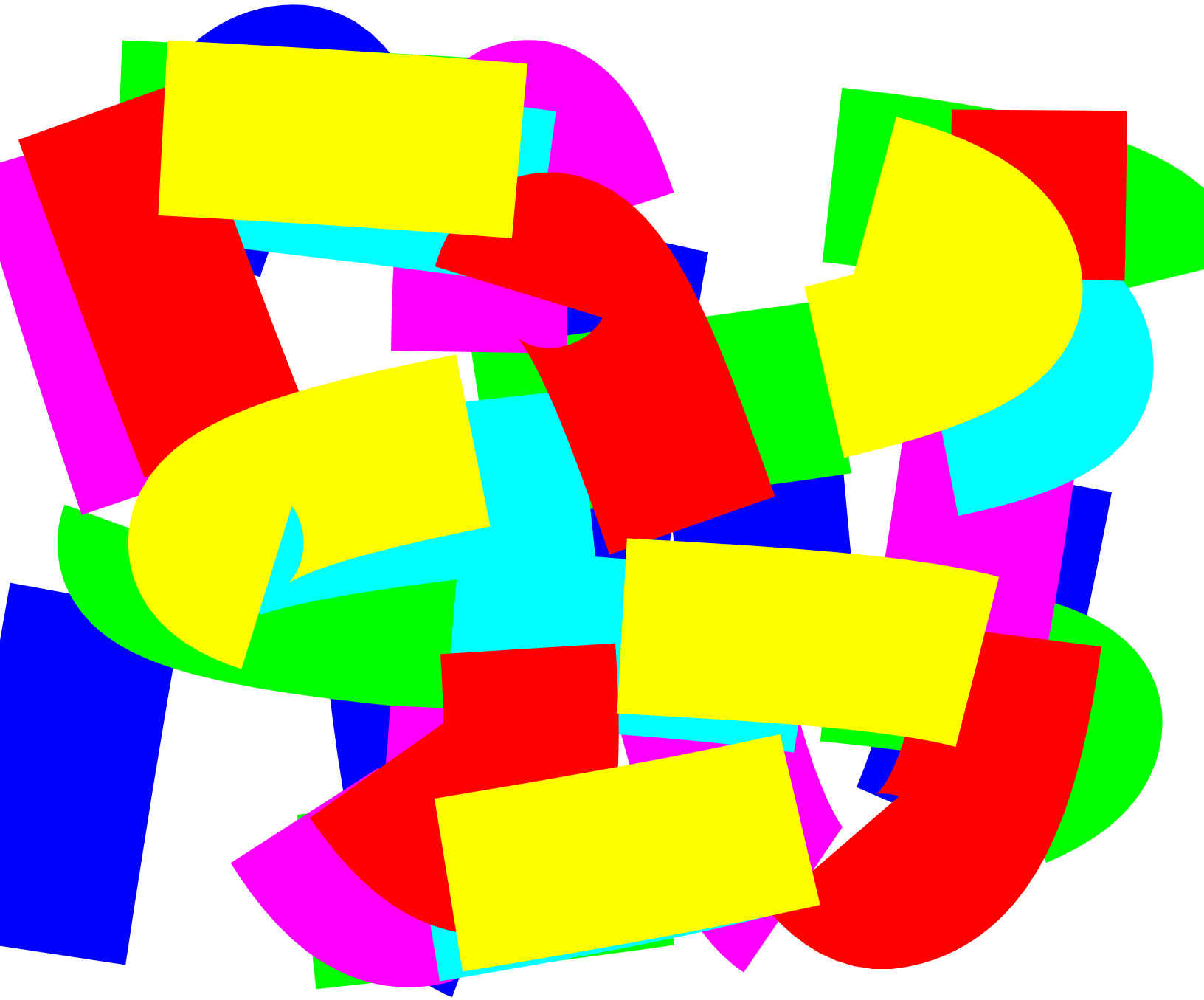


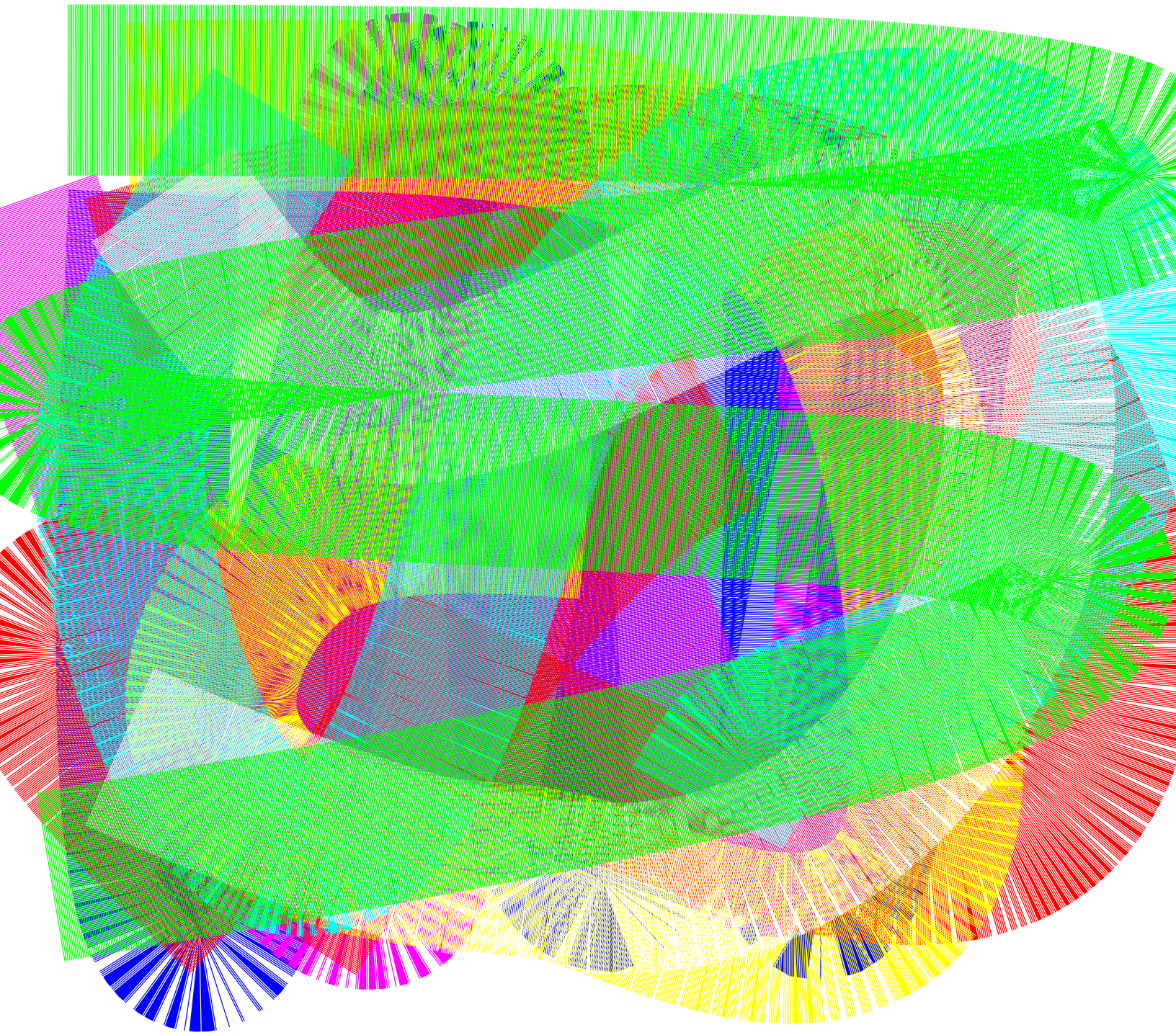

y_0
y_2
y_1
y_3
w_2^0
w_2^0
w_4^0
1/w_4
+
+
+
+
*
*
*
*
-
-
-
-

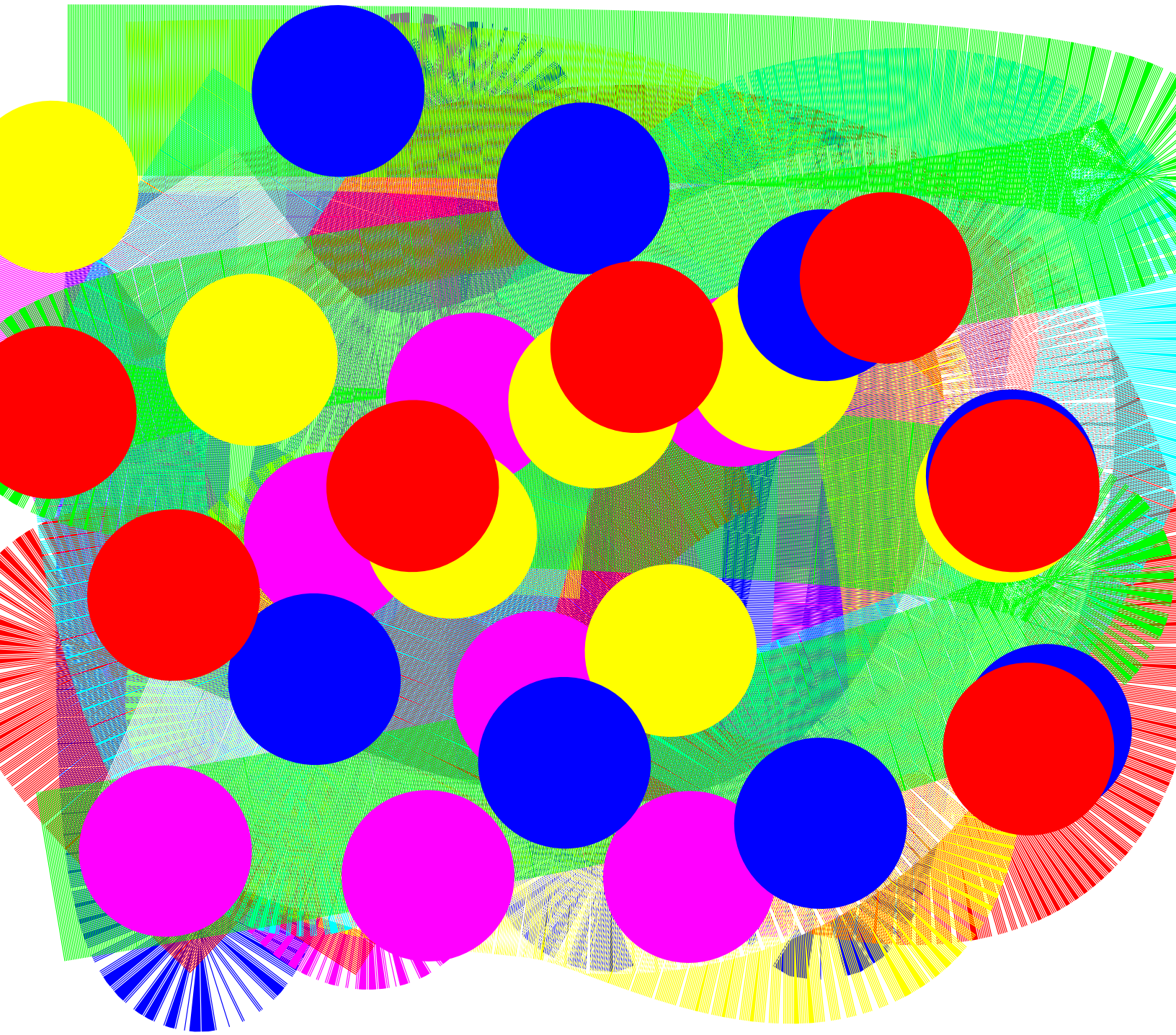
INVERTING THE FOURIER TRANSFORM

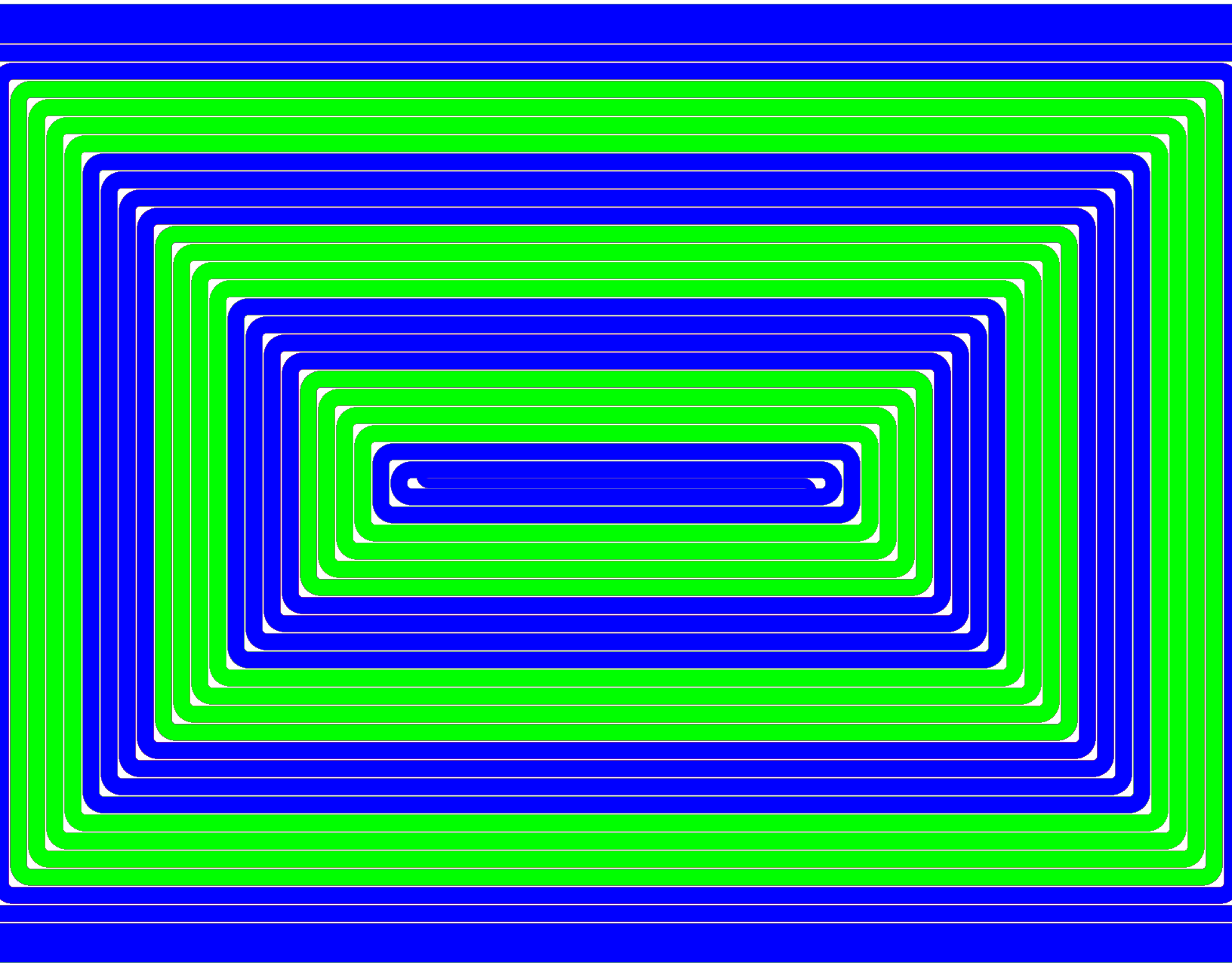


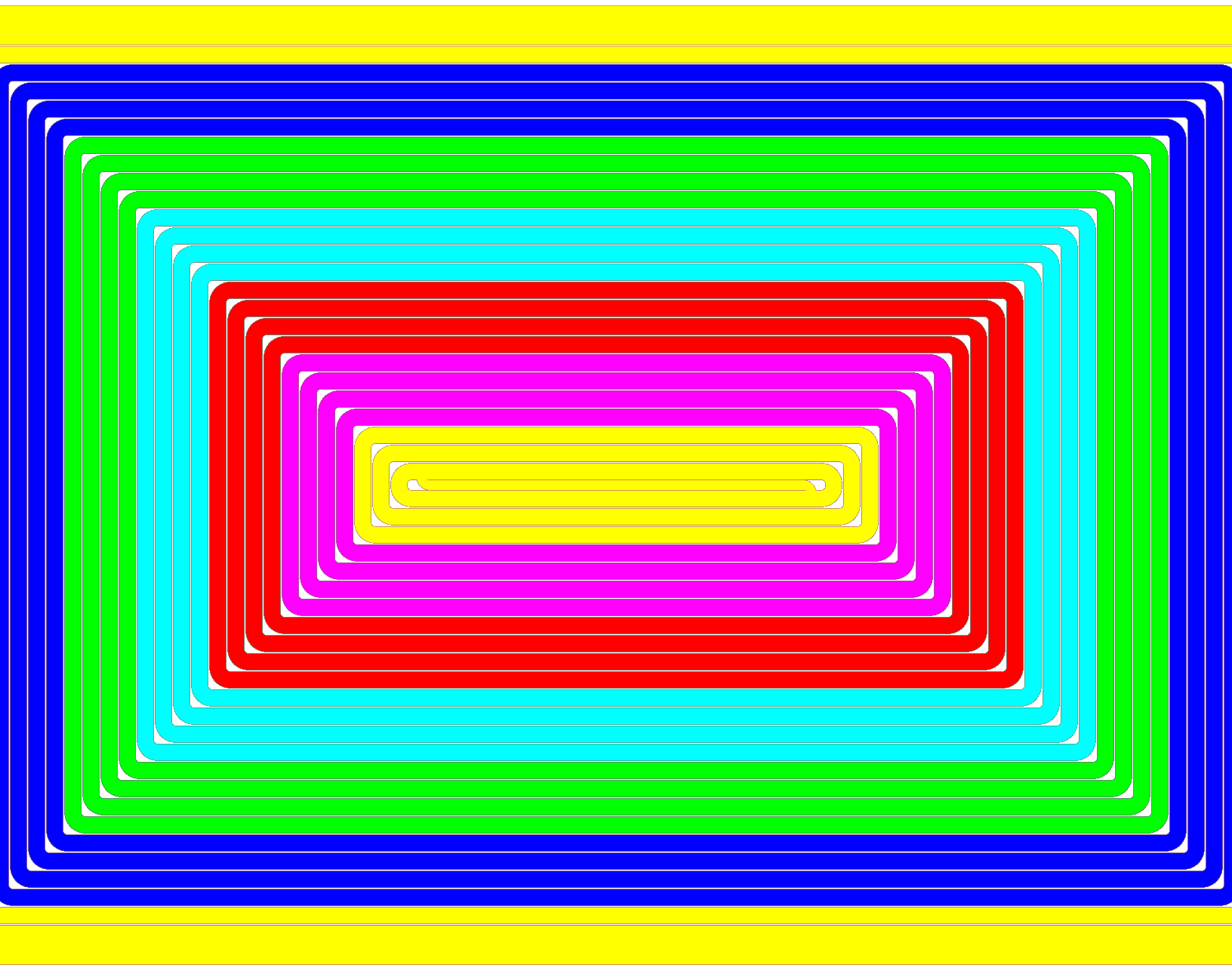


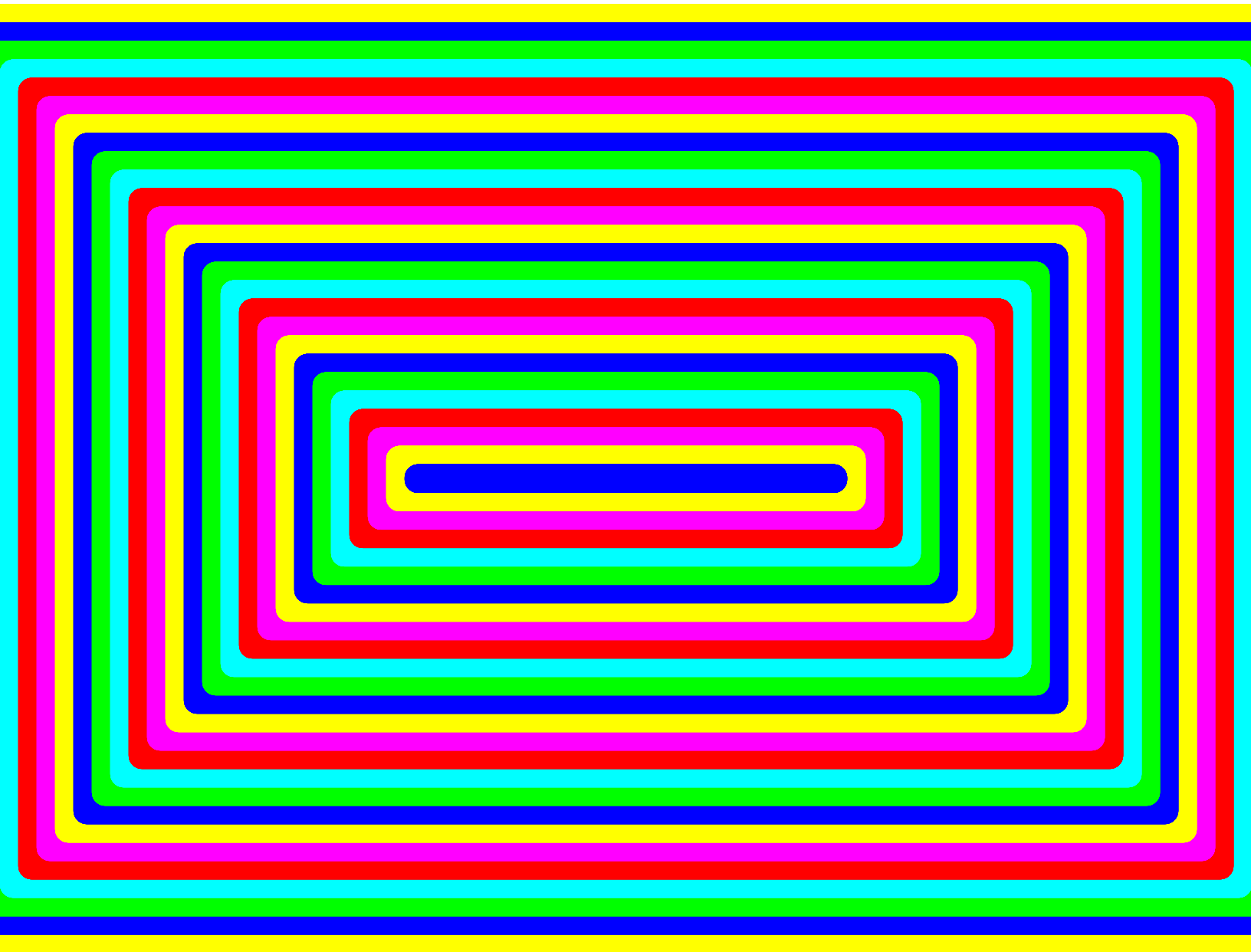


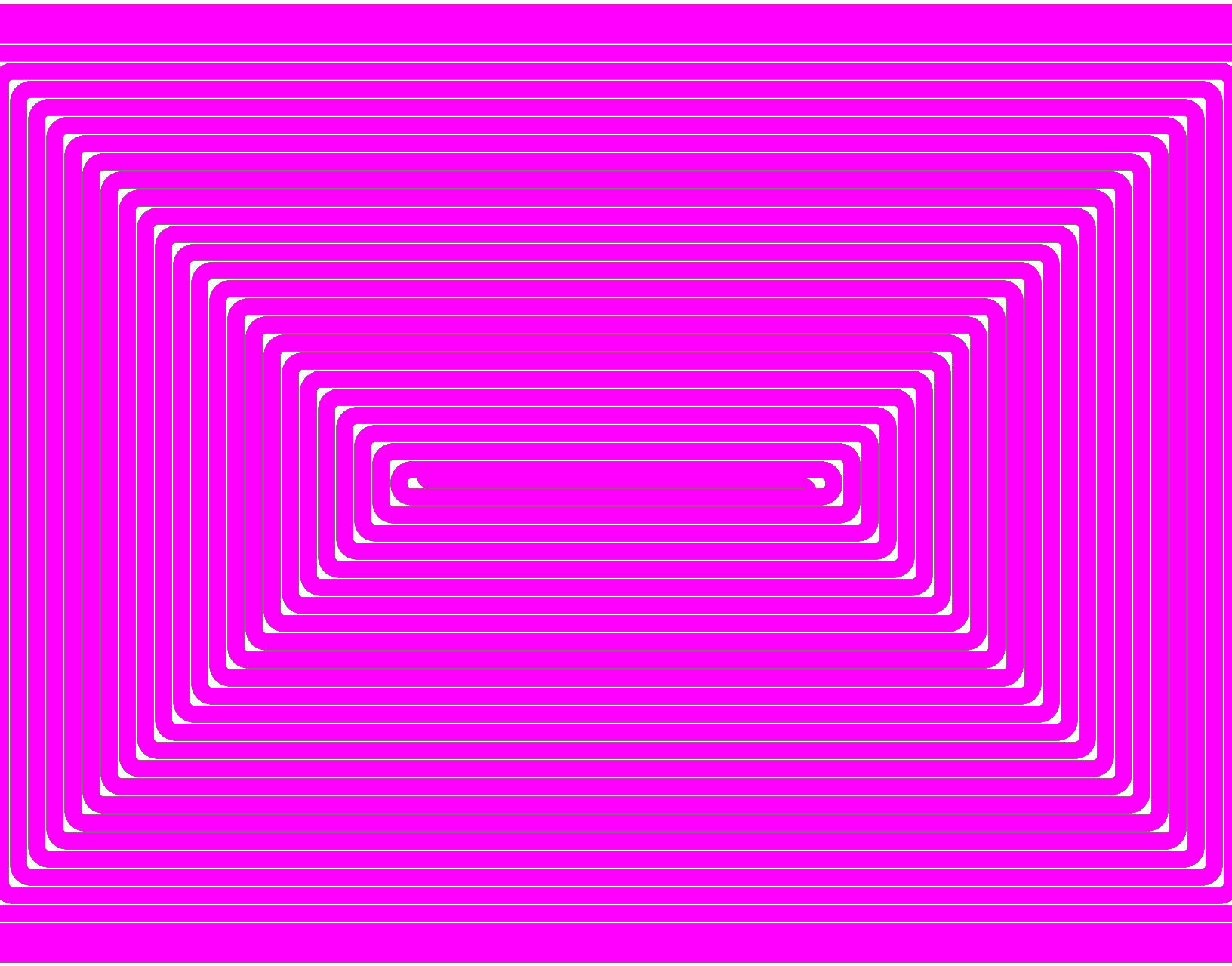


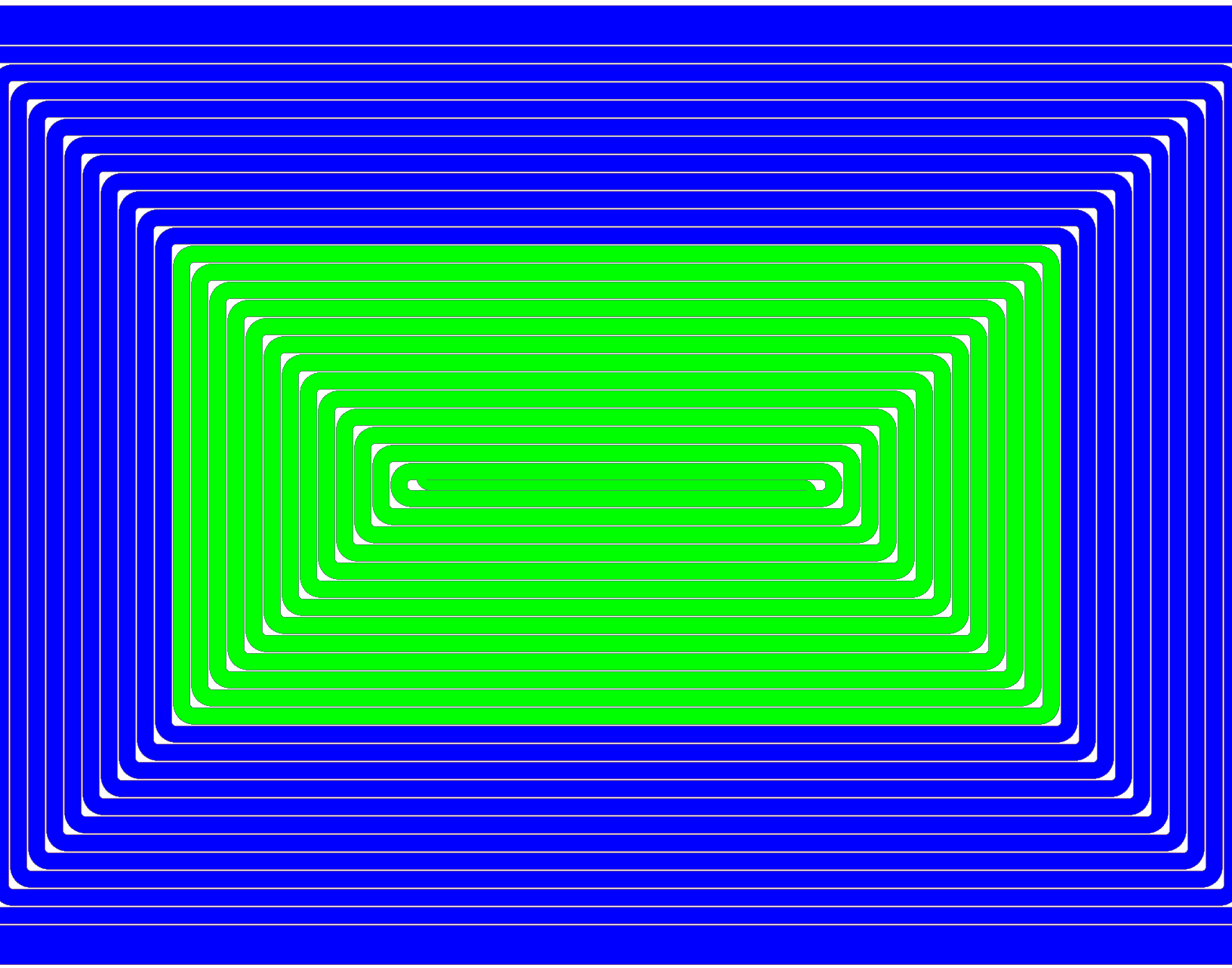


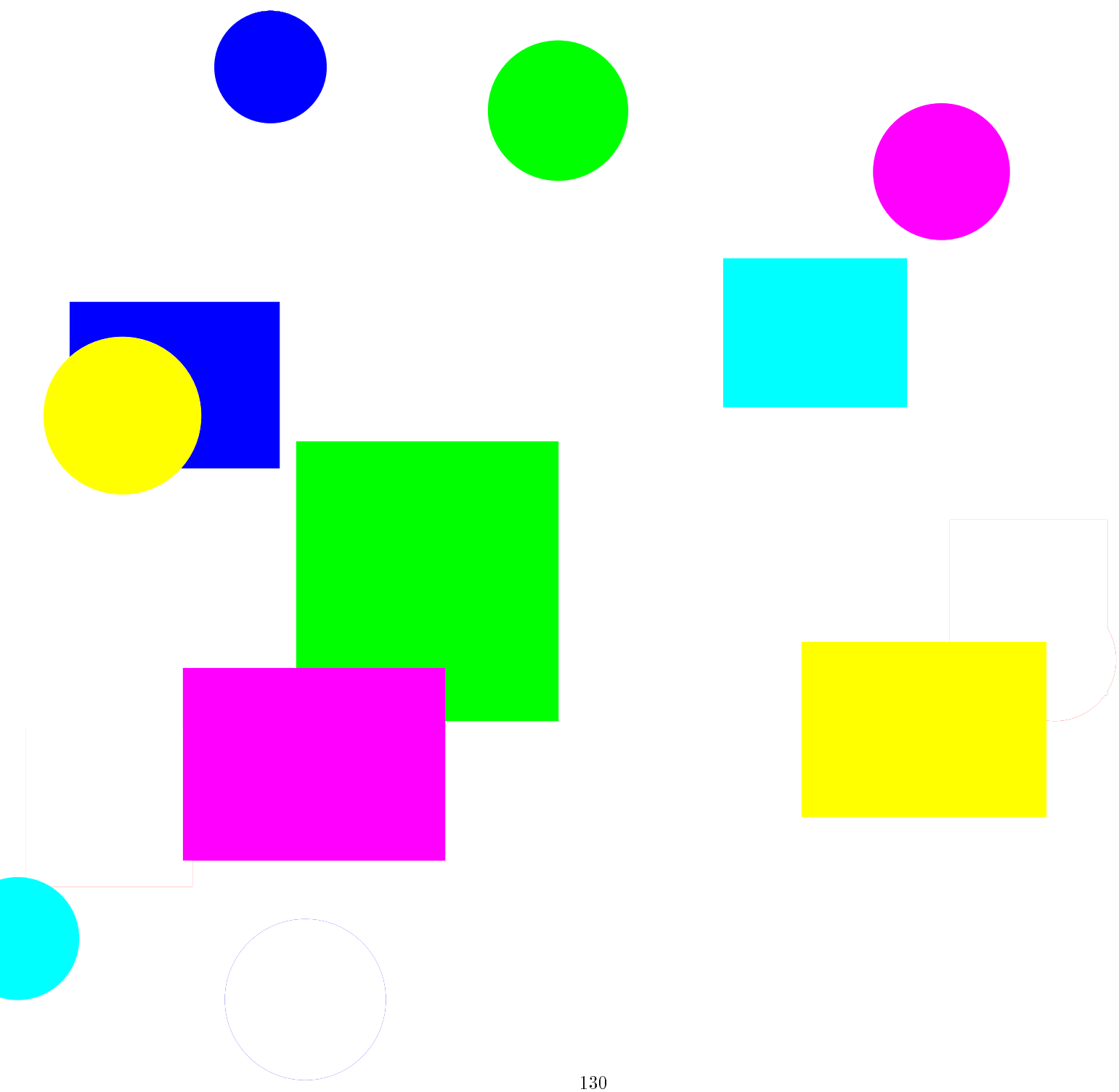












THE END