Brute force reduction

Yulia Burkatovskaya

Outline

- Independent and covering sets
- Backtracking
- Generation of independent sets of vertices
- Maximum independent set
- Branch-and-bound method
- Shortest vertex cover
- Travelling salesman problem

Independent and covering sets

- Covering sets
- Cover numbers
- Independent sets
- Independence numbers
- Cover and independence numbers theorem

Covering sets

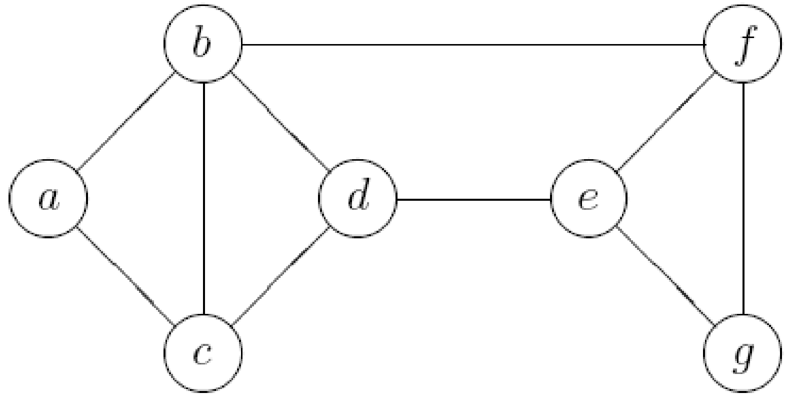
A vertex **covers** an edge if they are incident.

An edge **covers** a vertex if they are incident.

Example.

The vertex b covers the edges ab, bc, bd, bf

The edge ab covers the vertices a and b



Covering sets

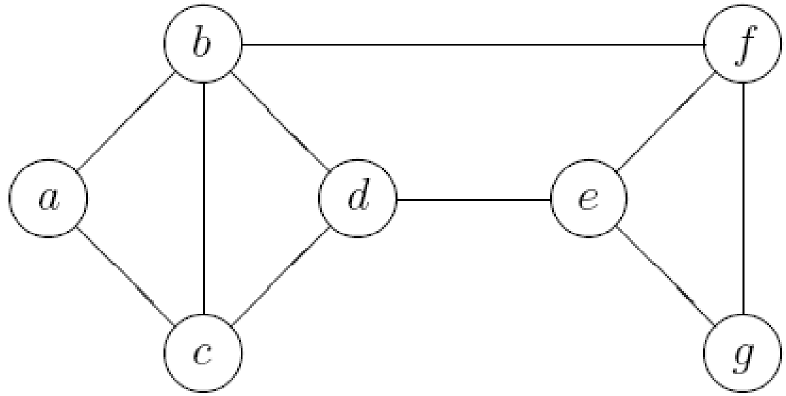
A **vertex covering set (vertex cover)** is a set of vertices of G covering all edges of G.

An **edge covering set (edge cover)** is a set of edges of G covering all vertices of G.

Example.

{a,b,d,e,f} – a vertex covering set.

{ab,ac,de,fg} – an edge covering set.



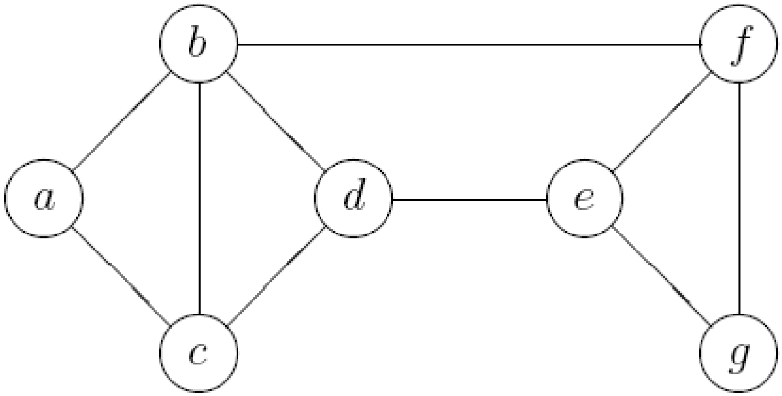
Covering sets

A cover is called **shortest** when it contains the smallest possible number of vertices (edges).

Example.

{a,b,c,d,e,f} is not a shortest vertex cover

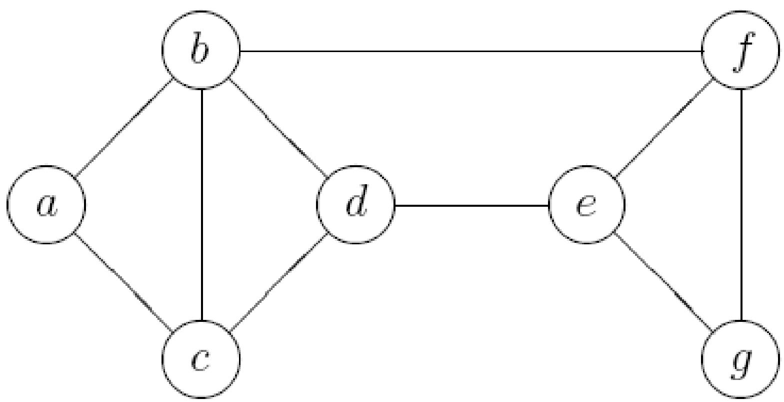
{b,c,e,g} is a shortest vertex cover.



Cover numbers

The **vertex cover number** α_0 of a graph G is the size of a shortest vertex cover in a graph, i.e., the minimum number of vertices covering all edges.

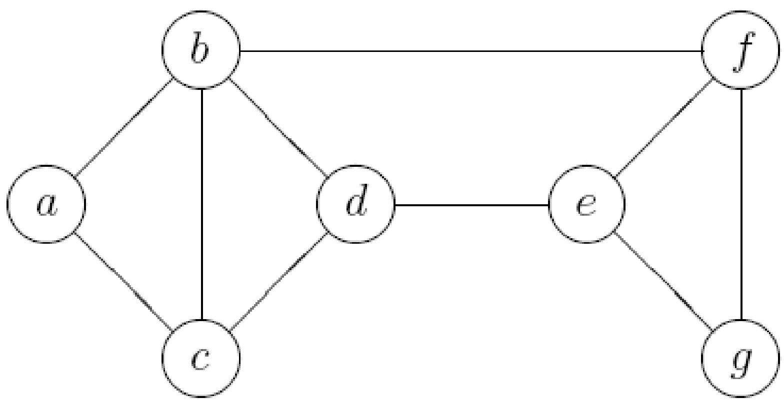
Example. α_0 =4, {b,c,e,f} – shortest vertex cover.



Cover numbers

The **edge cover number** α_1 of a graph G is the size of a shortest edge cover in a graph, i.e., the minimum number of edges covering all vertices.

Example. $\alpha_1 = 4$, {ab,cd,eg,ef} – shortest edge cover.



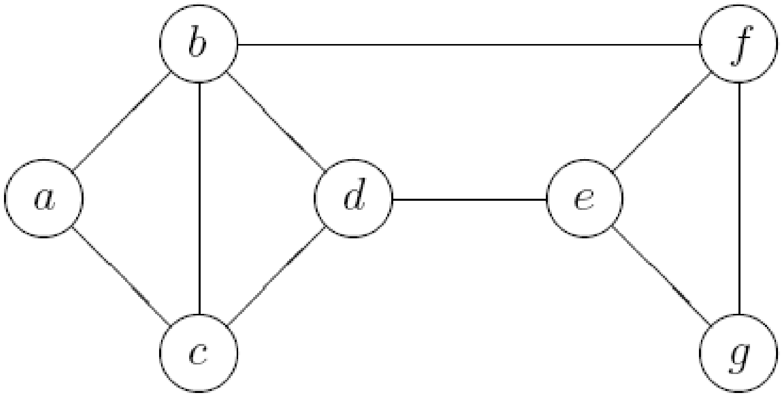
Independent sets

A **vertex (edge) independent set** is a set of vertices (edges) of G so that no two vertices (edges) of the set are adjacent.

Example.

{b,e} – independent vertex set.

{ab,cd,fg} – independent edge set.



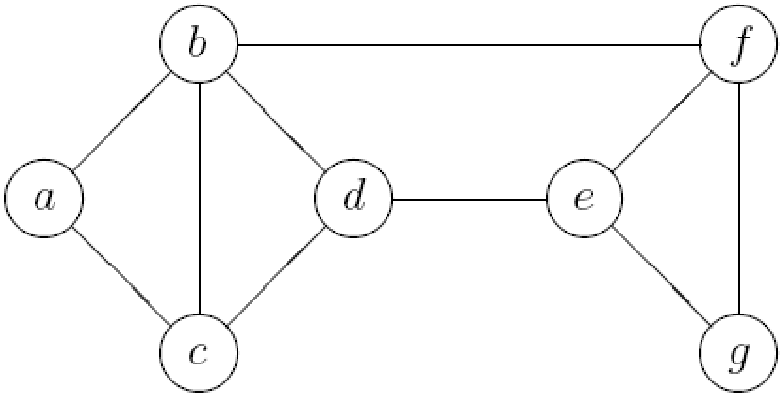
Independent sets

An independent set is called **maximum** when it contains the greatest number of vertices (edges).

Example.

{b,e} is not a maximum vertex independent set.

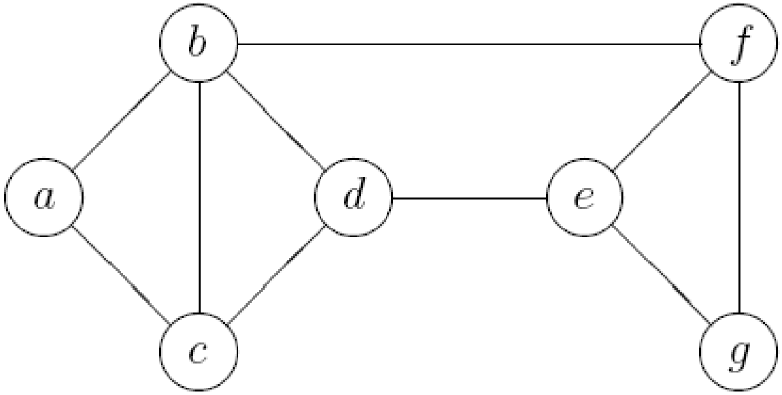
{a,d,f} is a maximum vertex independent set.



Independence numbers

The **vertex independence number** β_0 of a graph G is the maximum number of independent vertices.

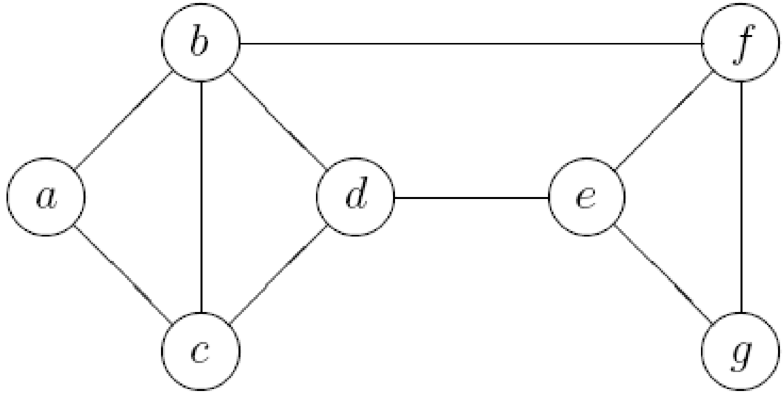
Example. $\beta_0 = 3$, {a,d,f} – independent vertex set.



Independence numbers

The **edge independence number** β_1 of a graph G is the maximum number of independent edges.

Example. $\beta_1 = 3$, {ab,cd,ef} – independent edge set.



Cover and independence numbers

	α_0	α_1	β_0	β_1
K_p				
$K_{m,n}$				
C_p				
Empty				

Cover and independence numbers

	α_0	α_1	β_0	β_1
K_p	p-1	p/2 (p–even), (p+1)/2 (p–odd)	1	p/2 (p–even), (p-1)/2 (p–odd)
$K_{m,n}$	min(m,n)	max(m,n)	max(m,n)	min(m,n)
C_p	p/2 (p–even), (p+1)/2 (p–odd)	p/2 (p–even), (p+1)/2 (p–odd)	p/2 (p–even), (p-1)/2 (p–odd)	p/2 (p–even), (p-1)/2 (p–odd)
Empty	0	no	p	0

Cover and independence number theorem

For every connected non-trivial graph

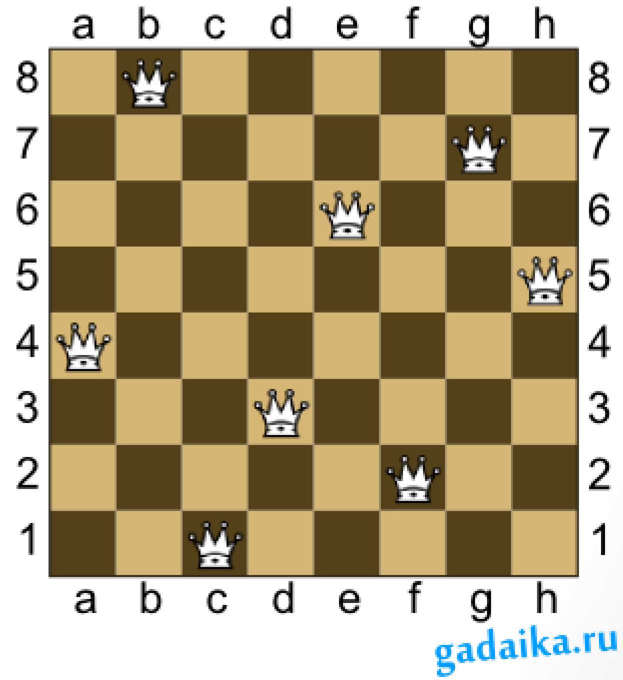
$$\alpha_0 + \beta_0 = \alpha_1 + \beta_1 = p.$$

Independent and covering sets of vertices

- Construction of independent sets
- Construction of covering sets
- Independent and covering sets
- Dominating sets
- Dominating and independent sets

Construction of independent sets

The **eight queens puzzle** is the problem of placing eight chess queens on an 8×8 chessboard so that no two queens threaten each other.

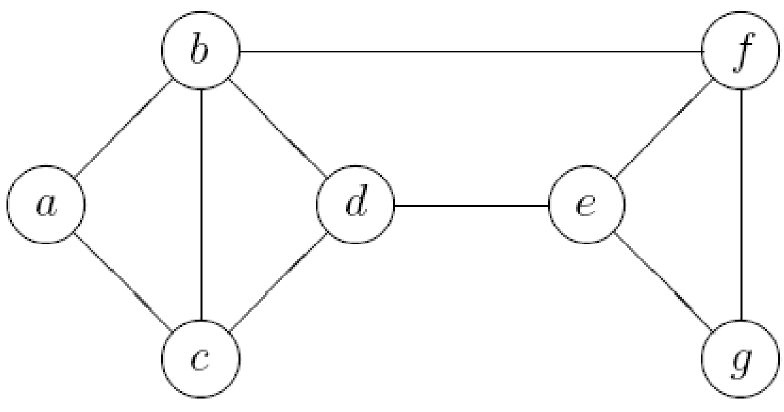


Construction of independent sets

An independent set is **maximal** if is not a subset of any other independent set.

In other words, there is no vertex outside the independent set that may join it.

Example. {a,d} is not a maximal independent set, {a,d,f} is a maximal independent set.



Construction of independent sets

- **Backtracking** is a general algorithm for finding all (or some) solutions to some computational problems, notably constraint satisfaction problems, that incrementally builds candidates to the solutions, and abandons a partial candidate ("backtracks") as soon as it determines that it cannot possibly be completed to a valid solution.

- https://www.youtube.com/watch?v=kX5frmc6B7c
- https://www.youtube.com/watch?v=xouin83ebxE

Construction of independent sets

Generalized Algorithm:

- Pick a starting point.
- While(Problem is not solved)
- For each path from the starting point.
  - check if selected path is safe,
  - if yes select it and make recursive call to rest of the problem
  - If recursive calls returns true, then return true. else undo the current move and return false.
- End For
- If none of the move works out, return false, NO SOLUTON.

Construction of independent sets

- S_k – obtained independent set of the cardinality k;
- Q_k – set of vertices that can be added to S_k ($\Gamma(S_k) \cap Q_k = \emptyset$);
- Q_k^- – vertices that have been used already to expand S_k;
- Q_k^+ – vertices that have not been used yet to expand S_k;

- Start: $k=0$, $S_k=\emptyset$, $Q_k^+=V$, $Q_k^-=\emptyset$.
- End:
  - if $Q_k^+=V$, $Q_k^-=\emptyset$ then the set can not be expand;
  - if there exists $u \in Q_k^-$ such as $\Gamma(u) \cap Q_k^+ = \emptyset$ then the obtaining set is not maximal as u can not be removed.

Construction of independent sets

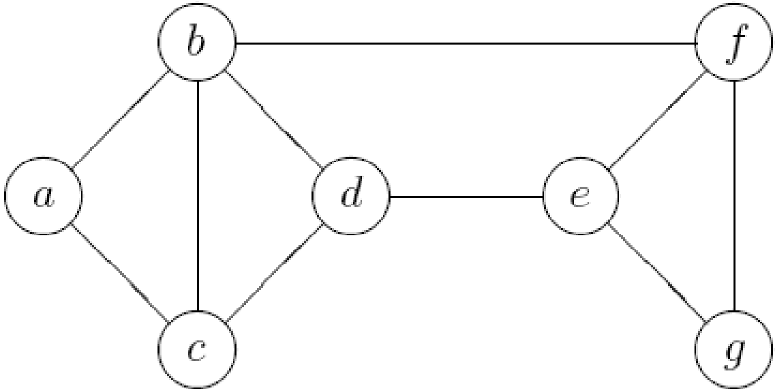
- Going ahead (from k to k+1):

$$S_{k+1} = S_k \cup \{v\};$$
$$Q^-_{k+1} = Q^-_k \setminus \Gamma(v);$$
$$Q^+_{k+1} = Q^+_k \setminus \{\Gamma(v) \cup v\};$$

- Going back (from k+1 to k):

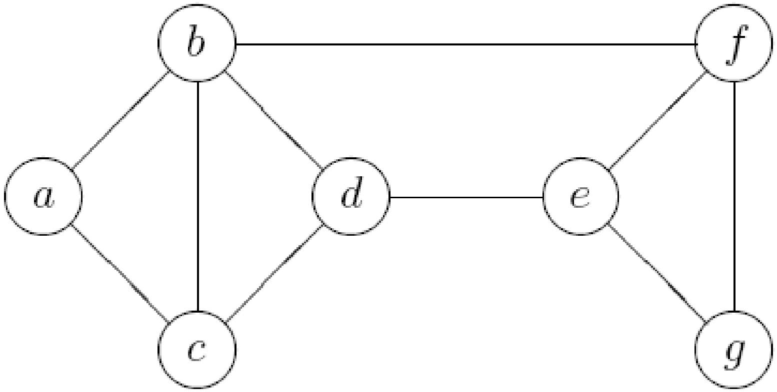
$$S_k = S_{k+1} \setminus \{v\};$$
$$Q^-_k = Q^-_k \cup \{v\};$$
$$Q^+_k = Q^+_k \setminus \{v\}.$$

Construction of independent sets



k	S_k	Q_k^+	Q_k^-
0	Ø	abcdefg	Ø
1	a	defg	Ø
2	ad	fg	Ø
3	adf	Ø	Ø
2	ad	g	f
3	adg	Ø	Ø
2	ad	Ø	fg
1	a	efg	d
2	ae	Ø	Ø

Construction of independent sets



k	S_k	Q_k^+	Q_k^-
2	ae	Ø	Ø
1	a	fg	ed
0	Ø	bcdefg	a
1	b	eg	Ø
2	be	Ø	Ø
1	b	g	e
2	bg	Ø	Ø
1	b	Ø	eg
0	Ø	cdefg	ab

Construction of covering sets

- ξ_j = 1 if and only if the vertex j belongs to the covering set;
- I is the incidence matrix;

The problem can be converted to the search of the shortest cover for the incidence matrix.

$$\sum_{j=1}^{p} c_j \xi_j \to \min;$$

$$\sum_{j=1}^{p} I_{jk} \xi_j \geq 1, \quad \forall k = 1, \ldots, q;$$

$$\xi_j \in \{0, 1\}.$$

Shortest cover of a Boolean matrix

Consider Boolean matrix Q.

Row i **covers** column j if $q_{ij}=1$, i.e. if row i contains 1 in column j.

A **cover** of a Boolean matrix is any set of its rows covering all its columns.

The **length** of a cover is the number of rows in the cover.

A **shortest cover** is a cover of the minimal length.

Shortest cover of a Boolean matrix

Example.

Row H covers columns 2, 3, 6, 8.

1	1									A
		1	1							B
				1	1					C
							1	1		D
						1			1	E
				1		1				F
								1	1	G
	1	1			1		1			H
1	2	3	4	5	6	7	8	9	10	

Shortest cover of a Boolean matrix

Example.

Covers: {A,B,C,D,E,F}, {A,B,C,E,G,H}…

Shortest cover: {A,B,C,D,E}

1	1									A
		1	1							B
				1	1					C
							1	1		D
						1			1	E
				1		1				F
								1	1	G
	1	1			1		1			H
1	2	3	4	5	6	7	8	9	10	

Essential row rule

An **essential row** is a row covering a column contained one and only one 1.

Essential row rule. If a Boolean matrix has an essential row hence this row is contained in any cover. An essential row is deleted from the matrix with all the columns covered by the row.

Essential row rule

Example.

Essential rows: A, B.

1	1									A
		1	1							B
				1	1					C
							1	1		D
						1			1	E
				1		1				F
								1	1	G
	1	1			1		1			H
1	2	3	4	5	6	7	8	9	10	

Essential row rule

Example.

After applying the essential row rule.

$$
Q' = \begin{array}{|c|c|c|c|c|c|l}
\hline
1 & 1 & & & & & C \\ \hline
 & & & 1 & 1 & & D \\ \hline
 & & 1 & & & 1 & E \\ \hline
1 & & 1 & & & & F \\ \hline
 & & & & 1 & 1 & G \\ \hline
 & 1 & & 1 & & & H \\ \hline
\multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} & \multicolumn{1}{c}{9} & \multicolumn{1}{c}{10} &
\end{array}
$$

Petrick's method

Column cover function is the disjunction of variables corresponding to rows covering the column.

Matrix cover function is the conjunction of all the column cover functions, i.e., conjunctive normal form (CNF).

If the CNF of the matrix cover function is transformed to the disjunctive normal form (DNF) then every conjunction of the DNF gives us a cover of the matrix. The shortest disjunction gives us the shortest cover.

Petrick's method

Example.

$$Q' = \begin{array}{|c|c|c|c|c|c|l}
\hline
1 & 1 & & & & & C \\ \hline
 & & & 1 & 1 & & D \\ \hline
 & & 1 & & & 1 & E \\ \hline
1 & & 1 & & & & F \\ \hline
 & & & & 1 & 1 & G \\ \hline
 & 1 & & 1 & & & H \\ \hline
\multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} & \multicolumn{1}{c}{9} & \multicolumn{1}{c}{10} &
\end{array}$$

$$(C \vee F)(C \vee H)(E \vee F)(D \vee H)(D \vee G)(E \vee G).$$

$$= CDE \vee CEGH \vee CDFG \vee DEFH \vee FGH$$

Predecessor row rule and successor column rule

Boolean vector $\alpha=a_1a_2\ldots a_n$ **precedes** Boolean vector $\beta=b_1b_2\ldots bn$ if for every $i=1,\ldots, n$:

$$a_i \leq b_i.$$

Here vector α is the **predecessor**, vector β is the **successor**.

Predecessor row rule. If in a Boolean matrix row α precedes row β then predecessor row α is deleted from the matrix. The shortest cover does not lost because in every cover row α can be replaced by row β.

Successor column rule. If in a Boolean matrix column γ precedes column δ then successor column δ is deleted from the matrix. The shortest cover does not lost because every cover of column γ also covers column δ..

Predecessor row rule and successor column rule

Example.

$$A \preceq C$$

$$Q_1'' = \begin{array}{|c|c|c|c|c|c|c}
\hline
 & 1 & & 1 & & & A \\ \hline
1 & 1 & & 1 & & & C \\ \hline
1 & & & & 1 & 1 & D \\ \hline
1 & 1 & & & 1 & & E \\ \hline
 & & 1 & 1 & & 1 & F \\ \hline
 & & 1 & & 1 & 1 & G \\ \hline
\multicolumn{1}{c}{1} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} &
\end{array}
\qquad
Q_2'' = \begin{array}{|c|c|c|c|c|c|c}
\hline
1 & 1 & & 1 & & & C \\ \hline
1 & & & & 1 & 1 & D \\ \hline
1 & 1 & & & 1 & & E \\ \hline
 & & 1 & 1 & & 1 & F \\ \hline
 & & 1 & & 1 & 1 & G \\ \hline
\multicolumn{1}{c}{1} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} &
\end{array}$$

Predecessor row rule and successor column rule

Example. 4≤1.

$$
Q_2'' = \begin{array}{|c|c|c|c|c|c|c}
\hline
1 & 1 &   & 1 &   &   & C \\
\hline
1 &   &   &   & 1 & 1 & D \\
\hline
1 & 1 &   &   & 1 &   & E \\
\hline
  &   & 1 & 1 &   & 1 & F \\
\hline
  &   & 1 &   & 1 & 1 & G \\
\hline
\multicolumn{1}{c}{1} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} &
\end{array}
\qquad
Q_3'' = \begin{array}{|c|c|c|c|c|c}
\hline
1 &   & 1 &   &   & C \\
\hline
  &   &   & 1 & 1 & D \\
\hline
1 &   &   & 1 &   & E \\
\hline
  & 1 & 1 &   & 1 & F \\
\hline
  & 1 &   & 1 & 1 & G \\
\hline
\multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} &
\end{array}
$$

Algorithm

Step 1. If there is a row covering all columns then add this row to the shortest cover and go to the end.

Step 2. Apply the essential row rule. If there were core rows then go to step 1.

Step 3. Apply the predecessor row rule. If there were predecessor rows then go to step 2.

Step 4. Apply the successor column rule. If there were successor columns then go to step 3.

Step 5. Write the CNF of the matrix cover function and transform it into DNF.

Step 6. Choose the shortest conjunction of the DNF and add core rows to the rows from this conjunction. The shortest cover is obtained.

Algorithm

Example. 8=5.

$$Q''_4 = \begin{array}{|c|c|c|c|c|l}
\hline
1 & & 1 & & & C \\
\hline
1 & & & 1 & & E \\
\hline
& 1 & 1 & & 1 & F \\
\hline
& 1 & & 1 & 1 & G \\
\hline
\multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} &
\end{array}
\qquad
Q''_5 = \begin{array}{|c|c|c|c|l}
\hline
1 & & 1 & & C \\
\hline
1 & & & 1 & E \\
\hline
& 1 & 1 & & F \\
\hline
& 1 & & 1 & G \\
\hline
\multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} &
\end{array}$$

The shortest c

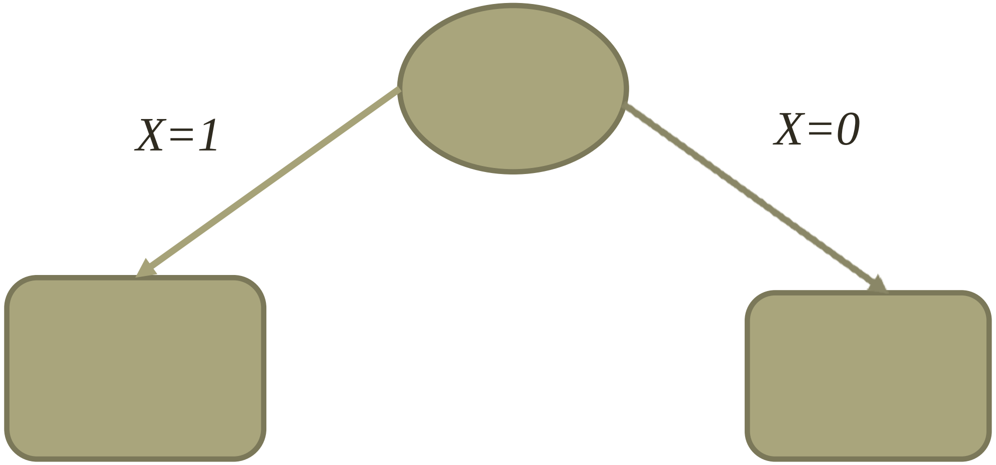
$$(C \vee E)(F \vee G)(C \vee F)(E \vee G) =$$
$$= (C \vee EF)(G \vee EF) = CG \vee EF$$

Direct tree search (branch-and-bound method)

Consider Boolean matrix M and row X.

X=1: row X is included into a cover. Delete row X and all columns covered by it, hence obtain matrix M'.

X=0: row X is not included into a cover. Delete row X, hence obtain matrix M'.



Direct tree search

Example. A, B – essential rows.

1	1									A
		1	1							B
				1	1					C
							1	1		D
						1			1	E
				1		1				F
								1	1	G
	1	1			1		1			H
1	2	3	4	5	6	7	8	9	10	

Direct tree search

Example. Choose row C. Matrix N: C=1, matrix P: C=0.

$$Q = \begin{array}{|c|c|c|c|c|c|l}
\hline
1 & 1 & & & & & C \\ \hline
 & & & 1 & 1 & & D \\ \hline
 & & 1 & & & 1 & E \\ \hline
1 & & 1 & & & & F \\ \hline
 & & & & 1 & 1 & G \\ \hline
 & 1 & & 1 & & & H \\ \hline
\multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} & \multicolumn{1}{c}{9} & \multicolumn{1}{c}{10} &
\end{array}$$

Direct tree search

Example. Choose row C. Matrix N: C=1, matrix P: C=0.

$$N = \begin{array}{cccc|c}
 & 1 & 1 & & D \\
1 & & & 1 & E \\
1 & & & & F \\
 & & 1 & 1 & G \\
 & 1 & & & H \\
\hline
7 & 8 & 9 & 10 &
\end{array}
\qquad
P = \begin{array}{cccccc|c}
 & & & 1 & 1 & & D \\
 & & 1 & & & 1 & E \\
1 & & 1 & & & & F \\
 & & & & 1 & 1 & G \\
 & 1 & & 1 & & & H \\
\hline
5 & 6 & 7 & 8 & 9 & 10 &
\end{array}$$

Direct tree search

Example. Consider matrix N.

$$N = \begin{array}{|c|c|c|c|l} \hline & 1 & 1 & & D \\ \hline 1 & & & 1 & E \\ \hline 1 & & & & F \\ \hline & & 1 & 1 & G \\ \hline & 1 & & & H \\ \hline \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} & \multicolumn{1}{c}{9} & \multicolumn{1}{c}{10} & \end{array}$$

$$N = \begin{array}{|c|c|c|c|l} \hline & 1 & 1 & & D \\ \hline 1 & & & 1 & E \\ \hline & & 1 & 1 & G \\ \hline \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} & \multicolumn{1}{c}{9} & \multicolumn{1}{c}{10} & \end{array}$$

$$F \preceq E,\ H \preceq D$$

Direct tree search

Example. E – an essential row.

$$
N = \begin{array}{|c|c|c|c|c}
\cline{1-4}
 & 1 & 1 & & D \\
\cline{1-4}
1 & & & 1 & E \\
\cline{1-4}
 & & 1 & 1 & G \\
\cline{1-4}
\multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} & \multicolumn{1}{c}{9} & \multicolumn{1}{c}{10} &
\end{array}
\qquad\qquad
N = \begin{array}{|c|c|c}
\cline{1-2}
1 & 1 & D \\
\cline{1-2}
 & 1 & G \\
\cline{1-2}
\multicolumn{1}{c}{8} & \multicolumn{1}{c}{9} &
\end{array}
$$

Row D covers all columns. We obtain a cover {A,B,C,D,E}.

Direct tree search

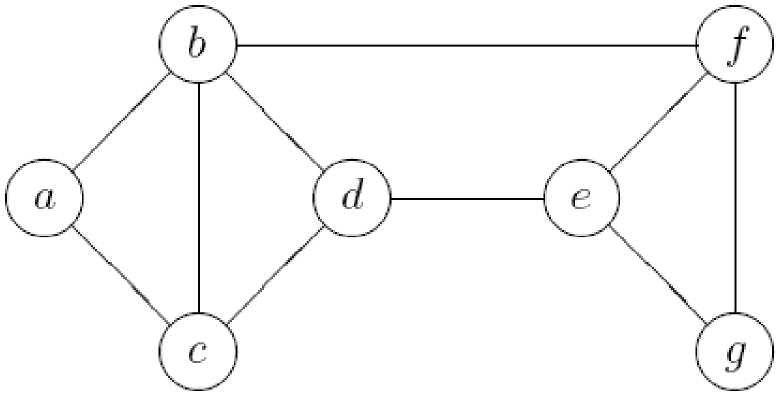
Example. Consider matrix P. F, H – essential rows.

$$P = \begin{array}{cc} \left[\begin{array}{cccccc} & & & 1 & 1 & \\ & & 1 & & & 1 \\ 1 & & 1 & & & \\ & & & & 1 & 1 \\ & 1 & & 1 & & \end{array}\right] & \begin{array}{c} D \\ E \\ F \\ G \\ H \end{array} \\ \begin{array}{cccccc} 5 & 6 & 7 & 8 & 9 & 10 \end{array} & \end{array}$$

$$P = \begin{array}{cc} \left[\begin{array}{cc} 1 & \\ & 1 \\ 1 & 1 \end{array}\right] & \begin{array}{c} D \\ E \\ G \end{array} \\ \begin{array}{cc} 9 & 10 \end{array} & \end{array}$$

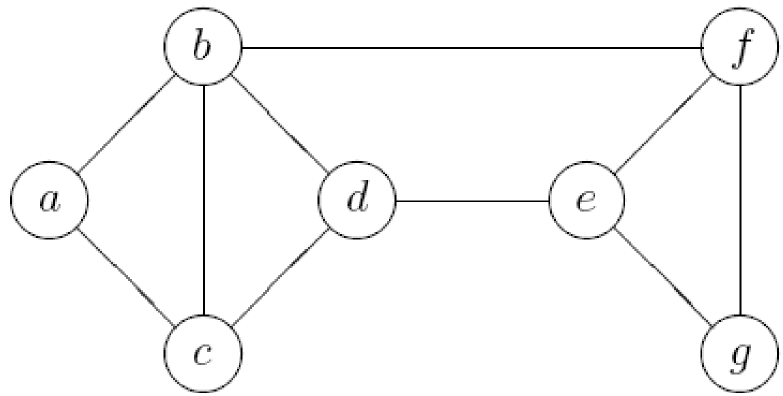
Row G covers all columns. We obtain a cover {A,B,F,G,H}.

Construction of covering sets



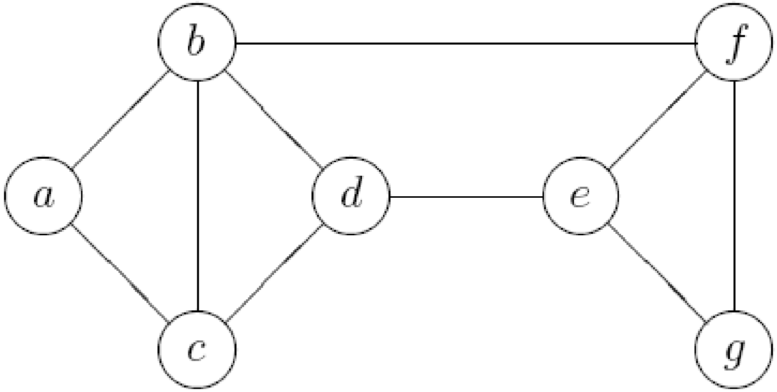
a	1	1								
b	1		1	1		1				
c		1	1		1					
d				1	1		1			
e							1	1	1	
f						1		1		1
g									1	1

Construction of covering sets



a	1	1								
c		1	1		1					
d				1	1		1			
e							1	1	1	
f						1		1		1
g									1	1

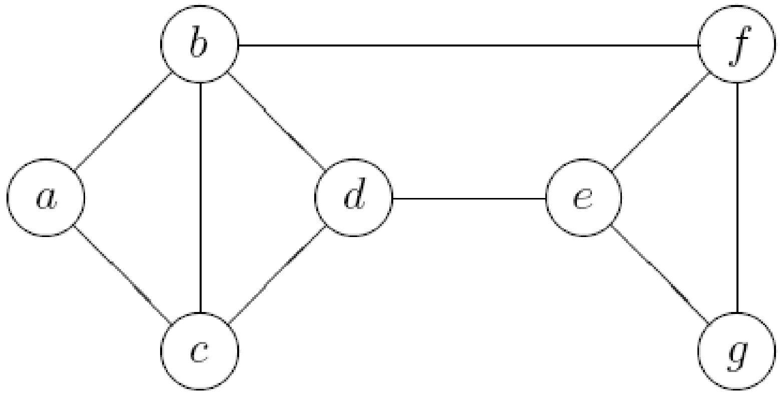
Construction of covering sets



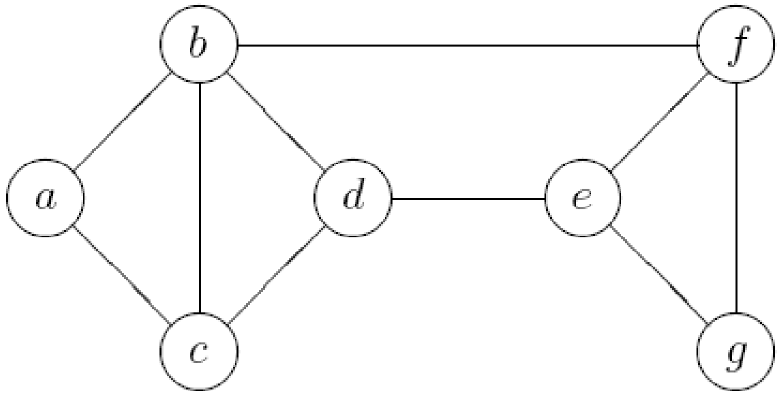
e	1
g	1

Construction of covering sets

- Cover: acdef

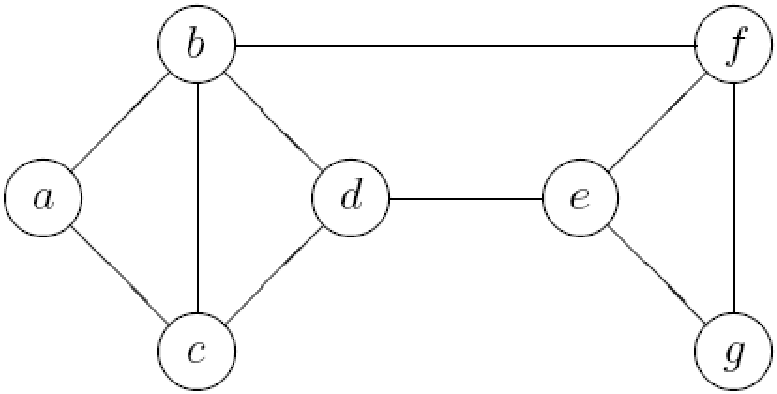


Construction of covering sets



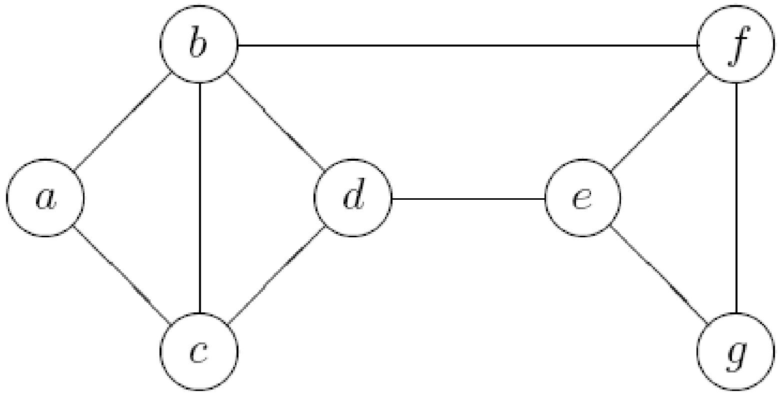
a	1	1								
b	1		1	1		1				
c		1	1		1					
d				1	1		1			
e							1	1	1	
f						1		1		1
g									1	1

Construction of covering sets



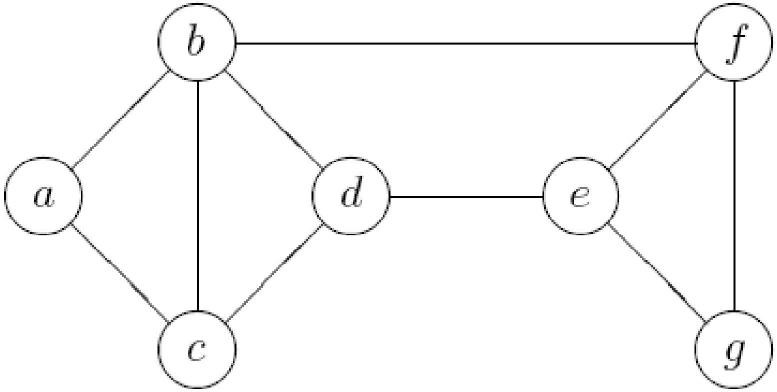
a	1					
c	1	1				
d		1	1			
e			1	1	1	
f				1		1
g					1	1

Construction of covering sets



a	1		
c	1	1	
d		1	
f			1
g			1

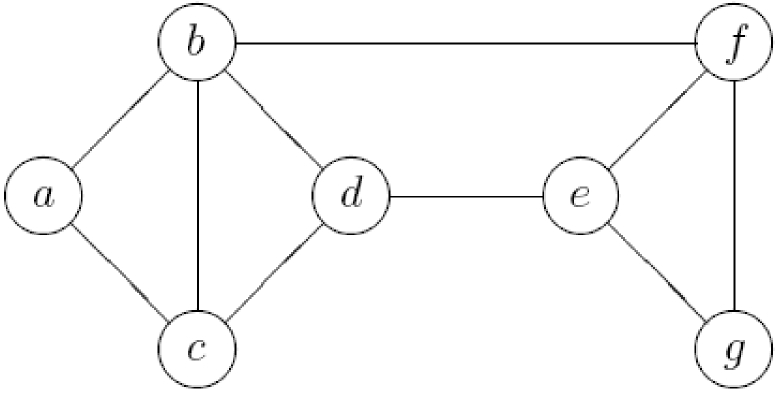
Construction of covering sets



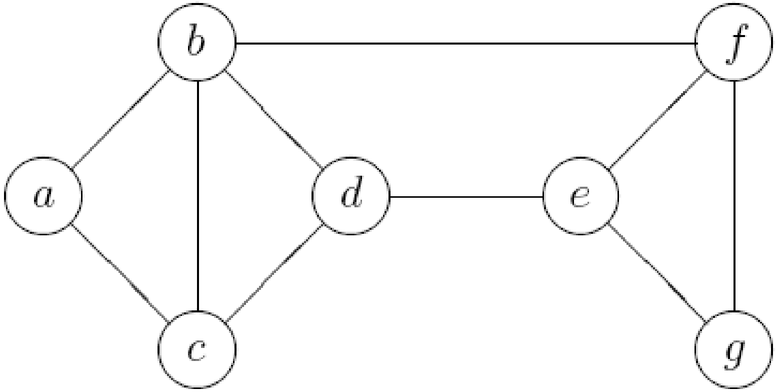
c	1	1	
f			1

Construction of covering sets

- Cover: bcef

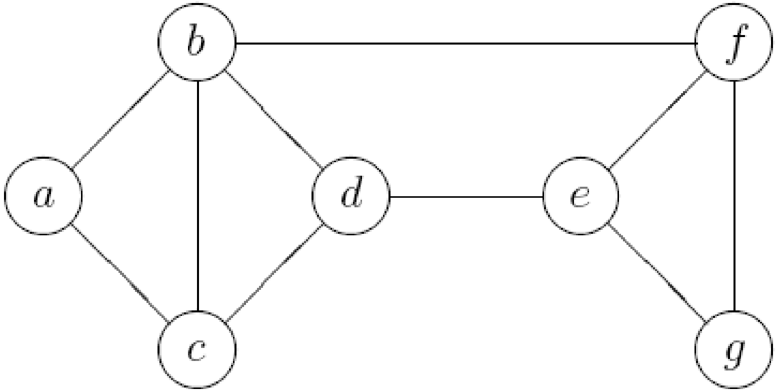


Construction of covering sets



a	1					
c	1	1				
d		1	1			
f				1		1
g					1	1

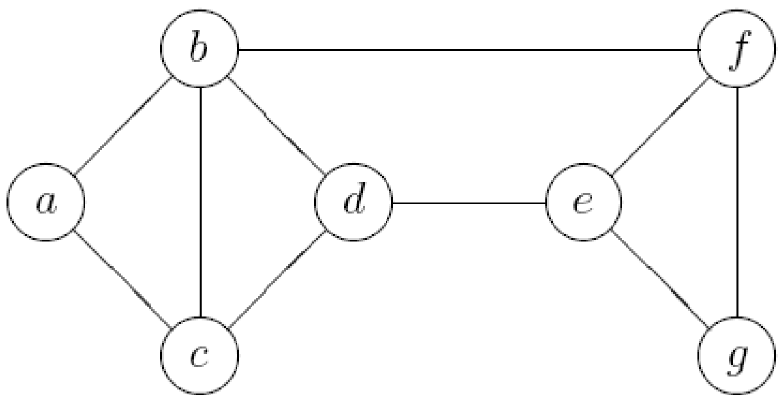
Construction of covering sets



a	1
c	1

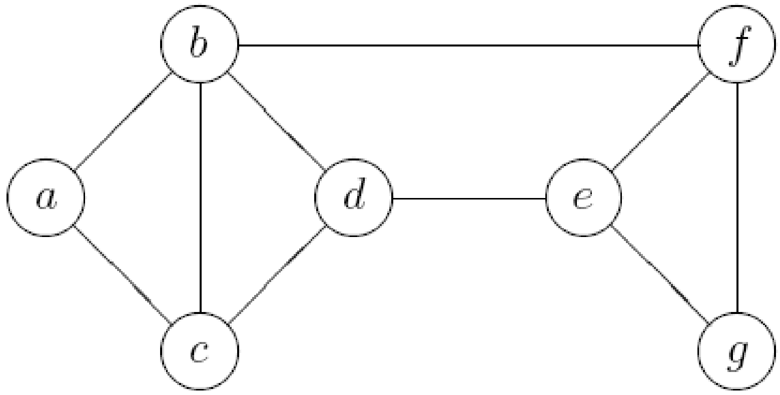
Construction of covering sets

- Cover: abdfg

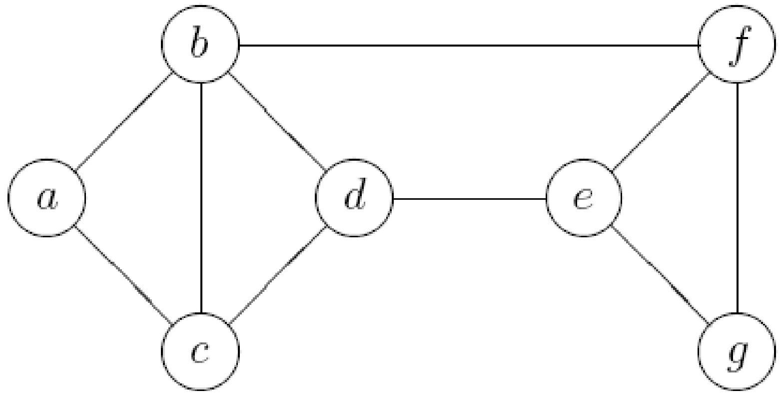


Construction of covering sets

- Shortest cover: bcef



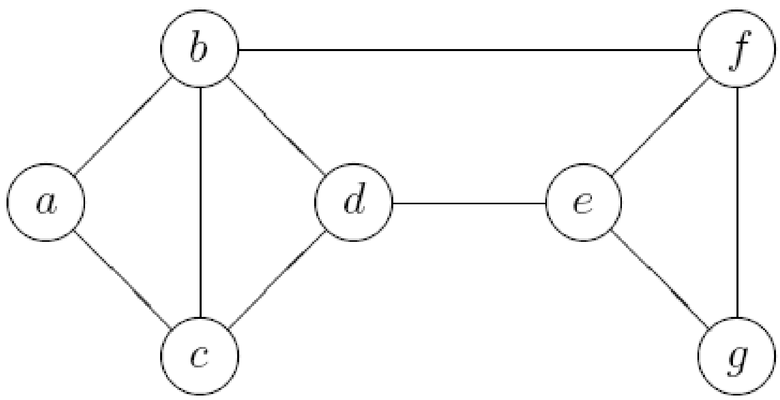
Construction of covering sets



a	1	1								
b	1		1	1		1				
c		1	1		1					
d				1	1		1			
e							1	1	1	
f						1		1		1
g									1	1

Construction of covering sets

- Shortest cover: bcef

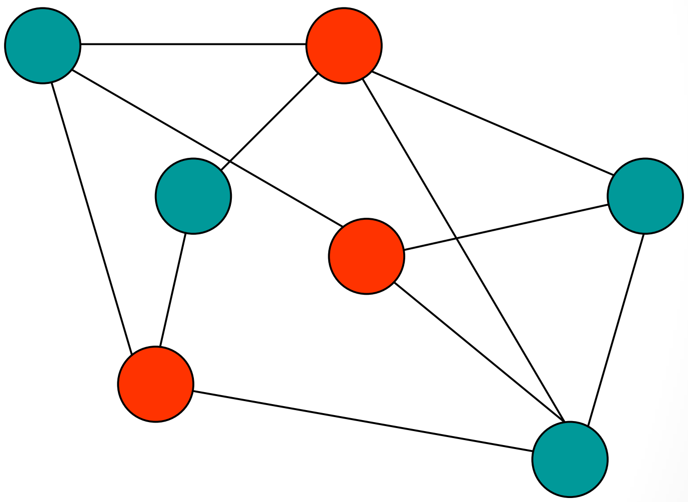


Independent and covering sets

A set is independent if and only if its complement is a vertex cover.

A set is covering if and only if its complement is an independent set.

Example. Red – independent, blue – covering.

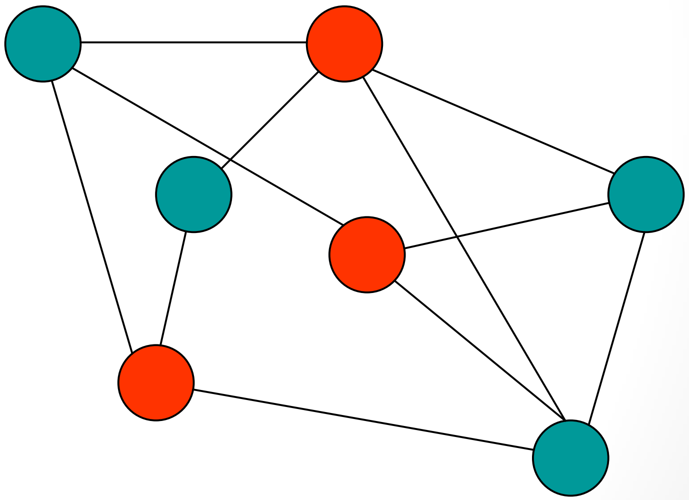


Independent and covering sets

The complement of a maximum independent set is a minimum vertex cover.

The complement of a minimum vertex cover is a maximum independent set.

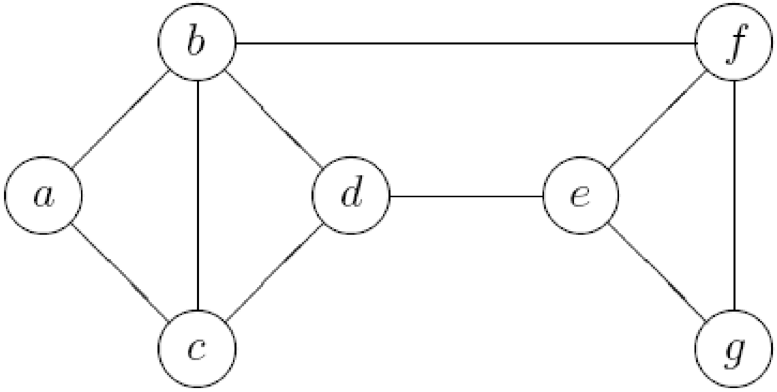
A solution of one problem gives a solution of another problem.



Dominating sets

A **dominating set** for a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one member of D.

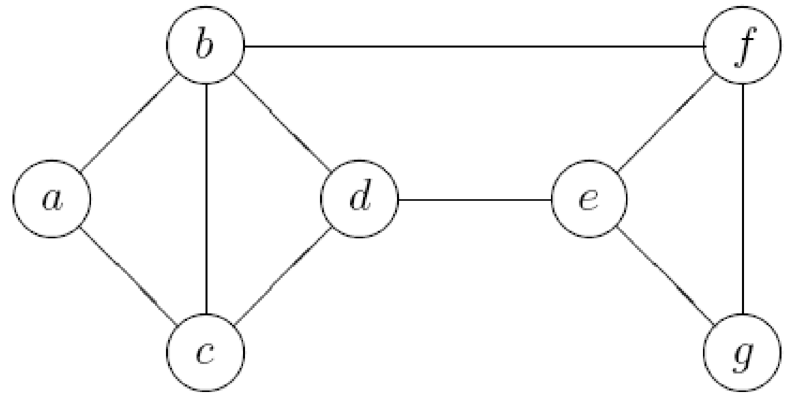
Example. {a,d,f} – dominating set.



Dominating sets

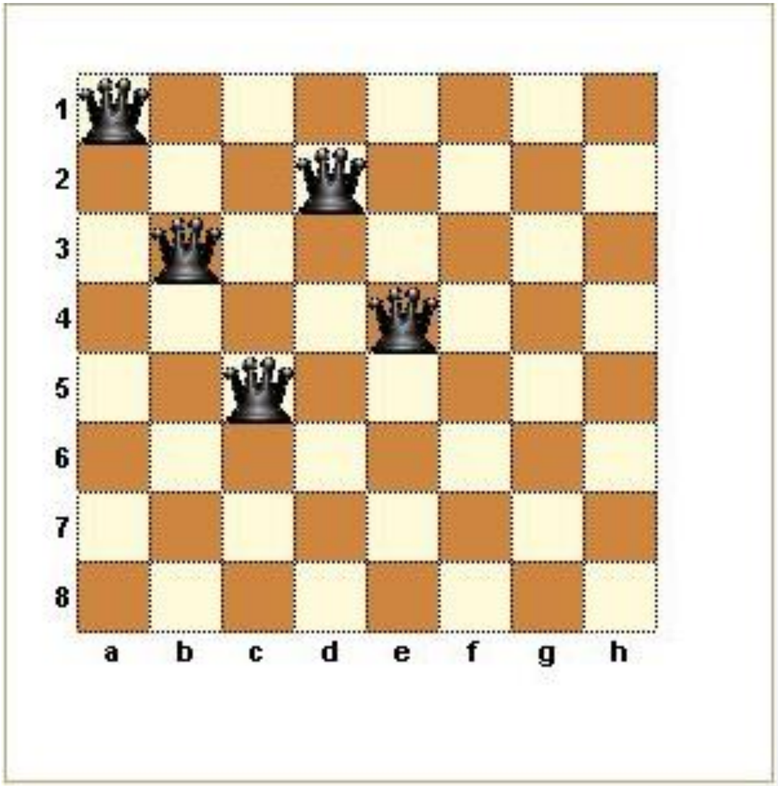
The **domination number** γ(*G*) is the number of vertices in a smallest dominating set for *G*. The set is called as **minimum dominating set**.

Example. {b,f} – a minimum dominating set, γ(*G*)=2.



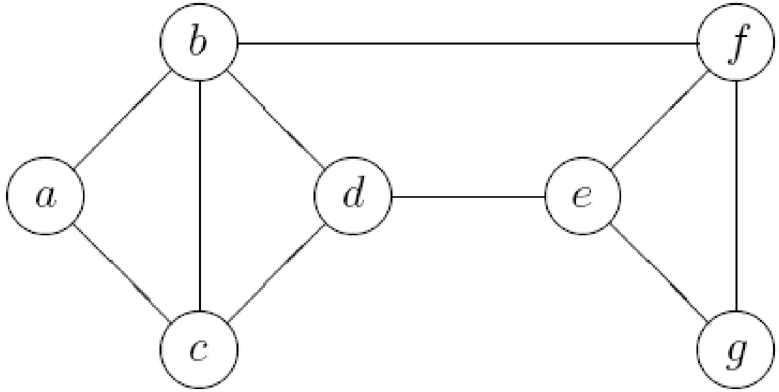
Dominating sets

The **five queens puzzle** is the problem of placing five chess queens on an 8×8 chessboard so that the queens can attack all the board.



Dominating sets

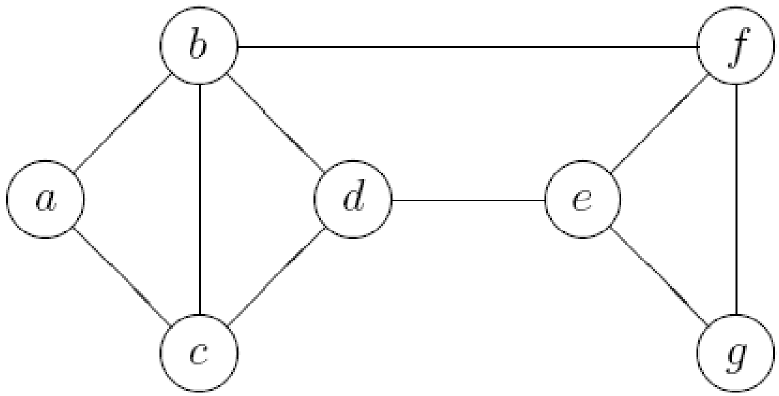
The **domination matrix** for a graph G(V,E) is its adjacency matrix where all elements of the main diagonal are equal to unity.



	a	b	c	d	e	f	g
a	1	1	1				
b	1	1	1	1		1	
c	1	1	1	1			
d		1	1	1	1		
e				1	1	1	1
f		1			1	1	1
g					1	1	1

Dominating sets

The **minimum dominating set** correspond to the shortest cover of the domination matrix.

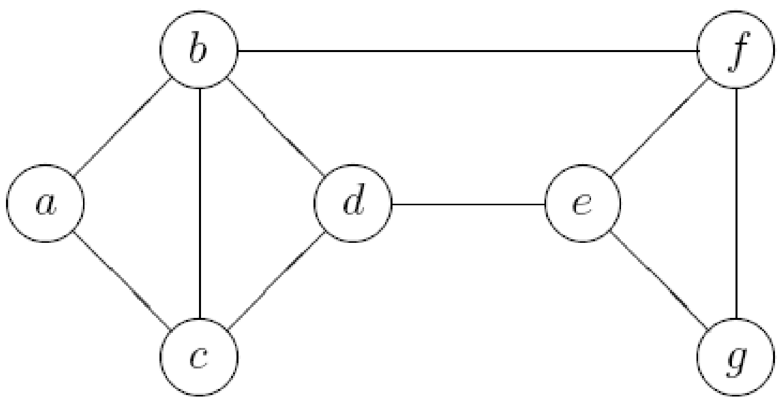


	a	b	c	d	e	f	g
a	1	1	1				
b	1	1	1	1		1	
c	1	1	1	1			
d		1	1	1	1		
e				1	1	1	1
f		1			1	1	1
g					1	1	1

Dominating sets

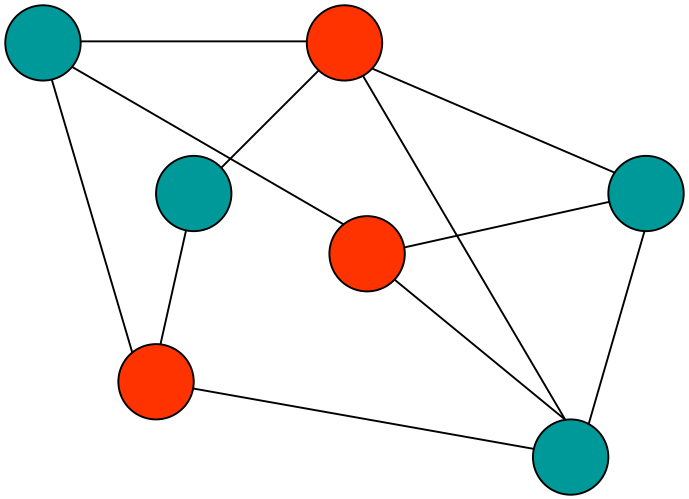
The **minimal dominating set** is a dominating set that does not contain any other dominating set.

Example. {b,e,f} is not a minimal dominating set, {b,f} is a minimal dominating set.



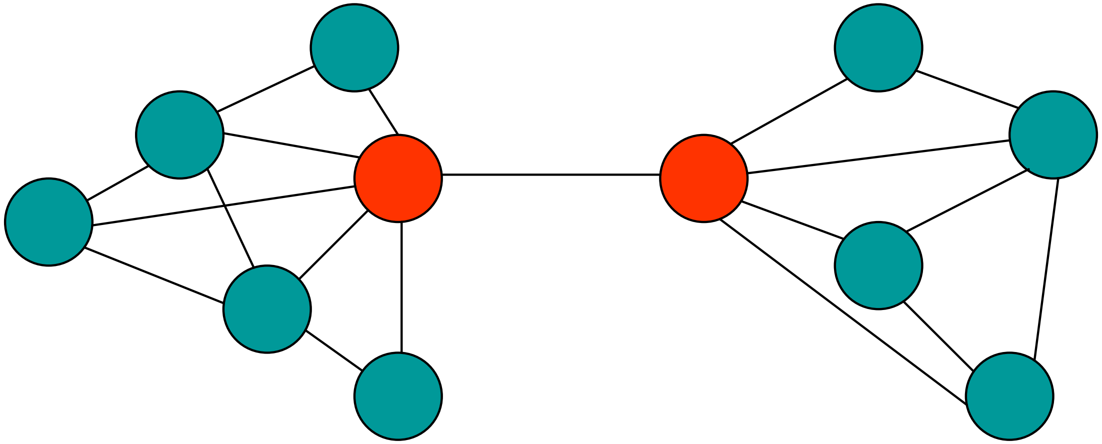
Dominating and independent sets

An **independent set** is also a **dominating set** if and only if it is a **maximal independent set**, so any **maximal independent set** in a graph is also a **minimal dominating set**.



Dominating and independent sets

A **dominating set** is not necessary an **independent set**.

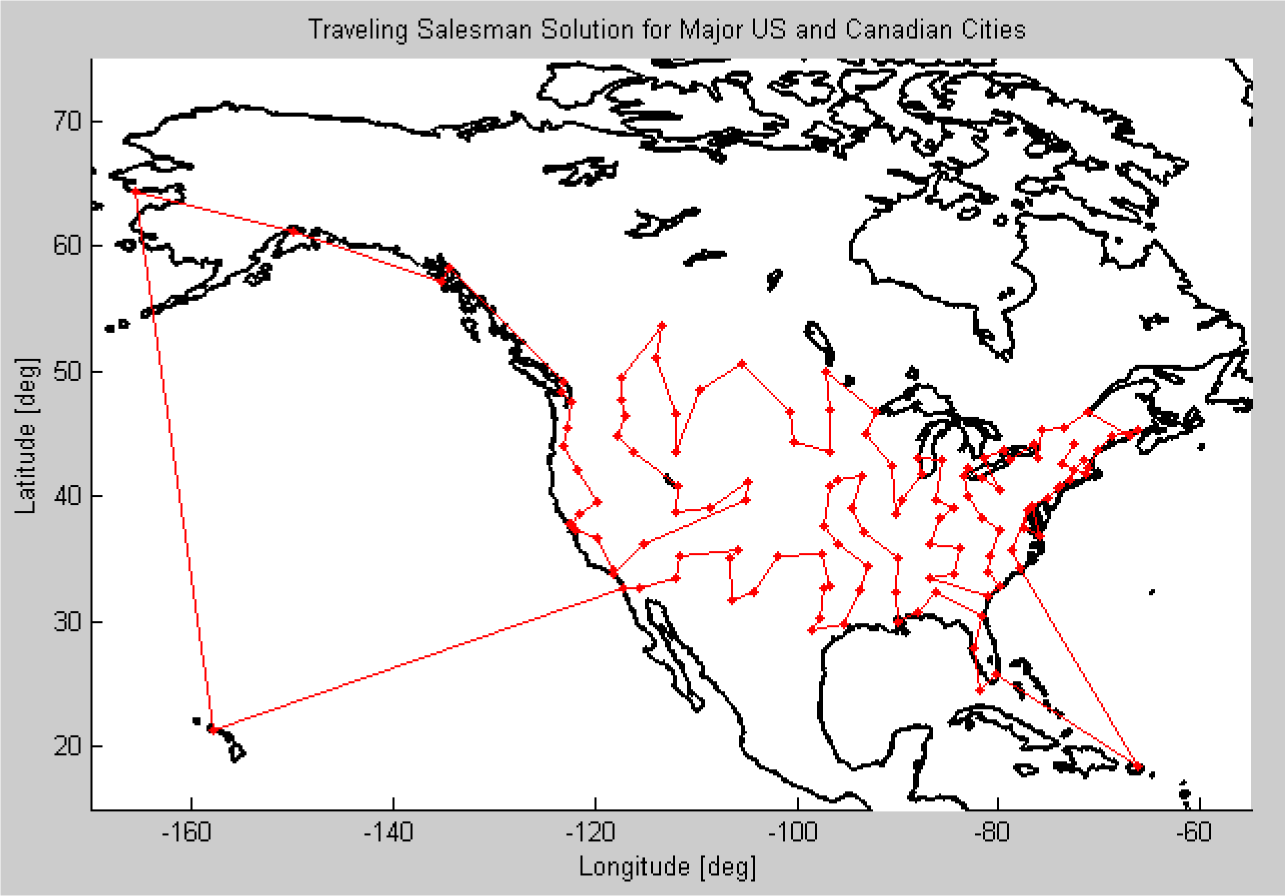


Traveling Salesman Problem

The **travelling salesman problem** (**TSP**) asks the following question:

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

Traveling Salesman Problem



Traveling Salesman Problem

Consider graph G(V,E) in which each vertex represents a city and each edge represents a road connecting two cities, and $d(x,y)$ is the weight of the edge (x,y) standing by the distance between cities x and y.

Problem: finding a cycle in G which visits each vertex once in minimum total distance.

This problem is **NP-hard**.

Traveling Salesman Problem

The **travelling salesman problem** (**TSP**) asks the following question:

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

Branch-and-bound method

- Low bound
- Branching
- Cutting branches

Low bound

Consider a graph G(V,E) with the matrix of weights W.

The minimum element of the row *i* gives us the minimum distance from *i* to another vertex. It can be understood as the minimum price we pay to leave the city *i*.

Example.

The minimum element

are written on the right.

	a	b	c	d	e	f	
a		8	4	6	12	3	3
b	7		13	9	1	7	1
c	2	13		16	11	6	2
d	5	7	18		3	14	3
e	10	2	12	4		13	2
f	2	8	3	11	10		2

Low bound

Then we subtract the minimum elements from the elements of the corresponding rows.

The minimum element of the column *j* gives us the minimum distance from another vertex to *j*. It can be understood as the minimum price we pay to come to the city *j*.

Example.

The minimum element are written underneath.

	a	b	c	d	e	f
a		5	1	3	9	0
b	6		12	8	0	6
c	0	11		14	9	4
d	3	5	16		0	11
e	8	0	10	2		11
f	0	6	1	9	8	
	0	0	1	2	0	0

Low bound

Then we subtract the minimum elements from the elements of the corresponding columns. This process is called **reduction** of the matrix. The obtained matrix is called **reduced matrix**.

Example.

$$A = \begin{array}{c|c|c|c|c|c|c|} & a & b & c & d & e & f \\ \hline a & & 5 & 0 & 1 & 9 & 0 \\ \hline b & 6 & & 11 & 6 & 0 & 6 \\ \hline c & 0 & 11 & & 12 & 9 & 4 \\ \hline d & 3 & 5 & 15 & & 0 & 11 \\ \hline e & 8 & 0 & 9 & 0 & & 11 \\ \hline f & 0 & 6 & 0 & 7 & 8 & \\ \hline \end{array}$$

Low bound

The low bound of the TSP solution is the sum of the minimum elements found during reducing of the weight matrix.

Example.

$$\underline{C}(A) = 3 + 1 + 2 + 3 + 2 + 2 + 0 + 0 + 1 + 2 + 0 + 0 = 16.$$

	a	b	c	d	e	f	
a		8	4	6	12	3	3
b	7		13	9	1	7	1
c	2	13		16	11	6	2
d	5	7	18		3	14	3
e	10	2	12	4		13	2
f	2	8	3	11	10		2

	a	b	c	d	e	f
a		5	1	3	9	0
b	6		12	8	0	6
c	0	11		14	9	4
d	3	5	16		0	11
e	8	0	10	2		11
f	0	6	1	9	8	
	0	0	1	2	0	0

Branching

A problem M is divided into two subproblems:

1) we include the edge (i,j) to the tour; then we delete the row i and the column j from the matrix M. The obtained matrix M' is reduced and the sum of the minimum elements of its rows and columns is added to the low bound of the problem M, the result is the low bound of the problem M';
2) we don't include the edge (i,j) to the tour; then we replace the element (i,j) of the matrix M by infinity. The obtained matrix M'' is reduced: the minimum elements of the row i and of the column j are subtracted from the row and the column respectively and added to the low bound of the problem M, the result is the low bound of the problem M".

Branching

How to choose an edge of the branching?

1) M(i,j)=0.
2) The edge (i,j) cannot close a cycle with other edges included in the tour except of the last edge.
3) The **index** of the edge (i,j) is the sum of the minimum elements of the row i and the column j except of M(i,j):

$$\delta(i,j) = \min_{k \neq j} M(i,k) + \min_{k \neq i} M(k,j).$$

4) We choose the edge with the maximum index to obtain the maximum value of the low bound of the subproblem M”. If the edge closes a cycle with other edges included in the tour then we replace it by infinity and reduce the matrix.

Branching

Example. Calculate the indexes of the elements of the matrix M. The edge (b,e) has the maximum index.

$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{|c|c|c|c|c|c|} \multicolumn{1}{c}{a} & \multicolumn{1}{c}{b} & \multicolumn{1}{c}{c} & \multicolumn{1}{c}{d} & \multicolumn{1}{c}{e} & \multicolumn{1}{c}{f} \\ \hline & 5 & 0(0) & 1 & 9 & 0(4) \\ \hline 6 & & 11 & 6 & 0(6) & 6 \\ \hline 0(4) & 11 & & 12 & 9 & 4 \\ \hline 3 & 5 & 15 & & 0(3) & 11 \\ \hline 8 & 0(5) & 9 & 0(1) & & 11 \\ \hline 0(0) & 6 & 0(0) & 7 & 8 & \\ \hline \end{array}$$

$$\delta(a,c) = 0 + 0 = 0;$$
$$\delta(a,f) = 0 + 4 = 4;$$
$$\delta(b,e) = 6 + 0 = 6;$$
$$\delta(c,a) = 4 + 0 = 4;$$
$$\delta(d,e) = 3 + 0 = 3;$$
$$\delta(e,b) = 0 + 5 = 5;$$
$$\delta(e,d) = 0 + 1 = 1;$$
$$\delta(f,a) = 0 + 0 = 0;$$
$$\delta(f,c) = 0 + 0 = 0.$$

Branching

Example. Include the edge (b,e) into the tour.

$B =$

	a	b	c	d	f	
a		5	0	1	0	0
c	0	11		12	4	0
d	3	5	15		11	3
e	8	0	9	0	11	0
f	0	6	0	7		0

	a	b	c	d	f
a		5	0	1	0
c	0	11		12	4
d	0	2	12		8
e	8	0	9	0	11
f	0	6	0	7	

$$\underline{C}(B) = \underline{C}(A) + 3 = 16 + 3 = 19.$$

Branching

Example. Don't include (b,e) into the tour.

$C =$

	a	b	c	d	e	f
a		5	0	1	9	0
b	6		11	6		6
c	0	11		12	9	4
d	3	5	15		0	11
e	8	0	9	0		11
f	0	6	0	7	8	

$C =$

	a	b	c	d	e	f
a		5	0	1	9	0
b	0		5	0		0
c	0	11		12	9	4
d	3	5	15		0	11
e	8	0	9	0		11
f	0	6	0	7	8	

$$\underline{C}(C) = \underline{C}(A) + 6 = 16 + 6 = 22.$$

Branching

Example. The decision tree.

$$\begin{tikzcd} & A(16) \arrow[dl, "{(b,e)}"] \arrow[dr, "\overline{(b,e)}"] & \\ B(19) & & C(22) \end{tikzcd}$$

Cutting branches

To cut brunches we need the upper bound of the TSP solution. For example, it can be the distance of any tour. A branch is cut if its lower bound is not less then the upper bound.

Example.

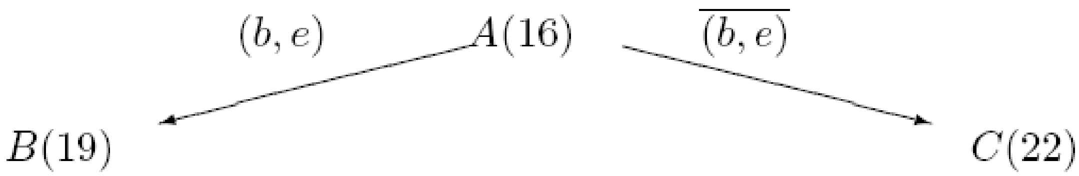
	a	b	c	d	e	f
a		8	4	6	12	3
b	7		13	9	1	7
c	2	13		16	11	6
d	5	7	18		3	14
e	10	2	12	4		13
f	2	8	3	11	10	

$$abcdefa$$

$$\overline{C} = 8 + 13 + 16 + 3 + 13 + 2 = 55.$$

Cutting branches

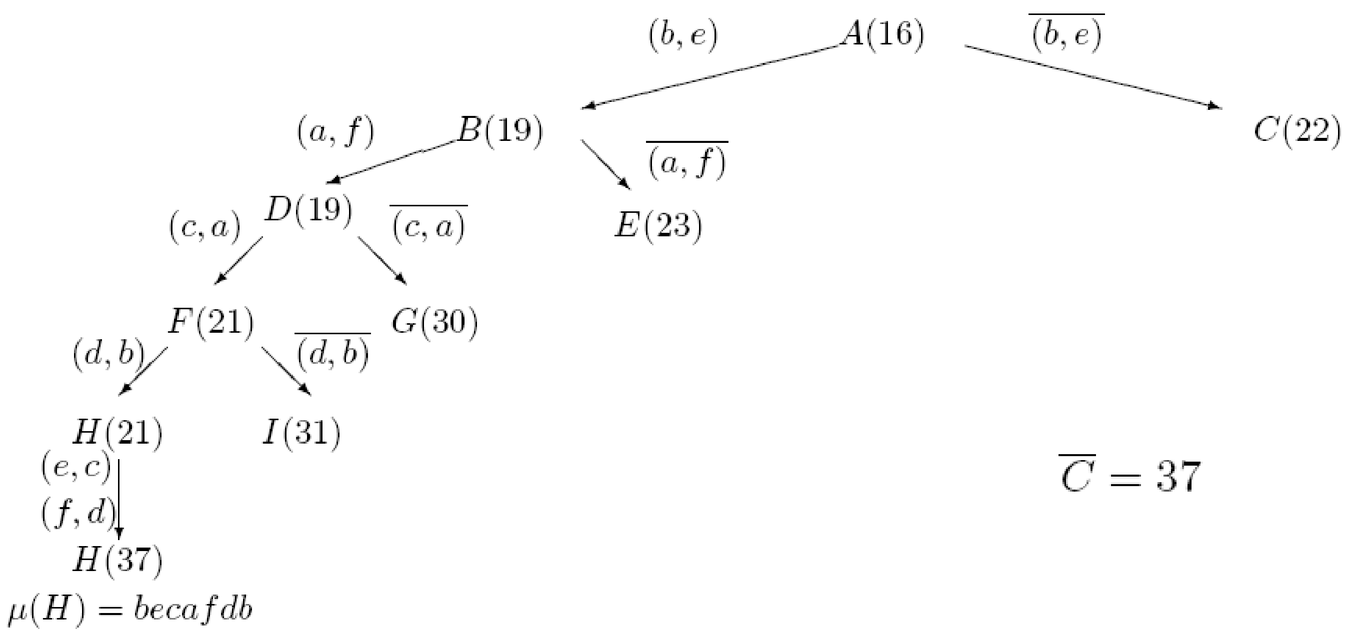
Example. Subproblems B and C have the lower bounds les then 55 so we don't cut the branches.



Cutting branches

If a new tour is obtained and its lower bound is less then the upper bound then we change the upper bound.

Example.



P-median problem

X_p – multimedian (p-median)
$v \in X_p$ – **median vertex**
$v \notin X_p$ – **non-median vertex**

Vertex j is **allocated** to vertex i if vertex i is a median vertex and

$$d(Xp,j)=d(i,j).$$

Any median vertex i is allocated to vertex i itself.

P-median problem

$$\xi_{i,j} = \begin{cases} 1, & \text{if vertex } j \text{ is allocated to vertex } i; \\ 0, & \text{otherwise.} \end{cases}$$

$$\begin{aligned}
&\sum_{i \in V} \sum_{j \in V} d(i,j)\xi_{i,j} \to \min_{\xi_{i,j}}; \\
&\sum_{i \in V} \xi_{i,j} = 1, \quad \forall j \in V; \\
&\sum_{i \in V} \xi_{i,i} = p; \\
&\xi_{i,j} \le \xi_{i,i}, \qquad \forall i \in V,\ j \in V; \\
&\xi_{i,j} \in \{0,1\}, \ \forall i \in V,\ j \in V.
\end{aligned}$$

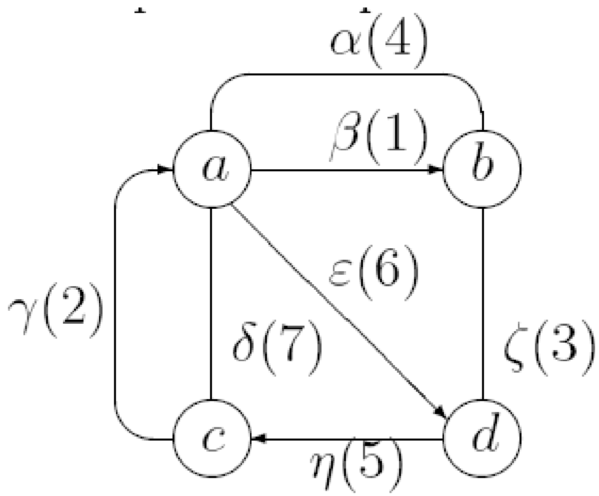
Direct tree search

Set up a matrix M:$n \times n$ the j-th column of which contains all the vertices of the graph G arranged in ascending order of their distance to vertex j. Thus, if m_{ij} = k, then there are i–1 vertices, such that the distance from them to vertex j does not exceed $d(k,j)$ and n–i vertices, such that the distance from them to vertex j is not less than $d(k,j)$.

Obviously, the nearest vertex to vertex j is itself, i.e. m_{1j} =j.

Direct tree search

Example.



	a	b	c	d
a	0	1	7	4
b	4	0	8	3
c	2	3	0	6
d	7	3	5	0

$$M = \begin{array}{c|cccc} & a & b & c & d \\ \hline 1 & a & b & c & d \\ 2 & c & a & d & b \\ 3 & b & c & a & a \\ 4 & d & d & b & c \end{array}$$

Direct tree search

For every vertex j we define index k_j as a number of a row of matrix M.

$$x_j = m_{k_j,j}$$

At the subproblem under consideration, vertex x_j is the best variant for vertex j to be allocated to.

A lower bound of the cost of the optimal solution

$$\underline{C} = \sum_{j \in V} d(x_j, j).$$

Direct tree search

For the start problem for every vertex $j \in V$

$$k_j = 1,\ x_j = j, \quad d(x_j, j) = 0.$$

An upper bound of the cost of the optimal solution

$$\min_j \mathrm{TVV}(j).$$

Two new subproblems are generated from the current subproblem choosing variable ξ_{ij} and setting $\xi_{ij} = 1$ and $\xi_{ij} = 0$.

Direct tree search

S^{+} – set of median vertices;

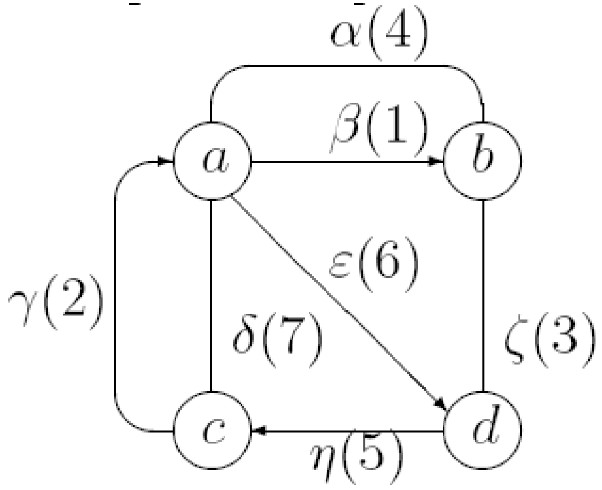
S^{-} – set of non-median vertices;

F – set of non-allocated vertices.

Every median vertex is allocated to itself, thus, $S^{+} \cap F = \varnothing$.

Direct tree search

Example. Start problem A (p=2).



	a	b	c	d
a	0	1	7	4
b	4	0	8	3
c	2	3	0	6
d	7	3	5	0

Direct tree search

S+(A)=∅;

S–(A)= ∅;

F(A)={a,b,c,d};

C(A)=0.

$$M(A) = \begin{array}{c|cccc} & a & b & c & d \\ \hline 1 & \mathbf{a} & \mathbf{b} & \mathbf{c} & \mathbf{d} \\ 2 & c & a & d & b \\ 3 & b & c & a & a \\ 4 & d & d & b & c \end{array}$$

Choose variable ξ_{aa}.

Direct tree search

$S^+(B) = \{a\};$
$S^-(B) = \emptyset;$
$F(B) = \{b, c, d\};$
$\xi_{a,a} = 1.$

$$M(B) = \begin{array}{c|cccc} & a & b & c & d \\ \hline 1 & \mathbf{a} & \mathbf{b} & \mathbf{c} & \mathbf{d} \\ 2 & c & a & d & b \\ 3 & b & c & a & a \\ 4 & d & d & b & c \end{array} \qquad \underline{C}(B) = 0.$$

$S^+(C) = \emptyset;$
$S^-(C) = \{a\};$
$F(C) = \{a, b, c, d\};$
$\xi_{a,a} = 0.$

$$M(C) = \begin{array}{c|cccc} & a & b & c & d \\ \hline 1 & a & \mathbf{b} & \mathbf{c} & \mathbf{d} \\ 2 & \mathbf{c} & a & d & b \\ 3 & b & c & a & a \\ 4 & d & d & b & c \end{array} \qquad \underline{C}(C) = 2.$$

Direct tree search