§The Searching Strategies

e.g. satisfiability problem

x_1	x_2	x_3
F	F	F
F	F	T
F	T	F
F	T	T
T	F	F
T	F	T
T	T	F
T	T	T

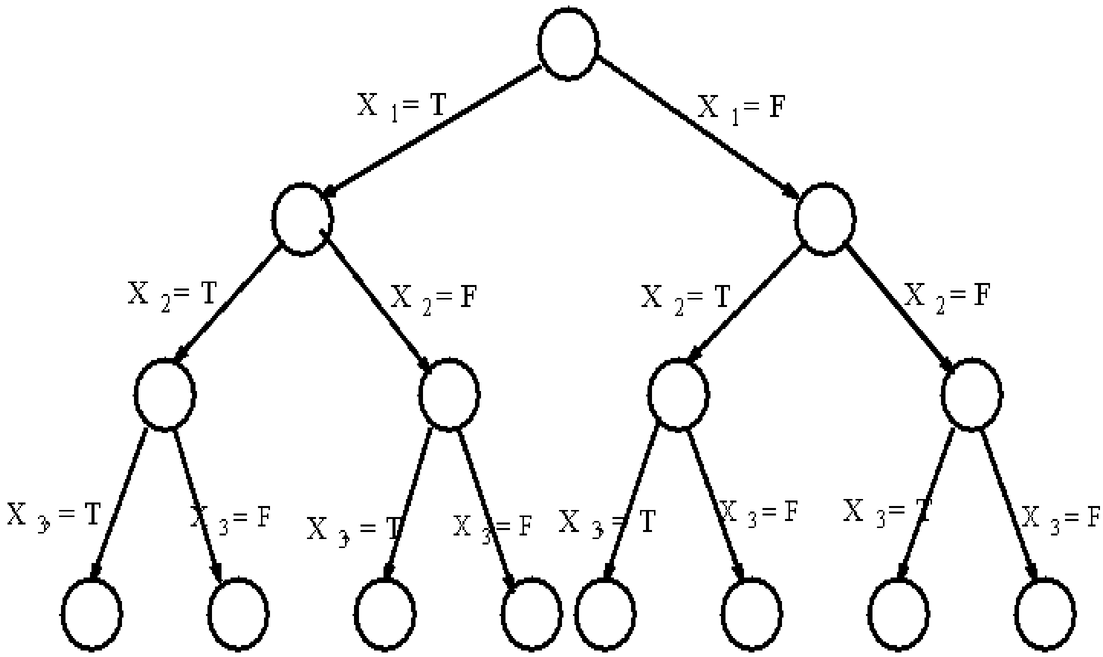


Fig. 6-1 Tree Representation of Eight Assignments.

If there are n variables x_1, x_2, …,x_n, then there are 2^n possible assignments.

an instance:

$-x_1$ ……………………… (1)
x_1 ……………………… (2)
$x_2 \vee x_5$ ……………… (3)
x_3 ……………………… (4)
$-x_2$ ……………………… (5)

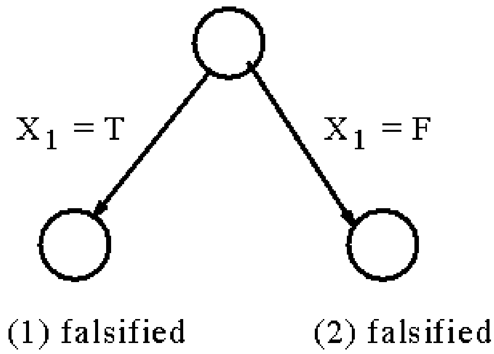


Fig. 6-2 A Partial Tree to Determine the Satisfiability Problem.

We may not need to examine all possible assignments.

e.g. the Hamiltonian circuit problem

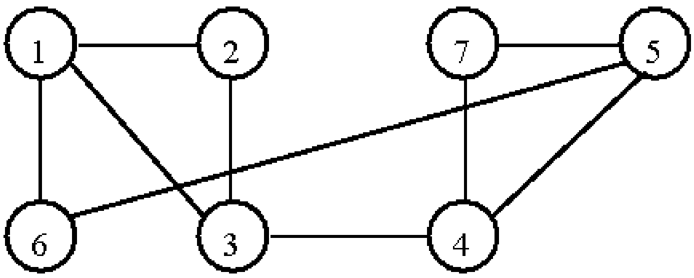


Fig. 6-6 A Graph Containing a Hamiltonian Circuit

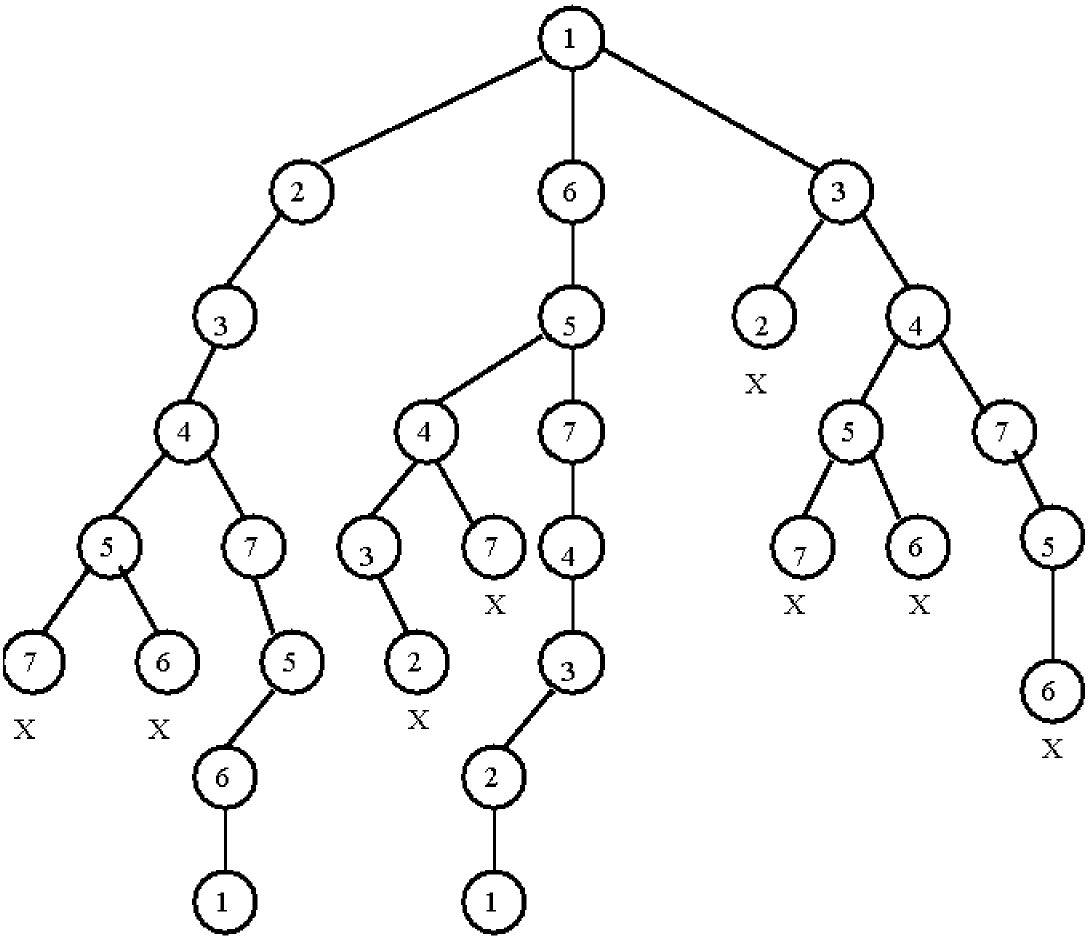


Fig. 6-8 The Tree Representation of Whether There Exists a Hamiltonian Circuit of the Graph in Fig. 6-6

● The breadth-first search

e.g. 8-puzzle problem

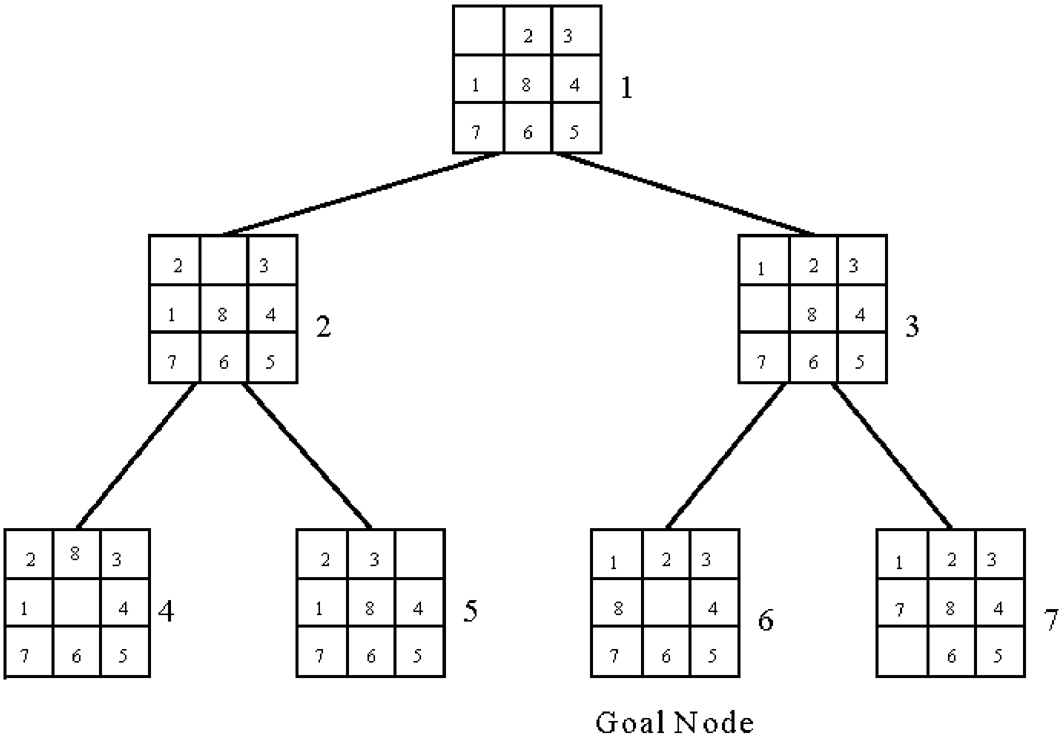


Fig. 6-10 A Search Tree Produced by a Breadth-First Search

The breadth-first search uses a queue to holds all expanded nodes.

● The depth-first search

e.g. sum of subset problem

S={7, 5, 1, 2, 10}

∃ S' ⊆ S ∋ sum of S' = 9 ?

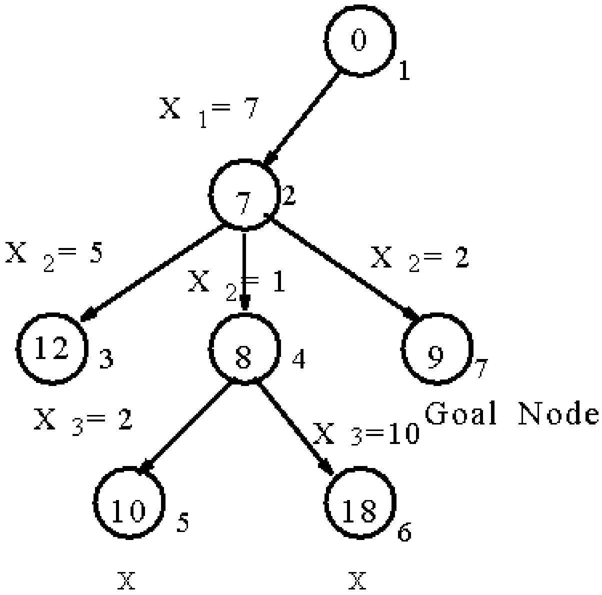


Fig. 6-11 A Sum of Subset Problem Solved by Depth-First Search.

A stack can be used to guide the depth-first search.

● Hill climbing

a variant of depth-first search

The method selects the locally optimal node to expand.

e.g. 8-puzzle problem

evaluation function $f(n) = d(n) + w(n)$

where $d(n)$ is the depth of node n

$w(n)$ is # of misplaced tiles in node n.

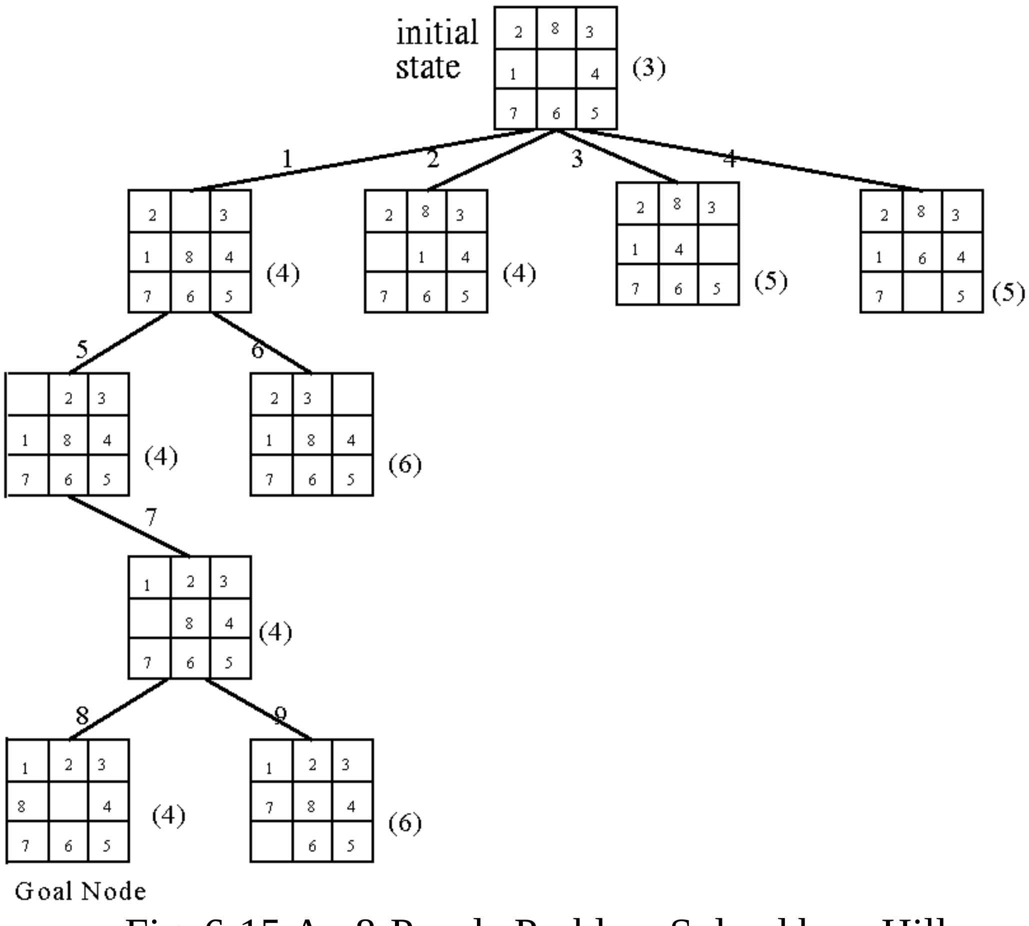


Fig. 6-15 An 8-Puzzle Problem Solved by a Hill Climbing Method

● Best-first search strategy

Combing depth-first search and breadth-first search

Selecting the node with the best estimated cost among all nodes.

This method has a global view.

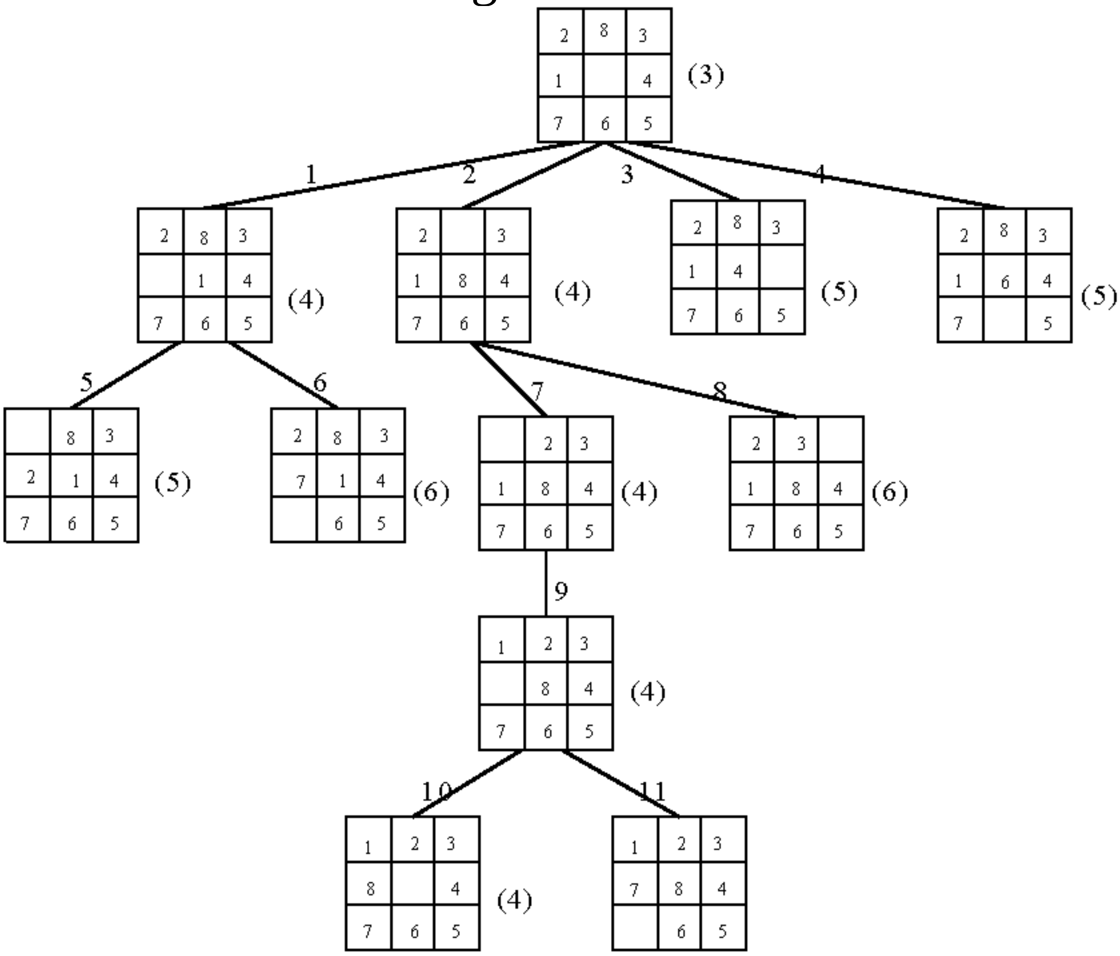


Fig. 6-16 An 8-Puzzle Problem Solved by a Best-First Search Scheme

Best-First Search Scheme

Step1: Form a one-element list consisting of the root node.

Step2: Remove the first element from the list. Expand the first element. If one of the descendants of the first element is a goal node, then stop; otherwise, add the descendants into the list.

Step3: Sort the entire list by the values of some estimation function.

Step4: If the list is empty, then failure. Otherwise, go to Step 2.

- **The branch-and-bound strategy**

  This strategy can be used to solve optimization problems. (DFS, BFS, hill climbing and best-first search can not be used to solve optimization problems.)

e.g.

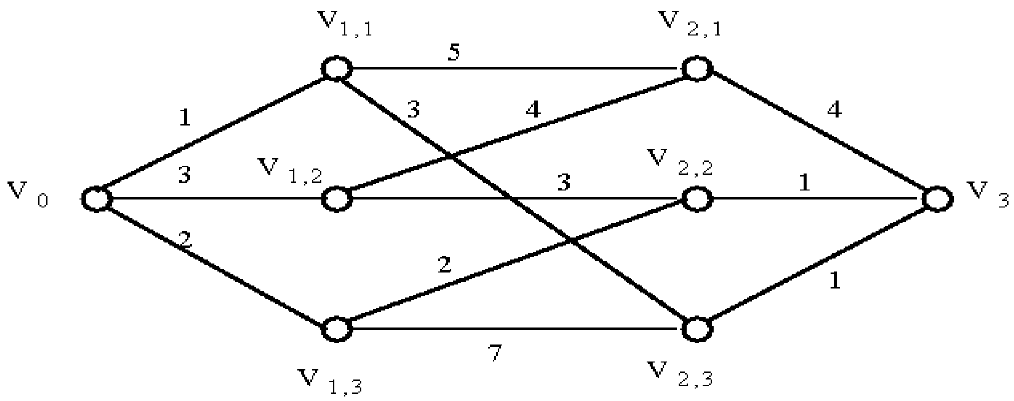
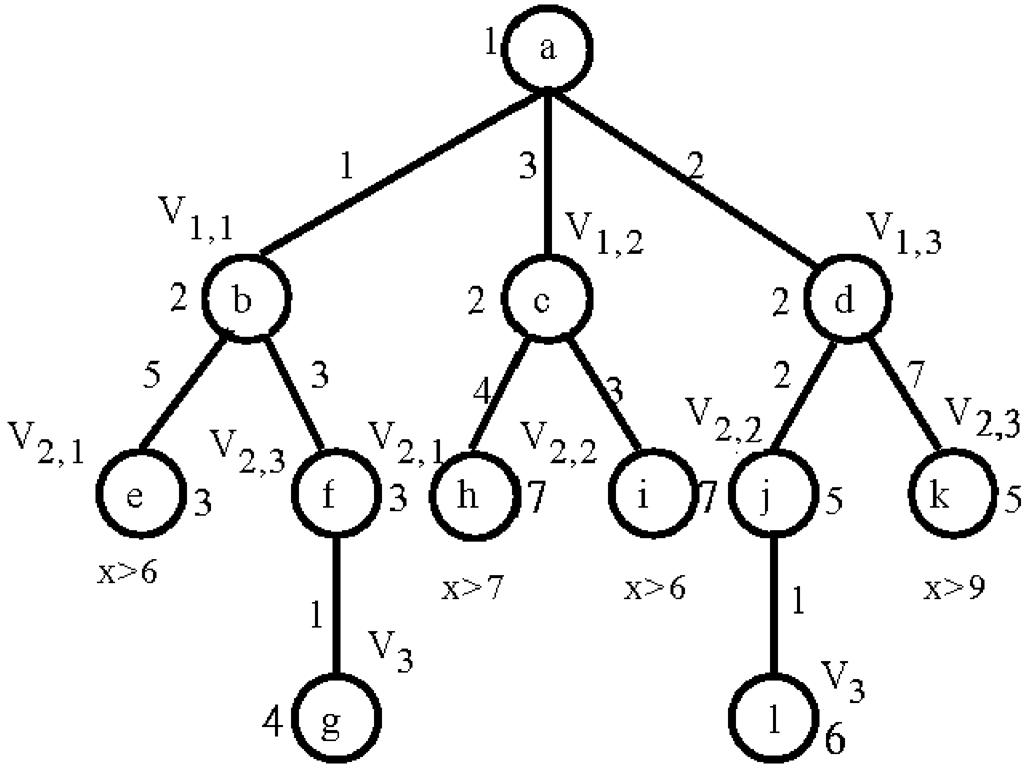


Fig. 6-17 A Multi-Stage Graph Searching Problem.

Solved by branch-and-bound:



- **The personnel assignment problem**

a linearly ordered set of persons $P=\{P_1, P_2, \ldots, P_n\}$

where $P_1<P_2<\ldots<P_n$

a partially ordered set of jobs $J=\{J_1, J_2, \ldots, J_n\}$

Suppose that P_i and P_j are assigned to jobs $f(P_i)$ and $f(P_j)$ respectively. If $f(P_i) \leq f(P_j)$, then $P_i \leq P_j$. Cost C_{ij} is the cost of assigning P_i to J_j. We want to find a feasible assignment with the min. cost. i.e.

$X_{ij} = 1$ if P_i is assigned to J_j and $X_{ij} = 0$ otherwise.

Minimize $\sum_{i,j} C_{ij}X_{ij}$

e.g.

$$\begin{tikzcd} J_1 \arrow[d] \arrow[dr] & J_2 \arrow[d] \\ J_3 & J_4 \end{tikzcd}$$

Fig. 6-21 A Partial Ordering of Jobs

After topological sorting, one of the following topologically sorted sequences will be generated:

J_1, J_2, J_3, J_4
J_1, J_2, J_4, J_3
J_1, J_3, J_2, J_4
J_2, J_1, J_3, J_4
J_2, J_1, J_4 J_3

one of feasible assignments:

$P_1 \rightarrow J_1$, $P_2 \rightarrow J_2$, $P_3 \rightarrow J_3$, $P_4 \rightarrow J_4$

cost matrix:

Jobs / Persons	1	2	3	4
1	29	19	17	12
2	32	30	26	28
3	3	21	7	9
4	18	13	10	15

Table 6-1 A Cost Matrix for a Personnel Assignment Problem

P.11-A

P.11-B

reduced cost matrix:

subtract a constant from each row and each column respectively such that each row and each column contains at least one zero.

Jobs / Persons	1	2	3	4	
1	17	4	5	0	(-12)
2	6	1	0	2	(-26)
3	0	15	4	6	(-3)
4	8	0	0	5	(-10)
		(-3)			

Table 6-2 A Reduced Cost Matrix

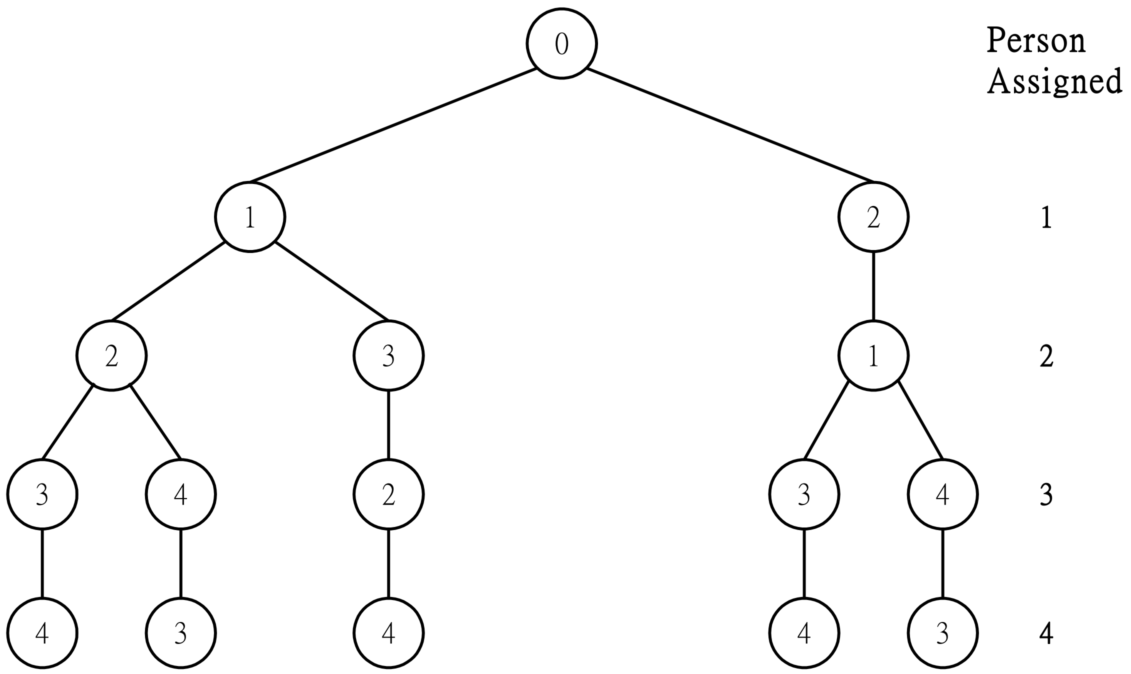
total cost subtracted: 12+26+3+10+3 = 54

This is a lower bound of our solution.

P.11-C

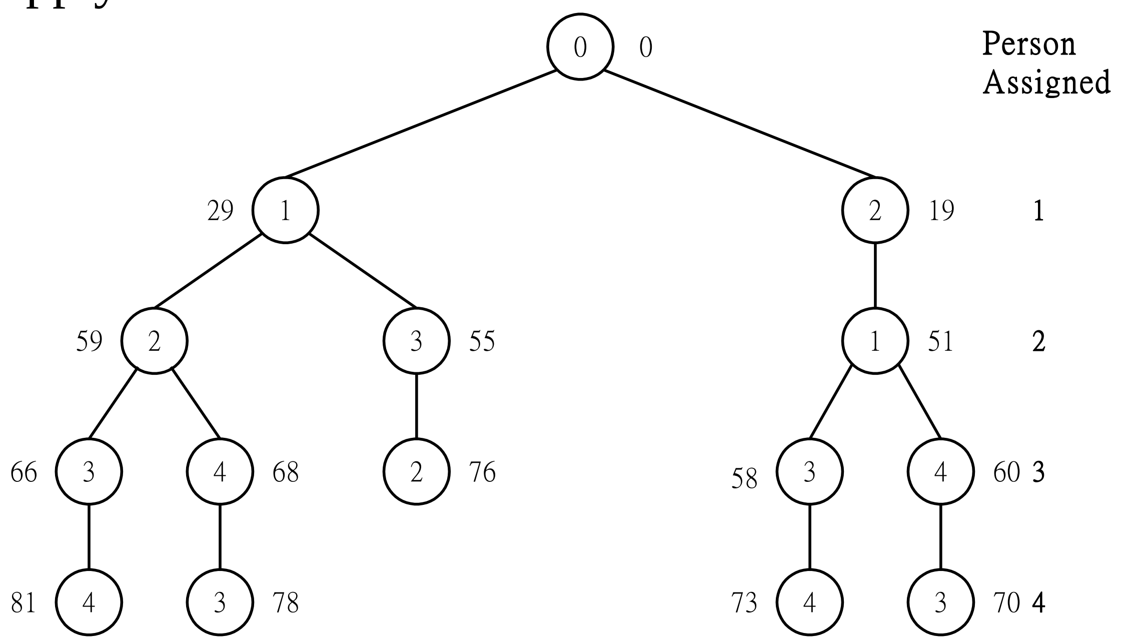
P11-A

Solution tree:



P11-B

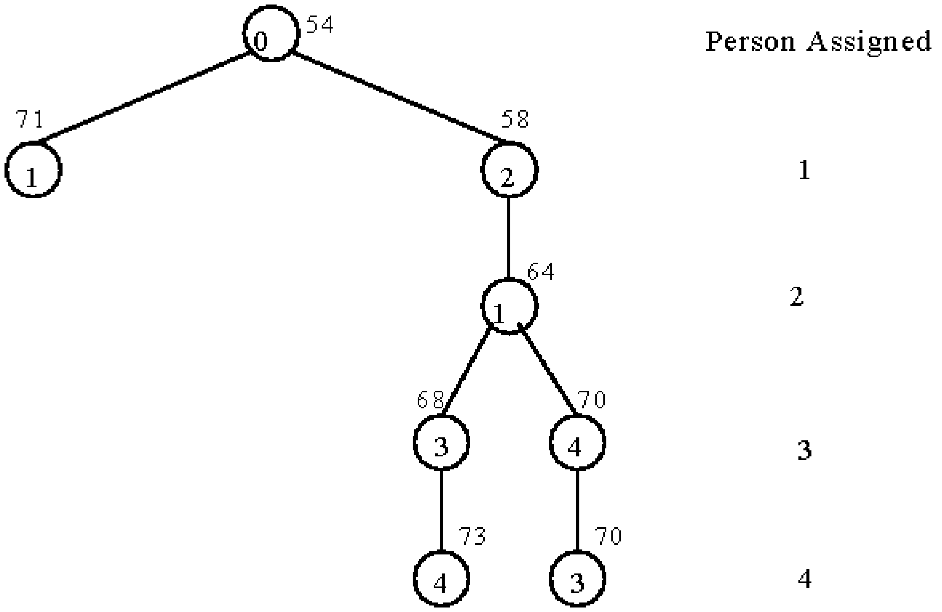
Apply the best-first search scheme:



Only one node is pruned away.

P11-C

bounding of subsolutions:



- **The traveling salesperson optimization problem**

It is NP-complete

e.g. cost matrix:

i \ j	1	2	3	4	5	6	7
1	∞	3	93	13	33	9	57
2	4	∞	77	42	21	16	34
3	45	17	∞	36	16	28	25
4	39	90	80	∞	56	7	91
5	28	46	88	33	∞	25	57
6	3	88	18	46	92	∞	7
7	44	26	33	27	84	39	∞

Table 6-3 A Cost Matrix for a Traveling Salesperson Problem.

Reduced cost matrix:

i \ j	1	2	3	4	5	6	7	
1	∞	0	90	10	30	6	54	(-3)
2	0	∞	73	38	17	12	30	(-4)
3	29	1	∞	20	0	12	9	(-16)
4	32	83	73	∞	49	0	84	(-7)
5	3	21	63	8	∞	0	32	(-25)
6	0	85	15	43	89	∞	4	(-3)
7	18	0	7	1	58	13	∞	(-26)

reduced:84

Table 6-4 A Reduced Cost Matrix.

i \ j	1	2	3	4	5	6	7
1	∞	0	83	9	30	6	50
2	0	∞	66	37	17	12	26
3	29	1	∞	19	0	12	5
4	32	83	66	∞	49	0	80
5	3	21	56	7	∞	0	28
6	0	85	8	42	89	∞	0
7	18	0	0	0	58	13	∞
			(-7)	(-1)			(-4)

Table 6-5 Another Reduced Cost Matrix.

total cost reduced: 84+7+1+4 = 96 (lower bound)

decision tree:

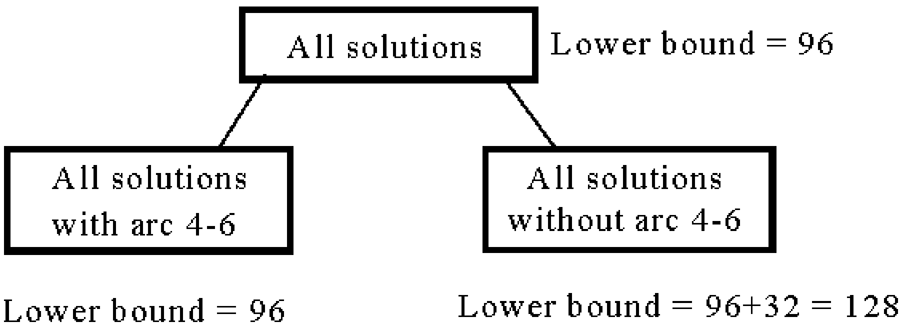


Fig. 6-25 The Highest Level of a Decision Tree.

If we use arc 3-5 to split, the difference on the lower bounds is 17+1 = 18.

i \ j	1	2	3	4	5	7
1	∞	0	83	9	30	50
2	0	∞	66	37	17	26
3	29	1	∞	19	0	5
5	3	21	56	7	∞	28
6	0	85	8	∞	89	0
7	18	0	0	0	58	∞

Table 6-6 A Reduced Cost Matrix. If Arc 4-6 is Included.

The cost matrix for all solution with arc 4-6:

i \ j	1	2	3	4	5	7	
1	∞	0	83	9	30	50	
2	0	∞	66	37	17	26	
3	29	1	∞	19	0	5	
5	0	18	53	4	∞	25	(-3)
6	0	85	8	∞	89	0	
7	18	0	0	0	58	∞	

Table 6-7 A Reduced Cost Matrix for that in Table 6-6.

total cost reduced: 96+3 = 99 (new lower bound)

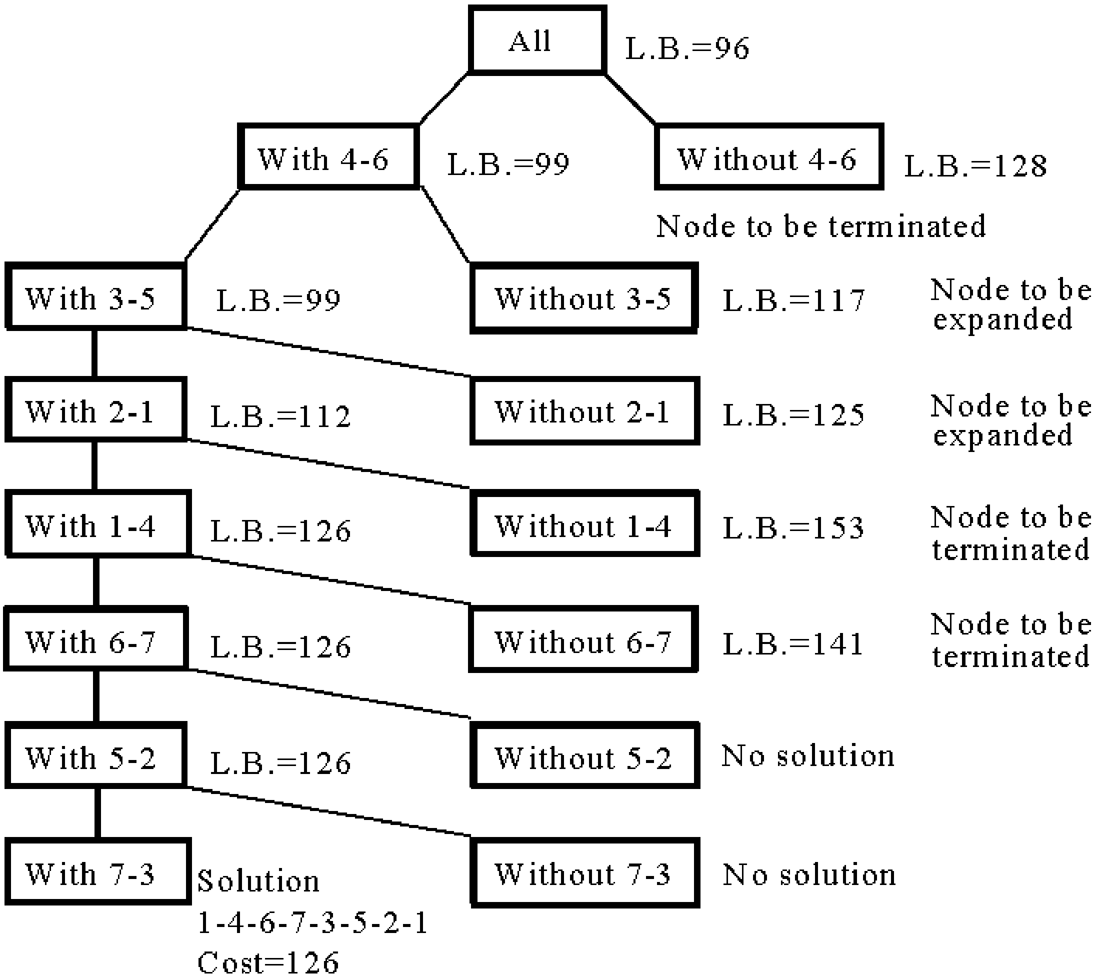


Fig 6-26 A Branch-and-Bound Solution of a Traveling Salesperson Problem.

● The 0/1 knapsack problem

positive integer $P_1, P_2, \ldots, P_n$ (profit)

$W_1, W_2, \ldots, W_n$ (weight)

M (capacity)

maximize $\sum_{i=1}^{n} P_i X_i$

subject to $\sum_{i=1}^{n} W_i X_i \le M$ $\quad X_i = 0$ or 1, $i = 1, \ldots, n$.

The problem is modified:

minimize $-\sum_{i=1}^{n} P_i X_i$

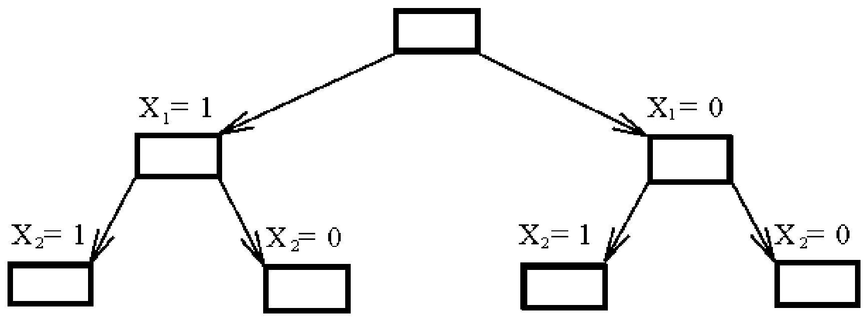


Fig. 6-27 The Branching Mechanism in the Branch-and-Bound Strategy to Solve 0/1 Knapsack Problem.

e.g. n = 6, M = 34

i	1	2	3	4	5	6
P_i	6	10	4	5	6	4
W_i	10	19	8	10	12	8

$(P_i/W_i \ge P_{i+1}/W_{i+1})$

a feasible solution: $X_1 = 1, X_2 = 1, X_3 = 0, X_4 = 0,$

$X_5 = 0, X_6 = 0$

$-(P_1+P_2) = -16$ (<u>upper bound</u>)

Any solution higher than -16 can not be an optimal solution.

Relax our restriction from $X_i = 0$ or 1 to $0 \leq X_i \leq 1$ (knapsack problem)

Let $-\sum_{i=1}^{n} P_i X_i$ be an optimal solution for 0/1 knapsack problem and $-\sum_{i=1}^{n} P_i X_i'$ be an optimal solution for knapsack problem. Let $Y = -\sum_{i=1}^{n} P_i X_i$,

$Y' = -\sum_{i=1}^{n} P_i X_i'$.

$\Rightarrow Y' \leq Y$

We can use the greedy method to find an optimal solution for knapsack problem:

$X_1 = 1$, $X_2 = 1$, $X_3 = 5/8$, $X_4 = 0$, $X_5 = 0$, $X_6 = 0$

$-(P_1+P_2+5/8P_3) = -18.5$ (lower bound)

-18 is our lower bound. (only consider integers)

$\Rightarrow -18 \leq$ optimal solution ≤ -16

optimal solution: $X_1 = 1$, $X_2 = 0$, $X_3 = 0$, $X_4 = 1$, $X_5 = 1$, $X_6 = 0$

$-(P_1+P_4+P_5) = -17$

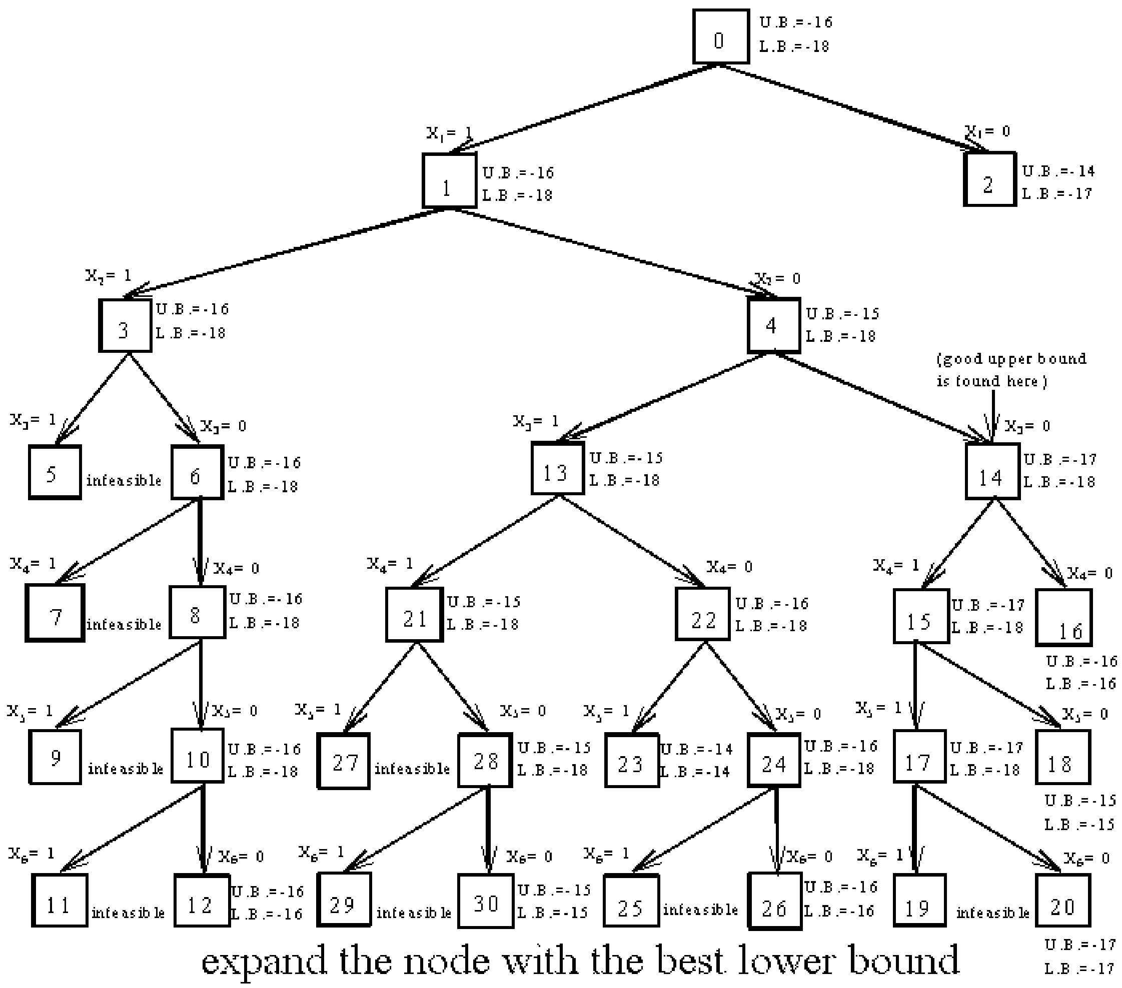


Fig. 6-28 0/1 Knapsack Problem Solved by Branch-and-Bound Strategy

- **The A* algorithm**

  used to solve optimization problems.
  using the best-first strategy.
  If a feasible solution (goal node) is obtained, then it is optimal and we can stop.

cost function of node n: f(n)

$$f(n) = g(n) + h(n)$$

g(n): cost from root to node n.
h(n): estimated cost from node n to a goal node.
$h^*(n)$: "real" cost from node n to a goal node.

$h(n) \le h^*(n)$

$\Rightarrow f(n) = g(n) + h(n) \le g(n)+h^*(n) = f^*(n)$

e.g.

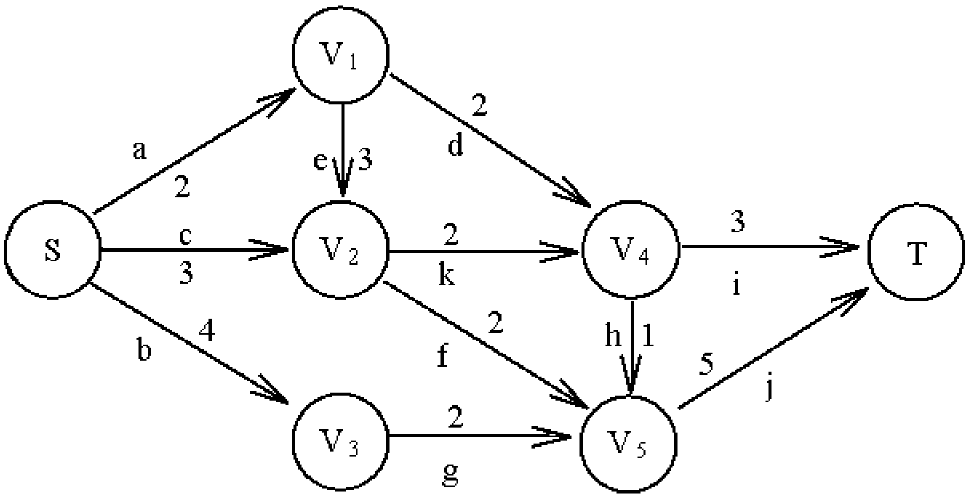
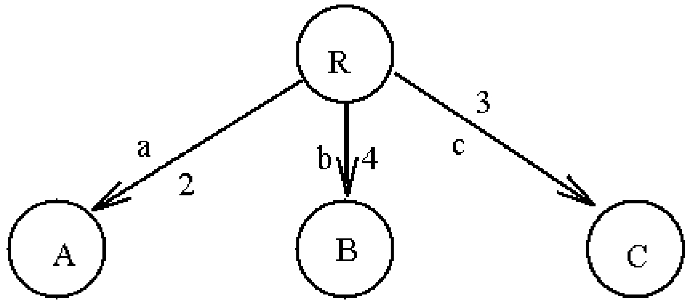


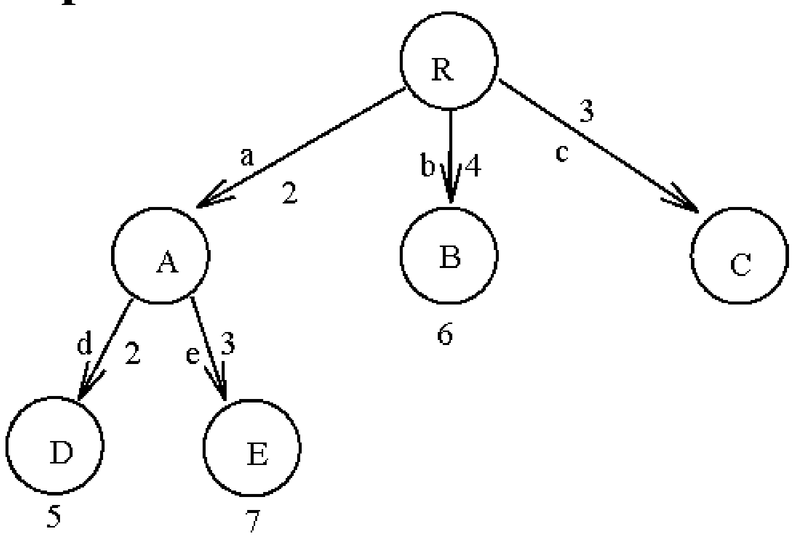
Fig. 6-36 A Graph to Illustrate A* Algorithm.

Step 1.



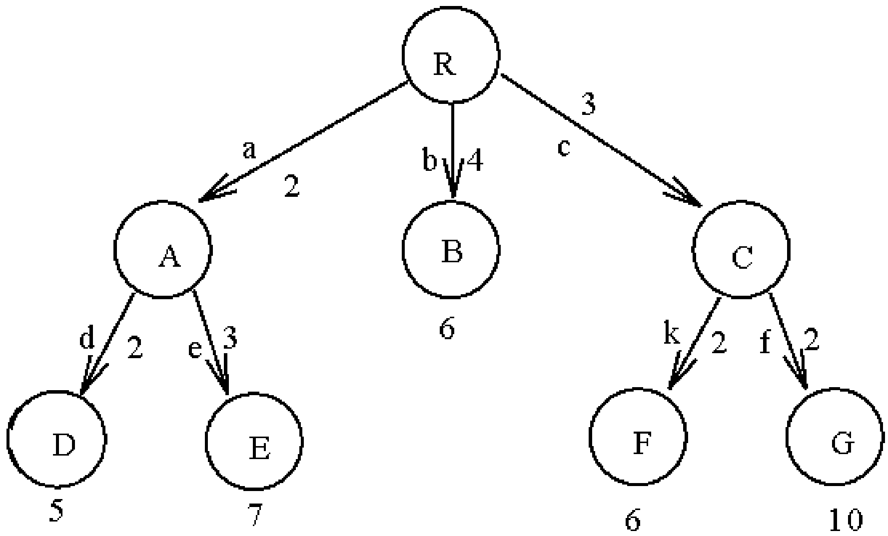
g(A)=2	h(A)=min{2,3}=2	f(A)=2+2=4
g(B)=4	h(B)=min{2}=2	f(B)=4+2=6
g(C)=3	h(C)=min{2,2}=2	f(C)= 3+2=5

Step 2. Expand A



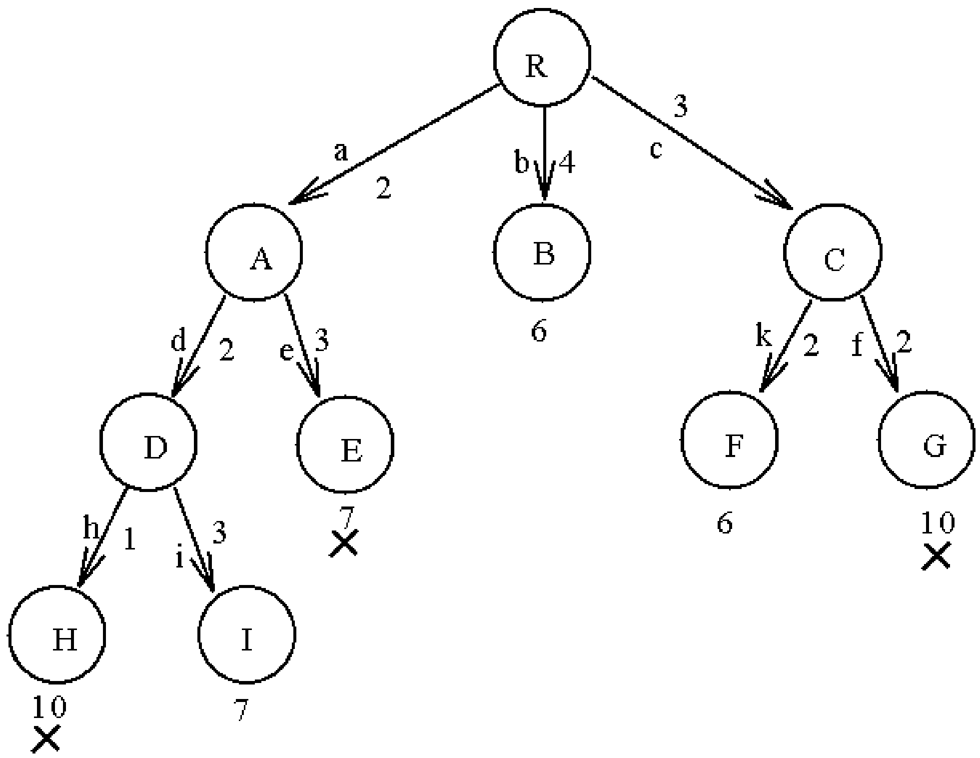
g(D)=2+2=4	h(D)=min{3,1}=1	f(D)=4+1=5
g(E)=2+3=5	h(E)=min{2,2}=2	f(E)=5+2=7

Step 3. Expand C



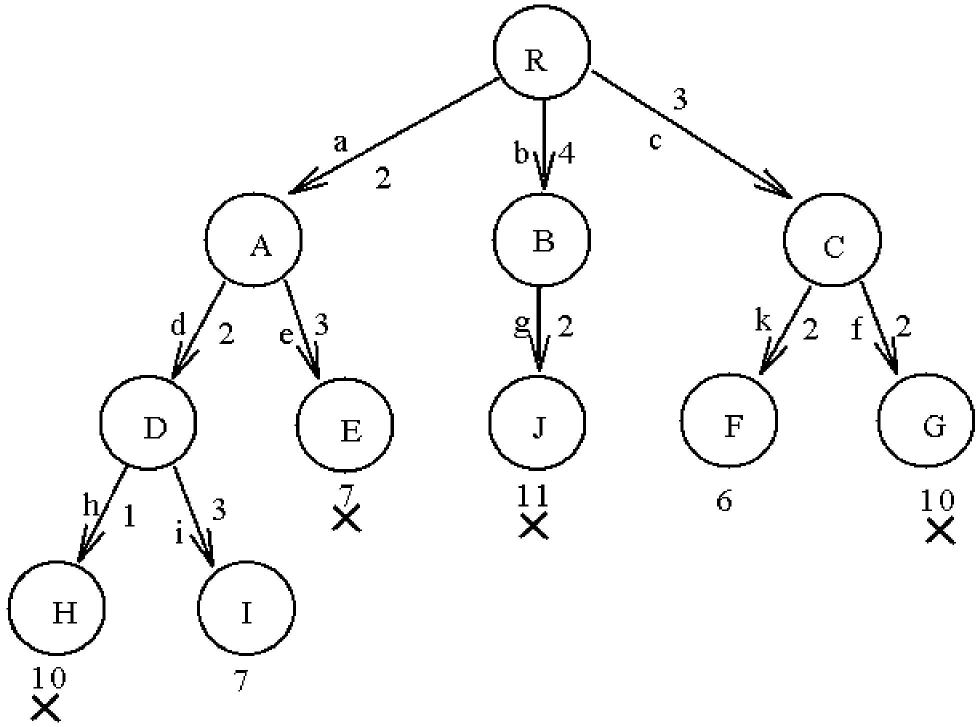
g(F)=3+2=5 h(F)=min{3,1}=1 f(F)=5+1=6
g(G) =3+2=5 h(G)=min{5}=5 f(G) =5+5=10

Step 4. Expand D



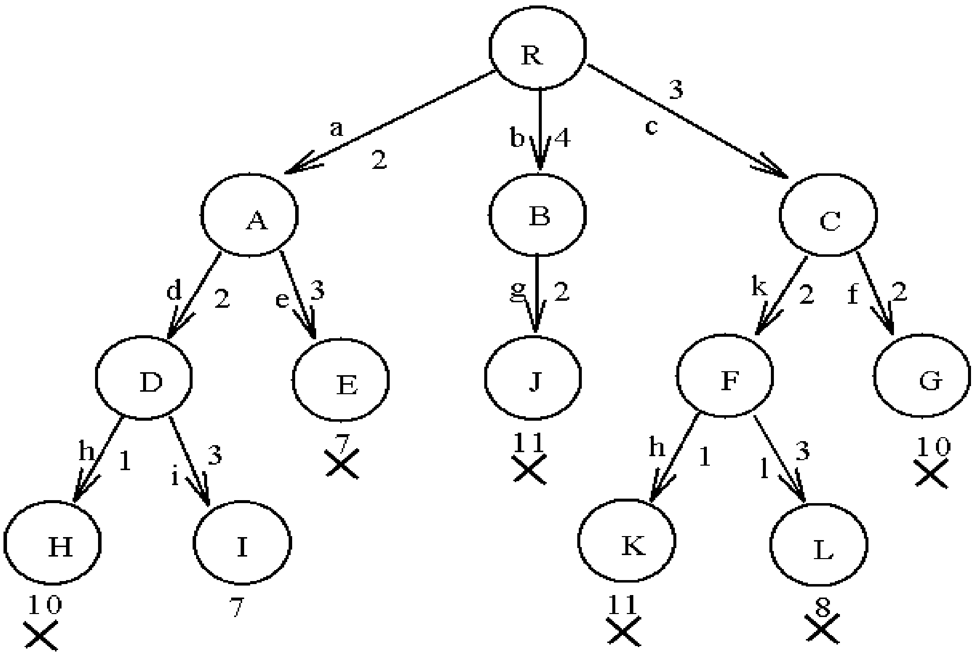
g(H)=2+2+1=5 h(H)=min{5}=5 f(H)=5+5=10
g(I)=2+2+3=7 h(I)=0 f(I)=7+0=7

Step 5. Expand B



g(J)=4+2=6　　　h(J)=min{5}=5　　f(J)=6+5=11

Step 6. Expand F



g(K)=3+2+1=6　　h(K)=min{5}=5　　f(K)=6+5=11

g(L)=3+2+3=8　　h(L)=0　　　　　　f(L)=8+0=8

● The channel routing problem

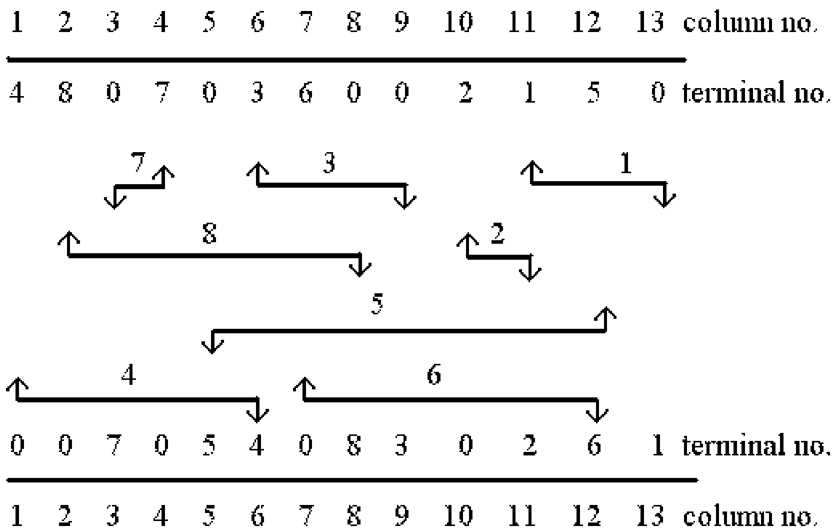
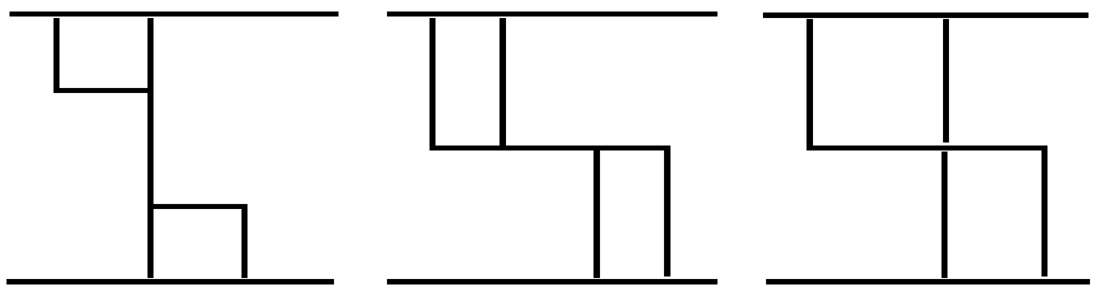
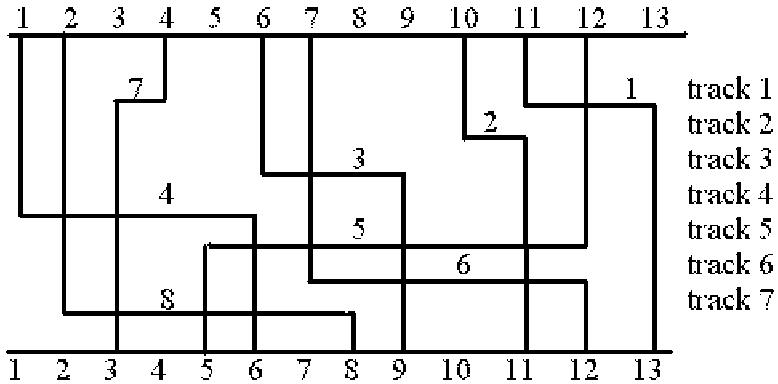


Fig. 6-40 A Channel Specification

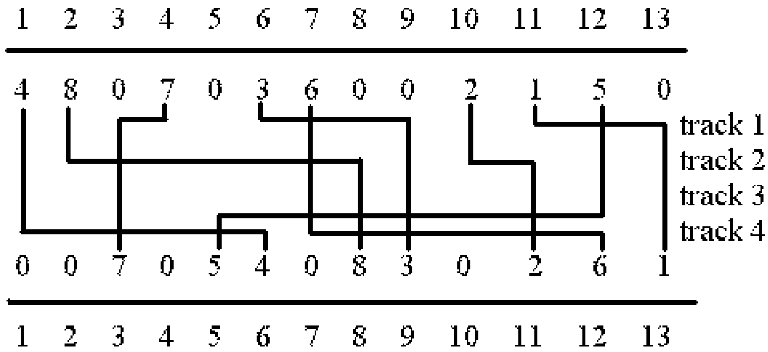
illegal wirings:



We want to find a layout which minimizes the number of tracks.
a feasible layout:

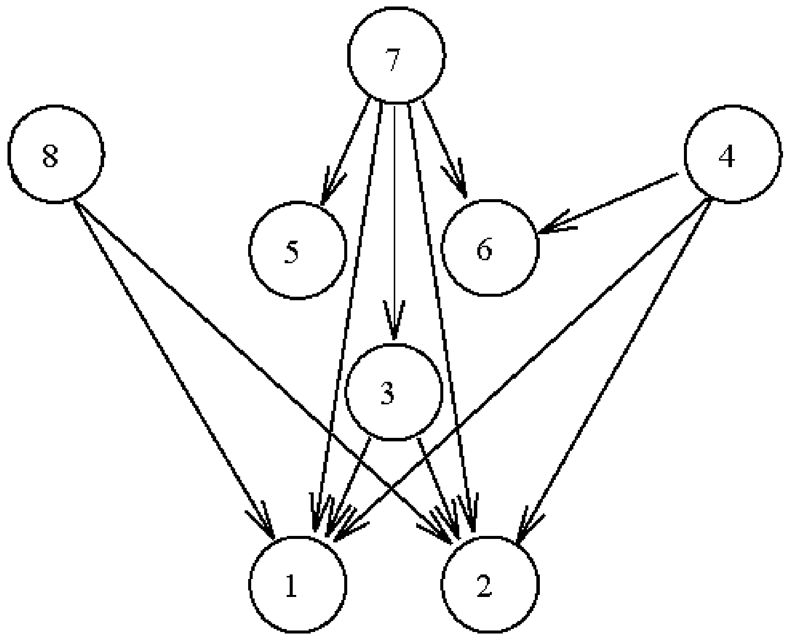


an optimal layout:



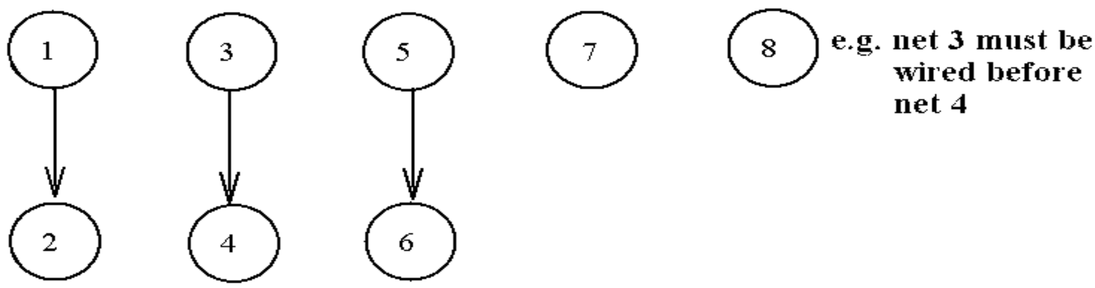
This problem is NP-complete.

horizontal constraint graph(HCG)



e.g. net 8 must be to the left of net 1 and net 2 if they are in the same track.

vertical constraint graph:



max. cliques in HCG: {1,8}, {1,3,7}, {5,7}
Each max. clique can be assigned to a track.

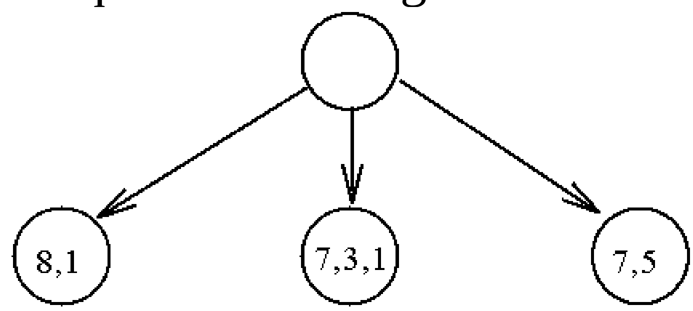


Fig. 6-46 The First Level of a Tree to Solve a Channel Routing Problem

$f(n) = g(n) + h(n)$, g(n): the level of the tree
h(n): maximal local density

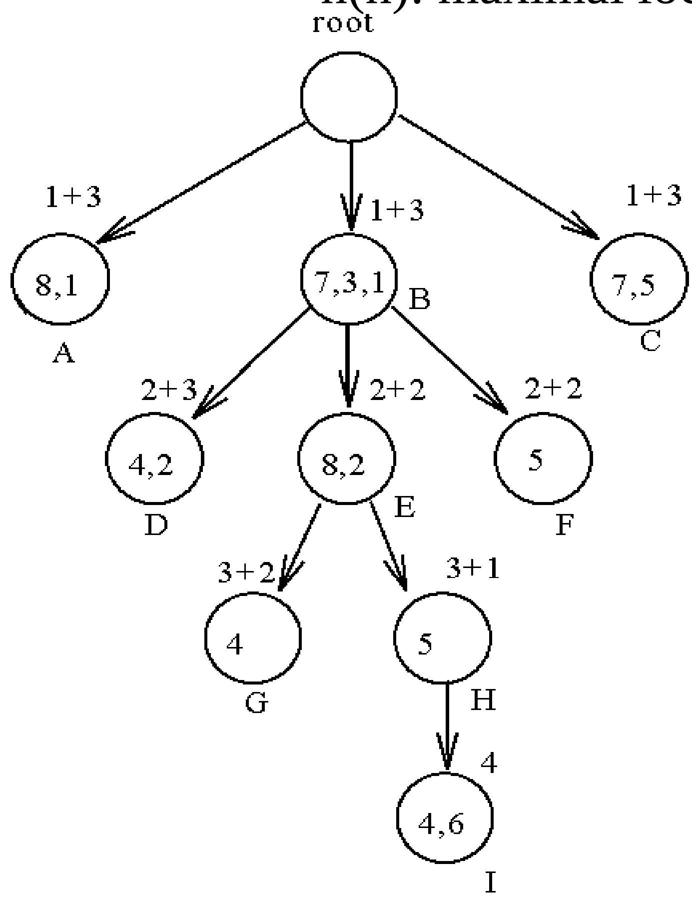


Fig 6-48 A Partial Solution Tree for the Channel Routing Problem by Using A^* Algorithm.