Formal Derivation of Covering Problem

- Each choice is represented with a Boolean variable x_i.
- $x_i = 1$ implies choice has been included in the solution.
- $x_i = 0$ implies choice has not been included in the solution.
- Covering problem is expressed as a product-of-sums, F.
- Each product (or *clause*) represents a constraint.
- Each clause is sum of choices that satisfy the constraint.
- Goal: find x_i's which satisfy all constraints with minimum cost.

$$cost = \min \sum_{i=1}^{t} w_i x_i \tag{1}$$

Example Covering Problem

$$
\begin{aligned}
f \quad = \quad & x_1 \overline{x_2} (\overline{x_3} + x_4)(\overline{x_3} + x_4 + x_5 + x_6)(\overline{x_1} + x_4 + x_5 + x_6) \\
& (\overline{x_4} + x_1 + x_6)(\overline{x_5} + x_6)
\end{aligned}
$$

Unate versus Binate

- *Unate covering problem* - choices appear only in their positive form (i.e., uncomplemented).
- *Binate covering problem* - choices appear in both positive and negative form (i.e., complemented).
- Algorithm presented here considers the more general case of the binate covering problem, but solution applies to both.

Constraint Matrix

- f is represented using a *constraint matrix*, A.
- Includes a column for each x_i variable.
- Includes a row for every clause.
- Each entry of the matrix a_{ij} is:
  - '-' if the variable x_i does not appear in the clause,
  - '0' if the variable appears complemented, and
  - '1' otherwise.
- i^{th} row of A is denoted a_i.
- j^{th} column is denoted by A_j.

Constraint Matrix Example

$$f = x_1\overline{x_2}(\overline{x_3}+x_4)(\overline{x_3}+x_4+x_5+x_6)(\overline{x_1}+x_4+x_5+x_6)$$
$$(\overline{x_4}+x_1+x_6)(\overline{x_5}+x_6)$$

$$\mathbf{A} = \begin{array}{c c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} & \\ \left[\begin{array}{cccccc} 1 & - & - & - & - & - \\ - & 0 & - & - & - & - \\ - & - & 0 & 1 & - & - \\ - & - & 0 & 1 & 1 & 1 \\ 0 & - & - & 1 & 1 & 1 \\ 1 & - & - & 0 & - & 1 \\ - & - & - & - & 0 & 1 \end{array}\right] & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \end{array}$$

Binate Covering Problem

- The binate covering problem is to find an assignment to **x** of minimum cost such that for every row a_i either
  1. $\exists j \,.\, (a_{ij} = 1) \wedge (x_j = 1)$; or
  2. $\exists j \,.\, (a_{ij} = 0) \wedge (x_j = 0)$.

BCP Algorithm

```
bcp(A,x,b)
  (A,x) = reduce(A,x);
  L = lower_bound(A,x);
  if (L ≥ cost(b)) then return(b);
  if (terminalCase(A)) then
    if (A has no rows) return(x); else return(b);
  c = choose_column(A);
  x_c = 1; A^1 = select_column(A,c); x^1 = bcp(A^1,x,b)
  if (cost(x^1)<cost(b)) then
    b = x^1;
    if (cost(b) = L) return(b);
  x_c = 0; A^0 = remove_column(A,c); x^0 = bcp(A^0,x,b)
  if (cost(x^0)<cost(b)) then b = x^0;
  return(b);
```

Reduce Algorithm

```
reduce(A,x)
  do
    A' = A;
    (A,x) = find_essential_rows(A,x);
    A = delete_dominating_rows(A);
    (A,x) = delete_dominated_columns(A,x);
  while (A ≠ ∅ and A ≠ A');
  return(A,x);
```

Essential Rows

- A row a_i of A is *essential* when there exists exactly one j such that a_{ij} is not equal to '-'.
- This cooresponds to clause consisting of a single literal.
- If the literal is x_j (i.e., $a_{ij} = 1$), the variable is *essential*.
- If the literal is $\overline{x_j}$ (i.e., $a_{ij} = 0$), the variable is *unacceptable*.
- The matrix A is reduced with respect to the essential literal.
- This variable is set to value of literal, column is removed, and any row where variable has same value is removed.

Essential Rows Example

$$
\begin{aligned}
f &= x_1\overline{x_2}(\overline{x_3}+x_4)(\overline{x_3}+x_4+x_5+x_6)(\overline{x_1}+x_4+x_5+x_6) \\
&\quad (\overline{x_4}+x_1+x_6)(\overline{x_5}+x_6)
\end{aligned}
$$

$$
\mathbf{A} = \begin{array}{c}
\begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\
\left[\begin{array}{cccccc}
1 & - & - & - & - & - \\
- & 0 & - & - & - & - \\
- & - & 0 & 1 & - & - \\
- & - & 0 & 1 & 1 & 1 \\
0 & - & - & 1 & 1 & 1 \\
1 & - & - & 0 & - & 1 \\
- & - & - & - & 0 & 1
\end{array}\right]
\end{array}
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array}
$$

Essential Rows Example

$$f = \overline{x_2}(\overline{x_3} + x_4)(\overline{x_3} + x_4 + x_5 + x_6)(x_4 + x_5 + x_6)(\overline{x_5} + x_6)$$

$$\mathbf{A} = \begin{array}{c c}
\begin{array}{ccccc} x_2 & x_3 & x_4 & x_5 & x_6 \end{array} & \\
\left[\begin{array}{ccccc}
0 & - & - & - & - \\
- & 0 & 1 & - & - \\
- & 0 & 1 & 1 & 1 \\
- & - & 1 & 1 & 1 \\
- & - & - & 0 & 1
\end{array}\right] & \begin{array}{c} 2 \\ 3 \\ 4 \\ 5 \\ 7 \end{array}
\end{array}$$

$$x_1 = 1$$

Essential Rows Example

$$f = (\overline{x_3} + x_4)(\overline{x_3} + x_4 + x_5 + x_6)(x_4 + x_5 + x_6)(\overline{x_5} + x_6)$$

$$\mathbf{A} = \begin{array}{c} \begin{array}{cccc} x_3 & x_4 & x_5 & x_6 \end{array} \\ \left[\begin{array}{cccc} 0 & 1 & - & - \\ 0 & 1 & 1 & 1 \\ - & 1 & 1 & 1 \\ - & - & 0 & 1 \end{array} \right] \end{array} \begin{array}{c} \\ 3 \\ 4 \\ 5 \\ 7 \end{array}$$

$$x_1 = 1, x_2 = 0$$

Row Dominance

- A row a_k *dominates* another row a_i if it has all 1's and 0's of a_i.
- Row a_k dominates another row a_i if for each column A_j of **A**, one of the following is true:
  - $a_{ij} = -$
  - $a_{ij} = a_{kj}$
- Removing dominating rows does not affect set of solutions.

Row Dominance Example

$$f = (\overline{x_3} + x_4)(\overline{x_3} + x_4 + x_5 + x_6)(x_4 + x_5 + x_6)(\overline{x_5} + x_6)$$

$$\mathbf{A} = \begin{array}{c c} \begin{array}{cccc} x_3 & x_4 & x_5 & x_6 \end{array} & \\ \left[\begin{array}{cccc} 0 & 1 & - & - \\ 0 & 1 & 1 & 1 \\ - & 1 & 1 & 1 \\ - & - & 0 & 1 \end{array}\right] & \begin{array}{c} 3 \\ 4 \\ 5 \\ 7 \end{array} \end{array}$$

$$x_1 = 1, x_2 = 0$$

Row Dominance Example

$$f = (\overline{x_3} + x_4)(x_4 + x_5 + x_6)(\overline{x_5} + x_6)$$

$$\mathbf{A} = \begin{array}{c c} \begin{array}{cccc} x_3 & x_4 & x_5 & x_6 \end{array} & \\ \left[\begin{array}{cccc} 0 & 1 & - & - \\ - & 1 & 1 & 1 \\ - & - & 0 & 1 \end{array} \right] & \begin{array}{c} 3 \\ 5 \\ 7 \end{array} \end{array}$$

$$x_1 = 1, x_2 = 0$$

Column Dominance

- A column A_j *dominates* another column A_k if for each clause a_i of A, one of the following is true:
  - $a_{ij} = 1$;
  - $a_{ij} = -$ and $a_{ik} \neq 1$;
  - $a_{ij} = 0$ and $a_{ik} = 0$.
- Dominated columns can be removed without affecting the existence of a solution.
- When removing a column, the variable is set to 0 which means any rows including that column with a 0 entry can be removed.

Column Dominance Example

$$f = (\overline{x_3} + x_4)(x_4 + x_5 + x_6)(\overline{x_5} + x_6)$$

$$\mathbf{A} = \begin{array}{cc} \begin{array}{cccc} x_3 & x_4 & x_5 & x_6 \end{array} & \\ \left[\begin{array}{cccc} 0 & 1 & - & - \\ - & 1 & 1 & 1 \\ - & - & 0 & 1 \end{array}\right] & \begin{array}{c} 3 \\ 5 \\ 7 \end{array} \end{array}$$

$$x_1 = 1, x_2 = 0$$

Column Dominance Example

$$f = (x_4 + x_6)$$

$$\mathbf{A} = \begin{array}{cc} x_4 & x_6 \\ \left[\begin{array}{cc} 1 & 1 \end{array}\right] \end{array} \quad 5$$

$$x_1 = 1, x_2 = 0, x_3 = 0, x_5 = 0$$

Checking Weights

- If weights are not equal, it is necessary to also check the weights of the columns before removing dominated columns.
- If weight of dominating column, w_j, is greater than weight of dominated column, w_k, then x_k should not be removed.
- Assume $w_1 = 3$, $w_2 = 1$, and $w_3 = 1$.

$$\mathbf{A} = \begin{array}{c} \\ \\ \end{array}\begin{array}{c} x_1 \quad x_2 \quad x_3 \\ \left[\begin{array}{ccc} 1 & 1 & - \\ - & 0 & 1 \end{array}\right] \end{array} \begin{array}{c} \\ 1 \\ 2 \end{array}$$

BCP Algorithm

```
bcp(A,x,b)
  (A,x) = reduce(A,x);
  L = lower_bound(A,x);
  if (L ≥ cost(b)) then return(b);
  if (terminalCase(A)) then
    if (A has no rows) return(x); else return(b);
  c = choose_column(A);
  x_c = 1; A^1 = select_column(A,c); x^1 = bcp(A^1,x,b)
  if (cost(x^1) < cost(b)) then
    b = x^1;
    if (cost(b) = L) return(b);
  x_c = 0; A^0 = remove_column(A,c); x^0 = bcp(A^0,x,b)
  if (cost(x^0) < cost(b)) then b = x^0;
  return(b);
```

Bounding

- If solved, cost of solution can be determined by Equation 1.
- Reduced matrix may have a *cyclic core*.
- Must test whether or not a good solution can be derived from partial solution found up to this point.
- Determine a lower bound, L, on the final cost, starting with the current partial solution.
- If L is greater than or equal to the cost of the best solution found, the previous best solution is returned.

Maximal Independent Set

- Finding exact lower bound is as difficult as solving the covering problem.
- Satisfactory heuristic method is to find a *maximal independent set* (MIS) of rows.
- Two rows are independent when it is not possible to satisfy both by setting a single variable to 1.
- Any row which contains a complemented variable is dependent on any other clause, so we must ignore these rows.

```
lower_bound(A, x)
  MIS = ∅
  A = delete_rows_with_complemented_variables(A);
  do
    i = choose_shortest_row(A);
    MIS = MIS ∪ {i};
    A = delete_intersecting_rows(A, i);
  while (A ≠ ∅);
  return(|MIS| + cost(x));
```

Bounding Example

$$\mathbf{A} = \begin{array}{cc} \begin{array}{ccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \end{array} & \\ \left[\begin{array}{ccccccccc} 1 & 1 & - & - & - & - & - & - & - \\ 1 & - & 1 & - & - & - & - & - & - \\ - & - & - & 1 & 1 & - & - & - & - \\ - & - & - & 1 & - & 1 & - & - & - \\ - & - & 1 & - & 1 & 1 & - & - & - \\ - & - & 1 & - & - & - & 1 & - & - \\ - & 1 & - & - & - & - & 1 & - & - \\ - & - & - & 1 & - & - & - & 1 & - \\ - & - & - & 1 & - & - & - & - & 1 \\ - & 1 & - & - & - & - & - & 1 & 1 \end{array}\right] & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} \end{array}$$

Bounding Example

$$
\mathbf{A} = \begin{array}{c}
\begin{array}{ccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \end{array} \\
\left[\begin{array}{ccccccccc}
- & - & - & 1 & 1 & - & - & - & - \\
- & - & - & 1 & - & 1 & - & - & - \\
- & - & 1 & - & 1 & 1 & - & - & - \\
- & - & 1 & - & - & - & 1 & - & - \\
- & - & - & 1 & - & - & - & 1 & - \\
- & - & - & 1 & - & - & - & - & 1
\end{array}\right]
\end{array}
\begin{array}{c} \\ 3 \\ 4 \\ 5 \\ 6 \\ 8 \\ 9 \end{array}
$$

$$MIS = \{1\}$$

Bounding Example

$$
\mathbf{A} = \begin{array}{c}
\begin{array}{ccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \end{array} \\
\left[\begin{array}{ccccccccc} - & - & 1 & - & - & - & 1 & - & - \end{array}\right]
\end{array} \quad 6
$$

$$MIS = \{1, 3\}$$

Bounding Example

$$
\mathbf{A} = \begin{array}{c c}
\begin{array}{ccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \end{array} & \\
\left[\begin{array}{ccccccccc}
1 & 1 & - & - & - & - & - & - & - \\
1 & - & 1 & - & - & - & - & - & - \\
- & - & - & 1 & 1 & - & - & - & - \\
- & - & - & 1 & - & 1 & - & - & - \\
- & - & 1 & - & 1 & 1 & - & - & - \\
- & - & 1 & - & - & - & 1 & - & - \\
- & 1 & - & - & - & - & 1 & - & - \\
- & - & - & 1 & - & - & - & 1 & - \\
- & - & - & 1 & - & - & - & - & 1 \\
- & 1 & - & - & - & - & - & 1 & 1
\end{array}\right] &
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array}
\end{array}
$$

$$MIS = \{1, 3, 6\}$$

```
bcp(A, x, b)
    (A, x) = reduce(A, x);
    L = lower_bound(A, x);
    if (L ≥ cost(b)) then return(b);
    if (terminalCase(A)) then
        if (A has no rows) return(x); else return(b);
    c = choose_column(A);
    x_c = 1; A^1 = select_column(A, c); x^1 = bcp(A^1, x, b)
    if (cost(x^1) < cost(b)) then
        b = x^1;
        if (cost(b) = L) return(b);
    x_c = 0; A^0 = remove_column(A, c); x^0 = bcp(A^0, x, b)
    if (cost(x^0) < cost(b)) then b = x^0;
    return(b);
```

Termination

- If A has no more rows, then all the constraints have been satisfied by x, and it is a terminal case.
- If no solution exists, it is also a terminal case.

Infeasible Problems

$$f = (x_1 + x_2)(\overline{x_1} + x_2)(x_1 + \overline{x_2})(\overline{x_1} + \overline{x_2})$$

$$\mathbf{A} = \begin{array}{c} \\ \\ \\ \\ \end{array}\begin{array}{cc} x_1 & x_2 \\ \left[\begin{array}{c} 1 \\ 0 \\ 1 \\ 0 \end{array}\right. & \left.\begin{array}{c} 1 \\ 1 \\ 0 \\ 0 \end{array}\right] \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}$$

BCP Algorithm

```
bcp(A, x, b)
  (A, x) = reduce(A, x);
  L = lower_bound(A, x);
  if (L ≥ cost(b)) then return(b);
  if (terminalCase(A)) then
    if (A has no rows) return(x); else return(b);
  c = choose_column(A);
  x_c = 1; A^1 = select_column(A, c); x^1 = bcp(A^1, x, b)
  if (cost(x^1) < cost(b)) then
    b = x^1;
    if (cost(b) = L) return(b);
  x_c = 0; A^0 = remove_column(A, c); x^0 = bcp(A^0, x, b)
  if (cost(x^0) < cost(b)) then b = x^0;
  return(b);
```

Branching

- If **A** is not a terminal case, matrix is *cyclic*.
- To find minimal solution, must determine column to branch on.
- A column intersecting short rows is prefered for branching.
- Assign a weight to each row that is inverse of row length.
- Sum the weights of all the rows covered by a column.
- Column x_c with highest value is chosen for case splitting.

Branching Example

$$\mathbf{A} = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\begin{array}{c} x_1 \; x_2 \; x_3 \; x_4 \; x_5 \; x_6 \; x_7 \; x_8 \; x_9 \\ \left[\begin{array}{ccccccccc}
1 & 1 & - & - & - & - & - & - & - \\
1 & - & 1 & - & - & - & - & - & - \\
- & - & - & 1 & 1 & - & - & - & - \\
- & - & - & 1 & - & 1 & - & - & - \\
- & - & 1 & - & 1 & 1 & - & - & - \\
- & - & 1 & - & - & - & 1 & - & - \\
- & 1 & - & - & - & - & 1 & - & - \\
- & - & - & 1 & - & - & - & 1 & - \\
- & - & - & 1 & - & - & - & - & 1 \\
- & 1 & - & - & - & - & - & 1 & 1
\end{array}\right]\end{array}
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array}$$

$$\mathbf{A} = \begin{array}{cc} \begin{array}{ccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \end{array} & \\ \begin{bmatrix} 1 & 1 & - & - & - & - & - & - & - \\ 1 & - & 1 & - & - & - & - & - & - \\ - & - & - & 1 & 1 & - & - & - & - \\ - & - & - & 1 & - & 1 & - & - & - \\ - & - & 1 & - & 1 & 1 & - & - & - \\ - & - & 1 & - & - & - & 1 & - & - \\ - & 1 & - & - & - & - & 1 & - & - \\ - & - & - & 1 & - & - & - & 1 & - \\ - & - & - & 1 & - & - & - & - & 1 \\ - & 1 & - & - & - & - & - & 1 & 1 \end{bmatrix} & \begin{array}{cc} 1 & 1/2 \\ 2 & 1/2 \\ 3 & 1/2 \\ 4 & 1/2 \\ 5 & 1/3 \\ 6 & 1/2 \\ 7 & 1/2 \\ 8 & 1/2 \\ 9 & 1/2 \\ 10 & 1/3 \end{array} \end{array}$$

Branching Example

$$
\begin{array}{c}
\begin{array}{ccccccccc}
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \\
1.0 & 1.3 & 1.3 & 2.0 & 0.8 & 0.8 & 1.0 & 0.8 & 0.8
\end{array}
\\
\mathbf{A} = \left[\begin{array}{ccccccccc}
1 & 1 & - & - & - & - & - & - & - \\
1 & - & 1 & - & - & - & - & - & - \\
- & - & - & 1 & 1 & - & - & - & - \\
- & - & - & 1 & - & 1 & - & - & - \\
- & - & 1 & - & 1 & 1 & - & - & - \\
- & - & 1 & - & - & - & 1 & - & - \\
- & 1 & - & - & - & - & 1 & - & - \\
- & - & - & 1 & - & - & - & 1 & - \\
- & - & - & 1 & - & - & - & - & 1 \\
- & 1 & - & - & - & - & - & 1 & 1
\end{array} \right]
\begin{array}{cc}
1 & 1/2 \\
2 & 1/2 \\
3 & 1/2 \\
4 & 1/2 \\
5 & 1/3 \\
6 & 1/2 \\
7 & 1/2 \\
8 & 1/2 \\
9 & 1/2 \\
10 & 1/3
\end{array}
\end{array}
$$

Branching

- x_c is added to the solution and constraint matrix is reduced.
- *bcp* is called recursively and result assigned to $\mathbf{x}^1$.
- If $\mathbf{x}^1$ better than best, record it.
- If $\mathbf{x}^1$ meets lower bound L, it is minimal.
- If not, remove x_c from solution and call *bcp*.
- If $\mathbf{x}^0$ better than best, return it.

BCP Algorithm

```
bcp(A,x,b)
  (A,x) = reduce(A,x);
  L = lower_bound(A,x);
  if (L ≥ cost(b)) then return(b);
  if (terminalCase(A)) then
    if (A has no rows) return(x); else return(b);
  c = choose_column(A);
  x_c = 1; A^1 = select_column(A,c); x^1 = bcp(A^1,x,b)
  if (cost(x^1) < cost(b)) then
    b = x^1;
    if (cost(b) = L) return(b);
  x_c = 0; A^0 = remove_column(A,c); x^0 = bcp(A^0,x,b)
  if (cost(x^0) < cost(b)) then b = x^0;
  return(b);
```

Branching Example

$$\mathbf{A} = \begin{array}{c} \begin{array}{ccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \end{array} \\ \left[\begin{array}{ccccccccc} 1 & 1 & - & - & - & - & - & - & - \\ 1 & - & 1 & - & - & - & - & - & - \\ - & - & - & 1 & 1 & - & - & - & - \\ - & - & - & 1 & - & 1 & - & - & - \\ - & - & 1 & - & 1 & 1 & - & - & - \\ - & - & 1 & - & - & - & 1 & - & - \\ - & 1 & - & - & - & - & 1 & - & - \\ - & - & - & 1 & - & - & - & 1 & - \\ - & - & - & 1 & - & - & - & - & 1 \\ - & 1 & - & - & - & - & - & 1 & 1 \end{array} \right] \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array}$$

$$\mathbf{A} = \begin{array}{cc} \begin{array}{cccccccc} x_1 & x_2 & x_3 & x_5 & x_6 & x_7 & x_8 & x_9 \end{array} & \\ \left[\begin{array}{cccccccc} 1 & 1 & - & - & - & - & - & - \\ 1 & - & 1 & - & - & - & - & - \\ - & - & 1 & 1 & 1 & - & - & - \\ - & - & 1 & - & - & 1 & - & - \\ - & 1 & - & - & - & 1 & - & - \\ - & 1 & - & - & - & - & 1 & 1 \end{array}\right] & \begin{array}{r} 1 \\ 2 \\ 5 \\ 6 \\ 7 \\ 10 \end{array} \end{array}$$

$$x_4 = 1$$

Branching Example

$$\mathbf{A} = \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & x_2 & x_3 & x_7 \end{array} \\ \left[\begin{array}{cccc} 1 & 1 & - & - \\ 1 & - & 1 & - \\ - & - & 1 & - \\ - & - & 1 & 1 \\ - & 1 & - & 1 \\ - & 1 & - & - \end{array}\right] \end{array} \begin{array}{r} \\ 1 \\ 2 \\ 5 \\ 6 \\ 7 \\ 10 \end{array}$$

$$x_4 = 1, x_5 = 0, x_6 = 0, x_8 = 0, x_9 = 0$$

Branching Example

$$x_2 = 1, x_3 = 1, x_4 = 1, x_5 = 0, x_6 = 0, x_8 = 0, x_9 = 0$$

$\text{cost}(\mathbf{x}^1) = 3$

Recall that $L = 3$

Therefore, we are done.

Branching Example

$$
\mathbf{A} = \begin{array}{c}
\begin{array}{ccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \end{array} \\
\left[\begin{array}{ccccccccc}
1 & 1 & - & - & - & - & - & - & - \\
1 & - & 1 & - & - & - & - & - & - \\
- & - & - & 1 & 1 & - & - & - & - \\
- & - & - & 1 & - & 1 & - & - & - \\
- & - & 1 & - & 1 & 1 & - & - & - \\
- & - & 1 & - & - & - & 1 & - & - \\
- & 1 & - & - & - & - & 1 & - & - \\
- & - & - & 1 & - & - & - & 1 & - \\
- & - & - & 1 & - & - & - & - & 1 \\
- & 1 & - & - & - & - & - & 1 & 1
\end{array}\right]
\end{array}
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array}
$$

Branching Example

$$
\mathbf{A} = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}
\begin{array}{c}
\begin{array}{cccccccc} x_1 & x_2 & x_3 & x_5 & x_6 & x_7 & x_8 & x_9 \end{array} \\
\left[\begin{array}{cccccccc}
1 & 1 & - & - & - & - & - & - \\
1 & - & 1 & - & - & - & - & - \\
- & - & - & 1 & - & - & - & - \\
- & - & - & - & 1 & - & - & - \\
- & - & 1 & 1 & 1 & - & - & - \\
- & - & 1 & - & - & 1 & - & - \\
- & 1 & - & - & - & 1 & - & - \\
- & - & - & - & - & - & 1 & - \\
- & - & - & - & - & - & - & 1 \\
- & 1 & - & - & - & - & 1 & 1
\end{array}\right]
\end{array}
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array}
$$

$$x_4 = 0$$

Branching Example

$$\mathbf{A} = \begin{array}{c} \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & x_2 & x_3 & x_7 \end{array} \\ \left[\begin{array}{cccc} 1 & 1 & - & - \\ 1 & - & 1 & - \\ - & - & 1 & 1 \\ - & 1 & - & 1 \end{array} \right] \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 6 \\ 7 \end{array}$$

$$x_4 = 0, x_5 = 1, x_6 = 1, x_8 = 1, x_9 = 1$$