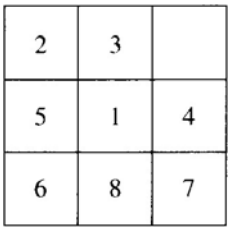
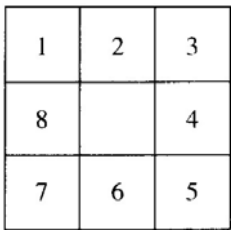
Tree Searching Strategies

There are many problems whose solution spaces (x_1, x_2, …, x_n) can be represented as trees, and their optimal solutions (x_1^*, x_2^*, …, x_n^*) can be obtained by systematically searching the trees.

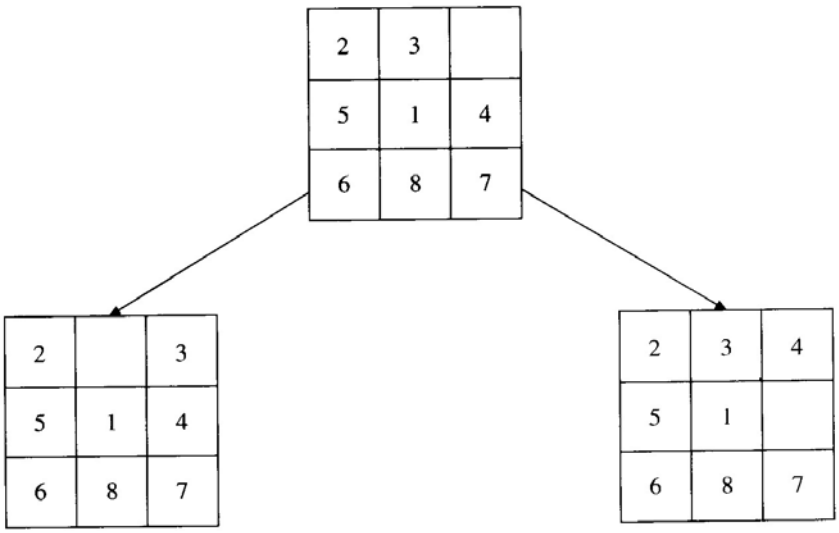
Ex. The 8-puzzle problem.



initial state

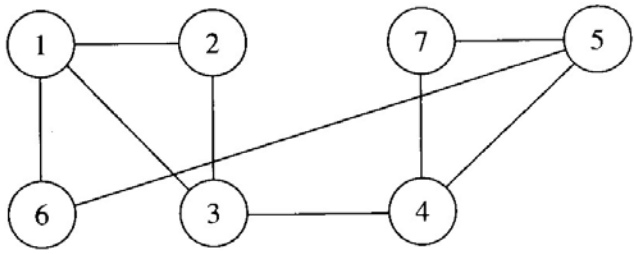


goal state

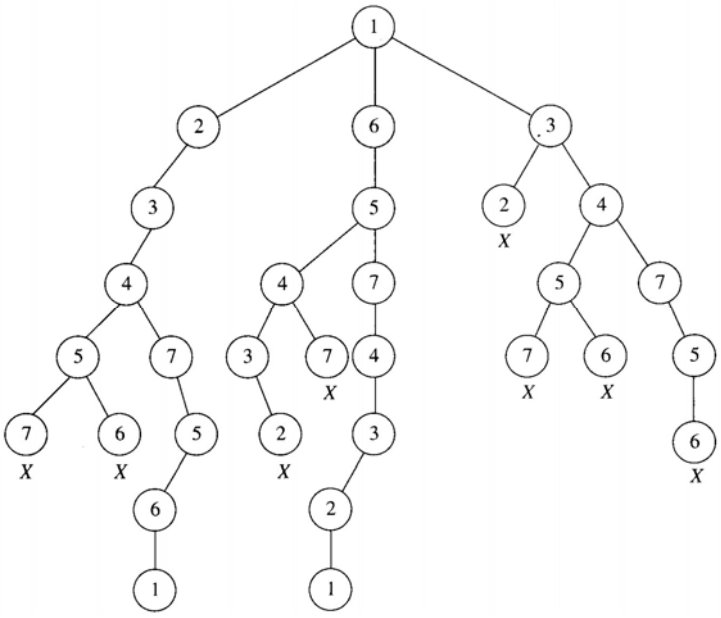


two possible moves from the initial state

Ex. The hamiltonian cycle problem.



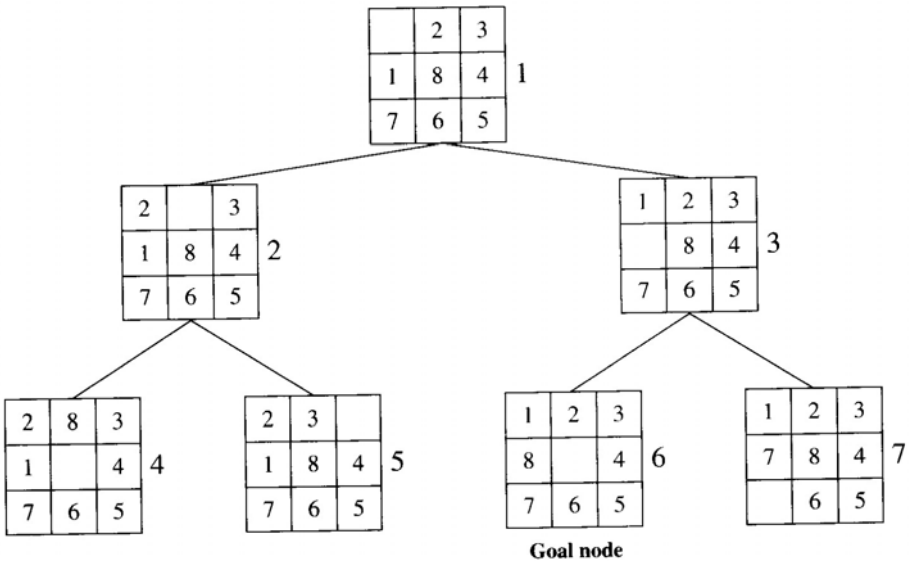
(1, 2, 3, 4, 7, 5, 6, 1) is a hamiltonian cycle.



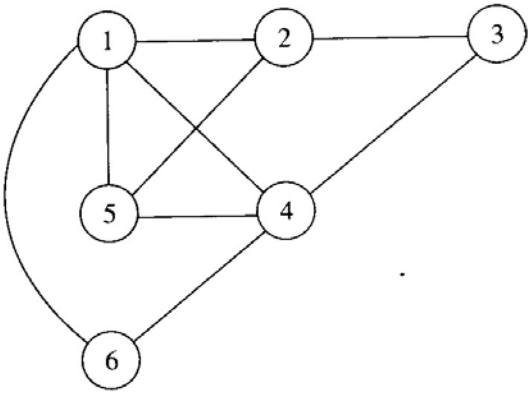
A tree representation of the solution space.

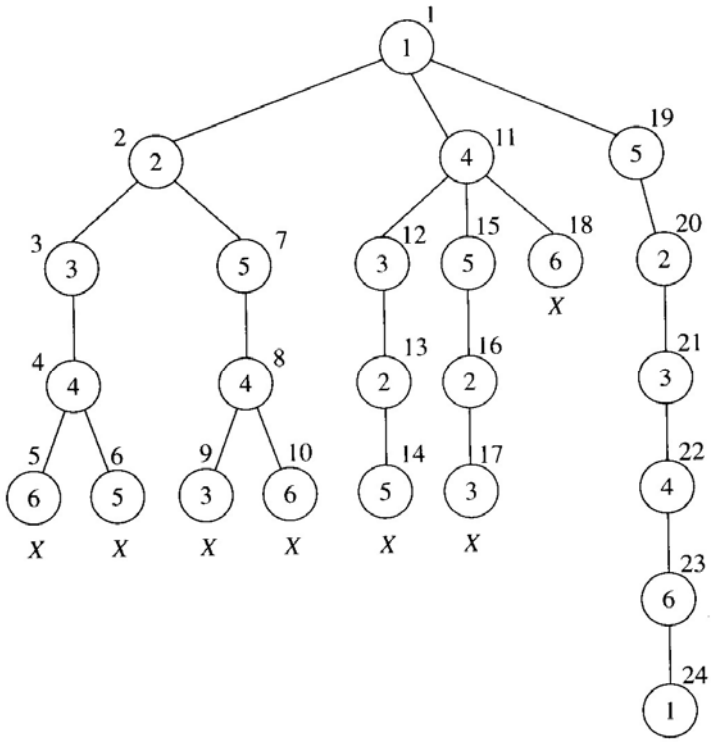
- **Breadth-First Search**

Nodes are searched one level after another.



- **Depth-First Search (Backtracking)**



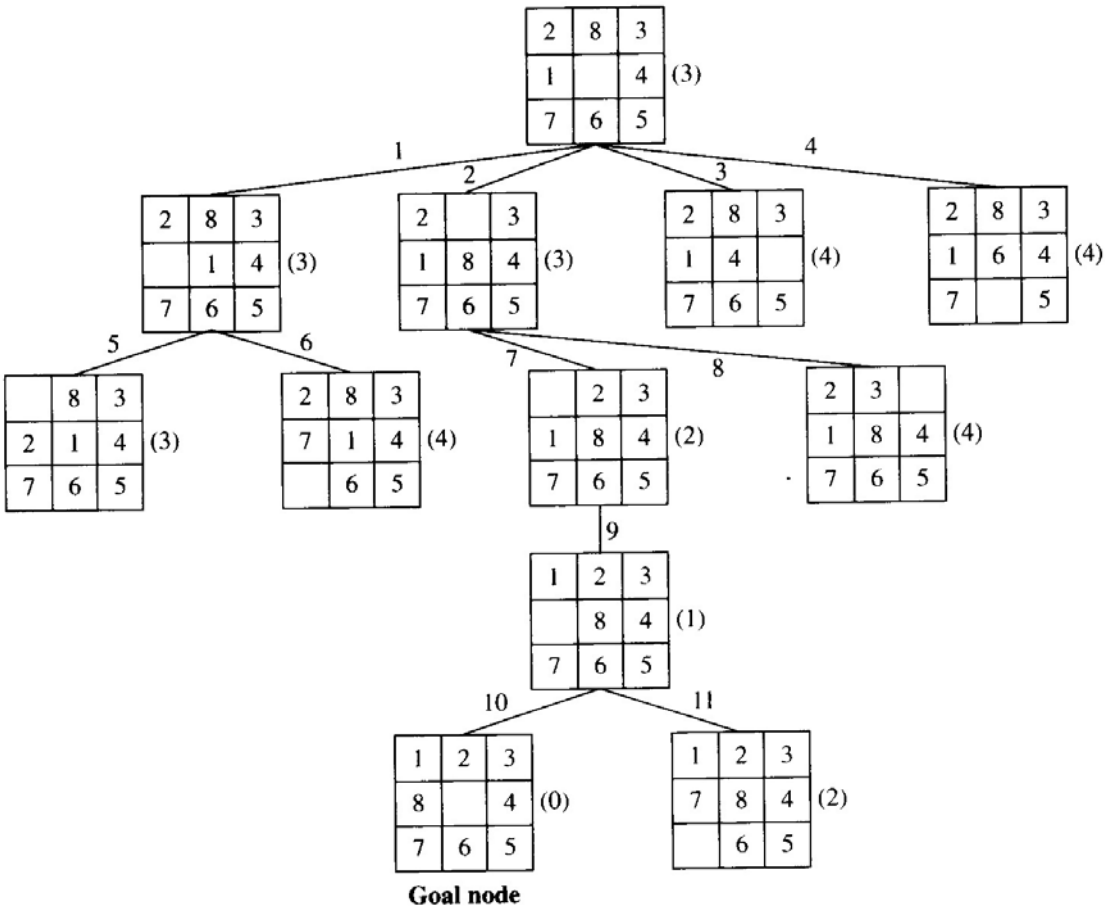


Finding a hamiltonian cycle by depth-first search.

- **Best-First Search**
  (Least-Cost-First Search)

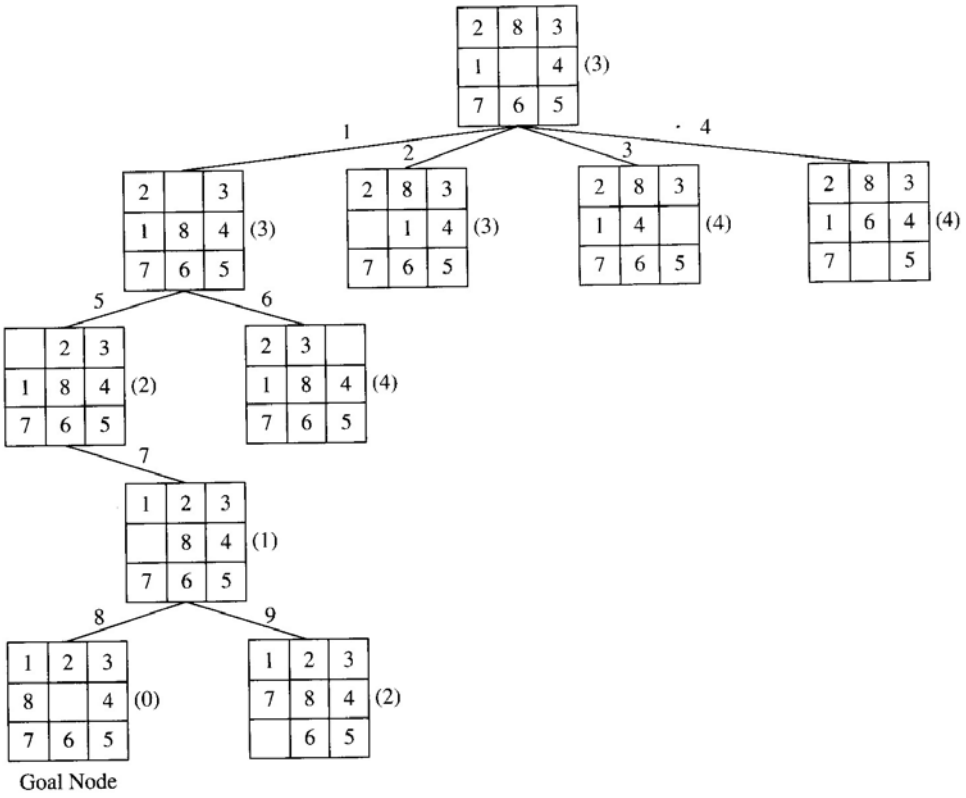
Select the next expanding node by the aid of an evaluation function

For example, the number of misplacements between a node and the goal state can be considered in solving the 8-puzzle problem.



- # Hill Climbing

A variant of depth-first search in which some heuristic method (or evaluation function) is applied for node branching



- **Branch-and-Bound**

  Solve optimization problems by means of “hill climbing” (“branch”) and “bound”

Consider a min-problem (max-problem) and the corresponding search tree.

Each node evaluates a lower (upper) bound on the best value that can be obtained from the subtree rooted at the node.

The smallest (greatest) feasible value obtained thus far is maintained as an upper bound on the optimal value.

“branch” : **expand the node with the least lower bound (greatest upper bound), similar to best-first search (i.e., select the branch with the highest probability of reaching an optimal node)**

“bound” : **fathom a node if the lower bound ≥ the upper bound (it is impossible to get a better solution than the upper bound below the node)**

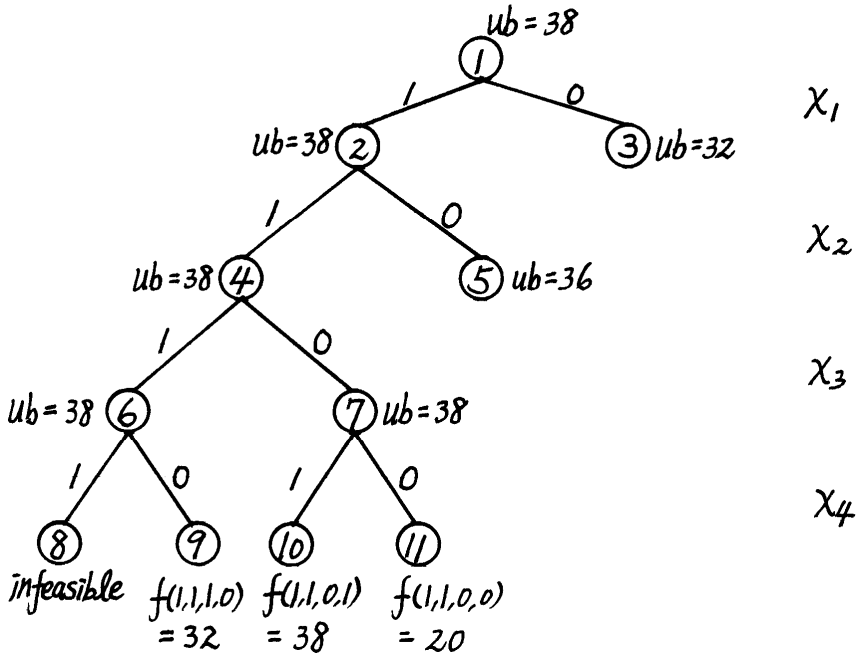
Ex. The 0/1 knapsack problem.

Consider the following instance.

$$z^* = \max 10x_1 + 10x_2 + 12x_3 + 18x_4$$

$$\text{s.t. } 2x_1 + 4x_2 + 6x_3 + 9x_4 \le 15$$

$$x_1, x_2, x_3, x_4 \in \{0, 1\}$$



Nodes ③ and ⑤ are fathomed.

nodes	the lower bound (lb)
①	32
②	32
③	32
④	32
⑤	32
⑥	32
⑦	38
⑧	38
⑨	38
⑩	38
⑪	38

Assume that the values of $x_1, x_2, \ldots, x_k$ of the node have been determined.

Computation of lb :

1. **Sequentially examine $x_{k+1}, x_{k+2}, \ldots$ and assign 1 (0) to x_i $(i \geq k+1)$ if the constraint is satisfied (else).**

   ((x_1, x_2, x_3, x_4) thus obtained is a feasible solution)

2. **Compute $f(x_1, x_2, x_3, x_4)$.**

3. **lb $\leftarrow$ max{lb, $f(x_1, x_2, x_3, x_4)$}.**

Computation of the upper bound (ub) :

$$\text{ub} = \sum_{i=1}^{k} p_i x_i + z$$

$$z = \max \sum_{i=k+1}^{n} p_i x_i$$

$$\text{s.t.} \quad \sum_{i=k+1}^{n} w_i x_i \leq M - \sum_{i=1}^{k} w_i x_i$$

$$0 \leq x_i \leq 1.$$

The value of *z* can be computed using the greedy method we proposed earlier.

Ex. The traveling salesman problem.

Consider the following instance.

from \ to	v_1	v_2	v_3	v_4	v_5
v_1	∞	20	30	10	11
v_2	15	∞	16	4	2
v_3	3	5	∞	2	4
v_4	19	6	18	∞	3
v_5	16	4	7	16	∞

cost matrix

$$\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

reduced cost matrix (reduced cost = 25)

A matrix is *reduced* if and only if every column and every row contain at least one zero and all remaining entries are nonnegative.

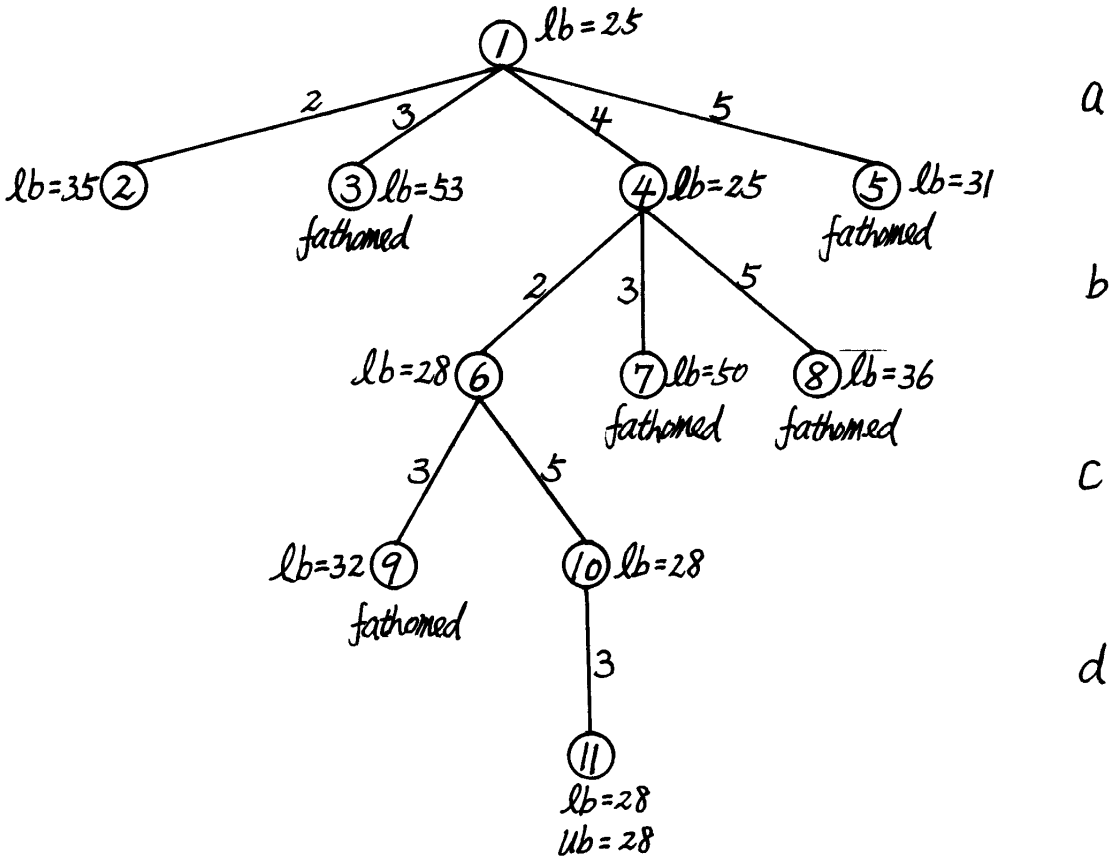
We can obtain the reduced cost matrix by subtracting constants from columns and rows of the cost matrix.

The total amount (reduced cost) subtracted is a lower bound of the minimum traveling mileage.

B&B (I) :

Assume that the traveling starts and ends at v_1.

Solution space : (v_1, a, b, c, d, v_1), where *abcd* is a permutation of v_2, v_3, v_4, v_5.



Every node is associated with a reduced cost matrix.

Let A be the reduced cost matrix for node R, and S be a child of R such that the branch (R, S) corresponds to edge (i, j).

The reduced cost matrix for S is computed as follows.

1. **Change all entries in row i and column j of A to ∞.**

   (preventing the use of other edges leaving i or entering j)

2. **Set $A(j, 1)$ to ∞ if S is not a leaf node.**

   (preventing the use of edge (j, 1))

3. **Further reduce the cost matrix.**
   Assume that the reduced cost is r.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ 0 & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix} \quad \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & \infty & 0 \\ 0 & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & 0 & \infty & \infty \end{bmatrix} \quad \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \end{bmatrix}$$

reduced cost matrices

for node 4, $r = 0$ for node 6, $r = 0$ for node 10, $r = 0$

Computation of lb :

$$\mathbf{lb}_S = \mathbf{lb}_R + A(i, j) + r,$$

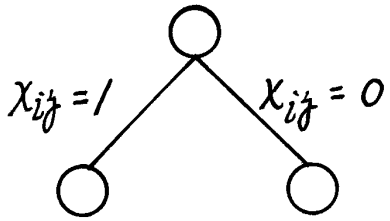
where $\mathbf{lb}_R$ ($\mathbf{lb}_S$) is the lower bound for the node R (S).

If S is a leaf node, then $\mathbf{lb}_S$ is the length of the tour represented by S.

B&B(II) :

solution space : $(x_{12}, x_{13}, \ldots, x_{54})$,

where $x_{ij} = 1$ **if edge** (i, j) **is included in the tour and** $x_{ij} = 0$ **else.**



Every tour contains five edges.

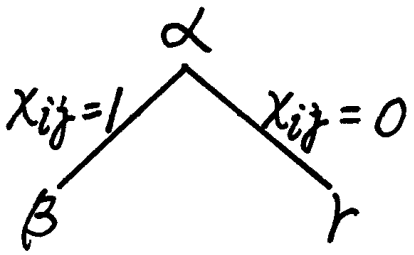
The following are two heuristics to select x_{ij} for branching :

1. **Select x_{ij} that has a maximal lb value for the right subtree (i.e., $x_{ij} = 0$).**
   (The lb value is evaluated all the same as B&B (I)).

2. **Select x_{ij} that has a maximal difference between the lb values for the left ($x_{ij} = 1$) and right ($x_{ij} = 0$) subtrees.**

(Other heuristics are also possible.)

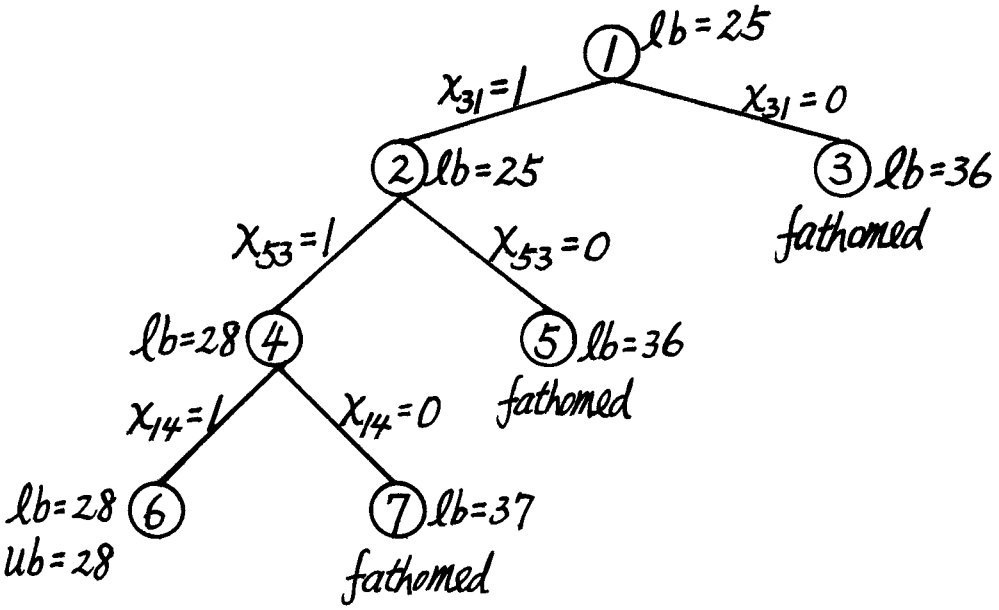
Assume that the heuristic 1 is adopted.



Let *A* be the reduced cost matrix for node α.

1. **The reduced cost matrix for node β is obtained all the same as B&B (I).**

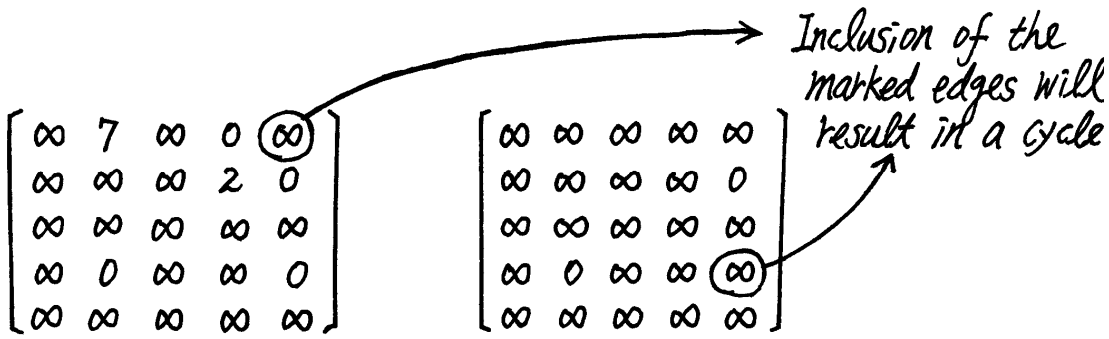
2. **The reduced cost matrix for node γ is obtained by changing $A(i, j)$ to ∞ and then reducing all rows and columns.**


1
lb = 25
$x_{31} = 1$
$x_{31} = 0$
2
lb = 25
3
lb = 36
fathomed
$x_{53} = 1$
$x_{53} = 0$
lb = 28
4
5
lb = 36
fathomed
$x_{14} = 1$
$x_{14} = 0$
lb = 28
ub = 28
6
7
lb = 37
fathomed

$$\begin{bmatrix} \infty & 10 & \infty & 0 & 1 \\ \infty & \infty & 11 & 2 & 0 \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 3 & 12 & \infty & 0 \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix} \qquad \begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 1 & \infty & 11 & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 4 & 3 & 12 & \infty & 0 \\ 0 & 0 & 0 & 12 & \infty \end{bmatrix}$$

reduced cost matrices

for node 2 **for node 3**



reduced cost matrices

for node 4 **for node 6**

Since every tour contains five edges, the remaining two available edges ((2, 5) and (4, 2)) at node 6 are selected.

Experimental results : the ratio of

$$1 - \frac{\text{the number of nodes generated}}{\text{the number of nodes in the complete tree}}$$

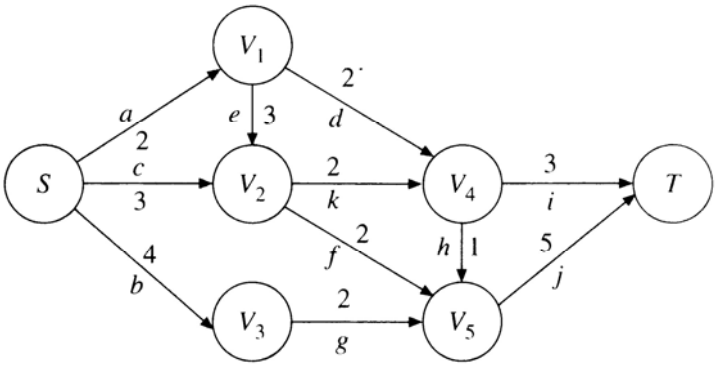
for the module assignment problem

		COMMUNICATION COSTS								
		[0,10]			[0,50]			[0,100]		
n	m	p=0.2	p=0.5	P=0.8	p=0.2	p=0.5	p=0.8	p=0.2	p=0.5	p=0.8
2	8	0.96047	0.94324	0.94011	0.95107	0.93072	0.91115	0.92915	0.92681	0.95655
	9	0.97087	0.95992	0.94584	0.93919	0.91964	0.94701	0.93137	0.94819	0.97204
	10	0.98114	0.96668	0.92447	0.92701	0.95789	0.97743	0.95280	0.96844	0.98114
	11	0.98539	0.97152	0.96683	0.95228	0.97015	0.98432	0.94661	0.97475	0.99291
	12	0.99313	0.98723	0.98151	0.98244	0.97384	0.98986	0.96574	0.98234	0.99499
	13	0.98985	0.98431	0.96492	0.97630	0.98292	0.99671	0.98057	0.99298	0.99737
	14	0.99552	0.98761	0.96805	0.97910	0.98860	0.99511	0.98453	0.99705	0.99822
	15	0.99419	0.98428	0.98315	0.98531	0.99272	0.99681	0.98669	0.99860	0.99894
	16	0.99732	0.98750	0.97985	0.98997	0.99576	0.99841	0.98377	0.99871	0.99956
	17	0.99773	0.98429	0.98527	0.99602	0.99685	0.99922	0.99282	0.99947	0.99968
	18	0.99713	0.99446	0.99496	0.98682	0.99847	0.99935	0.98794	0.99950	0.99980
	19	0.99862	0.99207	0.98998	0.99391	0.99868	0.99983	0.99741	0.99961	0.99992
	20	0.99888	0.99738	0.99420	0.99533	0.99934	0.99986	0.99819	0.99981	0.99995
3	5	0.95109	0.95109	0.94285	0.95109	0.92967	0.93461	0.91483	0.92802	0.93626
	6	0.97767	0.96669	0.96614	0.97054	0.94473	0.96120	0.94309	0.95846	0.96669
	7	0.98615	0.98414	0.97298	0.97774	0.95634	0.96274	0.98634	0.96969	0.98908
	8	0.99489	0.99325	0.99325	0.98831	0.97959	0.98611	0.98343	0.98252	0.99410
	9	0.99830	0.99634	0.99271	0.99242	0.98586	0.99592	0.98442	0.99100	0.99734
	10	0.99867	0.99789	0.99525	0.98919	0.98936	0.99701	0.99087	0.99728	0.99915
	11	0.99956	0.99802	0.99744	0.99502	0.99584	0.99933	0.99384	0.99787	0.99946
	12	0.99970	0.99765	0.99676	0.99667	0.99687	0.99961	0.99334	0.99900	0.99987
	13	0.99955	0.99896	0.99710	0.99782	0.99796	0.99990	0.99918	0.99950	0.99992
4	4	0.94780	0.94780	0.94545	0.94310	0.90557	0.93372	0.94310	0.93137	0.94545
	5	0.98344	0.97289	0.97465	0.97992	0.97875	0.97172	0.97113	0.96058	0.95882
	6	0.99454	0.99176	0.99088	0.98736	0.98443	0.98472	0.99102	0.97462	0.98209
	7	0.99775	0.99574	0.99515	0.99453	0.99003	0.99090	0.99398	0.99043	0.99306
	8	0.99923	0.99789	0.99750	0.99347	0.99165	0.99828	0.99217	0.99622	0.99752
	9	0.99957	0.99904	0.99871	0.99827	0.99671	0.99753	0.99495	0.99564	0.99951
	10	0.99982	0.99957	0.99883	0.99852	0.99736	0.99933	0.99869	0.99912	0.99971
5	3	0.89743	0.89743	0.89102	0.88461	0.87820	0.87820	0.89743	0.89743	0.89743
	4	0.97055	0.96414	0.96542	0.96927	0.96670	0.95390	0.96414	0.96158	0.95006
	5	0.99078	0.99052	0.98822	0.99180	0.98259	0.97388	0.98899	0.97772	0.98745
	6	0.99790	0.99703	0.99278	0.99170	0.98709	0.98540	0.99441	0.99139	0.99267
	7	0.99894	0.99858	0.99811	0.99751	0.99183	0.99298	0.99543	0.99145	0.99798
	8	0.99981	0.99955	0.99862	0.99832	0.99833	0.99725	0.99943	0.99675	0.99879
	9	0.99993	0.99971	0.99889	0.99901	0.99835	0.99924	0.99849	0.99898	0.99979
6	3	0.92664	0.92664	0.92664	0.92200	0.92664	0.92200	0.92200	0.92664	0.92200
	4	0.98083	0.98392	0.97852	0.98315	0.97543	0.96540	0.97852	0.97389	0.96308
	5	0.99629	0.99372	0.99539	0.99217	0.98883	0.98626	0.99320	0.98844	0.99191
	6	0.99921	0.99820	0.99794	0.99440	0.99460	0.99631	0.99625	0.99610	0.99687
	7	0.99937	0.99864	0.99886	0.99763	0.99801	0.99620	0.99867	0.99860	0.99838
	8	0.99992	0.99961	0.99885	0.99962	0.99788	0.99926	0.99960	0.99811	0.99975

- **A^* Algorithms**

a special kind of branch-and-bound algorithms

Ex. Find a shortest path from *S* to *T* in the following graph.



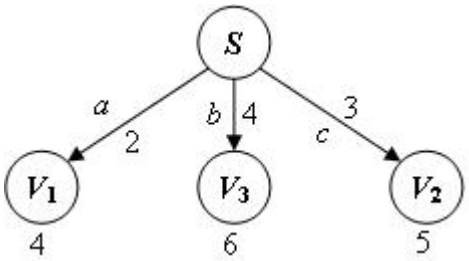
A cost function *f*(*i*) is evaluated for each node *i*, which is required not to be greater than the length of any *S*-to-*T* path passing node *i*.

Actually, *f*(*i*) is a lower bound of node *i* in terms of branch-and-bound algorithms.

A feasible *f*(*i*) is defined as follows :

> **the length of the *S*-to-*i* path plus the minimum distance from *i* to its neighboring nodes (exclusive of the one contained in the *S*-to-*i* path).**

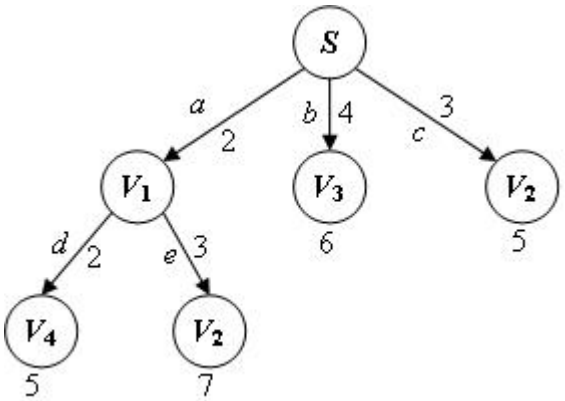
Branching by the least-cost-first scheme :

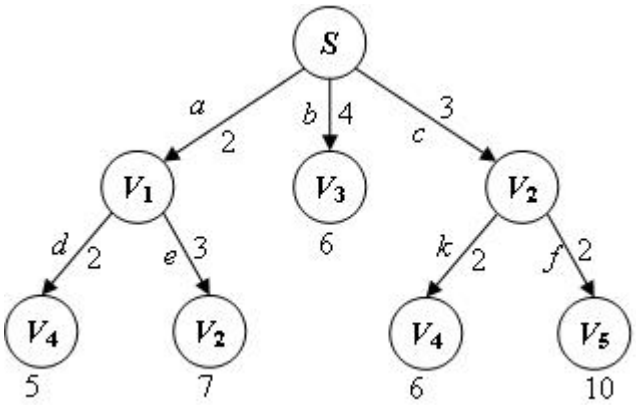


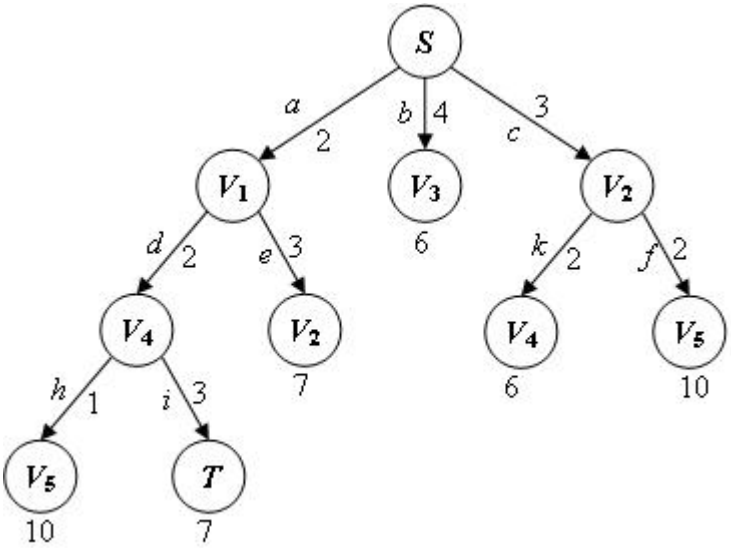
$$f(V_1) = 2 + 2 = 4$$

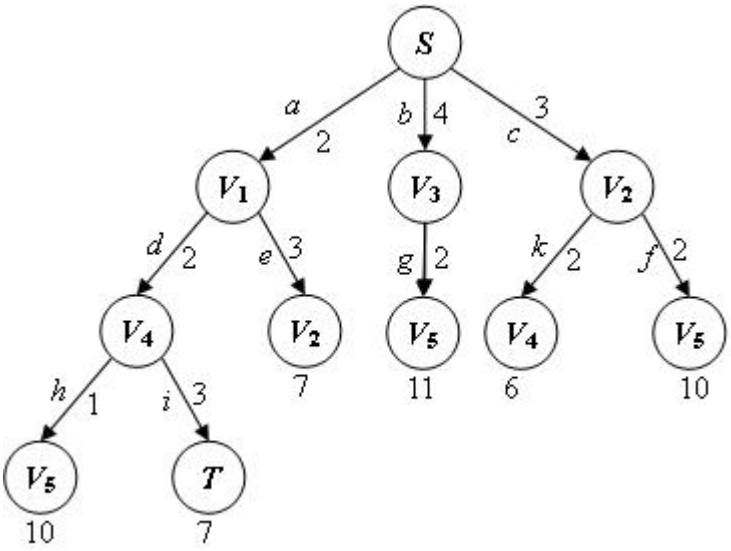
$$f(V_2) = 3 + 2 = 5$$

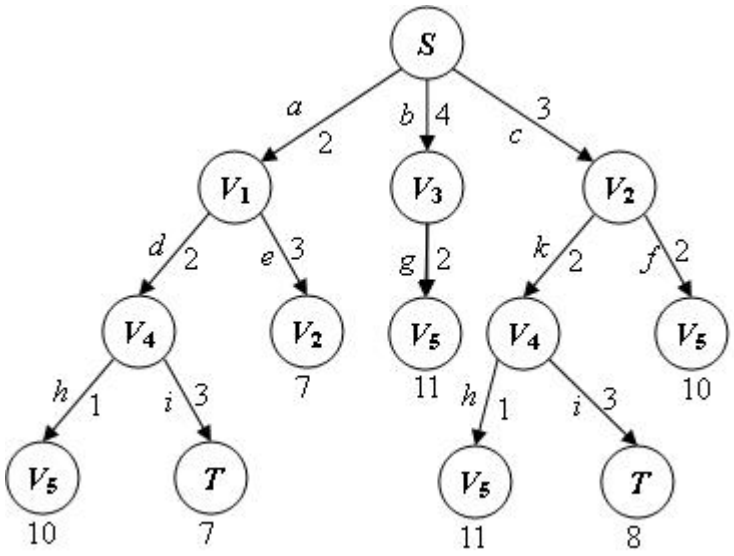
$$f(V_3) = 4 + 2 = 6$$




S
a
2
b
4
c
3
V1
V3
V2
6
d
2
e
3
k
2
f
2
V4
V2
V4
V5
5
7
6
10


S
a
2
b
4
c
3
V1
V3
V2
6
d
2
e
3
k
2
f
2
V4
V2
V4
V5
7
6
10
h
1
i
3
V5
T
10
7


S
a
2
b
4
c
3
V1
V3
V2
d
2
e
3
g
2
k
2
f
2
V4
V2
V5
V4
V5
7
11
6
10
h
1
i
3
V5
T
10
7



The next branching occurs at a goal node.

$\Rightarrow$ **the execution terminates with the shortest path**

$$S \rightarrow V_1 \rightarrow V_4 \rightarrow T$$

The cost of a goal node is an upper bound in terms of branch-and-bound algorithms.

Program Assignment 2 :

Write an executable program to solve the 0/1 knapsack problem and compute the ratio of

$$1 - \frac{\text{the number of nodes generated}}{\text{the number of nodes in the complete tree}}.$$